LOCALIZATION FOR ROBOTIC ASSEMBLIES
WITH POSITION UNCERTAINTY

by

SIDDHARTH R. CHHATPAR

Submitted in partial fulfillment of the requirements
For the degree of Doctor of Philosophy

Dissertation Advisor: Dr. Michael S. Branicky

Department of Mechanical and Aerospace Engineering
CASE WESTERN RESERVE UNIVERSITY

January 2006

Table of Contents

Table of Contents	iv
List of Tables	vii
List of Figures	viii
Acknowledgments	xviii
Abstract	xix
1 Introduction	1
1.1 Problem Formulation and Statement	6
1.2 Solution: General Localization Strategy	11
1.3 Previous Work	11
1.3.1 Planning Under Uncertainty	12
1.3.2 Peg-in-Hole Assemblies with Position Uncertainty . . .	13
1.3.3 Related Work	20
1.4 Contributions of this Dissertation	22
1.5 Dissertation Organization	23
2 Experimental Testbed Overview	26
2.1 The ParaDex Robot	27
2.2 Natural Admittance Controller	27
2.3 Example Assemblies	29
2.3.1 Circular Peg-in-Hole Assembly	30
2.3.2 Square Peg-in-Hole Assembly	30
2.3.3 Pentagonal Peg-in-Hole Assembly	32
2.3.4 Key-Lock Assembly	33
2.3.5 Summary of Example Assemblies	36
3 From Exhaustive Searching to Intelligent Localization	37
3.1 Analysis of Assembly Uncertainty and Exhaustive Search Strategies	38
3.1.1 Configuration Space View of the Search Region	39
3.1.2 Exhaustive Coverage of Search Region	40
3.1.3 Exhaustive Search Path Examples	41

3.2	Intelligent Localization Strategies	47
3.2.1	The Precession Strategy	47
3.2.2	The Sweep Strategy	51
3.3	General Localization Strategy	55
3.4	Conclusions	63
4	Assembly Localization Using Analytical Maps	64
4.1	Generating an Analytical Map	65
4.1.1	Circular Peg-in-Hole Assembly	66
4.1.2	Square Peg-in-Hole Assembly	71
4.2	Localization Using Analytical Maps	74
4.2.1	Localization for Circular Peg-in-Hole Assembly	76
4.2.2	Localization for Square Peg-in-Hole Assembly	83
4.3	Conclusions	87
5	Assembly Localization Using Sampled Maps	88
5.1	Sampled Contact Hyper-Surface Map	89
5.2	Assembly Sufficiency, Goal Region, and Approximate Localization	91
5.3	Sampling Conditions	95
5.4	Sampling the Contact C-Space Map	98
5.5	Localization Using Sampled Maps	101
5.5.1	Matching Observations to the C-Space Map	104
5.6	Change in Problem Formulation for Localization Using Sampled Maps	109
5.7	Results: Localization using Sampled Maps	111
5.7.1	Planar Assembly Example	111
5.7.2	Circular Peg-in-Hole Assembly Simulations	117
5.7.3	Square Peg-in-Hole Assembly	123
5.8	Conclusions	128
6	Assembly Localization Using the Cell Approach	130
6.1	The Cell Approach	131
6.1.1	Algorithm for Localization Using Cell Approach	137
6.1.2	Localization in Stages of Increasing Resolution	143
6.2	Simulation Results for Probe-on-Surface	145
6.2.1	Implementation Details	145
6.2.2	Simulation Results with Fixed Cell Size	148
6.2.3	Simulation Results with Changing Cell Size	153
6.3	Conclusions	154

7	Assembly Localization Using Particle Filtering	156
7.1	Introduction	156
7.1.1	The Probabilistic Framework	157
7.1.2	Particle Filters	159
7.2	Localization Strategy Implementation Using Particle Filtering	167
7.2.1	Computer Simulations	167
7.2.2	Robotic Assemblies	168
7.2.3	Active Localization	178
7.3	Conclusions	179
8	Conclusions and Future Work	182
8.1	Summary	182
8.2	Conclusions	183
8.3	Recommendations	186
8.4	Future Work	187
	Bibliography	191

List of Tables

2.1	Assembly clearance values for the various example assemblies.	36
2.2	Hole-configuration uncertainty regions for the various example assemblies.	36
8.1	Recommendations for the localization algorithm to be used based on the requirements and parameters of the assembly to be implemented.	188

List of Figures

1.1	Example vehicular transmission assemblies from Ford Motor Co.	2
1.2	Peg contacting hole.	7
1.3	Example 1 (top): localizing a probe on a curve (two-dimensional uncertainty). Example 2 (bottom): localizing a probe on a surface (three-dimensional uncertainty).	9
1.4	Example 3: Planar peg-hole localization (three-dimensional uncertainty).	10
1.5	An illustration of the concept of backward chaining. The states labelled as “sub-goal” and “sub-sub-goal” form the pre-images of the “final goal” and “sub-goal” states, respectively. Thus, starting from the final goal and chaining backward, paths can be created to reach the final goal starting from all possible initial conditions.	13
1.6	Peg-in-maze assembly: A keyed-peg has to be inserted through characteristically-shaped holes on five layers stacked in a container.	17
2.1	The ParaDex robot is a novel closed-chain mechanism with six parallel prismatic actuators. High stiffness, low friction, and a high payload make the ParaDex especially suitable for force control applications.	27
2.2	An example of a GUI that allows users to generate attractor trajectories and input parameters such as the virtual spring and damper values used by the controller.	28
2.3	Illustration of how the hybrid control scheme allows accurate tracking in selected dimensions ((x, y) -plane for the example shown) and compliance in other dimensions (vertical (z) axis for the example shown).	29
2.4	The circular peg-in-hole assembly.	30
2.5	The square peg-in-hole assembly.	32

2.6	The change in assembly clearance area with orientation error.	33
2.7	The pentagonal peg-in-hole assembly.	34
2.8	Set-up for key-lock assembly.	35
3.1	Assembly characterization in terms of C-space.	39
3.2	An example of an exhaustive coverage set: covering each point in the set guarantees exhaustive coverage of the search region w.r.t. assembly clearance.	41
3.3	Example of a search path that does not cover the search area exhaustively.	42
3.4	Solution path of approximate length $128c$ for a circular search area of radius $7c$, found using simulated annealing for an example exhaustive coverage set, $B \in \mathcal{A}$	43
3.5	Path consisting of concentric circles joined by a straight line. Path length is $85c$ for a circular search area of radius $7c$	44
3.6	An example spiral search path. Path length is approximately $102c$ for a circular search region of radius $7c$	46
3.7	The precession strategy: (a) The peg is initially tilted by 15° . (b) The peg touches the hole surface, and peg height h_1 is recorded. (c) As the tilt axis is rotated, the peg precesses. (d) The peg dips into the hole, height $h_2 < h_1$	49
3.8	Plot of the change in peg height as it precesses by 360° . The angle of the tilt-axis where the minimum height is recorded can be used to compute the direction of the hole center w.r.t. the peg-position. Moreover, the value of the change in height can be compared against a pre-acquired table to find the distance of the hole center from the peg-center.	50
3.9	Plot of the minimum peg height (z) that would be recorded during precession for different relative peg-hole positions in (x, y) sampled every 0.5 mm distance.	51
3.10	The sweep strategy: (a) The peg is initially tilted by 15° about a diagonal, (b) Peg is brought into contact with the hole-surface, (c) The peg is swept over the hole while maintaining contact, and peg height is recorded throughout, (d) Peg dips into hole and contact point on the peg shifts from vertex A to peg-edge.	53
3.11	Peg-height record (or, sweep signature) for the sweep as shown in Figure 3.10(a).	54
3.12	Both the actual hole configuration (a), and the mirror image (b), would produce the same sweep signature.	55

3.13	Worst-case scenario: Hole is at the edge of the uncertainty region. The sweep is always conducted so that vertex A passes through the center of the hole uncertainty region. If the hole uncertainty region is bounded by a square of edge-length l , then even for worst-case, the sweep-line will be distance of $l/2$ from the hole center, and vertex A will pass over the edge of the hole as shown.	56
3.14	Coordinate frames associated with the square peg-in-hole. . .	58
3.15	Three-dimensional contact hyper-surface slice for a square peg-in-hole assembly for the fixed relative orientation between peg and hole, as shown in (a).	59
3.16	Algorithm for implementing the general localization strategy. .	60
3.17	(a) Continuous surface and probe moves. (c) (Subset of) possible candidates for probe move. (e) Advancing the previous candidates. (b–f) Successive pruning of candidate transformations.	61
4.1	Analytical map for a round peg-in-hole assembly with matching relative orientations between the peg and the hole.	66
4.2	The region to be mapped encompasses the hole uncertainty region augmented by the peg move region.	67
4.3	Coordinate frames associated with the peg and the hole. . . .	68
4.4	Peg-hole contact hyper-surface with tilted peg.	69
4.5	The different regions of contact on the peg and hole that contribute to corresponding regions on the contact surface map for a circular peg-in-hole assembly.	70
4.6	The wire-figure approximation of the peg projected into the hole-plane. Intersection points between the hole-circle and peg-ellipse are computed to find the contact height of the peg.	71
4.7	Top and bottom views of the peg-hole contact surface, with the peg tilted by 30° w.r.t. the hole.	71
4.8	Contact hyper-surface slices for fixed orientation differences between peg and hole.	73
4.9	The only peg- and hole-edges that will come into intersection for relative orientations within $[0, \pi/2]$ rad.	75

4.10	Top view of the (x, y) -plane: the tilted peg contacting the hole. The actual position of the hole in the (x, y) -plane is not known. (It is only shown here for the purpose of illustration.) From the measured contact z -height, an (x, y) -slice of the contact surface map (as shown in Figure 4.11) forms the first belief set, $\mathcal{B}_1$. $\mathcal{B}_1$ represents all the possible peg-positions w.r.t. the hole. The actual—and, as yet, unknown—peg position is marked with a red/medium-gray circle.	79
4.11	An (x, y) -slice of the contact surface map at the measured contact z -height.	79
4.12	New peg position, $T_{p_2}^w$, after the first peg move. The belief set, $\mathcal{B}_1$, is also updated by translating each point in $\mathcal{B}_1$ by the peg move.	80
4.13	Another (x, y) -slice of the contact surface map at the measured contact z -height.	80
4.14	Set $\overline{\mathcal{B}}_2$ contains all the points from the second slice of the contact surface map, and represents peg-position estimates from the second peg-hole contact z measurement. It is intersected with the updated $\mathcal{B}_1$ to obtain the new belief set, $\mathcal{B}_2$	81
4.15	Third (x, y) -slice of the contact surface map at the measured contact z -height.	81
4.16	The actual relative position of the peg w.r.t. the hole is successfully localized after the third iteration.	82
4.17	First observation matched to the map: a slice through the four-dimensional map of the contact hyper-surface at the observed z -height produces a three-dimensional surface in x, y, θ_z . Each configuration here represents a possible current peg configuration w.r.t. the hole.	84
4.18	The second measurement produces a new set of configurations, $\overline{\mathcal{B}}_2$	85
4.19	The possible relative peg-hole configurations recorded after the first probe move, $\mathcal{B}_1$, are updated by moving them according to the peg-move in (x, y, θ_z) . Then, the updated set is intersected with the new slice, $\overline{\mathcal{B}}_2$, to produce the new belief set, $\mathcal{B}_2$. The updated $\mathcal{B}_1$ and the new slice, $\overline{\mathcal{B}}_2$ are shown here.	86

5.1	Simplified illustration of the sampling of the peg-hole contact hyper-surface map for a circular peg-in-hole assembly. Sampled points in the (x, y) -space are shown. The hole is fixed at T_h^w w.r.t. the robot coordinate frame. Hence, the robot moves the peg to configuration T_p^w w.r.t. the robot coordinate frame and brings it into contact with the hole to sample the relative peg-hole configuration T_h^p	90
5.2	Algorithm for sampling the contact hyper-surface map.	91
5.3	Planar assembly example with three-dimensional uncertainty: There is uncertainty in the initial configuration of the hole, but the hole is assumed to be stationary for the duration of the assembly. In (b), an example of assembly sufficiency is shown. The entire peg bottom edge is contained inside the hole.	93
5.4	The peg is aligned with the estimated configuration and lowered along the contact-axis into contact with the hole. As the actual hole-configuration is contained within the goal region of the estimated hole-configuration, when the peg contacts the hole assembly sufficiency is guaranteed to be satisfied.	95
5.5	A mapped configuration and its corresponding goal region. The y -bound and sampling tolerance are also shown. The mapped configuration in (x, y, θ_z) signifies the position and orientation of the hole w.r.t. the peg. The goal region is defined w.r.t. the hole configuration, and so is oriented w.r.t. the peg the same way as the hole. The goal region is unbounded along the contact-axis (y), but is shown truncated.	98
5.6	The region to be mapped is given by the hole-configuration uncertainty region augmented by the peg motion region. This ensures that in the condition when the hole is at the edge of its uncertainty region, and the peg moves in the opposite direction of the hole-configuration, the relative peg-hole configuration is still covered by the map.	99
5.7	Algorithm enumerating the various phases of the mapping cycle.	100
5.8	Hypothetical example showing how an irregularity can violate the sampling conditions. There can always be an “irregularity” between two mapped configurations, both of which satisfy the Sampling Condition 5.3.	101
5.9	Algorithm for general localization using sampled maps.	103

5.10	First peg move matched to the map: The actual relative peg-hole configurations before and after the first move, $T_h^{p_0}$ and $T_h^{p_1}$, respectively, are shown. Because the map fulfills the sampling conditions, we know that there will be mapped configurations, $(T_h^{p_0})_m$ and $(T_h^{p_1})_m$, such that $T_h^{p_0}$ and $T_h^{p_1}$ are contained within their respective sampling tolerances. But, $T_h^{p_1}$ will also be contained within the sampling tolerance of $T_{p_0}^{p_1} (T_h^{p_0})_m$. Hence, the intersection of the sampling tolerance regions of $T_{p_0}^{p_1} (T_h^{p_0})_m$ and $(T_h^{p_1})_m$ will contain $T_h^{p_1}$, and will not be an empty set. Therefore, $(T_h^{p_1})_m$ will be included in the new belief set.	105
5.11	Hypothetical example illustrating the benefit of additional matches: (a) Peg moves, α and β , (b) Matching move α to the mapped points.	106
5.12	Hypothetical example continued: (a) Matching move β to the map, (b) Matching the combined move α and β to the map.	107
5.13	Example explaining change in problem formulation.	110
5.14	The estimated hole-configurations, w.r.t. the actual, from 1000 assembly trials are plotted. As all the estimates fall within the goal region of the actual, all the trials localized successfully.	112
5.15	First peg-hole contact.	113
5.16	Iterations 1 and 2: as the peg moves (translates/rotates) into contact with the stationary hole, the belief set “moves” on the map.	114
5.17	Iterations 3 and 4: The belief set shrinks till the termination condition is satisfied after the 4th iteration; relative peg-hole configuration is estimated from the belief set $\mathcal{B}_4$	115
5.18	Assembly is attempted with the estimated hole-configuration, and is successful.	116
5.19	Localization results from the circular peg-in-hole assembly trials: 99.9% trials successful, where the estimated hole-configuration is within the goal region (assembly clearance) radius of 0.5 mm. Also shown is the hole-uncertainty region (radius: 25 mm) for comparison.	119
5.20	Comparison of localization progress in terms of uncertainty resolution and localization accuracy w.r.t. the number of iterations carried out, averaged over 1000 trials, for the circular peg-in-hole assembly with added observation error.	121

5.21	The goal region in (x, y, θ_z) for the square peg-in-hole assembly is diamond-shaped. The smallest rectangular prism containing the goal region is shown. On the other hand, a prism contained inside the goal region can be of various different sizes.	126
5.22	Localization results from the square peg-in-hole assembly trials: 73% trials successful, where the estimated hole-configuration is within the goal region of the actual.	127
5.23	The residual uncertainty volume (averaged over 100 trials) after the 5th, 10th, 15th, 20th, and 25th iterations for the square peg-in-hole assembly. The uncertainty volume is computed as the volume of the rectangular prism containing all the configurations in the belief set.	128
6.1	Key-lock contact surface map sampled using the ParaDex robot, and plotted using the <code>mesh</code> command in Matlab [31]. Each sample is the position of the key w.r.t. the lock coordinate frame.	132
6.2	Three-dimensional surface with two probe moves shown; we sample and store a map of the probe-surface contact transformations.	133
6.3	(a) A sine curve sampled with fine-grid discretization. Localization on the curve starting with global uncertainty and using the sampled map requires heavy computation in the first few iterations. The computational load tapers off as localization proceeds. (b) Using the cell approach, the mapped configurations on the sine curve are grouped into larger cells, and localization is implemented on the cells instead of the individual configurations. As the number of cells is smaller, the computational load is reduced.	135
6.4	(a) After a few iterations of localization using the large cells, the uncertainty is resolved to within four cells. Hence, the computational load is much reduced. (b) Then the four large cells can be decomposed into smaller cells, thus increasing the resolution in stages. The computational load is again raised to the original level.	136
6.5	The surface is divided into a square grid of points that form the centers of the cells. The cells represent all the points that fall within the x, y -squares occupied by them.	137
6.6	An example of how the square prisms are circumscribed with cylinders to facilitate the matching computations.	138

6.7	Cylinder representing a portion of the surface shown. The entire z -range of the surface is covered by the cylinder. . . .	139
6.8	Matching of the first move, α , for one of the cells, l^0 , in the (x, y) -plane is shown. There is also a z -match, which is not shown here. There is global orientation uncertainty. Hence, the entire peripheral region that can be reached from l^0 with the length of move α is checked. This region intersects cell l^1 . If a z -match between l^0 and l^1 based on move α is also satisfied, then sequence $[l^0, l^1]$ is included in the belief set. . .	140
6.9	Algorithm for implementing the cell approach.	141
6.10	Some of the possible cell sequences for the probe moves shown in Figure 6.2. The actual cell sequence is included.	142
6.11	After the second move, β , both move β and the combined move, $\alpha + \beta$, are matched to advance sequence $[l^0, l^1]$. From the two cells, l^0 and l^1 , we compute the possible region of current occupation of the probe. The matching process in the (x, y) -plane is shown here. The computation is made easier because we have chosen to restrict the probe moves to be along a straight line.	144
6.12	Different cell-sequences can have the same end-cell with different orientation bounds. (a) For non-overlapping orientation bounds, two copies of the same end-cell are added to the belief set with the individual bounds. (b) For overlapping orientation bounds, a super-bound, given by the union of the overlapping bounds is assigned to the end-cell (duplicates of the end-cell are not required).	146
6.13	Block diagram of simulation code: the environment module gets peg motion commands from the controller, updates the position of the probe on the surface and returns the probe height- <i>observation</i> to the controller. The controller uses the observations to localize the probe position on the surface. . . .	147
6.14	When the probe runs into the boundary of the surface, it is first turned to its right (turned by -90° about θ_z). If the probe is still heading into the boundary, then it is turned by 180° , and now the probe will most probably head out onto the surface.	148
6.15	Initial position of the probe on surface for a trial localization run. There is global uncertainty in the position and orientation of the probe w.r.t. the surface.	150

6.16	After the first probe move, cell sequences that accommodate the move are computed and selected to the belief set. The possible cells of current occupation are highlighted here. . . .	150
6.17	The second probe move results in a steep climb up to a high point on the surface leading to a rapid reduction in uncertainty.	151
6.18	The third probe move and the resulting end-cells.	152
6.19	After the fourth probe move, the number of end-cells is higher than before, meaning greater position uncertainty. However, the overall uncertainty, in terms of position and orientation, has been reduced.	152
6.20	Once the probe hits the boundary, the uncertainty is quickly resolved to within three end-cells. The localization would thus proceed until sufficient resolution of uncertainty is achieved. .	153
6.21	Change in CPU time, and average estimation error with varying cell-size.	154
7.1	Algorithm for implementing the general localization strategy using particle filtering.	161
7.2	Two dimensional peg-in-hole localization example: tilted peg in contact with hole surface; initial relative peg-hole configuration is $T_h^{p_0}$	162
7.3	The map is a collection of hole positions relative to the peg with the peg in contact with the hole.	163
7.4	First peg move: represented as the new peg position w.r.t. the old, $T_{p_1}^{p_0}$	164
7.5	Particles updated according to first move, $T_{p_0}^{p_1}$	164
7.6	After resampling, the particles bunch up around the original particles that generated low errors.	165
7.7	Random localization: Peg move is selected randomly.	166
7.8	Active localization: Peg move is selected so as to optimally resolve the current uncertainty.	166
7.9	Even if the other set of particles had represented the actual configuration, the same actively selected peg move would produce maximum discrimination between the two groups of particles.	166
7.10	Comparison of variation in average error and number of iterations with change in the number of particles and σ	168
7.11	Results from 30 assembly trials for the circular peg-in-hole assembly.	171
7.12	Results from 29 assembly trials for the square peg-in-hole assembly: Estimated hole positions w.r.t. the actual in (x, y) . . .	172

7.13	Results from 29 assembly trials for the square peg-in-hole assembly: Estimated hole orientations w.r.t. the actual (θ_z). . .	173
7.14	Estimated positions of the lock w.r.t. the actual for 47 trials of the lock-key assembly.	175
7.15	The updated belief sets after the first and fourth observations for the lock-key assembly. (a,c) Key contacting lock-surface. (b,d) Particles in the belief set.	176
7.16	(a) Key contacting the lock-surface. (b) The updated belief set after the eighth observation for the lock-key assembly. Successive observations shrink the belief set. After eight iterations, lock-position is estimated from the weighted average of the particles. (c) Assembly accomplished by compliant phase using estimated lock-position.	177
7.17	Cumulative distribution of the number of trials ending successfully after k iterations of the circular peg-in-hole assembly.	179
7.18	Average configuration uncertainty after each iteration, averaged over 10 assembly trials for the square peg-in-hole assembly.	180

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Michael Branicky, for the inspiration and mentoring without which this dissertation would not have been possible. I would also like to thank my dissertation committee members, Dr. Wyatt Newman, Dr. Roger Quinn and Dr. Cenk Cavusoglu, for their help and patience.

This work was supported by the NSF Embedded and Hybrid Systems program, under the direction of Dr. Helen Gill (grant CCR-0208919). This support is gratefully acknowledged. The early work on the precession and sweep strategies was performed under the support of the National Institute of Standards and Technology's Advanced Technology Program (Cooperative Agreement Number 70NANB7H3024). I also received support from the Department of Mechanical and Aerospace Engineering and the School of Graduate Studies (Endowment Mentorship Program).

I greatly appreciate the work done by Ravi Hebbar, Daniel Morris, and others in setting up the programming environment on the ParaDex and the hybrid force-control scheme, which were invaluable to this research.

I would also like to thank all my lab mates over the past 9 years who made life in the lab bearable. Special thanks to Craig Birkhimer for all the lively discussions.

Thanks to my squash buddies, Joe Pahl, Henry Boom, Murat Tasan, and all the others for helping me vent through squash and keeping me sane.

A special thank you to my very good friend, Jasmine, for everything.

Finally, I offer my deepest gratitude to Indira, Raojibhai, and Krishna for making me what I am today.

Localization for Robotic Assemblies with Position Uncertainty

Abstract

by

Siddharth R. Chhatpar

This dissertation deals with the class of robotic assemblies where position uncertainty far exceeds assembly clearance, and visual assistance is not available to resolve the uncertainty. Our research is motivated by actual assemblies from vehicular transmissions that fall under this class. For this class of assemblies, the focus shifts from the dynamics of the assembly to the problem of searching for part alignment. A novel idea is introduced to transform the search for part alignment into one of localizing the peg-hole misalignment on the hyper-surface formed in the peg-hole contact configuration space (C-space). This idea is developed into an intelligent localization strategy for resolving the uncertainty in the relative configuration of parts. The strategy is to explore the assembly contact C-space and match observations to a pre-acquired map of the C-space.

The implementation of our localization strategy is described using both analytical and sampled maps of the contact C-space. Thus, one can either model the contact C-space using equations of the three-dimensional volumetric intersections of the mating parts, or sample it using a robot or CAD model. However, a sampled map does not provide a complete representation of the continuous contact C-space. Hence, the concepts of assembly suffi-

ciency, goal region, and approximate localization are introduced to help in localizing sufficiently for assembly.

With increasing dimensionality of the assembly uncertainty and small assembly clearances, the computational load becomes large and uneven over the localization period. An algorithm, termed the cell approach, is developed to implement the localization strategy in stages of increasing resolution, thus distributing the computational load more evenly.

To make the localization strategy more robust, the application of particle filtering for robotic assemblies with position uncertainty was pioneered in this dissertation. Particle filtering is a probabilistic scheme that maintains a set of weighted particles, where each particle represents an estimate of the relative peg-hole configuration; it can handle errors in actuation and observation, and also errors in mapping. Moreover, the number of particles can be adjusted to accommodate the computational resources available.

The ideas presented in this dissertation were validated with mathematical analyses, computer simulations, and actual robotic assemblies.

Chapter 1

Introduction

This dissertation deals with the class of robotic assemblies where the position uncertainty of the mating parts far exceeds the assembly clearance. Examples of such assemblies that motivate our research include the **forward clutch** assembly and the **planetary gear** assembly, from a vehicular transmission. See Figure 1.1. To assemble the forward clutch, a toothed-hub has to be inserted through a series of five grooved-clutch plates. However, there is uncertainty in the initial position and rotation of the clutch plates, which are stacked vertically in a container. This uncertainty being much larger than the assembly clearance between the hub and each clutch plate, there is no pre-defined trajectory for insertion other than an exhaustive search covering the entire uncertainty region.

A seemingly straightforward solution would be to use vision to resolve the uncertainty. However, for the forward clutch and the planetary gear (and many other such target assemblies), the relative positioning of parts occludes a good view of the mating elements, thus ruling out the use of visual assistance. Under this scenario, we have to undertake a search for the alignment of mating parts. This search can be conducted either in a *blind* manner, where the uncertainty region is searched exhaustively, or in an *intelligent* manner, where the misalignment of the mating parts is localized

without an exhaustive search.

In this dissertation, our approach is to *utilize clues from contact measurements between the mating parts to resolve the uncertainty in their relative configuration*. We validate this strategy using mathematical analyses and computer simulations. We also implement it on actual robotic assemblies including a key-lock assembly plus circular, square, and pentagonal peg-in-hole assemblies. These assemblies are used to focus our research and for conducting experiments. However, the concepts introduced here are generally applicable to other assemblies as well.

(a) forward clutch

(b) planetary gear

Figure 1.1: Example vehicular transmission assemblies from Ford Motor Co.

Our initial attempts to *intelligently* utilize clues from contact measurements to localize the misalignment of parts resulted in the **precession** and **sweep strategies**. In the precession strategy, the peg is tilted and precessed while maintaining contact with the hole-surface. The resulting trajectory of the peg is analyzed online to compute the position of the hole w.r.t. the peg. In the sweep strategy, a tilted peg is swept across the hole while maintaining contact with it. Again, the resulting trajectory of the peg can be analyzed to derive the configuration of the hole w.r.t. the sweep. These preliminary strategies inspired the creation of the general localization strategy that is the focus of this dissertation.

In the **general localization strategy**, the peg-hole misalignment is localized by exploring the contact configuration space (C-space) and matching the individual contact observations to a pre-acquired map of the contact C-space. Once the relative peg-hole configuration is known, assembly becomes easy. To explore the contact C-space, the peg is re-posed (translated and rotated) to different configurations and brought into contact with the hole. The various contact configurations thus measured are recursively matched to a map of the contact C-space. The localization strategy can work with *both analytical and sampled maps*.

A general analytical map would be six-dimensional (three dimensions each in position and orientation), and would take the form of a collection of functions describing the configurations of the peg w.r.t. the hole on contact (or vice versa), applicable in different regions of the contact configuration space (C-space). An analytical map is relatively easy to generate for regular assemblies, e.g., the circular and square peg-in-holes. The general localization strategy was tested on these assemblies using their corresponding analytical maps.

However, producing an analytical map requires painstaking effort to generate equations for three-dimensional volumetric intersections for each new

assembly. On the other hand, a contact C-space map can be easily sampled using either a robot or a CAD model. But, a sampled map does not provide a complete representation of the continuous contact C-space. Hence, the probability of finding an exact match of a peg-hole contact configuration on the map is reduced to zero. We introduce the concepts of **assembly sufficiency** and **goal region** to show how *approximate localization using a sampled map* can guarantee assembly. The idea of the goal region is based on the concept of assembly clearance, but is applicable in higher-dimensions (unlike assembly clearance, which is usually defined in the plane of the assembly). The resulting localization strategy using sampled maps was tested successfully in computer simulations of the circular and square peg-in-hole assemblies. After adding observation errors to the simulations to mimic physical assemblies, we modified the strategy to make it robust to these errors.

However, with increasing dimensionality of the assembly uncertainty and small assembly clearances, the fine-grained maps required to satisfy assembly sufficiency conditions grow intractably large. This increases the computation time required to match the large number of initial configurations to the map, creating an uneven computational load over the localization period.

We introduce the **cell approach** algorithm to tackle this problem by implementing the localization strategy in stages of increasing resolution. Thus, computational load is more evenly distributed. In the cell approach, small neighborhoods of configurations are combined into monolithic **cells**, which then represent the entire neighborhood. Localization can then proceed using these cells instead of the individual mapped configurations. The added advantage is that the cells can be dilated to *accommodate errors from actuation and measurement, or incomplete maps*. Thus, it provides a *worst-case approach*. Several different cell-sizes were experimented with and their respective advantages and shortcomings were compared. In general, with larger cells, computation time per iteration is reduced, but the failure rate is higher,

and localization resolution is poor. On the other hand, with smaller cells, each iteration has a higher computational load, especially in the early stages of localization. However, as the smaller cells approximate the surface better, their failure rate is lower and the resolution of localization is improved. To combine the advantages of the different cell-sizes, localization was initiated with a larger cell-size, thus reducing computational load. As localization proceeds and the uncertainty is reduced to a small number of cells, the computational requirement decreases. At this time, the remaining cells can be decomposed into smaller cells, thus increasing the resolution of localization while maintaining a level computational load.

Another approach that accommodates limited computational resources and distributes computational load evenly is the **particle filtering** algorithm. The application of particle filtering for robotic assemblies with position uncertainty was pioneered in this dissertation. With this algorithm, instead of keeping track of all the possible configuration estimates that match the current contact position measurement, a fixed number of the best estimates (called **particles**) are tracked. The advantage of using particle filters is that the number of particles can be adjusted depending on the computational resources available. Also particle filtering is a probabilistic technique that can handle errors from actuation and observation, as well as incomplete maps. Another advantage of particle filtering is that it can simultaneously track multiple solutions, which abound in our problem. Obviously, localization cannot be guaranteed using particle filtering. Moreover, with the coarsely-sampled maps that were employed in localization, only approximate localization is possible. Our goal in using the particle filtering implementation was to reduce large uncertainties in the hole-configuration to smaller values in a short time. The small residual uncertainty can then be accommodated using either an exhaustive search, or some form of compliant assembly. We were able to obtain excellent results with the actual physical assemblies of

the circular, square, and pentagonal peg-in-hole assemblies, and the lock-key assembly.

Organization. The remainder of this chapter is organized as follows. In Section 1.1, we formulate the problem of localization for robotic assemblies with position uncertainty, and state it formally. We also give several examples of localization problems that will be used for illustration purposes later in the dissertation. Section 1.2 briefly states our solution to this problem—a general localization strategy, and shifts the focus to the search for algorithms to implement this strategy. Section 1.3 presents previous work in the field of robotic assemblies with position uncertainty, and also related work in the field of mobile robot localization, geometric pattern-matching, workpiece localization, etc. In Section 1.4, the contributions of this dissertation are highlighted. Finally, Section 1.5 describes the organization of the rest of the dissertation.

1.1 Problem Formulation and Statement

For the example transmission assemblies, the uncertainty is limited to the location of the hole (each clutch plate) in the horizontal plane and its orientation about the vertical axis; the vertical placement of the hole and the orientation of the hole-plane about the horizontal axes are determined by the fixturing of the container. During assembly, the peg (clutch hub) is held by a robot and its configuration is known.

Although these transmission assemblies motivate our research, in this dissertation we will tackle the more complete problem of position and orientation uncertainty in the full six-dimensional space. Throughout, however, we will assume that there is bounded uncertainty in the configuration of the hole, while the configuration of the peg—held by a robot—is known. An-

other important assumption implicit in our approach is that the hole remains stationary throughout the search process.

Problem 1.1 *For an arbitrarily-shaped peg and hole, with bounded uncertainty in the position and orientation of the hole, achieve insertion of peg into hole.*

For example, consider the problem of inserting a cubical peg into a similarly shaped hole, with bounded uncertainty in the position (x, y, z) and the orientation (rotations about x, y, z) of the hole. Figure 1.2 depicts the assembly task.

Figure 1.2: Peg contacting hole.

In the example shown, the peg z -axis is the axis of insertion. To avoid excessive skewing of the hole orientation w.r.t. the insertion axis, tight bounds are placed on the orientation uncertainty along the (θ_x, θ_y) dimensions. On the other hand, the orientation uncertainty about the vertical axis (θ_z) is usually allowed to be large (even unbounded in many cases). The bounds on the position uncertainty are generally 100 times as large as the assem-

bly clearance. These bounds are justified uncertainties encountered in our motivating assemblies.

The above problem statement is sufficient if our goal is simply to look for *exhaustive* search strategies. But we are more interested in intelligent solutions that focus on localizing the hole configuration (position and orientation) relative to the peg, so as to be able to guide the peg into the hole. Below, we formulate a rigorous statement of this localization problem.

Let $\mathcal{F}_h$ be the coordinate frame attached to the hole. We assume the hole is fixtured, and so $\mathcal{F}_h$ is stationary. But, there is uncertainty in the position and orientation of the hole, and hence, also $\mathcal{F}_h$. Let $\mathcal{F}_p$ represent the coordinate frame attached to the peg. The controller has full knowledge of the peg-configuration, and hence, $\mathcal{F}_p$. The problem of locating the hole can be represented in terms of finding the transformation from $\mathcal{F}_p$ to $\mathcal{F}_h$. Let T_p^h be the 4×4 matrix associated with the homogeneous transformation¹ from frame $\mathcal{F}_p$ to frame $\mathcal{F}_h$; then a position vector a_p in $\mathcal{F}_p$ is transformed to a_h in $\mathcal{F}_h$ by $a_h = T_p^h a_p$. This coordinate transformation can also be interpreted as the position and orientation of the peg w.r.t. the hole coordinate frame. The reformulated problem is stated below.

Problem 1.2 *For an arbitrarily-shaped peg and hole, with bounded uncertainty in the position and orientation of the hole, find the coordinate transformation, T_p^h , from $\mathcal{F}_p$ to $\mathcal{F}_h$.*

Note that we can also formulate the problem as localizing the hole-configuration w.r.t. the peg coordinate frame, i.e., T_h^p . This formulation is equivalent to the one given above. However, as will be seen later, the latter formulation is better suited for localization using sampled (discretized) maps.

¹The homogeneous transformation matrix is explained in further detail in Section 3.3.

We present several examples below to better illustrate Problem 1.2. These examples will be utilized frequently throughout the dissertation for illustration purposes.

Example 1.1 *Consider the two-dimensional curve shown in Figure 1.3(top), traversed by a point probe. There is uncertainty in the position and orientation of the curve, which can also be interpreted as follows. The position of the probe on the curve in (x, z) , and its orientation (i.e., whether the probe is travelling in the $+x$ or $-x$ direction on the curve) are not known. Thus, the problem has three-dimensional uncertainty, (x, z, θ) , where θ equals either $+1$ or -1 . Our goal is to find the transformation from the coordinate frame attached to the curve to the coordinate frame attached to the probe.*

Figure 1.3: Example 1 (top): localizing a probe on a curve (two-dimensional uncertainty). Example 2 (bottom): localizing a probe on a surface (three-dimensional uncertainty).

Example 1.2 *The second example consists of a surface in (x, y, z) , shown in Figure 1.3(bottom), traversed by a point probe. Again, the position and orientation of the surface under the probe are not known. In this case the uncertainty is four-dimensional, (x, y, z, θ) , where $\theta \in [0, 2\pi]$. The goal of localizing the probe on the surface can be formalized as the problem of finding the transformation from the surface coordinate frame to the probe coordinate frame.*

Example 1.3 *In this example, we have a planar peg-in-hole set-up. A drawing of the example planar assembly is shown in Figure 1.4. There is bounded uncertainty in the configuration of the hole: position (x, y) and orientation (θ) . Thus, it is a three-dimensional uncertainty problem. The goal is to localize the hole-configuration to facilitate assembly of the peg into the hole.*

Figure 1.4: Example 3: Planar peg-hole localization (three-dimensional uncertainty).

1.2 Solution: General Localization Strategy

Our solution to Problem 1.2 is to find the coordinate transformation between the peg and hole by exploring the peg-hole contact configuration space and matching observations to a pre-acquired map of the contact C-space. We call this the **general localization strategy**. This leads us to the formulation of the problem of implementing this general localization strategy.

Problem 1.3 *For a peg-in-hole assembly with bounded uncertainty in the position and orientation of the hole, explore the peg-hole contact C-space. Match the resulting observations to a pre-acquired map of the contact C-space to localize the coordinate transformation from $\mathcal{F}_p$ to $\mathcal{F}_h$.*

Thus, we have converted the problem of searching for peg-hole alignment into one of localizing a point in the relative peg-hole contact C-space. We term this the **assembly localization problem**.

1.3 Previous Work

The problem of peg-in-hole assembly under position uncertainty has been of central importance in the field of robotics and the subject of much research. This problem has often been categorized under the broader subject of *motion planning under position uncertainty*, a comprehensive review of which is presented in [18]. Compliant manipulation has long been recognized as an essential tool in programming robots to overcome uncertainty in assembly [48, 19, 36, 26, 35]. A formal approach for dealing with uncertainties arising from control error and inaccurate environment models is reviewed in the next section. Then, in Section 1.3.2, we describe the previous work on peg-in-hole assemblies with position uncertainty. The literature on peg-in-hole assemblies can be divided into two major groups depending on the essence of the strategy used. The first group concentrates on setting up favorable

conditions for the peg to drop into the hole *by chance*. On the other hand, the second group uses *intelligent* techniques to resolve the uncertainty in the hole configuration, which then helps in simple assembly.

1.3.1 Planning Under Uncertainty

A seminal paper by Lozano-Perez, Mason, and Taylor [30] presented a formal approach for dealing with uncertainty based on the geometry of configuration space. **Configuration space** can be defined as the simplified space of all the states that a system can attain. For instance, for a serial-link robot with rotary joints, the space formed by the joint angles is its configuration space. Any pose of the robot can be described as a point in this space given by the individual joint-angle values. Lozano-Perez *et al.* identify the major requirements for a framework to deal with autonomous assembly under uncertainty:

- Configuration space, or **C-space**, which allows the moving part to be treated as a point,
- Model of control with uncertainty,
- Model of initial and allowable final uncertainty,
- Termination predicate (essentially a model of sensor uncertainty).

Starting with a known goal configuration, they identify a set of positions, called the **pre-image**, from which the goal can be reached with a single motion. Then they identify the sensor conditions and control parameters required to achieve the goal from the pre-image. Thus, chaining backward from the goal to the initial conditions (in the manner of dynamic programming), they establish a series of commands and constraints to reach the goal. An illustration of the concept of **backward chaining** is shown in Figure 1.5. The states labelled as “sub-goal” form the pre-image of the “final goal”. The states labelled “sub-sub-goal” form the pre-images for the sub-goal states. Thus, starting from the final goal and chaining backward, a path can be cre-

Figure 1.5: An illustration of the concept of backward chaining. The states labelled as “sub-goal” and “sub-sub-goal” form the pre-images of the “final goal” and “sub-goal” states, respectively. Thus, starting from the final goal and chaining backward, paths can be created to reach the final goal starting from all possible initial conditions.

ated to reach the final goal starting from all possible initial conditions and taking into account all possible sensor measurements. In this dissertation, we are interested in finding strategies for dealing with uncertainties arising from the unknown configurations of mating parts.

1.3.2 Peg-in-Hole Assemblies with Position Uncertainty

Assembly By Chance

Daniel Whitney has made significant contribution to the field of “peg-in-hole” assembly through his work at the Charles Stark Draper Laboratory Inc., Cambridge, MA. In [48], Whitney analyzed a force feedback strategy for fine motions to guide a simple, cylindrical peg-in-hole assembly. In 1982, Whitney developed quasi-static conditions to avoid wedging and jamming of the peg-in-hole assembly [49]. These conditions allow the peg to correct its position and orientation with respect to the hole with the simple

application of a downward force on the peg. The conditions are true for two-dimensional, chamfered peg-in-hole assemblies, and assume quasi-static assembly with small initial peg-hole misalignment. This research led to the development of the Remote Center Compliance (RCC) device. This is a passive device attached between the gripper and the peg. It simulates compliance about a remote (adjustable) center that is usually placed at the lower tip of the peg. Thus, the RCC substitutes for a force-guided strategy, involving a force sensor and an active control loop. However, if the parts are not chamfered, or if the position uncertainties exceed the chamfer width, the RCC is ineffective. In general, if the force-guided strategy to be invoked cannot be realized in terms of a passive mechanism, then the RCC approach is inadequate.

More recently, Joseph Schimmels has patented a passive mechanism designed to encode a force-guided strategy to overcome position uncertainty in assembly [38]. This mechanism was designed specifically for assembling the forward clutch (see Figure 1.1(a)). The mechanism has to overcome uncertainty in the $(x-y)$ horizontal plane to align the hub with each ring, as well as rotational uncertainty about the vertical (z) to mesh teeth on the hub with slots on the ring. It was successful in assembling the forward clutch while limiting the contact forces between the assembling parts to a minimum to prevent damage to the final product [38]. The fundamental difference between the RCC and Schimmels' mechanism is that, while the RCC relies on chamfered surfaces to generate appropriate forces, which can be used to align the peg with the hole, Schimmels' mechanism needs a search strategy to hunt for initial alignment. Yet, according to David Gravel of Ford Motor Co. Advanced Manufacturing Technology Development Center (AMTD) [19]:

The behaviors of these [Schimmels'] passive compliant devices were very effective in producing assemblies for the specific part

they had been designed for, such as a forward clutch hub. However, they lacked a general utility and could not cover a large class of assembly tasks without resorting to another part-specific device.

Thus, passive mechanisms, substituting for active force-feedback controlled, force-guided strategies, can guarantee stability, but are not easily adaptable to varying assembly requirements.

In our previous work, we reported on successful assemblies of the forward clutch and other gear-meshing assemblies involving large initial position uncertainties [6, 35, 34, 25]. The **forward clutch** assembly (Figure 1.1(a)) consists of two main parts: the clutch pack (Figure 1.1(a) bottom) and the splined clutch hub (Figure 1.1(a) top). The clutch pack holds five rings (clutch plates), stacked vertically and separated by high-friction disks. The splined clutch hub is to be aligned with the splines on each ring for insertion. Each ring is capable of independent movement in the (x, y) -plane (within the confines of the clutch pack) and (unlimited) rotation around z . During assembly, although the clutch pack is fixtured in a known location, the position and orientation of each ring is arbitrary and unknown. The requirement for inserting the hub into each ring includes bringing the hub within assembly-clearance distance of the ring-center, and simultaneously aligning the hub-rotation with the ring for spline-meshing.² We cannot rely on visual assistance for localizing the positions and orientations of the rings for several reasons. The primary reason is that the top ring occludes a good view of the rings below. Also, once the assembly starts, the hub occludes any view of the rings below it, and visual localization is no longer possible.

²While the (x, y) -plane radial assembly clearance between the hub and any ring is 0.23 mm, the position uncertainty of a ring can be as large as 0.8 mm. Also, the rotational alignment clearance for spline-meshing is 0.005 rad, while the rotational uncertainty of each ring is unbounded (though periodic with the spline-pitch of 0.13 rad). Thus, the uncertainties in the position and rotation of the rings are much larger than the assembly clearance.

We used a low-level impedance-based control algorithm (natural admittance control [33]) and integrated it with higher-level search strategies to achieve time-efficient assemblies with low contact forces. A “spiral” path was used to exhaustively search for hub-clutch plate alignment in the horizontal plane integrated with a search in rotation about the vertical for teeth meshing.

The forward clutch and other example transmission assemblies belong to a larger class of assemblies, that we call the general multi-tiered peg-in-hole assembly class, or simply, **peg-in-maze**. In related work [7], we built a representative peg-in-maze assembly as a first step towards a conceptual assembly that would encompass the characteristics of the example assemblies. This assembly was useful for illustration and experimentation purposes. Figure 1.6 shows this representative peg-in-maze assembly. It abstracts the peculiarities of both the forward clutch and the planetary gear assemblies described above.

By construction, the example peg-in-maze assembly is analogous to the forward clutch. The clutch plates are represented by *layers* and the hub is represented by a peg with a key on the bottom rim (Figure 1.6(c)). The layers are stacked vertically inside a container and are capable of individual movement in the plane and rotation about the vertical axis, within the container. As the position of each layer in the plane and its orientation are initialized arbitrarily, and are not known to the controller, there is three-dimensional (x, y, θ_z) uncertainty in the assembly. The peg has to assemble through a characteristically-shaped hole in each layer as it moves down. While some holes are simply circular, others have slots on them. Figures 1.6(d) and (e) show two layer-types with slots: single-slot hole and double-slot hole, respectively. The peg can insert into the slotted holes only if its key is aligned with the slot. Hence, the peg has to undertake a rectangular search in the (x, y) -plane combined with a rotational search about the vertical (z) to align

Figure 1.6: Peg-in-maze assembly: A keyed-peg has to be inserted through characteristically-shaped holes on five layers stacked in a container.

its key with any slots. The layers and the container are designed so that the uncertainty in layer configuration is much larger than the assembly clearance between the peg and each hole. The peg-in-maze assembly also possesses a jam condition that requires retraction and selection of alternate assembly routes. This property mirrors the planetary gear assembly, which is prone to jamming and requires jam-recognition and retraction.

The peg-in-maze assembly requires switching between **continuous behaviors**, such as, searching for peg-hole alignment, inserting the peg into the hole, and recovering from a jam, based on **discrete events**, such as, peg-hole contact is established, peg-hole alignment is achieved, and jam condition is

recognized. Hence, we applied a hybrid systems approach to build a general framework that facilitated switching between the continuous behaviors based on the detection of discrete events.

Intelligent Techniques for Assembly

Some early work on non-chamfered, round peg-in-hole assembly with large initial misalignment made use of position or force/moment sensing to compute the exact location of the hole with respect to the peg. Specifically, Simunovic used moment information generated on the peg [40]. Even with large misalignments between the peg and the hole, some portion of the peg bottom edge is supported by the hole-edge while the rest overhangs the hole. Vertical downward force on the peg causes a moment about a specific horizontal axis (attempting to tilt the peg into the hole) that can be measured. The peg can be aligned with the hole by moving it in the direction normal to the axis of the measured moment. Inoue uses peg-hole contact position information to compute hole location [26]. Inoue starts with a tilted peg, which insures an initial configuration with some part of the peg inside the hole. Now, the peg is moved from side to side until it contacts the hole, and the two contact positions are noted. The hole is located on the line bisecting the two recorded positions and normal to the line joining them. However, both the above strategies (Simunovic and Inoue) were limited to round peg-in-hole assemblies. We are interested in a general strategy applicable to arbitrary peg-in-hole assemblies.

Among recent research, Caine and Strip use the contact C-space generated by a tilted peg to implement rectangular peg-in-hole assemblies. Caine starts by enumerating the set of contact configurations for a rectangular peg tilted about all three axes, assuming only point contacts [5]. He then leads the assembly through a sequence of configurations that increasingly limit the degrees of freedom of the peg in contact with the hole, thus reducing the

uncertainty of the system. Strip extends this strategy in combination with a general insertion strategy developed in [42] to generalize it for all convex three-dimensional pegs [43]. The strategy starts with a tilted peg and the remainder is based roughly on the approach used by humans in inserting tight-fitting parts. The peg is moved in contact with the hole with translational compliance perpendicular to the direction of the applied force and rotational compliance about the axis in line with the direction of insertion. But both Caine and Strip assume close initial alignment of peg with hole. According to Strip [43],

Simple fixturing or a crude vision system should allow us to align the z -axis position to within 0.25 inches and rotational angular displacement within a degree or two.

This assumption does not hold true for our problem. Moreover, Caine requires a knowledgeable user to construct a complete set of contact configurations and a valid path through the configurations leading to assembly. Similarly, Strip requires one to encode the human strategy to be followed by the controller. Our goal is to find a strategy requiring minimal intervention when introducing a new assembly.

Peshkin introduced the concept of mapping sensed forces characterized by assembly error conditions onto motions that would reduce the corresponding force (error) [36]. As a consequence of this research, the “accommodation matrix” was developed by Peshkin and Schimmels, which is an attractive method for encoding force-guided assembly strategies [39]. Under this method, sensed contact forces are mapped onto corresponding velocities through an accommodation matrix. A nominal velocity imposed over this strategy leads the assembly towards a specific goal. Unfortunately, no actual experimental assemblies using an accommodation matrix have been reported. The disadvantage of the accommodation matrix strategy is that

the manipulator’s response to sensed forces (the accommodation matrix) has to be constructed by a knowledgeable operator.

Birkhimer *et al.* report on learning strategies for robotic assembly inspired by humans performing the same assemblies without the aid of vision [2]. The goal is to automatically synthesize assembly *trajectories* from atomic behaviors by analyzing position and force-feedback records of human strategies. Successful assembly of a square peg-in-hole using this technique was reported in [24].

1.3.3 Related Work

The assembly localization problem, stated in Problem 1.3, is similar to the **mobile robot localization problem**, in particular, the **global localization problem** [21, 20, 4, 45]. Global localization involves determining the robot’s **pose** (position and orientation) by matching observations to a map of the environment, starting from global initial uncertainty. The problem is complicated by uncertainties in the action and perception of mobile robots. Most approaches to the mobile robot localization problem use a probabilistic formulation to represent and update the pose of the robot. In particular, Markovian methods use a discrete representation of the distribution over the pose-space. Fine-grained grid approximations, while providing higher resolution, are computationally expensive, and recently Monte Carlo methods, like particle filtering, have been employed to overcome this problem [46]. An alternative approach to the localization problem called **multiple hypothesis tracking** allows a Kalman filter-based solution to the global problem by tracking every possible hypothesis [27]. In our problem, the actions and observations add little uncertainty. This is because of the high resolution of the ParaDex robot, which we use for our assembly experiments. Current localization techniques are either **landmark-based** or use **dense sensor matching**. In our problem, there are no easily identifiable observational

cues, and hence, we are interested in the dense sensor matching scheme, which attempts to use whatever sensor information is available.

Another area of related research is the localization problem for workpiece fixturing. A comparison of various geometric algorithms developed for **workpiece localization** is given in [13]. These algorithms use a least squares formulation to minimize an objective function to converge on the required Euclidean transformation between the workpiece coordinate frame and some reference coordinate frame. But, specific initial conditions have to be selected to prevent the algorithms from getting trapped in local minima [13]. In our problem, besides the abundance of local minima, there is complete initial uncertainty, making it difficult to set up favorable conditions to ensure convergence.

Although not entirely similar, our problem can also be compared to the **geometric pattern-matching problem** [17]. In geometric pattern-matching, one is given two sets of points, $\mathcal{A}$ and $\mathcal{B}$, in d dimensions, and the problem is to determine the rigid transformation that brings set $\mathcal{A}$ closest to set $\mathcal{B}$, under some distance measure. In applying this to our case, we can consider the set of points forming the contact map to be set $\mathcal{B}$ and the sampled exploration points to be set $\mathcal{A}$. Then, as with the geometric pattern-matching problem, we are looking for a rigid transformation, T , that brings every point of $\mathcal{A}$ within some Euclidean distance of a unique point of $\mathcal{B}$. The formula can be written as follows. For every $a \in \mathcal{A}$, there exists a unique $b \in \mathcal{B}$, such that $\|Ta - b\| \leq \epsilon$, where ϵ is a pre-determined threshold. For our problem, it does not have to be an exact match, only within some ϵ -threshold, because, for every assembly with a positive clearance, there is some allowable error in estimation that does not affect successful assembly. But, our problem is different in two ways. First, we seek all the rigid transformations (and there generally *is* more than one) that qualify. Second, our problem is incremental, i.e., we build set $\mathcal{A}$ over time as the robot collects more data; yet we

would like to start the matching process as soon as the second observation is made. To accommodate these considerations, we would search in the full space of transformations for our first match after the second observation and build a set of qualifying transformations. For subsequent matches, we would only search this restricted set of transformations, and thus refine the set of qualifying transformations with each observation.

Among other related research is the work done towards sensing by **probing** and **point sampling** [28]. The main goal of work in this area is to identify part pose by probing efficiently with a minimum number of probe points. The probing can be done using a set of mechanical probes or using a set of point light emitters and detectors to avoid contact with the part. In our research, we use the peg itself as the probe to contact the hole. This poses various difficulties since a tilted peg presents a much more complex probing surface (volume) than a point probe. (For a point probe on a surface, a change in orientation will produce a map that is identical to a map with a different orientation, only rotated by the orientation difference, while for a square peg-in-hole, for example, each orientation produces a different map.)

1.4 Contributions of this Dissertation

Below is a summary of the contributions made by this dissertation.

- For assemblies with position uncertainty, we analyze search space coverage in terms of the assembly clearance, and establish conditions for a search path to provide exhaustive coverage [8].
- We introduce a novel idea to transform the problem of searching for peg-hole alignment into one of localizing a point on a hyper-surface corresponding to all the relative configurations of the peg and the hole in contact [8].
- We develop an *intelligent* localization strategy for resolving the uncer-

tainty in hole configuration, leading to successful assembly [8]. The strategy is to explore the assembly contact configuration-space and match observations to a pre-acquired map of the C-space.

- The introduction of the concepts of **assembly sufficiency**, **goal region**, and approximate localization, lead to guaranteed localization even with a sampled map of the contact C-space.

- We present the **cell approach** algorithm developed to implement our localization strategy in stages of increasing resolution for computational efficiency [9].

- The localization strategy was implemented using particle filtering [11]. Particle filtering is a probabilistic scheme that maintains a set of weighted particles, where each particle represents an estimate of the relative peg-hole configuration. The number of particles can be adjusted to accommodate the computational resources available. Particle filtering can handle errors in actuation and observation, and also errors in mapping.

- Active localization was implemented successfully in conjunction with particle filtering [11]. Active localization involves computing the peg move that optimally resolves the current uncertainty.

- Our localization strategies were validated with mathematical analyses, computer simulations, and actual robotic assemblies [25, 24].

1.5 Dissertation Organization

In this chapter, we introduced the problem of robotic peg-in-hole assemblies with position uncertainty. The real-world transmission assemblies that motivate our research were described. We formulated the problem of searching for the alignment of mating parts as a localization task, and provided a formal problem statement. We also summarized the previous research done in the field of peg-in-hole assemblies under uncertainty. The rest of the dissertation

is organized as follows.

In Chapter 2, we give an overview of the experimental test-bed, including the robot, the controller, and example peg-in-hole assemblies.

In Chapter 3, we start with transforming the search space for peg-hole alignment to configuration space, which facilitates the subsequent analysis of the exhaustive coverage of the search space. We establish conditions for exhaustive coverage of the continuous search space with a finite set of points, based on the assembly clearance. We present various search paths that fulfill these conditions, and discuss their merits in terms of implementation using a robot. Moving away from these exhaustive search strategies, we describe our *intelligent* localization strategy with an application on a simple two-dimensional example.

Chapter 4 gives a detailed description of the localization strategy using analytical maps of the contact hyper-surface. Successful applications on circular and square peg-in-hole assemblies are presented to validate the strategy.

Creating an analytical map for the peg-hole contact hyper-surface requires painstaking effort to generate equations for three-dimensional volumetric intersections, which we would like to avoid. Moreover, for an actual assembly, its contact hyper-surface might not be represented accurately by a *perfect* analytical map. Hence, our goal is to localize using a sampled map of the contact hyper-surface, which can be easily generated using a robot. This is described in Chapter 5. However, a sampled map does not provide a complete representation of the contact hyper-surface. To overcome this problem, we introduce the concepts of assembly sufficiency, goal region, and approximate localization. We also develop sampling conditions and show how localization can be guaranteed even with a sampled map.

However, with increasing dimensionality of the assembly uncertainty and small assembly clearances, the computational load becomes large and uneven over the localization period. We introduce the cell approach algorithm in

Chapter 6, which implements the localization strategy in stages of increasing resolution. Thus, computational load is more evenly distributed.

In Chapter 7, we describe the implementation of our localization strategy using particle filtering. As particle filtering is a probabilistic technique it can handle the errors arising from a coarsely-sampled map, as well as actuation and observation errors. By changing the number of particles employed, we can accommodate the algorithm to the available computational resources. Moreover, particle filters can track multiple solutions simultaneously, which abound in our problem.

Finally, in Chapter 8, we conclude and discuss future work.

Chapter 2

Experimental Testbed Overview

This chapter gives an overview of the testbed used for experimental verification of the localization strategy. This includes the robot, the controller, and the various test-assemblies. The localization strategy was also tested on computer simulations of the circular and square peg-in-hole assemblies, before the actual assembly experiments. The simulations were coded in Matlab [31]. The experiments reported in this dissertation were performed on a ParaDex robot [47] at the Center for Automation and Intelligent Systems Research (CAISR), Case Western Reserve University. The low-level controller on the ParaDex is a flavor of impedance control called **natural admittance control (NAC)** [33]. The particular implementation of natural admittance control used simulates hybrid force-position control. A number of example assemblies were set up, including, a circular peg-in-hole, a square peg-in-hole, a pentagonal peg-in-hole, and a key-lock assembly. A brief description of each of these components is given below, and suggestions are provided for further reading for more details.

2.1 The ParaDex Robot

A picture of the ParaDex robot is shown in Figure 2.1(a). The ParaDex is a closed-chain mechanism [41] with six parallel prismatic actuators. A seventh degree of freedom from a central tool roll (about an axis parallel to the six actuators) provides kinematic redundancy. High stiffness, low friction, and a high payload make the ParaDex especially suitable for force control applications. For more details, see [47]. Figure 2.1(b) shows a diagram of the axis definition for the ParaDex.

(a) The ParaDex Robot

(b) Axis definition for the ParaDex

Figure 2.1: The ParaDex robot is a novel closed-chain mechanism with six parallel prismatic actuators. High stiffness, low friction, and a high payload make the ParaDex especially suitable for force control applications.

2.2 Natural Admittance Controller

In natural admittance control, a model of the robot, incorporating estimates of actual inertias, is simulated in real time to respond to sensed end-effector forces, plus virtual forces defined with respect to a computed attractor. The physical robot is servoed to track the idealized simulation. The attractor is defined in Cartesian space, and is connected to the end-effector through a set

of virtual springs and dampers. User commands to move the end-effector in space are input through the attractor, which, in turn, generates virtual forces on the end-effector through the springs and dampers. Figure 2.2 shows an example of a GUI allowing a user to generate attractor trajectories [44]. The figure shows a recursive spiral trajectory being constructed and parameterized with virtual spring and damper values. The hybrid force-position control implementation allows arbitrary axes to be locked into position control while maintaining natural admittance control (NAC) on the other axes.

Figure 2.2: An example of a GUI that allows users to generate attractor trajectories and input parameters such as the virtual spring and damper values used by the controller.

Consider the following scenario as an example where hybrid force-position control is required. The robot is holding and manipulating a peg. The peg needs to accurately trace a path in the horizontal (x, y) -plane while maintaining contact with the surface and being compliant along the vertical axis (z). (Needless to say, we make sure that the peg will not encounter an obstacle in the horizontal plane that can not be accommodated by the vertical compliance.) To achieve this, the horizontal axes are locked under position

control and the vertical axis is kept under NAC. The attractor is made to follow the required path in the horizontal plane, while it is pulled beneath the surface in the vertical to maintain contact. This is shown in Figure 2.3. If the peg encounters a hill while following the prescribed horizontal path, the vertical compliance allows it to climb or lower itself without leaving the path.

Figure 2.3: Illustration of how the hybrid control scheme allows accurate tracking in selected dimensions ((x, y) -plane for the example shown) and compliance in other dimensions (vertical (z) axis for the example shown).

2.3 Example Assemblies

The example assemblies used to test the localization strategies developed are described here. For the peg-in-hole assemblies, the peg is gripped by the ParaDex in its end-effector. Since the **tool-transform** (coordinate transform from the end-effector to the tool, or in this case, peg) is known, both the position and orientation of the peg can be considered to be known in the robot coordinate frame. For the key-lock assembly, the key is held by the

ParaDex, and again the position and orientation of the key can be considered to be known in the robot coordinate frame.

2.3.1 Circular Peg-in-Hole Assembly

The simplest of the example assemblies is a circular peg-in-hole assembly, shown in Figure 2.4, consisting of a hole of 100 mm diameter and a peg of 99.5 mm diameter. If the initial uncertainty in the position of the hole is limited to a 100 mm-diameter circle, then the task would be to find alignment coordinates that lie within a 0.5 mm-diameter circle located somewhere within a 100 mm-diameter bound. Note that an exhaustive spiral strategy would require a path of length more than 10 m to cover the entire uncertainty region!

Figure 2.4: The circular peg-in-hole assembly.

2.3.2 Square Peg-in-Hole Assembly

The second example is a square peg-in-hole assembly, shown in Figure 2.5. A 69.5 mm-edge peg is to be inserted into a 70 mm-edge hole. The initial posi-

tion uncertainty of the hole is bounded within a 70 mm-edge square. What makes this assembly more difficult than the circular peg-in-hole is that the orientation of the hole is also unknown. The maximum assembly clearance in this case is 0.5 mm along an edge. But, for the square peg-in-hole, the assembly clearance allows the peg to insert into the hole even if the peg has a slightly different orientation than the hole. Yet, as the relative orientation between the peg and the hole increases, the assembly clearance reduces proportionately. Thus, maximum assembly clearance (0.5 mm) is available only when the peg and hole have the same orientation. The assembly clearance for a relative orientation of θ between the peg and hole is given by

$$c = 70 - 69.5[\cos(\theta) + \sin(\theta)].$$

Thus, the assembly clearance reduces to zero for a maximum allowable orientation difference of 0.0072 rad. For greater differences in orientation between the peg and hole, assembly is not possible. Figure 2.6 shows a plot of the change in assembly clearance with changes in relative orientation between the peg and the hole.

The full search is carried out in three-dimensional configuration space corresponding to locating simultaneously the orientation (z -rotation) alignment and the position ((x, y) -plane) alignment to within assembly clearance. To find a lower bound on the ratio between the uncertainty volume and the assembly clearance volume, we find the smallest cube that completely encloses the assembly clearance volume. Note that Figure 2.6 shows only half the assembly clearance volume—there is an inverted pyramid attached at the base to the one shown, corresponding to negative relative orientations between the peg and hole. The volume of the smallest enclosing cube is then given by $0.5 \times 0.5 \times (2 \times 0.0072) = 0.0036$. If the uncertainty in the hole-position (x, y) is limited to a circle of diameter 50 mm and hole-orientation (θ_z) is limited to a range of $[0, \frac{\pi}{2}]$, then the assembly task would require alignment

Figure 2.5: The square peg-in-hole assembly.

of the peg and hole in position and orientation to within a unit assembly clearance volume located anywhere within an uncertainty volume more than 3.42×10^6 times its size.

2.3.3 Pentagonal Peg-in-Hole Assembly

The third example is a pentagonal peg-in-hole assembly, shown in Figure 2.7. Each side of the pentagonal peg is 37.5 mm in length, and each side of the hole is 37.9 mm in length. Hence, the assembly clearance along each side is 0.4 mm. The maximum orientation error that can be permitted is 0.0148 rad. With this orientation error, the planar assembly clearance reduces to zero, i.e., the peg and hole centers have to exactly aligned for assembly to be possible. The uncertainty in the position of the hole ranges over a circle of radius 15 mm, while the orientation uncertainty is unbounded (effectively, in the range $[0, \frac{2\pi}{5}]$ rad considering the symmetry of the pentagon).

Figure 2.6: The change in assembly clearance area with orientation error.

2.3.4 Key-Lock Assembly

The fourth example is a key-lock assembly. There is bounded uncertainty in the position of the lock in (x, y, z) . The key is held by the robot and is to be inserted into the key-hole on the lock. A photograph of the set-up is shown in Figure 2.8. While the assembly clearance between the key and the key-hole is on the order of several hundred microns, there is a spherical depression (of approximately 2.0 mm radius) around the key-hole, designed to guide the key into the key-hole. Hence, if the key is placed within a 2.0 mm-radius of the key-hole, and vertical pressure is applied while permitting horizontal compliance, the key is automatically guided into the key-hole.

If the lock-position uncertainty region is limited to a circle of radius 10 mm, then to locate the key-hole within this uncertainty region, an ex-

Figure 2.7: The pentagonal peg-in-hole assembly.

haustive spiral search path of length more than 113 mm would be required. However, while conducting a spiral search path, it might not be helpful to find the key-hole depression, because the tangential velocity of the key as it passes over the depression would prevent the compliant slipping of the key into the hole. Hence, when searching exhaustively a more conservative approach geared at finding key-hole alignment might be necessary, which would require the spiral path to be much longer than 113 mm.

Figure 2.8: Set-up for key-lock assembly.

2.3.5 Summary of Example Assemblies

The various example assemblies with their respective assembly clearances and uncertainty regions are listed in Tables 2.1 and 2.2. For the circular peg-in-hole assembly, two different position uncertainty regions were used in the experiments. Hence, these assemblies are distinguished with separate labels in Table 2.2. For example, in the Circular Peg-in-Hole I assembly, the hole-position uncertainty ranges over a circle of radius 50 mm; while in the Circular Peg-in-Hole II assembly, the hole-position uncertainty ranges over a circle of radius 25 mm.

<i>Assembly</i>	<i>Maximum Clearance</i>	
	Position: (x, y) -plane	Orientation: (θ_z)
Circular Peg-in-Hole	0.5 mm-radius circle	—
Square Peg-in-Hole	0.5 mm-edge square	± 0.0072 rad
Pentagonal Peg-in-Hole	0.4 mm-edge pentagon	± 0.0148 rad
Lock-Key	2.0 mm-radius circle	—

Table 2.1: Assembly clearance values for the various example assemblies.

<i>Assembly</i>	<i>Uncertainty region</i>	
	Position: (x, y) -plane	Orientation: (θ_z)
Circular Peg-in-Hole I	50 mm-radius circle	—
Circular Peg-in-Hole II	25 mm-radius circle	—
Square Peg-in-Hole I	20 mm-radius circle	$[0, \frac{\pi}{2}]$ rad
Square Peg-in-Hole II	10 mm-radius circle	$[0, \frac{\pi}{2}]$ rad
Pentagonal Peg-in-Hole	15 mm-radius circle	$[0, \frac{2\pi}{5}]$ rad
Lock-Key	10 mm-radius circle	—

Table 2.2: Hole-configuration uncertainty regions for the various example assemblies.

Chapter 3

From Exhaustive Searching to Intelligent Localization

For the class of assemblies with position uncertainties larger than the assembly clearance, there is no definite *a priori* trajectory, other than an exhaustive search throughout the uncertainty region, that will lead to successful alignment of the peg and hole. For these assemblies, the focus shifts from the dynamics of the assembly to the problem of searching for part alignment. In this chapter, we tackle the two basic methods that can be employed to search for part alignment. The first method concentrates on exhaustive searches through the uncertainty space for peg-hole alignment. The second method uses *intelligent* techniques to resolve the uncertainty in the hole configuration, which then helps in simple assembly.

We start with an analysis of the assembly uncertainty, and discuss the requirements for exhaustive searches through the uncertainty space to guarantee peg-hole alignment. Some example search paths for exhaustive coverage are presented, along with a comparison of their relative merits.

We then move on to describe intelligent strategies to localize peg-hole misalignment based on peg-hole contact position measurements. First, the **precession strategy** and the **sweep strategy** are presented. These strategies

are then subsumed into the **general localization strategy**. The precession strategy involves precessing a tilted peg while maintaining contact with the hole. The height of the peg is recorded during the precession, and this record is then used to compute the hole position relative to the peg. For the sweep strategy, the hole-configuration relative to the sweep can be computed from the recorded peg-heights as the peg is tilted and swept across the hole while maintaining contact with it. Both the precession and sweep strategies record contact measurements while executing pre-defined trajectories. The general localization strategy differs from these strategies in the fact that it records individual contact measurements and matches them to a pre-acquired map of the contact C-space.

3.1 Analysis of Assembly Uncertainty and Exhaustive Search Strategies

For assemblies with bounded uncertainty, the hole is within a certain region around its nominal position, which forms our **search region**. If the peg is moved to every point within this region, peg-hole alignment is guaranteed and assembly will be successful.¹ This is called **exhaustive coverage** of the search region. However, for a peg-in-hole assembly with positive assembly clearance, every point in the search region need not be visited to guarantee peg-hole alignment. We will show how a finite set of points can form an equivalent representation of the continuous search region. From here on in, exhaustive coverage refers to covering an equivalent finite set of points instead of the continuous search region. Drawing from this analysis, various search paths for exhaustive coverage of the search region will be presented.

¹The important assumption here is that the hole does not move during the search.

3.1.1 Configuration Space View of the Search Region

In the case of the forward clutch, described in Chapter 1, the assembly task consists of inserting a hub through five rings. For each ring, the hub has to be aligned simultaneously in position and rotation within assembly clearance. For simplicity, consider a modified forward clutch with just one ring and no teeth, which reduces it to a round peg-in-hole assembly. A drawing of this setup is shown in Figure 3.1(a). While the container-location is known, the position of the ring (and thus, the hole) in the container is not known. The amount of uncertainty in the hole-position is given by the mobility of the ring inside the container. Shifting to a C-space viewpoint, the planar dimensions of the hub and hole are reduced by the hub-diameter (i.e., consider the hub to be a point). The reduced hole-radius is equal to the assembly clearance (Figure 3.1(b)). Similarly, the ring and container dimensions are reduced by the ring-diameter. The reduced container area forms the search region (Figure 3.1(c)). The following section describes the reduction of the continuous search region to a finite set of search points based on the assembly clearance area, and derives the conditions to be imposed on a search strategy for search region coverage to guarantee assembly.

Figure 3.1: Assembly characterization in terms of C-space.

3.1.2 Exhaustive Coverage of Search Region

For the sake of clarity, the search area is labelled as H , and the assembly clearance area as C .² Since the assembly clearance is circular for the example being considered, let c be the radius of the assembly clearance circle. From the C-space viewpoint, established previously, the requirement for assembly is to get the peg-center within distance c of the hole-center. But, the hole-center is located anywhere within its region of mobility. Thus, the search area, H , or the uncertainty region, is the same as the area of mobility of the hole-center. Also, the peg and hole centers do not have to be exactly aligned for assembly, but only brought to within distance c of each other. Thus, it can be seen that the peg-center need not sweep H entirely to cover it exhaustively. Specifically, if a point covered by the peg-center fails to lead to peg-hole alignment, this ensures that the hole-center does not lie within distance c of that point. That is, for every point that the peg-center **visits**, it covers a part of H equaling the assembly clearance area C . Thus, H is reduced by C . Strictly speaking, the peg-center only needs to visit a certain set of points, that we term an **exhaustive coverage set**, such that for every point, P , in H , there is a point in the set within distance c of P .³ There is an infinity of exhaustive coverage sets that can be constructed by reducing H w.r.t. C in different ways. We denote the collection of these sets by $\mathcal{A}$. An example exhaustive coverage set is illustrated in Figure 3.2.

Next, the conditions for valid paths that cover the search area exhaustively are determined. An **exhaustive search path** is defined as any path that satisfies the requirement that every point in the search area is within distance c of some point on the path. In other words, an exhaustive path—when dilated by c —should cover the entire search region. We call this the

²For the simple example here, assembly clearance is two-dimensional. In higher dimensions, the notion of assembly clearance must be generalized to that of goal region, as explained in Chapter 5.

³In technical terms, the “dispersion” of the points in the set should be less than c [29].

Figure 3.2: An example of an exhaustive coverage set: covering each point in the set guarantees exhaustive coverage of the search region w.r.t. assembly clearance.

“spray paint” technique, by analogy to covering an entire surface with paint while traversing a path. As a counterexample, Figure 3.3 shows a path that is not exhaustive. There is at least one point in the search area (as shown in Figure 3.3) that is not within distance c of any point on the path. Hence, if the hole-center is located at this point, then the peg, traversing the given path, will never be within c distance of the hole-center, and assembly will not be possible.

3.1.3 Exhaustive Search Path Examples

This section looks at some examples of search paths that cover the search region exhaustively. It has already been determined that a path that visits every point in an exhaustive coverage set $B \in \mathcal{A}$ satisfies this requirement. Initially, several different paths are compared for shorter lengths. However, the significant criterion is assembly time, and shortest path length does not

Figure 3.3: Example of a search path that does not cover the search area exhaustively.

always translate to least coverage-time, as explained below. Hence, later, the comparison focuses on search paths that are not the shortest in length but are time-efficient.

Search Paths from Joining Points in Exhaustive Coverage Sets

Here, search paths obtained by joining the points in an exhaustive coverage set are analyzed. It is important to note that for any peg-in-hole assembly with fixed search region and assembly clearance, there is an infinite number of exhaustive coverage sets possible. The length-efficiency of search paths through these sets can be investigated at two different levels. First, the exhaustive coverage set with the smallest number of points can be found. Next, the shortest length path joining the points in this set can be computed. But, this does not guarantee that the resulting path will be of shortest length among all possible search paths. Generally, paths that are hand-crafted for a particular search area tend to be more efficient in terms of path-length, as shown later. However, a rigorous analysis of shortest path lengths is outside the scope of this dissertation. Here, a simple description of finding a short

path that contains a given set of points, $B \in \mathcal{A}$, is presented.⁴ In other words, the shortest route connecting all the points in B needs to be computed. This is analogous to the travelling salesman problem (TSP), although the restriction that a point should only be visited once is not necessary. The computation time for an exact solution to the TSP increases exponentially with the number of vertices (grid points in B), becoming rapidly prohibitive in cost as the vertices increase. To get an approximate solution to the TSP, simulated annealing was used [37]. One of the results is shown in Figure 3.4 below.

Figure 3.4: Solution path of approximate length $128c$ for a circular search area of radius $7c$, found using simulated annealing for an example exhaustive coverage set, $B \in \mathcal{A}$.

The length for the search path shown in Figure 3.4 is $128c$ for a circular search area of radius $7c$. But, as mentioned before, this solution does not guarantee shortest path length among all valid paths. Moreover, the path shown in Figure 3.4 has many corners and for a robot to traverse that path accurately it will need to decelerate to a complete stop at every corner. This will slow the robot down considerably, and even though the path is short,

⁴Here, B was obtained by dividing the circular search area into a rectangular grid with side lengths of $\sqrt{2}c$. The gridding strategy can be selected based on the shape of the search area for more efficiency.

the robot will take a long time in traversing it. So, the requirement that a path be smooth needs to be enforced, i.e., it should have a large radius of curvature at all points. It should be noted here, that the search path shown in Figure 3.4 can be made smoother by rounding the corners at each point. As long as the robot passes through each point in the set, the resulting search path will be valid.

Concentric Circles

The search path for a particular search area can be hand-crafted based on the shape of the area. A path made of concentric circles joined by a straight line is illustrated here for a circular search area. If the circular search region is covered with concentric circles spaced at $2c$ distance radially, then for every point in the search region, there is at least one point on the circles within distance c . Thus, a path composed of concentric circles satisfies the requirement for an exhaustive search path. An example path is illustrated in Figure 3.5. To make this path as short as possible, the outermost circle should be of radius $r - c$, where r is the radius of the search area-circle.

Figure 3.5: Path consisting of concentric circles joined by a straight line. Path length is $85c$ for a circular search area of radius $7c$.

Assuming the radius of the circular search area is $7c$, three circles of radii $6c$, $4c$, and $2c$, respectively, connected with a straight line extending to the center of the search area, will be required to cover the area exhaustively. Hence, the total path length will be $2\pi(6 + 4 + 2)c + 6c$, or approximately $81c$. Again, to make the path smoother, the line connecting the circles can be made more tangential to the circles instead of perpendicular as considered here.

Spiral Path

A spiral search path is short in length and has smooth curvature throughout. A spiral path can be parameterized to satisfy the requirement for an exhaustive search path by setting the pitch, p , equal to the assembly clearance diameter, $2c$. Then, for every point in the search region, there is at least one point on the spiral path within distance c . The dimensions of the spiral can be determined from the search area and the assembly clearance as follows:

$$\dot{r} = p \left(\frac{\dot{\theta}}{2\pi} \right). \quad (3.1)$$

Using this and fixing a constant path speed, $ds/dt = v$, we can derive

$$\begin{aligned} \dot{x} &= \dot{r} \cos \theta - r \dot{\theta} \sin \theta, \\ \dot{y} &= \dot{r} \sin \theta + r \dot{\theta} \cos \theta, \\ v^2 &= \dot{x}^2 + \dot{y}^2, \\ \Rightarrow v^2 &= \dot{r}^2 + r^2 \dot{\theta}^2. \end{aligned}$$

Substituting from Equation (3.1), we obtain:

$$\begin{aligned} v^2 &= \left(\frac{p}{2\pi} \right)^2 \dot{\theta}^2 + r^2 \dot{\theta}^2, \\ \Rightarrow \dot{\theta} &= \frac{v}{\sqrt{\left(\frac{p}{2\pi} \right)^2 + r^2}}. \end{aligned} \quad (3.2)$$

Figure 3.6: An example spiral search path. Path length is approximately $102c$ for a circular search region of radius $7c$.

Thus, $\dot{r}$ and $\dot{\theta}$ are obtained in terms of the assembly clearance by setting $p = 2c$:

$$\dot{\theta} = \frac{v}{\sqrt{\left(\frac{c}{\pi}\right)^2 + r^2}}, \quad (3.3)$$

$$\dot{r} = \frac{v}{\sqrt{1 + \frac{r^2}{(c/\pi)^2}}}, \quad (3.4)$$

To minimize the length of the spiral, these equations should be solved with initial conditions: $r(0) = R + c$ and $\theta(0) = 0$, where R is the radius of the search region. Applying these, the length of a spiral path covering a circular search region of radius $7c$ (as shown in Figure 3.6) can be computed approximately as $2\pi(7 + 5 + 3 + 1)c$, or approximately $101c$. The spiral search path was extensively used in robotic assemblies of the transmission components (for more details see [35]).

However, there is a drawback to employing exhaustive search strategies in combination with force-feedback control algorithms. Because we use NAC, the search path will be encoded as position commands to the attractor. While the robot follows the attractor, the control algorithm also makes it compliant

to environmental forces. Hence, the actual robot position is not guaranteed to follow the attractor accurately, especially in the presence of contact with the environment. Under this scenario, although the attractor executes an exhaustive search path, the robot is not guaranteed to do so.

3.2 Intelligent Localization Strategies

The analysis of search region coverage, described above, makes use of the fact that if a point covered by the peg-center fails to lead to peg-hole alignment, then this ensures that the hole-center does not lie within assembly clearance distance of that point. Thus, each assembly attempt at a different point can be interpreted as an observation that helps to reduce the uncertainty. Then the question is, can we extract more information out of each such observation for a more rapid reduction in uncertainty? In this context, we present our *intelligent* localization strategies, which are based on peg-hole contact position measurements, and allow us to more quickly resolve the uncertainty in hole-configuration. Once the peg-hole misalignment has been localized, it can be corrected, and assembly becomes easy. We initially developed the precession and sweep strategies, described in the next two subsections. We later subsumed these strategies under the general localization strategy, described in Section 3.3, that is the focus of this dissertation.

3.2.1 The Precession Strategy

The **precession strategy** is an intelligent localization strategy based on measurements of the peg position as it precesses while maintaining contact with the hole. To execute a precise precession trajectory the robot needs to be under stiff position control in the $(x, y, \theta_x, \theta_y)$ dimensions, i.e., positions along and rotations about the x - and y -axes. On the other hand, to maintain soft contact between the peg and hole, the robot needs to be under compliant

control along the vertical (z) axis. This combination of position and compliant control on selective axes is achieved using the hybrid control scheme described in Section 2.2. The precession strategy is described below in the context of a circular peg-in-hole assembly with hole position uncertainty in (x, y) .

The first step is to tilt the peg about a tilt axis, by the tilt-angle, θ_{tilt} . As shown in Figure 3.7(a), the tilt-axis, initially aligned with the negative x -axis, passes through the bottom center of the peg. Next, the peg is lowered into contact with the hole surface (Figure 3.7(b)). Using the hybrid compliant controller described in Section 2.2, a steady downward force is applied by the robot through the peg, while the tilt axis is rotated about the vertical axis, so the tilted peg precesses as shown in Figure 3.7(c).

Consider the initial condition shown in Figure 3.7(b).

The point of contact between the peg and the hole is on the hole surface. As the peg precesses, the contact point moves along the perimeter of the peg, and on a corresponding circular path on the hole surface until it reaches the hole edge. During this interval, the nominal height of the peg is constant. As the peg dips into the hole, the peg height decreases until a critical point where the peg is in contact with the hole edge in two places. With further precession, the peg rises out of the hole and the peg height increases. The change in peg height over the 360° precession is shown in Figure 3.8. This change in peg height reveals not only the direction of the hole center relative to the peg-position, but also the distance. The direction of the hole center is given by the vector perpendicular to the tilt-axis at the moment of minimum peg height. The distance of the hole center from the peg-center can either be calculated analytically from the decrease in peg height, or looked up from a table of sampled peg height values corresponding to peg-hole distance. A visualization of such a table is shown in Figure 3.9. The minimum peg height values recorded during precession for different relative peg-hole positions are

(a) Initial tilting of peg

(b) Contact with hole surface

(c) Peg precessed by 135°

(d) Contact with hole-edge

Figure 3.7: The precession strategy: (a) The peg is initially tilted by 15° . (b) The peg touches the hole surface, and peg height h_1 is recorded. (c) As the tilt axis is rotated, the peg precesses. (d) The peg dips into the hole, height $h_2 < h_1$.

plotted. Hence, for the experiment shown in Figure 3.8, the minimum peg-height of 14.91 mm would be matched to this table to obtain the possible peg positions relative to the hole. This is shown in Figure 3.9. The possible peg positions for the example experiment are 2 mm and -57 mm, respectively. With the relative peg-hole position localized to two possible values, we can proceed in a variety of different ways. One option is to select one of the two values and utilize it to compute the hole-configuration w.r.t. the peg, and

attempt assembly. If assembly fails, then we know for sure that the other value is the actual relative peg-hole position. Another option is to move the peg to a different position and repeat the precession strategy. The results from the two experiments analyzed together will be sufficient to localize the relative peg-hole position.

Figure 3.8: Plot of the change in peg height as it precesses by 360° . The angle of the tilt-axis where the minimum height is recorded can be used to compute the direction of the hole center w.r.t. the peg-position. Moreover, the value of the change in height can be compared against a pre-acquired table to find the distance of the hole center from the peg-center.

For the precession strategy to be successful, the precessing peg has to pass over the hole. For a circular peg-in-hole, this is guaranteed to happen if the hole-position is within distance $r_p \cos(\theta_{tilt})$ of the peg-center, where r_p is the radius of the peg, and θ_{tilt} is the angle of tilt of the peg. Hence, the hole-uncertainty region, U_r , has to be bounded by a circle of radius $r_p \cos(\theta_{tilt})$ for the precession strategy to be successful.

Figure 3.9: Plot of the minimum peg height (z) that would be recorded during precession for different relative peg-hole positions in (x, y) sampled every 0.5 mm distance.

3.2.2 The Sweep Strategy

Here, we present the **sweep strategy**, which is based on a principle similar to the precession strategy described above. However, instead of the tilted peg being precessed, it is swept across the search region (horizontally) while maintaining contact with the hole-surface. To follow the pre-planned sweep trajectory in the (x, y) -plane accurately, the robot needs to be under position control along the x and y dimensions. On the other hand, to maintain soft contact between the peg and hole during the sweep, the robot needs to be under compliant control along the vertical (z) axis. This combination of position and compliant control on selective axes is achieved using the hybrid control scheme described in Section 2.2.

Here, a square peg-in-hole assembly example is used to illustrate the sweep strategy. The goal is to resolve the (x, y, θ_z) uncertainty in the hole-configuration. The peg is tilted about a diagonal as shown in Figure 3.10(a),

and vertex A is brought into contact with the hole-surface. The sweep is conducted in a way such that the peg is dragged on the hole-surface as shown in Figure 3.10(a). As the peg starts the sweep outside the search region, its initial contact with the hole surface is outside the hole (Figure 3.10(b)). During the sweep, it encounters the hole, and the contact point then moves along the hole-edge. At this time, vertex A is inside the hole, and thus, the peg lowers into the hole. This is shown in Figure 3.10(c). Throughout the sweep, the peg height is sampled every 0.2 mm, and this record of peg heights is called the **sweep signature**. The sweep is brought to an end when the peg climbs back out of the hole, and vertex A establishes contact with the hole-surface. This state can be easily identified from the sweep signature.

Figure 3.11 shows the recorded peg heights for the entire sweep with the hole positioned w.r.t. the sweep as shown in Figure 3.10(a).

Now, for any hole-configuration (x, y, θ_z) , w.r.t. the peg, the peg height sweep signature is exactly the same as that for its **mirror image** hole-configuration, $(-x, y, -\theta_z)$, as shown in Figure 3.12.

However, every relative peg-hole configuration and its mirror image share a *unique* peg height sweep signature that is not shared by any other configuration. Thus, the recorded peg heights can be used to resolve the hole-configuration uncertainty down to two possible configurations. The tie between the two configurations can be broken in a variety of ways, including recording another sweep at a different location, or by making a single extra measurement. If another sweep is recorded, then out of the two resulting configurations, only one will match one of the configurations obtained from the previous sweep, and thus the real hole-configuration will be revealed. The hole-configuration can be computed analytically from the two sweeps recorded, or looked up from a table of sweep signatures corresponding to relative sweep-hole configurations.

In our implementation, the latter technique was employed. The four-

Figure 3.10: The sweep strategy: (a) The peg is initially tilted by 15° about a diagonal, (b) Peg is brought into contact with the hole-surface, (c) The peg is swept over the hole while maintaining contact, and peg height is recorded throughout, (d) Peg dips into hole and contact point on the peg shifts from vertex A to peg-edge.

dimensional contact hyper-surface map was first decomposed into several three-dimensional (x, y, z) -slices for fixed relative orientation values. Each (x, y, z) map was then sliced along (y, z) (y being the sweep direction) at discretized intervals in x to obtain a collection of sweep signatures with associated peg-hole misalignment values. To match an actual sweep signature to this table of sweep signatures, it is broken up into its component straight-line slopes. For instance, the sweep shown in Figure 3.11 would be broken up into

Figure 3.11: Peg-height record (or, sweep signature) for the sweep as shown in Figure 3.10(a).

five parts. The matching is then carried out in two steps:

1. A straight line is fit to each part of the sweep using a least squares fit, and this line is used to compute the slope of that part.
2. The sweep is then matched to the table of sweep signatures in a hierarchical manner as follows. First, the tabulated sweeps that have the same number of parts as the current sweep are culled (short-listed). Next, the sweep is matched to the culled table by comparing slopes. Depending on the number of slopes in the current sweep, a particular slope is picked as the crucial one for comparison. The tabulated sweep that has a slope closest in value to the crucial slope on the current sweep is then selected as the closest matching sweep.

The hole-configuration corresponding to the closest matching tabulated sweep is then the new hole-configuration estimate.

The amount of hole-configuration uncertainty that can be tolerated by the sweep strategy can be determined easily. The strategy will be successful

Figure 3.12: Both the actual hole configuration (a), and the mirror image (b), would produce the same sweep signature.

if, during the sweep, the peg contacts the hole-edge, and the vertex A lowers into the hole. Specifically, for a square peg-in-hole assembly the worst-case scenario occurs when the hole is oriented such that a hole-edge is parallel to the sweep direction. In this case, for a hole-edge of length l , the hole position cannot be at a distance more than $l/2$ from the sweep line. This scenario is shown in Figure 3.13.

3.3 General Localization Strategy

Consider the peg-in-hole assembly illustration of Figure 3.14. As shown in the figure, we attach body-fixed coordinate frames to the peg and the hole at points P and H , respectively. We denote the Cartesian coordinate frames at P and H as $\mathcal{F}_p$ and $\mathcal{F}_h$, respectively. Because we assume that the hole is stationary, $\mathcal{F}_h$ is invariant. The position of the peg coordinate frame w.r.t. the hole coordinate frame is given by the position vector $\overrightarrow{HP} \in \mathcal{R}^3$. The orientation of the peg coordinate frame w.r.t. the hole coordinate frame can be represented by a series of rotations about the hole axes, H_x , H_y , and H_z .

Figure 3.13: Worst-case scenario: Hole is at the edge of the uncertainty region. The sweep is always conducted so that vertex A passes through the center of the hole uncertainty region. If the hole uncertainty region is bounded by a square of edge-length l , then even for worst-case, the sweep-line will be distance of $l/2$ from the hole center, and vertex A will pass over the edge of the hole as shown.

Let θ_x , θ_y , and θ_z , be the rotations about the body-fixed axes, in that order. The orientation of the peg w.r.t. $\mathcal{F}_h$ can then be given by the rotation matrix $R = R(\theta_z)R(\theta_y)R(\theta_x)$, where $R(\theta_z)$, $R(\theta_y)$, and $R(\theta_x)$ are individual rotation matrices corresponding to the θ_z , θ_y , and θ_x rotations respectively. E.g., the rotation matrix $R(\theta_z)$ is of the form

$$R(\theta_z) = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Thus, the coordinate transformation from $\mathcal{F}_p$ to $\mathcal{F}_h$, denoted by T_p^h , can be put in either a six-element form, $(x, y, z, \theta_x, \theta_y, \theta_z)^5$, or a **homogeneous**

⁵In this dissertation, we use $(x, y, z, \theta_x, \theta_y, \theta_z)$ to denote the space of configurations unless otherwise mentioned

form [1] as follows:

$$T = \left[\begin{array}{ccc|c} & & & x \\ & R & & y \\ & & & z \\ \hline \bar{0} & \bar{0} & \bar{0} & \bar{1} \end{array} \right].$$

In the homogeneous form, multiple consecutive transformations can be combined into a single matrix by simple matrix multiplication. Hence, this form is extensively used throughout the dissertation. But, the elements of the homogeneous coordinate transformation matrix are not independent, as they represent only six degrees of freedom. On the other hand, the six-element form uses only three independent variables to represent orientation. Hence, this representation is well-suited for configuration space descriptions and the observations recorded therein, and is used for that purpose in this dissertation. An observation is recorded in the six-element form, and then converted to the homogeneous form for further manipulation. This conversion requires computing rotation matrices corresponding to the roll-pitch-yaw angles, and is relatively simple. On the other hand, the inverse transformation—from the homogeneous form to the six-element form—is more difficult.

In this dissertation, we use calligraphic symbols, e.g., $\mathcal{T}$, to denote sets of coordinate transformations or neighborhoods of configurations. Of course, ordinary set operations, like union and intersection, can be applied to these sets. We also make use of **transformations** of such sets. For a neighborhood of configurations (or, a set of transformations), $\mathcal{A}$, and a given coordinate transformation, T , the operation $T\mathcal{A}$ is defined as the transformation of the entire neighborhood of configurations:

$$T\mathcal{A} = \{S \mid S = TA, \text{ for some } A \in \mathcal{A}\}.$$

Consider the six-dimensional configuration space given by the position and orientation of the peg relative to the hole coordinate frame. The relative

Figure 3.14: Coordinate frames associated with the square peg-in-hole.

configurations possible⁶ are represented by a six-dimensional hyper-volume, which is bounded by a hyper-surface that corresponds to all the points where the peg is in contact with the hole. We refer to it as the **contact hyper-surface** in C-space. Figure 3.15(b) shows a three-dimensional (x, y, z) *slice* of this hyper-surface for fixed relative orientations between the square peg and hole (Figure 3.15(a)). Our strategy is based on localizing the relative peg-hole configuration on the contact hyper-surface by matching actual contact observations to a stored map of the hyper-surface. The map can be computed analytically or sampled using a robot (or, perhaps a CAD model).

Let us recall the formulation of the localization problem stated in Section 1.1. The goal is to find the transformation T_p^h , from $\mathcal{F}_p$ to $\mathcal{F}_h$, where $\mathcal{F}_p$ is the coordinate frame attached to the peg, and $\mathcal{F}_h$ is the coordinate frame attached to the hole. We want to do this by localizing the relative peg-hole configuration on a map of the contact hyper-surface.

Consider n contact point observations: $O_i = (x, y, z, \theta_x, \theta_y, \theta_z)_i$, $i = 1, \dots, n$. Each observation is a record of the peg configuration in some

⁶That is, configurations not resulting in volumetric interference.

Figure 3.15: Three-dimensional contact hyper-surface slice for a square peg-in-hole assembly for the fixed relative orientation between peg and hole, as shown in (a).

robot coordinate frame, say $\mathcal{F}_w$. Let the corresponding homogeneous coordinate transformation from $\mathcal{F}_p$ to $\mathcal{F}_w$ be given by $T_{p_i}^w$. $T_{p_i}^w$ represents the i th peg configuration in the robot coordinate frame. These observations result from the robot *re-posing* (translating/orienting) the peg to n distinct configurations in contact with the stationary hole. Hence, they are samples of the six-dimensional contact hyper-surface. But, with the hole-configuration unknown, the observations themselves have no basis. We can also represent each peg move as the coordinate transformation from the new (or $(i+1)$ th) peg configuration to the old (or, i th) peg configuration. Then, $T_{p_{i+1}}^w = T_{p_i}^w T_{p_{i+1}}^{p_i}$. Alternatively, we can find the peg move from the observations as, $T_{p_{i+1}}^{p_i} = (T_{p_i}^w)^{-1} T_{p_{i+1}}^w$.

The map, M , of the contact hyper-surface is a collection of coordinate transformations, T_p^h , from $\mathcal{F}_p$ to $\mathcal{F}_h$, that result in peg-hole contact. Although the actual observations have no relation to the map, the moves do. Each relative peg-hole contact configuration, $T_{p_i}^h$, is represented on the map. These configurations are related through the peg moves as follows:

$$T_{p_{i+1}}^h = T_{p_i}^h T_{p_{i+1}}^{p_i}.$$

We localize the actual relative peg-hole configuration on the map by incre-

Algorithm 3.1

- Step 0: • Set $k = 0$
- Initialize belief set, $\mathcal{B}_0$
 - Record observation: $T_{p_0}^w$
- Step 1: • Update $k = k + 1$
- Record observation: $T_{p_k}^w$; [0.5em] find $T_{p_k}^{p_{k-1}}$
 - For each configuration in the belief set, $T_b \in \mathcal{B}_{k-1}$:
 if $T_b T_{p_k}^{p_{k-1}} \in M$, then include $T_b T_{p_k}^{p_{k-1}}$ in $\mathcal{B}_k$
- Step 2: • Check if $\mathcal{B}_k$ is satisfactory $\Rightarrow$ goto Step 3
- Else check if too many iterations $\Rightarrow$ goto Step 4
 - Else goto Step 1
- Step 3: • (Localization successful) Assemble
- Step 4: • (Localization failed) Stop

Figure 3.16: Algorithm for implementing the general localization strategy.

mentally matching the peg moves to the map using the above relation. As more peg moves are matched, the number of qualifying configurations generally decreases, and the localization is improved.

The incremental algorithm for implementing the general localization strategy is presented in Figure 3.16. The initial uncertainty in the relative peg-hole configuration is given by the maximum possible uncertainty in hole-configuration. All the configurations on the map falling under this region are included in the *belief set*, $\mathcal{B}$, which is then updated after every iteration to represent the collection of estimates of the current relative peg-hole configuration. Each iteration, indexed by k , involves moving the peg from one contact configuration to another, making an observation, and matching the move to the map.

Localization in Two Dimensions

To gain a better understanding of the implementation of the localization strategy, let us revisit the simple two-dimensional problem described in Section 1.1. As shown in Figure 3.17, the probe moves along a surface in (x, z) .

Figure 3.17: (a) Continuous surface and probe moves. (c) (Subset of) possible candidates for probe move. (e) Advancing the previous candidates. (b–f) Successive pruning of candidate transformations.

The starting position of the probe on the surface is selected arbitrarily and is not known. Also, the direction of motion ($\pm x$) of the probe (i.e., orientation of the probe w.r.t. the surface) is unknown. We need to localize the position and orientation of the probe coordinate frame w.r.t. the coordinate frame attached to the surface. This is a transformation from the probe coordinate frame to the surface coordinate frame, which we denote T_p^s . The complete transformation space is in three dimensions (x, z, θ) . A map, M , of all possible configurations of the probe coordinate frame w.r.t. the surface coordinate frame (i.e., transformations T_p^s) with the probe in contact with the surface is available. Note: $\theta \in \{1, -1\}$, where $\theta = 1$ represents positive probe

orientation (the probe and surface x -axes point in the same direction) and, $\theta = -1$ represents negative probe orientation.

Initially, there is global uncertainty in the configuration of the surface coordinate frame. Hence, our initial belief set, $\mathcal{B}_0$, covers the entire transformation space map, M . This is illustrated by the grey-colored area in Figure 3.17(b).

The arrow marked ‘1’ in Figure 3.17(a) shows the first probe move. The observations O_0 and O_1 are recorded at the beginning and end of the move, which give the position and orientation of the probe w.r.t. a world frame: $T_{p_0}^w$ and $T_{p_1}^w$ respectively. We compute the probe move, $T_{p_1}^{p_0} = (T_{p_0}^w)^{-1} T_{p_1}^w$. The belief set is then updated by matching the probe move to the map as follows. We advance each configuration, T_b , in the belief set, $\mathcal{B}_0$, by the probe move, $T_{p_1}^{p_0}$, and check whether the resulting configuration, $T_b T_{p_1}^{p_0}$ belongs to the map. If it does, then the resulting configuration is included in the new belief set, $\mathcal{B}_1$. *Some* of the possible candidates for the first move are shown in Figure 3.17(c). The candidate configurations are shown in the transformation space of Figure 3.17(d). The actual current transformation has been circled.

The second move, marked as ‘2’ (Figure 3.17(a)), yields observation O_2 . Again we update the belief set by matching the new peg move, $T_{p_2}^{p_1}$, to the map. This is the same as advancing each of the previous candidate configurations by $T_{p_2}^{p_1}$, and selecting the ones that land on the map. As seen in Figure 3.17(e), most of the candidates fail to accommodate the new move (advancing them by the new move takes them away from the curve). We can see that the only surviving candidate is the true representative of actual probe motion. Figure 3.17(f) shows the pruned belief set, $\mathcal{B}_2$. In general, this procedure would be repeated until the belief set is reduced to a single relative probe-surface configuration, or all the remaining configurations are contained within the assembly clearance. In the latter case, any one of the remaining configurations can be selected as the estimated configuration. When the

peg is aligned with the estimate for insertion, it is guaranteed to be within assembly clearance of the actual, and assembly will be successful.

3.4 Conclusions

In this chapter, we tackled the two basic methods that can be employed to search for part alignment. The first method concentrates on exhaustive searches through the uncertainty space for peg-hole alignment. The second method uses *intelligent* techniques to resolve the uncertainty in the hole configuration.

We started with an analysis of the assembly uncertainty and discussed the requirements for exhaustive searches through the uncertainty space to guarantee peg-hole alignment. Some example search paths for exhaustive coverage were presented, along with a comparison of their relative merits.

We then moved on to describe intelligent strategies to localize peg-hole misalignment based on peg-hole contact position measurements. First, the **precession strategy** and the **sweep strategy** were presented. These strategies were then subsumed into the **general localization strategy**, which forms the basis of all the Algorithms presented in Chapters 4–7.

Chapter 4

Assembly Localization Using Analytical Maps

In Chapter 3, several intelligent localization strategies were introduced, including the precession and sweep strategies, and the general localization strategy. The general localization strategy involves matching individual peg-hole contact measurements to a map of the contact C-space. The map of the contact hyper-surface can be either analytical or sampled. In this chapter, the implementation of the general localization strategy using analytical maps is described.

The contact hyper-surface map is a record of all peg-configurations w.r.t. the hole coordinate frame, $\mathcal{F}_h$, that result in peg-hole contact. For a general assembly, with d -dimensional uncertainty, each configuration is described in d dimensions. An analytical form of the contact hyper-surface map would involve one or more functions derived from an analysis of volumetric intersections between the peg and hole of the type:

$$f_1(\underline{X}) = \gamma_1, \text{ for } \underline{X} \in \mathcal{X}_1, \quad (4.1)$$

$$f_2(\underline{X}) = \gamma_2, \text{ for } \underline{X} \in \mathcal{X}_2, \quad (4.2)$$

$$\vdots \quad (4.3)$$

$$f_m(\underline{X}) = \gamma_m, \text{ for } \underline{X} \in \mathcal{X}_m, \quad (4.4)$$

where each $f_i : \mathcal{R}^d \rightarrow \mathcal{R}$, $\underline{X} \in \mathcal{R}^d$, and each γ_i is a constant. Also, each $\mathcal{X}_i \in \mathcal{R}^d$ is a region of configurations where f_i is valid. For assemblies with simple geometrical shapes, e.g., round and square peg-in-hole assemblies, it is relatively easy to generate analytical maps of the contact hyper-surface. Below, we give a brief description of analytical maps generated for these assemblies. We also describe in detail the procedure for localizing the peg-hole misalignment for these assemblies using such an analytical map, and present results from computer simulations.

4.1 Generating an Analytical Map

As mentioned before, a general six-dimensional analytical map would take the form of a collection of functions applicable in appropriate regions of the configuration-space. For instance, consider a simplified circular peg-in-hole assembly where the uncertainty in hole-configuration is restricted to the horizontal plane (x, y) . The analytical map for this assembly would then be a collection of two functions and their corresponding constraint equations:

$$f_1 : \mathcal{R}^3 \rightarrow \mathcal{R}, \quad f_1(x, y, z) = \gamma_1, \text{ for } x^2 + y^2 \leq c^2, \quad (4.5)$$

$$f_2 : \mathcal{R}^3 \rightarrow \mathcal{R}, \quad f_2(x, y, z) = \gamma_2, \text{ for } x^2 + y^2 > c^2, \quad (4.6)$$

where (x, y) is the position of the peg relative to the hole, and c is the assembly clearance radius. The two regions represented by these functions are shown in Figure 4.1(b). The function f_1 represents all the peg configurations within assembly clearance of the hole, leading to assembly and the peg position, z_1 , after insertion. On the other hand, f_2 represents all the peg configurations not within assembly clearance of the hole, and the peg sits on the hole surface, as shown in Figure 4.1(a).

Figure 4.1: Analytical map for a round peg-in-hole assembly with matching relative orientations between the peg and the hole.

It is important to note here that while the entire transformation space would be infinitely large, only a finite portion is mapped. At first glance one would assume that the extents of the map should equal the bounds on the hole-configuration uncertainty. But, actually the map needs to be more extensive. To understand this, consider the following worst-case scenario. The initial hole configuration is at the edge of the uncertainty bound. If the first move takes the peg in the opposite direction, then the relative peg-hole configuration is greater than the hole uncertainty bound. Since all relative peg-hole configurations possible need to be mapped, the map needs to encompass the hole uncertainty region augmented by a **peg move region**, as shown in Figure 4.2. Here, the peg move region is centered at the initial peg configuration, and the peg is constrained to move within this region.

Below, we describe the analytical maps generated for circular and square peg-in-hole assemblies.

4.1.1 Circular Peg-in-Hole Assembly

For the general circular peg-in-hole assembly, the uncertainty space is reduced to five dimensions, as the rotation of the hole about the vertical axis

Figure 4.2: The region to be mapped encompasses the hole uncertainty region augmented by the peg move region.

need not be resolved. Hence, it would require a five-dimensional map in the $(x, y, z, \theta_x, \theta_y)$ -space of peg-configurations relative to the hole, and the analytical form of the contact surface map would be given by Equations (4.1)–(4.4) with $d = 5$.

The physical significance of the surface can be explained as follows. Consider the five-dimensional configuration space given by the position and orientation of the peg coordinate frame, $\mathcal{F}_p$, w.r.t. the hole coordinate frame, $\mathcal{F}_h$. See Figure 4.3. Any point $(x, y, z, \theta_x, \theta_y)$ in that space represents a peg-configuration w.r.t. $\mathcal{F}_h$, and can be obtained by translating $\mathcal{F}_p$ (initially aligned with $\mathcal{F}_h$) by (x, y, z) , and then rotating it about its x and y axes by θ_x and θ_y , respectively, in that order. The contact hyper-surface represents all such points that result in contact between the peg and hole.

Generating an analytical map of this five-dimensional hyper-surface would be a difficult task. Instead, consider a constrained problem where the hole configuration-uncertainty is restricted to the (x, y) -space. This means that the orientation of the hole (rotations about θ_x and θ_y) and its vertical po-

Figure 4.3: Coordinate frames associated with the peg and the hole.

sition are known.¹ Hence, only the space of (x, y, z) configurations of the peg w.r.t. $\mathcal{F}_h$ needs to be mapped.² Although this allows us the freedom to choose whatever relative orientation (between peg and hole) we prefer for the map, we need to ensure that the orientation during assembly matches our original choice. If we choose matching orientations for peg and hole (i.e., zero relative orientation), as shown in Figure 4.1(a), then the relative configuration space can be represented by two functions similar to the ones described in Section 4.1. But, this monotonous (featureless) contact surface, shown in Figure 4.1(b), would not provide any clues to our localization strategy—unless the peg falls into the hole and assembly is achieved. Instead, if we allow a small orientation difference between the peg and the hole about a fixed axis, the contact surface is much more informative, as shown in Figure 4.4. To generate an analytical map for this surface, we need to find the collection of functions that give the contact-height for all possible peg posi-

¹This assumption is justified, because, even with the hole-orientation about (θ_x, θ_y) unknown, using compliant control, the peg can be made to lie flat on the hole-surface. Once this is achieved, the peg's orientation about (θ_x, θ_y) matches that of the hole, and the hole-uncertainty in (θ_x, θ_y) is resolved. It also resolves the uncertainty in the vertical position of the hole-surface.

²It is possible to localize using a contact C-space map of (x, y) -configurations for a fixed z . However, this complicates the exposition.

Figure 4.4: Peg-hole contact hyper-surface with tilted peg.

tions relative to the hole. These functions also depend on the sizes of the peg and hole, and the amount of tilt.

For a hole of 100 mm diameter and a peg of 99.5 mm diameter, with a peg-tilt of 15° about the x -axis, the contact surface can be described by a collection of four functions. Each of these four functions is applicable in a different region of the surface as described in Figure 4.5. Region $\mathcal{X}_1$ corresponds to the peg contacting the hole surface outside the hole: $x^2 + y^2 > R^2$, and the corresponding function is $f_1(x, y, z) = \gamma_1$. Regions $\mathcal{X}_2$ and $\mathcal{X}_3$ are symmetric about the peg y -axis (or, the axis perpendicular to the axis of tilt) and correspond to the lower rim of the peg contacting the hole rim. Region $\mathcal{X}_4$ corresponds to contact between the cylindrical surface of the peg and the hole rim. For Region $\mathcal{X}_4$, computing the intersection point between a curved surface and a circle is computationally intensive, and so an approximate wire-figure of the peg is used, as shown in Figure 4.6(a). The wire-figure approximates the cylindrical surface of the peg that generates the surface in Region $\mathcal{X}_4$ with a line, which we term the **surface-line**. This approximation introduces only minor errors in the resulting contact surface compared to the original, because Region $\mathcal{X}_4$ forms a very small portion of the contact surface. To compute the contact height of the peg for a given position

Figure 4.5: The different regions of contact on the peg and hole that contribute to corresponding regions on the contact surface map for a circular peg-in-hole assembly.

w.r.t. the hole, the wire-figure-approximation is projected into the plane of the hole. The projection of the lower rim of the peg becomes an ellipse with the tilt axis as the major axis. This is shown in Figure 4.6(b). Then, the points where the hole-circle intersects the peg-ellipse and the surface-line are computed. Once these intersection points are determined, the height of the peg for each possible intersection is computed. The intersection that yields the largest peg-height represents the first contact between the peg and hole as the peg is lowered into contact with the hole. Since, the peg cannot be lowered any further after this contact is established, the other intersections can never occur. Hence, the largest peg-height is the actual peg-height on contact with the hole at the given (x, y) -position. Figure 4.7 shows two views of the peg-hole contact surface generated.

Figure 4.6: The wire-figure approximation of the peg projected into the hole-plane. Intersection points between the hole-circle and peg-ellipse are computed to find the contact height of the peg.

Figure 4.7: Top and bottom views of the peg-hole contact surface, with the peg tilted by 30° w.r.t. the hole.

4.1.2 Square Peg-in-Hole Assembly

A square peg-in-hole assembly presents a six-dimensional localization problem. The analytical map of the peg-hole contact hyper-surface takes the form of Equations (4.1)–(4.4) with $d = 6$. The physical significance of the surface can be explained as follows. Let $\mathcal{F}_p$ and $\mathcal{F}_h$ be body-fixed coordinate frames attached to the peg and hole, respectively. See Figure 3.14 for an illustration of the coordinate frames. If the position of $\mathcal{F}_p$ w.r.t. $\mathcal{F}_h$ (i.e., $\vec{HP}$) is

(x, y) , and the orientation of $\mathcal{F}_p$ w.r.t. $\mathcal{F}_h$ is given by rotations of θ_x , θ_y , and θ_z , about the P_x , P_y , and P_z axes, respectively, in that order, then the peg position relative to the hole on contact is z along the H_z -axis.

Generating an analytical form of the contact hyper-surface map for the circular peg-in-hole assembly required solving equations for the intersection of curves and even curved surfaces. However, for the square peg-in-hole, intersections are limited to the straight line edges of the peg and the hole. Although there are instances where the hole contacts the peg on one of its vertical faces, these can be identically modelled as contact on an edge of that face. All that said, we still constrain the square peg-in-hole problem so that the hole configuration-uncertainty is restricted to the (x, y, θ_z) space. This means that the orientation of the hole about any horizontal axis is known, and so is the vertical position of the hole-surface.³ Hence, only the space of (x, y, z, θ_z) configurations of the peg w.r.t. the hole needs to be mapped. Although this allows us the freedom to choose whatever relative orientation (between peg and hole) about any horizontal axis we prefer for the map, we need to ensure that the orientation during assembly matches our previous choice. If we choose matching orientations for peg and hole (i.e., zero relative orientation) about any horizontal axis, then one of two conditions is possible. Either the relative orientation between the peg and hole about the vertical (z -axis) is within maximum orientation clearance, or it is not. In the latter case, where the relative orientation is greater than maximum orientation clearance, all peg positions and orientations result in the same contact height, producing a flat, featureless surface. Even in the former case, all peg positions except for the small subset that fall within assembly clearance of the hole, produce the same flat surface. The small

³This assumption is justified, because, even with the hole-orientation about (θ_x, θ_y) unknown, using compliant control, the peg can be made to lie flat on the hole-surface. Once this is achieved, the peg's orientation about (θ_x, θ_y) matches that of the hole, and the hole-uncertainty in (θ_x, θ_y) is resolved. It also resolves the uncertainty in the vertical position of the hole-surface.

subset of peg-configurations within assembly clearance of the hole generate another flat surface at a different height. This is shown in Figure 4.8(b). This generally monotonous (featureless) contact surface does not provide any clues to our localization strategy, unless the peg falls into the hole and assembly is achieved. Instead, as with the circular peg-in-hole, if we allow a small orientation difference between the peg and the hole about a fixed axis, the contact surface is much more informative, as shown in Figure 4.8(d).

Figure 4.8: Contact hyper-surface slices for fixed orientation differences between peg and hole.

To generate an analytical map for this surface, we need to find the collection of functions that give the contact-height for all possible peg-positions in the horizontal plane and peg-orientations about the vertical axis relative to the hole. These functions also depend on the sizes of the peg and hole, the tilt-axis, and the amount of tilt.

For a square hole of 70 mm edge and a peg of 69.5 mm edge, with a

peg-tilt of 10° about a diagonal aligned with the x -axis, we can describe the contact surface with the help of four functions. For a fixed relative orientation between the peg and the hole, each function represents a plane arising from the intersection of a specific peg-edge with a hole-edge. Consider the illustrations in Figure 4.9 where the relevant peg- and hole-edges are highlighted. Because the square peg-in-hole is symmetrical in rotation about the vertical z -axis by a period of $\pi/2$, it is sufficient to only consider the interval $[0, \pi/2]$ rad of relative peg-hole orientations. Two relative orientations, $\frac{\pi}{4}$ rad and 0 rad, are shown in Figures 4.9(a) and 4.9(b), respectively. Within this interval, we can determine the possible intersections between the peg- and hole-edges. The only peg edges that will ever be in contact with the hole are: ab , ad , and ah . The possible intersections are: ah can only intersect AD , ab can intersect AB , and ad can intersect BC and CD . When ab intersects BC , ad has to be intersecting BC as well, and so taking one of the two intersections into account will be enough. There are other special cases where a peg-vertex will contact a hole-edge or a hole-vertex, in which case technically more than one peg-edge intersects a hole-edge or more than one hole-edge. However, for our purpose it is sufficient to consider one of those intersections.

Using these functions, an analytical map of the contact hyper-surface can be generated for a square peg-in-hole assembly. This map can then be used to implement the general localization strategy. Figure 4.8(d) shows a three-dimensional slice (in (x, y, z)) of the contact hyper-surface for fixed relative orientations between the peg and the hole.

4.2 Localization Using Analytical Maps

Once a map of the contact hyper-surface has been generated, the localization strategy can be implemented. The goal of the localization process is to locate the actual (unknown) coordinate transformation, T_p^h , from $\mathcal{F}_p$ to $\mathcal{F}_h$ on the

Figure 4.9: The only peg- and hole-edges that will come into intersection for relative orientations within $[0, \pi/2]$ rad.

map. In this section, we start with a general description of the localization strategy using analytical maps, given in Algorithm 3.1 (Figure 3.16). Later, the localization procedure will be described in detail for the constrained circular and square peg-in-hole examples discussed above.

Initially, there is global uncertainty in the hole-configuration, so the belief set, $\mathcal{B}_0$, is initialized to include all the transformations corresponding to the hole uncertainty region. Note, however, that this is just a portion of the mapped region. As discussed earlier (see Section 4.1) the mapped region, M , is the hole-uncertainty region augmented by the peg move region. Let the configurations in M that fall under the hole-uncertainty region be denoted by U . Then $U \subseteq M$, and we set $\mathcal{B}_0 = U$. The initial peg configuration is at the center of the hole-uncertainty region. Let the actual initial—but as yet unknown—relative peg-hole configuration be $T_{p_0}^h$. However, $T_{p_0}^h \in \mathcal{B}_0$. To begin the localization process, we bring the peg into contact with the hole and record the first observation, O_0 . This is a sampling of the six-dimensional contact hyper-surface, and is of the form $(x, y, z, \theta_x, \theta_y, \theta_z)$. Next, the peg is *re-posed* (translated and rotated) to a new contact point, and observation

O_1 is recorded. These observations correspond to peg configurations $T_{p_0}^w$ and $T_{p_1}^w$ w.r.t. some robot coordinate frame, $\mathcal{F}_w$. From these observations, we can compute the first peg move, $T_{p_1}^{p_0} = (T_{p_0}^w)^{-1}T_{p_1}^w$. To match this move to the map, we advance each configuration in the belief set by the move, and if the resulting configuration belongs to the map, we include it in the new belief set. Thus, for each $T_b \in \mathcal{B}_0$, if $T_b T_{p_1}^{p_0} \in M$, then we include $T_b T_{p_1}^{p_0}$ in $\mathcal{B}_1$. The new actual—but unknown—peg-hole configuration is $T_{p_1}^h = T_{p_0}^h T_{p_1}^{p_0}$. Note that $T_{p_1}^h \in M$, so that it will be included in $\mathcal{B}_1$.

In general, for iteration k , for each $T_b \in \mathcal{B}_{k-1}$, if $T_b T_{p_k}^{p_{k-1}} \in M$, then we include $T_b T_{p_k}^{p_{k-1}}$ in $\mathcal{B}_k$. This procedure is repeated until the belief set is pruned to a single configuration, or the remaining configurations are contained within the assembly clearance, at which point localization is complete. In the latter case, any one of the remaining configurations can be selected as the estimated configuration. When the peg is aligned with the estimated hole configuration for insertion, it is guaranteed to be within assembly clearance of the actual configuration, and assembly will be successful.

4.2.1 Localization for Circular Peg-in-Hole Assembly

The general circular peg-in-hole assembly is a five-dimensional uncertainty problem, as described in Section 4.1.1. Here, we describe localization for the circular peg-in-hole assembly using an analytical map. For ease of illustration, the general problem is simplified by assuming that the orientation of the hole-surface and its vertical position are known. In fact, we assume that the peg and hole have matching initial orientations. Thus, as the uncertainty in the relative peg-hole configuration is limited to the (x, y) space, the five-dimensional search space of coordinate transformations is reduced to two dimensions of simple translation in (x, y) . It was shown in Section 4.1.1 that the contact surface for matching peg-hole orientations is flat and monotonous, while, for mismatched orientations, we have a constantly varying and infor-

mative surface. To take advantage of this fact, we purposely tilt the peg by a known amount w.r.t. the hole. For the experiments here, the peg-tilt was fixed at 15° about its body-fixed x -axis w.r.t. the hole. With fixed relative orientations, the five-dimensional contact hyper-surface collapses to three-dimensions: (x, y, z) , as shown in Figure 4.7.

For this simplified problem, localization in the (x, y) space can be carried out as follows. The tilted peg is moved down along the hole z -axis to its first contact with the hole, as shown in Figure 4.10. The peg-position $T_{p_1}^w$ is recorded in the robot coordinate frame. Here, $T_{p_1}^w$ is an observation in (x, y, z) . As the vertical placement of the hole-surface is known to the controller, the recorded height in z , can be directly matched to the map. This corresponds to an (x, y) -slice of the contact surface map at the recorded z , as shown in Figure 4.11. The points thus obtained give all the peg (x, y) -positions relative to the hole that would result in the recorded height on peg-hole contact. This is shown in Figure 4.10. It can be seen that the actual peg position—which is unknown at this point—is included in this locus. Thus, we obtain our first belief set, $\mathcal{B}_1$, of (x, y) transformations from the peg frame to the hole frame.

Next, the peg is translated (preserving its orientation relative to the hole) to another contact point. A new observation $T_{p_2}^w$ is recorded. The new peg position is shown in Figure 4.12. The peg move $T_{p_2}^{p_1}$ can be computed from the two observations, $T_{p_1}^w$ and $T_{p_2}^w$. As the peg position has changed, the belief set, $\mathcal{B}_1$, estimating the current peg position, is updated by translating each point in the set by the peg move. The updated belief set, $\mathcal{B}_1 T_{p_2}^{p_1}$, is shown in Figure 4.12. The new recorded height corresponds to another slice through the contact surface map, as shown in Figure 4.13, yielding a second locus of points. The new set of points represents the position estimates based on the new measurement. We call this set $\overline{\mathcal{B}}_2$, as shown in Figure 4.14. Among the position estimates in $\overline{\mathcal{B}}_2$, only the ones that are consistent with the updated

$\mathcal{B}_1$ are valid. This can be checked as follows: for $T_b \in \mathcal{B}_1$, if $T_b T_{p_2}^{p_1} \in \overline{\mathcal{B}_2}$, then $T_b T_{p_2}^{p_1}$ is a valid candidate for the current peg-position w.r.t. the hole, and it is included in the new belief set $\mathcal{B}_2$. This is shown in Figure 4.14. Because the sets $\mathcal{B}_1$ and $\overline{\mathcal{B}_2}$ are in analytical form, the new belief set, $\mathcal{B}_2$, can be efficiently computed by translating $\mathcal{B}_1$ by $T_{p_2}^{p_1}$, and intersecting it with $\overline{\mathcal{B}_2}$. Although $\mathcal{B}_2$ *has* to include the actual relative peg-position, it can include other candidates that would yield the observed contact heights. As shown in Figure 4.14, the $\mathcal{B}_1$ and $\overline{\mathcal{B}_2}$ loci intersect in two points, either of which can be the actual peg-position w.r.t. the hole. Over successive iterations of peg moves and observation matches, the spurious candidates are pruned out, and the actual relative peg-position remains, as shown in Figure 4.16.

Figure 4.10: Top view of the (x, y) -plane: the tilted peg contacting the hole. The actual position of the hole in the (x, y) -plane is not known. (It is only shown here for the purpose of illustration.) From the measured contact z -height, an (x, y) -slice of the contact surface map (as shown in Figure 4.11) forms the first belief set, $\mathcal{B}_1$. $\mathcal{B}_1$ represents all the possible peg-positions w.r.t. the hole. The actual—and, as yet, unknown—peg position is marked with a red/medium-gray circle.

Figure 4.11: An (x, y) -slice of the contact surface map at the measured contact z -height.

Figure 4.12: New peg position, $T_{p_2}^w$, after the first peg move. The belief set, $\mathcal{B}_1$, is also updated by translating each point in $\mathcal{B}_1$ by the peg move.

Figure 4.13: Another (x, y) -slice of the contact surface map at the measured contact z -height.

Figure 4.15: Third (x, y) -slice of the contact surface map at the measured contact z -height.

Figure 4.16: The actual relative position of the peg w.r.t. the hole is successfully localized after the third iteration.

4.2.2 Localization for Square Peg-in-Hole Assembly

We have simplified the general square peg-in-hole, which is a six-dimensional uncertainty problem, by assuming that the orientation of the hole-surface about any axis in the horizontal plane and its vertical position are known. This assumption is not unjustified, because, under compliant control, the peg can be made to lie flat on the hole surface, thus revealing the orientation of the hole about the horizontal axes. In fact, we assume that the peg and hole have matching initial orientations. As the uncertainty in the relative peg-hole configuration is limited to a bounded region in the (x, y, θ_z) space, the six-dimensional search space of coordinate transformations from $\mathcal{F}_p$ to $\mathcal{F}_h$ is reduced to three dimensions: (x, y, θ_z) .

As shown before, the contact hyper-surface for matching peg-hole orientations is flat and monotonous, while, for mismatched orientations, we have a constantly varying and informative surface (see Section 4.1.2). To take advantage of this fact, we purposely tilt the peg by a known amount w.r.t. the hole. For the experiments here, the peg-tilt was fixed at 10° about a diagonal aligned with the x -axis, relative to the hole. With fixed relative orientations (θ_x, θ_y) , the six-dimensional contact hyper-surface collapses to four-dimensions: (x, y, z, θ_z) .

For this simplified problem, the localization in the (x, y, θ_z) space can be carried out as follows. The tilted peg is moved down along the hole z -axis to its first contact with the hole, and the peg-configuration T_{p1}^w is recorded in the robot coordinate frame. Here, T_{p1}^w is an observation in (x, y, z, θ_z) . As the vertical placement of the hole-surface is known to the controller, the recorded height, z , can be directly matched to the map. This corresponds to an (x, y, θ_z) -slice of the contact hyper-surface map, at the recorded z -height. The points thus obtained give all the peg-configurations (x, y, θ_z) relative to the hole that would result in the recorded height on peg-hole contact, as shown

in Figure 4.17(b). Figure 4.17(a) shows the peg-hole contact configuration.

Figure 4.17: First observation matched to the map: a slice through the four-dimensional map of the contact hyper-surface at the observed z -height produces a three-dimensional surface in x, y, θ_z . Each configuration here represents a possible current peg configuration w.r.t. the hole.

Corresponding to each relative θ_z possible between the peg and hole, there is a locus of relative (x, y) positions that would yield the recorded contact height. The (x, y) positions for the different θ_z -values are stacked along the vertical to produce the surface seen in Figure 4.17(b). The actual peg configuration is included on this surface (shown circled). Hence, we obtain our first belief set, $\mathcal{B}_1$, of (x, y, θ_z) transformations from $\mathcal{F}_p$ to $\mathcal{F}_h$. Now, the peg is translated and rotated (preserving its orientation relative to the hole about the (θ_x, θ_y) -axes) to another contact configuration. The new observation $T_{p_2}^w$ is recorded. The new recorded height corresponds to another slice through the hyper-surface map, yielding a second set of points. The new set of points represents the configuration estimates based only on the new measurement. We call this set $\overline{\mathcal{B}_2}$, as shown in Figure 4.18(b). The peg move $T_{p_2}^{p_1}$ can be computed from the two observations, $T_{p_1}^w$ and $T_{p_2}^w$. Among the configuration estimates in $\overline{\mathcal{B}_2}$, only the ones that are consistent with $\mathcal{B}_1$ are valid. This can be checked as follows: for $T_b \in \mathcal{B}_1$, if $T_b T_{p_2}^{p_1} \in \overline{\mathcal{B}_2}$, then $T_b T_{p_2}^{p_1}$ is a valid candidate for the

(a) Second peg-hole contact configuration

(b) Second (x, y, θ_z) -slice of the contact hypersurface yields the set $\overline{\mathcal{B}}_2$

Figure 4.18: The second measurement produces a new set of configurations, $\overline{\mathcal{B}}_2$.

current peg-configuration w.r.t. the hole, and it is included in the new belief set $\mathcal{B}_2$. Since the sets $\mathcal{B}_1$ and $\overline{\mathcal{B}}_2$ are in analytical form, the new belief set, $\mathcal{B}_2$, can be efficiently computed by transforming $\mathcal{B}_1$ by $T_{p_2}^{p_1}$, and intersecting it with $\overline{\mathcal{B}}_2$. Although $\mathcal{B}_2$ has to include the actual relative peg-configuration, it can include other candidates that would yield the observed contact heights. This can be seen in Figure 4.19; the $\mathcal{B}_1$ and $\overline{\mathcal{B}}_2$ loci will intersect in many points, any of which can be the actual peg-configuration w.r.t. the hole. Over repeated observing-matching cycles, the spurious candidates get pruned out, and the actual relative peg-configuration remains.

Figure 4.19: The possible relative peg-hole configurations recorded after the first probe move, $\mathcal{B}_1$, are updated by moving them according to the peg-move in (x, y, θ_z) . Then, the updated set is intersected with the new slice, $\overline{\mathcal{B}_2}$, to produce the new belief set, $\mathcal{B}_2$. The updated $\mathcal{B}_1$ and the new slice, $\overline{\mathcal{B}_2}$ are shown here.

4.3 Conclusions

We implemented the localization strategy described above on computer simulations of the circular and square peg-in-hole assemblies using analytical maps. For the circular peg-in-hole, localization was always successful and was generally achieved within three observation-matching cycles. We bounded the hole-uncertainty region to be slightly smaller in size than the hole, i.e., the radius of the hole-uncertainty circle is smaller than the radius of the hole. This ensures that when the peg is at the center of the hole-uncertainty region, peg-hole contact will be on the hole-edge and not on the hole-surface.

Localization for the square peg-in-hole assembly was also always successful, although it took more observation-matching cycles to resolve the three-dimensional uncertainty. For the square peg-in-hole, the hole-uncertainty region in (x, y) was set equal to the circle inscribing the hole (of diameter equal to the length of the hole-edge). In orientation, the hole-uncertainty bounds were set to $[0, \frac{\pi}{2})$, which covers the entire uncertainty range given the $\frac{\pi}{2}$ -rotational symmetry of the square peg-in-hole.

In general, for an arbitrary peg-in-hole assembly, we can carry out localization using analytical maps as described in Section 4.2. The analytical map can be generated by creating equations for volumetric intersections of the peg and hole. The map can then be used to implement the localization algorithm described above. Once the relative peg-hole configuration has been localized, assembly becomes easy.

Chapter 5

Assembly Localization Using Sampled Maps

In the previous chapter, we described localization using analytical maps. But, generating an analytical map for the peg-hole contact hyper-surface can require painstaking effort to produce equations of three-dimensional volumetric intersections. Moreover, for an actual assembly with its many possible imperfections, its contact hyper-surface might not be represented accurately by a *perfect* analytical map. Hence, our goal is to localize using sampled maps of the contact hyper-surface, which can be generated with minimum operator intervention. However, a sampled map does not provide a complete representation of the continuous contact hyper-surface. To overcome this problem, we introduce the concept of **assembly sufficiency** and show how sufficient assembly can be guaranteed through *approximate* localization using a sampled map. We also introduce the concept of a **goal region** based on assembly clearance, which helps develop sampling conditions to fulfill the above guarantee.

We start with a description of the sampling process to generate the contact hyper-surface map. Later, we describe the use of the sampled map to localize peg-hole misalignment for a general peg-in-hole assembly. Finally,

we describe results from simulated localization experiments on circular and square peg-in-hole assemblies using sampled maps.

5.1 Sampled Contact Hyper-Surface Map

As mentioned before, a sampled contact hyper-surface map can be easily generated either using a CAD model of the assembly parts, or using a robot to explore the peg-hole contact C-space and sample points. The map can be stored as a tabular collection of (six-dimensional) configurations. Although with a sampled map, we need to make a significant change in our problem formulation as described below.

According to our original formulation, the goal was to find the transformation from the peg coordinate frame, $\mathcal{F}_p$, to the hole coordinate frame, $\mathcal{F}_h$. In accordance with this goal, as described in the previous chapter, we mapped the space of peg-configurations relative to the hole, and we matched observations to this map to compute the actual (unknown) transformation, T_p^h . With a perfect analytical map and perfect matching, the above formulation was successful. But, with a sampled map, discretization errors are introduced and a change in our representation is required. This will be explained in more detail in Section 5.6. Of immediate concern is the required change in mapping. Instead of mapping peg-configurations relative to the hole, T_p^h , we map hole-configurations relative to the peg, T_h^p .

While the peg-hole contact hyper-surface is six-dimensional, it only has five degrees-of-freedom, meaning we can only choose independent values in five dimensions—the sixth is dependent on the five values chosen. For our map we let the height along the z -axis be the dependent dimension. That means we have to sample every combination of values in the five dimensions of $(x, y, \theta_x, \theta_y, \theta_z)$. We call this five-dimensional space the **free space**, and the dependent dimension, the **contact axis**, since its value is determined

Figure 5.1: Simplified illustration of the sampling of the peg-hole contact hyper-surface map for a circular peg-in-hole assembly. Sampled points in the (x, y) -space are shown. The hole is fixed at T_h^w w.r.t. the robot coordinate frame. Hence, the robot moves the peg to configuration T_p^w w.r.t. the robot coordinate frame and brings it into contact with the hole to sample the relative peg-hole configuration T_h^p .

by contact between the peg and the hole. This five-dimensional subspace can be discretized into a grid of points determined by the sampling criterion described in Section 5.3. Each point here represents a hole-configuration in five dimensions w.r.t. a fixed peg coordinate frame, T_h^p . However, since the hole is fixed and the peg is mobile, we have to move the peg to realize each relative configuration in the map. Obtaining the hole-configuration, T_h^p , w.r.t. the peg is equivalent to *posing* the peg in configuration $T_p^h = (T_h^p)^{-1}$ w.r.t. the hole. This is shown in Figure 5.1.

The sampling of the contact hyper-surface can be automated as described in Algorithm 5.1 (see Figure 5.2). Thus, we obtain contact hyper-points representing hole configurations in the peg coordinate frame. This information is tabulated, and represents a map of the relative peg-hole configuration space.

Algorithm 5.1

- Step 0: • Set $k = 0$.
- The operator initializes the robot with the peg inside the hole and their coordinate frames aligned. Hence, $T_h^w = T_p^w$ can be noted by the controller.
- Step 1: • Update $k = k + 1$.
- The peg is moved to the k th-configuration, $(T_h^p)_k$, on the sampling set.
- Step 2: • The peg is moved along its z -axis until it contacts the hole, and T_p^w is recorded. The hole-configuration w.r.t. the peg coordinate frame is computed as $T_h^p = T_w^p T_h^w$.
- Step 3: • If entire map sampled, Stop; else go to Step 1.

Figure 5.2: Algorithm for sampling the contact hyper-surface map.

5.2 Assembly Sufficiency, Goal Region, and Approximate Localization

In Chapter 4, we described how the controller, given a continuous (analytical) map of the contact hyper-surface, can generate a set of (all possible) transformations corresponding to hole configurations relative to the peg by matching peg moves to the map. This set was guaranteed to contain the actual transformation. But with a sampled map, it is highly improbable that the actual relative peg-hole configuration is contained in the map. Hence, exact localization with a sampled map is not possible. Below, we introduce the concept of “assembly sufficiency,” and show how approximate localization can lead to sufficient assembly.

Generally, a peg-in-hole assembly is considered complete when the peg is inserted all the way into the hole and comes in contact with the hole bottom.

But, during the assembly process there may come a stage when the assembly can be considered sufficiently progressed, so that assembly completion becomes a trivial process. We formalize this stage under the **assembly sufficiency condition**. For our problem, we consider the assembly sufficiency condition satisfied, and the peg sufficiently assembled into the hole, if the entire lower rim of the peg is contained within the hole. This is justified, because, beyond this stage, compliant control can take over and insert the peg further into the hole. For instance, among known methods, the Remote Center of Compliance wrist [49], described in Section 1.3, can automatically correct for small misalignments in position and orientation between the peg and hole.

To facilitate the illustration of the concepts introduced here, we use a simple planar peg-in-hole assembly example. A drawing of the planar assembly is shown in Figure 5.3(a). There is bounded uncertainty in the configuration of the hole: position (x, y) and orientation (θ_z) . Thus, it is a three-dimensional uncertainty problem. We assume accurate knowledge of the peg configuration at all times, as well as the capability to accurately alter its configuration.¹ The problem is to localize the hole configuration to facilitate assembly of the peg into the hole. Figure 5.3(b) shows an example configuration of the peg and hole where the assembly sufficiency condition is satisfied.

With the assembly sufficiency condition defined, we can show that exact localization is not required—that it is enough to localize approximately. If the estimated hole configuration is “close enough” to the actual hole configuration, then the assembly sufficiency condition will be satisfied. For example, for a circular peg-in-hole assembly with two-dimensional uncertainty in the hole position in the (x, y) -plane, if the estimated hole configuration is within assembly clearance distance of the actual hole configuration, then assembly

¹However, this assumption is not required for localization using particle filtering, described in Chapter 7.

Figure 5.3: Planar assembly example with three-dimensional uncertainty: There is uncertainty in the initial configuration of the hole, but the hole is assumed to be stationary for the duration of the assembly. In (b), an example of assembly sufficiency is shown. The entire peg bottom edge is contained inside the hole.

is possible. Here, we extend this two-dimensional assembly clearance concept to higher dimensions, and call it the “goal region.”

As mentioned above, the concept of the goal region is similar to that of assembly clearance—the difference being that while the latter is usually restricted to the translational clearance (in the (x, y) -plane) of the assembly, the goal region also takes into consideration the orientation clearances. This will be dealt with further later in this section. The **goal region** can be defined as the neighborhood of configurations around and w.r.t. the estimated hole configuration such that, if the actual configuration belongs to the goal region, then assembly will be successful. We denote this goal region as $\mathcal{T}$, because it represents a *set* of transformations. Two important clarifications need to be made here. First, the goal region is defined w.r.t. the free space of the (estimated) hole coordinate frame. Second, the goal region is unbounded along the contact axis of the hole coordinate frame, i.e., the actual hole position can be anywhere along the contact-axis of the esti-

mated hole configuration as long as it is within the goal region in the free space. When the peg is aligned with the estimated hole configuration, the free space and contact axis of the peg coincide with those of the estimated hole configuration. For the planar assembly example, (x, θ_z) forms the free space and y is the contact axis. Figure 5.4(a) shows a slice of the goal region for the planar assembly example for an estimated hole configuration at $(0,0)$ in (x, θ_z) . As noted, the actual hole position can be anywhere along the y -axis of the estimated configuration as long as it is within the goal region in (x, θ_z) . Thus, the goal region is three-dimensional: bounded in (x, θ_z) , as shown in Figure 5.4(a), and unbounded along the contact axis, y . Let us look at an example of how the goal region provides assembly sufficiency. The highlighted point within the goal region shown in Figure 5.4(a), corresponds to an actual hole configuration of $(-0.7, 0, -0.25)$ in (x, y, θ_z) w.r.t. the estimated configuration. Figure 5.4(b) shows the final condition when the peg is aligned with the estimated configuration and advanced in the insertion direction until contact. We can see that the assembly sufficiency condition has been satisfied. This would hold true for every point within the goal region. Note that the goal region is defined w.r.t. the hole coordinate frame. Hence, the goal region corresponding to a peg-hole configuration would be transformed accordingly as $T_h^p \mathcal{T}$. This will be explained in more detail in the next section.

The goal region discretizes the transformation search space much the same way as the assembly clearance was shown to discretize the assembly search space in Section 3.1.2. If we were searching through the transformation space exhaustively by checking each relative peg-hole configuration, then for each configuration we check, we would have essentially checked all the configurations within its goal region. Thus, we can cover the entire transformation space by checking a finite number of configurations. This helps us formulate our sampling condition below.

(a) Actual configuration within the goal region of the estimate

(b) Peg aligned with estimated configuration leads to assembly sufficiency

Figure 5.4: The peg is aligned with the estimated configuration and lowered along the contact-axis into contact with the hole. As the actual hole-configuration is contained within the goal region of the estimated hole-configuration, when the peg contacts the hole assembly sufficiency is guaranteed to be satisfied.

5.3 Sampling Conditions

With a sampled map of the contact hyper-surface, we cannot guarantee that the actual relative peg-hole configuration will be contained in the map. But, as mentioned above, even if we localize the peg-hole configuration approximately, we can fulfill the assembly sufficiency condition. So our goal is to track the actual configuration approximately with mapped configurations that contain the actual configuration within their respective goal regions. That is, as the actual configuration moves over the contact hyper-surface (corresponding to the peg moves), we ensure that the belief set contains at least one mapped configuration that contains the actual configuration within its goal region. For this, we have to first ensure that for every relative peg-hole configuration possible, there is at least one mapped configuration that contains it within its goal region.

Sampling Condition 5.1 *For every peg-hole configuration, T_h^p , on the contact hyper-surface, there should be at least one configuration, $(T_h^p)_m$, on the*

map such that T_h^p is within the goal region of $(T_h^p)_m$, i.e., $T_h^p \in (T_h^p)_m \mathcal{T}$.

Sampling Condition 5.1 takes care of limiting the discretization error in the free space $(x, y, \theta_x, \theta_y, \theta_z)$ ((x, θ_z) for the planar assembly example). But, to match the peg moves to the surface in all the dimensions, we also need a bound in the height difference along the contact axis between any configuration on the surface and a closely mapped configuration. For the planar assembly example the contact axis is y , so we need to bound the height difference along the y -axis. We define Sampling Condition 5.2 as an extension of Sampling Condition 5.1, to enforce a chosen y -bound, y_b .

Sampling Condition 5.2 *For every peg-hole configuration, T_h^p , on the contact hyper-surface, there should be at least one configuration, $(T_h^p)_m$, on the map such that (from Sampling Condition 1) $T_h^p \in (T_h^p)_m \mathcal{T}$, and also (Sampling Condition 2) T_h^p is within y_b -distance of $(T_h^p)_m$.*

The two sampling conditions are defined separately, because the mapping is carried out in more than one phase as explained below, and the conditions are applied in different phases. While Sampling Condition 5.1 is based on the goal region, and is enforced during the initial mapping, Sampling Condition 5.2 requires some knowledge of the contact hyper-surface to determine y_b , which can only be obtained after one phase of mapping, and so is enforced after the first mapping phase. The two conditions can be combined and enforced as one in the later mapping phases, as shown below. The bound y_b is selected according to the contact surface characteristics. Too large a value for y_b would fail to represent the true surface characteristics. On the other hand, too small a value would result in a large number of mapped configurations increasing the sampling time and the storage space required for the map. Figure 5.5 shows a hypothetical mapped configuration $(4, 5, \frac{\pi}{10})$ in (x, y, θ_z) . For ease of depiction, the goal region, $\mathcal{T}$, has been shown as a rectangular prism. The actual shape of the goal region in (x, θ_z) is as shown

in Figure 5.4(a). As can be seen, $\mathcal{T}$ is unbounded along the y -axis of the configuration. Also shown is the enforced y -bound, y_b . The region enclosed by $\mathcal{T}$ and y_b gives the goal region volume referred to by the combined sampling condition, Sampling Condition 5.3. Actually, for sampling we use a smaller subset of this volume that we call the **sampling tolerance**, $\mathcal{T}_s$, also shown in Figure 5.5. Using sampling tolerance instead of the goal region, the combined sampling condition can be stated as:

Sampling Condition 5.3 *For every peg-hole configuration, T_h^p , on the contact hyper-surface, there should be at least one configuration, $(T_h^p)_m$, on the map such that $T_h^p \in (T_h^p)_m \mathcal{T}_s$, where $(T_h^p)_m \mathcal{T}_s$ is the sampling tolerance region of configuration $(T_h^p)_m$.*

Note that $\mathcal{T}_s$ is obtained by reducing $\mathcal{T}$ in the free space and then keeping the portion enclosed by the bound along the contact axis (y_b for the planar example). As the sampling tolerance is a subset of the goal region, it satisfies the assembly sufficiency condition. One reason for using a sampling tolerance that is a subset of the goal region is that the goal region might not have a regular shape, making it difficult to use it to discretize the search space. Also, an irregular shape would make the matching process more difficult, especially the computation of inclusions and intersections. Moreover, as the sampling tolerance is reduced, the sampled surface approximates the actual surface more closely, and leads to faster localization as explained in Section 5.5. In an ideal scenario, the sampling tolerance would be infinitely small, producing a continuous map, and allowing exact matching. However, this would also mean an infinite number of configurations on the map. A larger map takes longer to sample, and also requires more storage space in computer memory. Hence, the sampling tolerance and the contact axis bound are selected to find the right balance between a large map and a good surface approximation.

Figure 5.5: A mapped configuration and its corresponding goal region. The y -bound and sampling tolerance are also shown. The mapped configuration in (x, y, θ_z) signifies the position and orientation of the hole w.r.t. the peg. The goal region is defined w.r.t. the hole configuration, and so is oriented w.r.t. the peg the same way as the hole. The goal region is unbounded along the contact-axis (y), but is shown truncated.

5.4 Sampling the Contact C-Space Map

In this section, we discuss the issues concerning the sampling of the map to fulfill the sampling conditions laid out above. The extent of the contact hyper-surface to be mapped is decided by the relative peg-hole configurations possible, and was shown to be greater than the hole uncertainty region (see Section 4.1). The region to be mapped is the hole uncertainty region augmented by the region of peg motion. This is shown in Figure 5.6 for the planar assembly example. The sequential procedure of the various mapping phases is outlined in Algorithm 5.2 in Figure 5.7, and is followed by a detailed explanation.

The free space region can be discretized according to Sampling Condition 5.1 using sampling tolerance equal to the goal region. After the first

Figure 5.6: The region to be mapped is given by the hole-configuration uncertainty region augmented by the peg motion region. This ensures that in the condition when the hole is at the edge of its uncertainty region, and the peg moves in the opposite direction of the hole-configuration, the relative peg-hole configuration is still covered by the map.

phase of sampling, Sampling Condition 5.1 is satisfied, however, there is no guarantee that the contact hyper-surface has been adequately mapped to fulfill the combined condition, Sampling Condition 5.3. There is also no infallible method to check for condition fulfillment. We suggest a couple of methods here that are easy to implement. We employ the first method (Step 1) after the first phase of sampling. It involves comparing each mapped configuration with all of its immediate neighbors to check if the difference in their contact axis “height” is greater than $2 \times y_b$. If this is true, then some configuration in the interval between the chosen configurations might not be within y_b of either, and Sampling Condition 5.2 would be violated. Again, there are two options for tackling the violation of this sampling condition. If there are numerous violations throughout the map, which is usually the case after the first sampling phase, we shrink the sampling tolerance in the free space (maintain y_b), and resample the entire surface with the new sampling tolerance (Step 2). The logic behind this strategy is that if we have to sample a large number of extra configurations, we might as well reap the benefit of

Algorithm 5.2

- Step 0: • First sampling phase: sample configurations obtained by setting the sampling tolerance equal to the goal region.
- Step 1: • Check the resulting map for sampling condition violations by comparing the contact-axis height of each mapped configuration with its immediate neighbors. If the number of violations is more than the set threshold, goto Step 2; else goto Step 3.
- Step 2: • Shrink the sampling tolerance and resample. Goto Step 1.
- Step 3: • Resample locally around the configurations where the sampling condition has failed. Goto Step 4.
- Step 4: • Check the resulting map for sampling condition violations by randomly selecting configurations on the actual contact hyper-surface. If any violations goto Step 3, else Stop.

Figure 5.7: Algorithm enumerating the various phases of the mapping cycle.

a smaller sampling tolerance. The second option, employed when there are only a handful of violations, is to resample locally around the configurations where the sampling condition fails (Step 3). Usually, after the second sampling phase, the resulting map may still not satisfy Sampling Condition 5.3 sufficiently, and local resampling will still have to be carried out. The second method, which we employ as a check after the final mapping phase, involves picking configurations on the actual contact hyper-surface at random and checking if there are mapped configurations that satisfy the sampling conditions for the selected configuration. If not, then we can resample locally around the selected configuration.

One important fact to note here is that we can never be certain that

Figure 5.8: Hypothetical example showing how an irregularity can violate the sampling conditions. There can always be an “irregularity” between two mapped configurations, both of which satisfy the Sampling Condition 5.3.

the final map fully satisfies Sampling Condition 5.2. There can always be an “irregularity” between two mapped configurations, both of which satisfy Sampling Condition 5.3. A hypothetical example is shown in Figure 5.8. However, for most regular peg-in-hole assemblies, such irregularities are absent, and we can safely assume that the Sampling Condition 5.3 has been satisfied.

5.5 Localization Using Sampled Maps

Using a sampled map, M , the localization strategy is to track the actual peg-hole configuration approximately. If localization terminates successfully, then the estimated peg-hole configuration should contain the actual within its sampling tolerance, $\mathcal{T}_s$, thus satisfying the assembly sufficiency condition. The algorithm to implement the general localization strategy was presented in Algorithm 3.1 (see Figure 3.16). The implementation of the algorithm was described using analytical maps in Chapter 4 (see Section 4.2). Because the algorithm used analytical maps, it was able to employ exact matching, with-

out any error bounds or approximations. However, as discussed above, sampled maps do not provide a complete representation of the peg-hole contact hyper-surface. Hence, to match a configuration to a sampled map, approximate matching with error bounds has to be employed. The error bounds required for matching were defined in the sampling conditions above as the sampling tolerance, $\mathcal{T}_s$. An improvised algorithm to implement the general localization strategy using sampled maps is presented in Algorithm 5.3 (Figure 5.9).

In Algorithm 5.3, the belief set, $\mathcal{B}_0$, is initialized to include all the mapped configurations corresponding to the hole-uncertainty region, given by U . Thus, $U \subseteq M$, and $\mathcal{B}_0 = U$. The sampled map is sorted and indexed so that when matching a configuration, T_a , to the map, the entire map does not have to be considered. This indexing facilitates quick selection of a small neighborhood of configurations on the map surrounding T_a for matching.

Each iteration involves making an observation, matching it to the map, and updating the belief set. After the first peg move, we have the observations $T_{p_0}^w$ and $T_{p_1}^w$, which give the peg-configurations w.r.t. some robot coordinate frame. We can compute $T_{p_0}^{p_1} = (T_{p_1}^w)^{-1} T_{p_0}^w$. Let the actual—but unknown—peg-hole configurations corresponding to these observations be $T_h^{p_0}$ and $T_h^{p_1}$. Hence,

$$T_h^{p_1} = T_{p_0}^{p_1} T_h^{p_0} \quad (5.1)$$

From Sampling Condition 5.3 we know that corresponding to the actual configurations, $T_h^{p_0}$ and $T_h^{p_1}$, there are mapped configurations, say $(T_h^{p_0})_m$ and $(T_h^{p_1})_m$, respectively, that contain the actual configurations within their respective sampling tolerances. That is,

$$T_h^{p_0} \in (T_h^{p_0})_m \mathcal{T}_s, \quad (5.2)$$

$$T_h^{p_1} \in (T_h^{p_1})_m \mathcal{T}_s. \quad (5.3)$$

Algorithm 5.3

Step 0: • Set $k = 0$

- Initialize belief set, $\mathcal{B}_0 = U$
- Record observation: $T_{p_0}^w$

Step 1: • Update $k = k + 1$

- Record observation: $T_{p_k}^w$
- Compute $T_{p_{k-1}}^{p_k}$
- For each configuration in the belief set: $b_{k-1} \in \mathcal{B}_{k-1}$,
 - Compute $T_{p_{k-1}}^{p_k} T_{b_{k-1}}$, and find the corresponding (indexed) neighborhood of mapped configurations, $\mathcal{A}$
 - For each configuration $a \in \mathcal{A}$,
 - if $T_a \mathcal{T}_s \cap T_{p_{k-1}}^{p_k} T_{b_{k-1}} \mathcal{T}_s \neq \emptyset$, then include $a \in \mathcal{B}_k$
- Delete any duplicates from the new belief set, $\mathcal{B}_k$
- For $i = 0$ to $k - 2$,
 - Compute $T_{p_i}^{p_k}$
 - For each configuration $b_k \in \mathcal{B}_k$,
 - if there is any configuration $b_i \in \mathcal{B}_i$, such that $T_{p_i}^{p_k} T_{b_i} \mathcal{T}_s \cap T_{b_k} \mathcal{T}_s \neq \emptyset$, then include $b_k \in \mathcal{B}'_k$
 - $\mathcal{B}_k = \mathcal{B}'_k$; $\mathcal{B}'_k = \emptyset$

Step 2: • Check if $\mathcal{B}_k$ is satisfactory $\Rightarrow$ goto Step 3

- Else check if too many iterations $\Rightarrow$ goto Step 4
- Else goto Step 1

Step 3: • (Localization successful) Attempt assembly

Step 4: • (Localization terminated) Attempt assembly

Figure 5.9: Algorithm for general localization using sampled maps.

Since $\mathcal{B}_0$ is initialized to include all the mapped configurations corresponding to the hole-uncertainty region, U , $(T_h^{p_0})_m \in \mathcal{B}_0$. In matching the peg move to the map, we want to ensure that the mapped configuration, $(T_h^{p_1})_m$, corresponding to the current peg-hole configuration, $T_h^{p_1}$, is included in the new belief set, $\mathcal{B}_1$. Each configuration, $b_0 \in \mathcal{B}_0$, is advanced by the peg move to obtain $T_{p_0}^{p_1} T_{b_0}$, and matched to the map. To match the configuration to the map, the indexed neighborhood of configurations, say $\mathcal{A}$, surrounding $T_{p_0}^{p_1} T_{b_0}$ is selected for matching. All the configurations, $a \in \mathcal{A}$, that satisfy the requirement, $T_{p_0}^{p_1} T_{b_0} \mathcal{T}_s \cap T_a \mathcal{T}_s \neq \emptyset$ are selected to the new belief set, $\mathcal{B}_1$. Let us check whether $(T_h^{p_1})_m$ will be included in the new belief set. Pre-multiplying both sides of Equation (5.2) with $T_{p_0}^{p_1}$ gives

$$T_{p_0}^{p_1} T_h^{p_0} \in T_{p_0}^{p_1} (T_h^{p_0})_m \mathcal{T}_s. \quad (5.4)$$

From Equations (5.1) and (5.4) we have

$$T_h^{p_1} \in T_{p_0}^{p_1} (T_h^{p_0})_m \mathcal{T}_s. \quad (5.5)$$

From Equations (5.3) and (5.5) we have

$$T_{p_0}^{p_1} (T_h^{p_0})_m \mathcal{T}_s \cap (T_h^{p_1})_m \mathcal{T}_s \neq \emptyset. \quad (5.6)$$

Thus, $(T_h^{p_1})_m$ is included in $\mathcal{B}_1$. See Figure 5.10.

5.5.1 Matching Observations to the C-Space Map

This section describes the matching process of Algorithm 5.3 in more detail. In Figure 5.10, the actual peg-hole configurations before and after the first peg move (translation and rotation), $T_{p_1}^{p_0}$, are labelled as $T_h^{p_0}$ and $T_h^{p_1}$, respectively. Both actual configurations fall within the sampling tolerance of mapped configurations, $(T_h^{p_0})_m$ and $(T_h^{p_1})_m$ as shown. This guarantees that the actual configuration $T_h^{p_1}$ falls within the sampling tolerance of $T_{p_0}^{p_1} (T_h^{p_0})_m$.

Figure 5.10: First peg move matched to the map: The actual relative peg-hole configurations before and after the first move, $T_h^{p_0}$ and $T_h^{p_1}$, respectively, are shown. Because the map fulfills the sampling conditions, we know that there will be mapped configurations, $(T_h^{p_0})_m$ and $(T_h^{p_1})_m$, such that $T_h^{p_0}$ and $T_h^{p_1}$ are contained within their respective sampling tolerances. But, $T_h^{p_1}$ will also be contained within the sampling tolerance of $T_{p_0}^{p_1} (T_h^{p_0})_m$. Hence, the intersection of the sampling tolerance regions of $T_{p_0}^{p_1} (T_h^{p_0})_m$ and $(T_h^{p_1})_m$ will contain $T_h^{p_1}$, and will not be an empty set. Therefore, $(T_h^{p_1})_m$ will be included in the new belief set.

Pruning Duplicates from the Belief Set (Step 1, part 5). Since in this algorithm, adapted for sampled maps, matching is carried out approximately using error bounds, it is possible that a configuration is selected to the belief set more than once. Hence, after a new belief set, $\mathcal{B}_k$, is formed, duplicate configurations are pruned out as shown in the algorithm.

Additional Matches (Step 1, part 6). After the second move, let $T_h^{p_2}$ be the actual peg-hole configuration and $(T_h^{p_2})_m$ the mapped configuration such that $T_h^{p_2} \in (T_h^{p_2})_m \mathcal{T}_s$. The new peg move to match to the map is $T_{p_1}^{p_2}$. Additionally, we match the combined peg move $T_{p_0}^{p_2} = T_{p_1}^{p_2} T_{p_0}^{p_1}$ to the map as well. With an analytical map this additional match would be redundant, because the individual move matches are perfect. But, with a sampled map, errant candidates accumulated in the individual matches can be pruned out with the additional match. Figures 5.11 and 5.12 help explain this better.

Figure 5.11: Hypothetical example illustrating the benefit of additional matches: (a) Peg moves, α and β , (b) Matching move α to the mapped points.

Consider the hypothetical example shown in Figure 5.11(a). The actual configuration moves from A to B to C ; we label the moves (transformations) as α and β , respectively. Also shown in each figure are the mapped configurations. One such configuration, A_m , contains configuration A within its circular sampling goal region, $T_{A_m} \mathcal{T}_s$, (shown as a shaded circle in Figure 5.11(a)). Let us assume that our initial belief set only contains one configuration, A_m , i.e., $\mathcal{B}_0 = A_m$. First, move α is matched to the map. Figure 5.11(b) shows the configurations, B_{m_i} , included in the new belief set, $\mathcal{B}_1$. $B_{m_i}, i = 1, \dots, 7$,

are the mapped configurations for which the following condition holds true: $T_{B_{m_i}} \mathcal{T}_s \cap T_\alpha T_{A_m} \mathcal{T}_s \neq \emptyset$. Clearly, most of these configurations do not contain the actual configuration within their sampling goal region.

Figure 5.12: Hypothetical example continued: (a) Matching move β to the map, (b) Matching the combined move α and β to the map.

Next, move β is matched to the map. Figure 5.12(a) shows the candidates, C_{m_i} , included in the new (but temporary) belief set, $\mathcal{B}'_2$. These are mapped configurations for which the following condition holds true (as shown in the figure): $T_{C_{m_i}} \mathcal{T}_s \cap T_\beta T_{B_{m_j}} \mathcal{T}_s \neq \emptyset$, for some B_{m_j} in the current belief set, $\mathcal{B}_1$. Next, the combined move $T_\beta T_\alpha$ is matched to the temporary belief set, $\mathcal{B}'_2$, which is pruned to obtain the new belief set, $\mathcal{B}_2$. Figure 5.12(b) shows the pruned belief set, $\mathcal{B}_2$. The number of candidates reduces from 18 in $\mathcal{B}'_2$ to 6 in $\mathcal{B}_2$. The remaining candidates are mapped configurations for which two conditions hold true. The first was checked by matching move β , and the second is checked by matching the combination of moves α and β : $T_{C_{m_i}} \mathcal{T}_s \cap T_\beta T_\alpha T_{A_m} \mathcal{T}_s \neq \emptyset$. Thus, with a sampled map, additional matches can prune out errant candidates accumulated through individual matches.

Termination Condition (Step 2). Every time a new move is matched to the map and the belief set is updated, we check whether the new belief set is sufficiently localized. The belief set is considered sufficiently localized if we can estimate a peg-hole configuration from it that is guaranteed to satisfy assembly sufficiency. To satisfy assembly sufficiency, an estimated configuration, T_{est} , has to contain the actual configuration, $(T_h^p)_{act}$, within its goal region, $\mathcal{T}$. That is, $(T_h^p)_{act} \in T_{est} \mathcal{T}$. (Note that $\mathcal{T}$ is the entire goal region and not the shrunk sampling tolerance, $\mathcal{T}_s$.) We know that for some $b_j \in \mathcal{B}$,

$$\begin{aligned} (T_h^p)_{act} &\in T_{b_j} \mathcal{T}_s \\ \Rightarrow (T_h^p)_{act} &\in \bigcup_{i=1}^n T_{b_i} \mathcal{T}_s, \end{aligned}$$

where $n = |\mathcal{B}|$ and $\bigcup_{i=1}^n b_i = \mathcal{B}$. If we can find an estimated configuration, T_{est} , such that

$$\bigcup_{i=1}^n T_{b_i} \mathcal{T}_s \subset T_{est} \mathcal{T},$$

then we are guaranteed that $(T_h^p)_{act} \in T_{est} \mathcal{T}$; and, if we attempt assembly with this estimated configuration, then the assembly sufficiency condition will be satisfied. Hence, at this point, localization is terminated and assembly is attempted.

The computation of T_{est} for a variety of assemblies is explained in detail in Section 5.7. Localization can also be terminated if more than a fixed number of iterations have been carried out without sufficient convergence of the belief set. (In which case, we compute the hole-configuration estimate from the centroid of the configurations remaining in the current belief set.) The reason behind this is that there is a drop in time-efficiency in localizing beyond a certain degree of accuracy [9]. Moreover, from an *assembly line* point of view, to accommodate critical cycle-time requirements [19], we need

to comply with a preset maximum localization time. Hence, after a pre-determined number of iterations we can switch from the localization strategy to either an exhaustive search strategy (as explained in Chapter 3), or a compliant form of assembly [2] that can handle the reduced uncertainty.

Now that the localization process using a sampled map has been explained, the reason behind the change in the problem formulation is clarified in the next section.

5.6 Change in Problem Formulation for Localization Using Sampled Maps

Our original problem formulation called for finding the transformation, T_h^p , from the peg coordinate frame, $\mathcal{F}_p$, to the hole coordinate frame, $\mathcal{F}_h$. That is, we sought to find the actual peg-configuration w.r.t. the hole. Although the actual hole configuration was not known, with an analytical map and perfect matching, we were able to always track $(T_p^h)_{act}$. But, with a sampled map, we resort to approximately tracking the actual relative peg-hole configuration. If we continue to use the original formulation, we would run into problems when we try to advance each candidate peg-configuration, $(T_p^h)_m \in \mathcal{B}$, with the known peg move. As the peg move is recorded in the peg coordinate frame, each candidate would be advanced according to its coordinate frame. Let $(T_p^h)_m$ be a mapped configuration such that, $(T_p^h)_{act} \in (T_p^h)_m \mathcal{T}_s$. If the orientations of $(T_p^h)_m$ and $(T_p^h)_{act}$ do not match exactly, then, after advancing $(T_p^h)_m$ by the peg move, we can no longer ensure that

$$(T_p^h)_{act} T_{p_{i+1}}^{p_i} \in (T_p^h)_m T_{p_{i+1}}^{p_i} \mathcal{T}_s. \quad (5.7)$$

This can be better explained with the following example.

Consider a square peg-in-hole assembly example. Suppose there is a small orientation error between $(T_p^h)_{act}$ and $(T_p^h)_m$, such that $(T_p^h)_{act} \in (T_p^h)_m \mathcal{T}_s$.

In Figure 5.13(a) we use the original problem formulation. The difference

Figure 5.13: Example explaining change in problem formulation.

between the actual and mapped peg-configurations is rotation θ about the vertical (z), i.e., $(T_p^h)_{act} = (T_p^h)_m T_{R(\theta)}$, where $T_{R(\theta)}$ is a homogeneous transformation matrix corresponding to the rotation matrix $R(\theta)$, and the position vector $\vec{0}$. After the peg has been moved to the new position shown, we need to update the mapped peg-configuration by the move. But, because of the existing orientation error, the update introduces a large translation error (proportional to the length of the move) between the updated actual and mapped configurations. That is, the actual configuration updates to $(T_p^h)_{act} T_{p_{i+1}}^{p_i} = (T_p^h)_m T_{R(\theta)} T_{p_{i+1}}^{p_i}$, while the mapped is updated to $(T_p^h)_m T_{p_{i+1}}^{p_i}$. Depending on θ and the length of the move, the translation error introduced can be large enough so that Equation 5.7 is no longer satisfied.

On the other hand, Figure 5.13(b) shows the new problem formulation, where we maintain the hole-configuration w.r.t. the peg coordinate frame, T_h^p . Since the original peg-configuration and the new peg move are accurately known in some robot coordinate frame, the update of the mapped hole configuration preserves the small orientation error and does not add any translation error. In this case, the actual configuration would update to

$T_{p_i}^{p_{i+1}} (T_h^p)_{act} = T_{p_i}^{p_{i+1}} (T_h^p)_m T_{R(\theta)}$, and the mapped configuration is updated to $T_{p_i}^{p_{i+1}} (T_h^p)_m$. Note that now Equation 5.7 is satisfied.

Hence, although the two formulations are equivalent for localization with analytical mapping, the new formulation is better suited for sampled maps, or maps with error.

5.7 Results: Localization using Sampled Maps

The localization strategy described above was tested in computer simulations on various assemblies, including the planar assembly example, and the circular and square peg-in-hole assemblies, described in Chapter 2. The results are presented below.

5.7.1 Planar Assembly Example

The localization strategy using a sampled map was implemented on a computer simulation of the planar assembly example. We conducted 1000 assembly trials with random initial uncertainty in hole-configuration within bounds. The peg moves were also selected randomly. Localization was terminated if the belief set was sufficiently localized, or the number of iterations exceeded a pre-determined limit. To decide whether the belief set is sufficiently localized we need to find an estimated configuration, T_{est} , such that, $\cup_{i=1}^n T_{b_i} \mathcal{T}_s \subset T_{est} \mathcal{T}$. Recall that $\mathcal{T}_s$ is the sampling tolerance, and $\mathcal{T}$ is the goal region. For the sake of computational ease, we implemented a more conservative approach as described below. We find the mean of all the configurations in $\mathcal{B}_k$, say T_m , and if the following condition holds true, $\cup_{i=1}^n T_{b_i} \mathcal{T}_s \subset T_m \mathcal{T}$, then we declare $\mathcal{B}_k$ sufficiently localized, and T_m becomes the estimated hole configuration. Now it is possible that even if the above condition does not hold true, $\mathcal{B}_k$ is sufficiently localized according to the former condition, and the chance to terminate localization will be missed.

If localization is terminated because the number of iterations has exceeded the pre-determined limit, then $\mathcal{B}_k$ is not sufficiently localized. However, an estimate of the relative peg-hole configuration is still computed from the mean of all the configurations in $\mathcal{B}_k$, and assembly is attempted using the computed estimate. Only if the estimated hole-configuration does not lead to the assembly sufficiency condition being fulfilled, do we consider the localization trial a failure.

Average localization error values were computed over 1000 assembly trials. Figure 5.14 shows the actual hole-configurations for the 1000 assemblies plotted w.r.t. the final estimated configuration. The gray points make up the goal region, $\mathcal{T}$. As can be seen, all the actual configurations are contained within the goal region. Hence, assembly was successful for all the trials. Actually, all the trials terminated successfully—well before the preset forced termination. The average number of iterations required for guaranteed localization was 8.36.

Figure 5.14: The estimated hole-configurations, w.r.t. the actual, from 1000 assembly trials are plotted. As all the estimates fall within the goal region of the actual, all the trials localized successfully.

Figures 5.15–5.18 show the peg-hole contacts and the computed belief set for all the iterations of a trial localization run.

Figure 5.15: First peg-hole contact.

Figure 5.16: Iterations 1 and 2: as the peg moves (translates/rotates) into contact with the stationary hole, the belief set “moves” on the map.

Figure 5.17: Iterations 3 and 4: The belief set shrinks till the termination condition is satisfied after the 4th iteration; relative peg-hole configuration is estimated from the belief set $\mathcal{B}_4$.

Figure 5.18: Assembly is attempted with the estimated hole-configuration, and is successful.

5.7.2 Circular Peg-in-Hole Assembly Simulations

Set Up

We simulated a restricted version of the Circular Peg-in-Hole II assembly problem (described in Section 2.3.5). The general circular peg-in-hole is a five-dimensional localization problem. However, we restricted the uncertainty in the hole configuration to the (x, y) -plane. That is, the orientation of the hole surface in the (θ_x, θ_y) -space and its vertical placement (z) are known to the controller. Hence, the contact hyper-surface map was sampled in the (x, y, z) dimensions.

This simulation was used to test the general localization strategy using sampled maps given in Algorithm 5.3 (see Figure 5.9). The hole and peg diameters chosen for the simulation were 100 mm and 99 mm respectively, which gave an assembly clearance radius of 0.5 mm. The initial uncertainty in the position of the hole ranged within a circle of radius 25 mm, making the uncertainty region 2500 times the goal region in area.

The contact-axis bound (along the vertical axis, z) was fixed at 0.1 mm. To fulfill Sampling Condition 5.3 for the given bounds, 979842 configurations had to be mapped.

Localization Termination Conditions

Localization was terminated if either of the following two conditions were met:

1. All the configurations in the belief set are contained within the goal region (same as the assembly clearance circle for the circular peg-in-hole).
2. The number of iterations has exceeded a pre-determined limit.

Hence, after every iteration, we need to determine whether all the configurations in the belief set can be contained within a circle equal to the assembly clearance. This is the same problem as that of finding the smallest circle enclosing a set of points in two dimensions, which has been solved. The solution we used was given by Chrystal in 1885 [12]. It involves finding the convex hull of the point set, and then iterating among the points on the convex hull to find the smallest enclosing circle. The maximum run-time for the main loop of the algorithm (iterating among the points on the convex hull) is proportional to the total number of points in the set plus the cube of the number of points on the convex hull. We chose this algorithm over other more recently proposed algorithms—the latest of which runs in linear time [32]—because of ease of implementation. If this convergence condition is satisfied, then localization is terminated, and the hole position estimate is set equal to the center of the enclosing circle.

If the above condition has not been met, then we check if the number of iterations has exceeded the pre-determined limit. If this is true, then localization is terminated, and an estimate of the hole position is computed from the mean of the configurations in the belief set.

In either case, whether localization terminates successfully or is forcibly terminated, the estimated hole position is compared against the actual. If the hole-position estimate is within assembly clearance distance of the actual, then assembly is declared successful. Otherwise, the trial is considered a failure.

Results

For the circular peg-in-hole, 10000 assembly trials were simulated. If the estimation error (distance between the actual and estimated hole positions) is greater than the assembly clearance radius (0.5 mm), then the assembly trial is considered a failure. There were a total of only 9 failures out of

10000 trials. Figure 5.19 shows the final estimated hole-position w.r.t. the actual. It also depicts the goal region (assembly clearance) and the hole-uncertainty region for comparison.

(a) Estimated hole-positions w.r.t. the actual

(b) Comparison of goal region and hole-uncertainty region

Figure 5.19: Localization results from the circular peg-in-hole assembly trials: 99.9% trials successful, where the estimated hole-configuration is within the goal region (assembly clearance) radius of 0.5 mm. Also shown is the hole-uncertainty region (radius: 25 mm) for comparison.

Localization was forcibly terminated after the 20th iteration. All 9 assembly failures were found to have been forcibly terminated. This shows that the localization algorithm is consistent. If the belief set converges and localization terminates successfully, then successful assembly is guaranteed.

Out of the 10000 trials, 146 trials were terminated forcibly. However, only 9 of these resulted in failure. This information can help us strike a good balance between the cost of further localization and the probability of the current estimate failing.

Simulations Adding Observation Errors

Modified Algorithm. The simulated assembly trials of the circular peg-in-hole reported above did not incorporate any observation errors that occur in the physical assemblies. Hence, to implement localization on physical

assemblies, we need to augment the algorithm to handle the associated observation errors. We accomplished this by accounting for the maximum error possible for each observation cycle. This was done by increasing the matching tolerance by the maximum observation error. While this makes the algorithm more conservative, and it takes a longer time to converge, it does allow us to handle the errors associated with a physical assembly. We tested the new algorithm on the computer simulation of the circular peg-in-hole assembly, which was designed to closely mimic the corresponding physical assembly by adding a random error to each observation. (Note that this method assumes prior knowledge of the maximum observation error possible.) While we may not be able to guarantee that an observation error will never exceed the maximum estimated, we may be able to put a good estimate on the probability of this happening. Then it is a matter of striking a good balance between the rate of convergence (higher for a lower maximum error) and the probability of an observation error exceeding the maximum, leading to localization failure.

For the simulated assembly trials, we set a maximum observation error of 0.2 mm—compared to an original matching tolerance (which is the same as the sampling tolerance) of 0.1 mm. Hence, the new dilated matching tolerance was 0.3 mm.

Results. We ran 1000 simulated assembly trials with added observation error. Because of the conservative matching tolerance to account for the observation error, 824 assemblies (82.4%) were terminated forcibly, which is much larger compared to 1.5% for the original trials.

Out of the 1000 trials, 711 ended successfully. Again, all the failures resulted from forcible termination of the trials, as the number of iterations exceeded the pre-determined limit of 50.

To gain some insight into the progress of localization over the iterations,

Figure 5.20: Comparison of localization progress in terms of uncertainty resolution and localization accuracy w.r.t. the number of iterations carried out, averaged over 1000 trials, for the circular peg-in-hole assembly with added observation error.

we recorded the uncertainty² after the 10th, 20th, 30th, 40th, and 50th iterations. Figure 5.20(a) shows a comparison of the residual uncertainty for the recorded iterations. As would be expected, there is rapid reduction in uncertainty in the first 20 iterations, after which further localization progresses very slowly. This analysis can help us determine the number of iterations to allow before localization is terminated forcibly to strike a good balance between uncertainty resolution and localization time.

We also estimated the relative peg-hole configuration from the current belief set after each of the above-mentioned iterations, and compared them with the actual to check whether assembly would have been successful if localization had been terminated after that iteration. Figure 5.20(b) shows the number of assembly failures plotted against the number of iterations. The results are counter-intuitive. Although there is more uncertainty after 10 iterations compared to a later stage, more assemblies would be successful if localization is stopped after the 10th iteration. That is, the mean of all

²The recorded uncertainty is determined by finding the smallest enclosing circle for the points in the belief set.

the remaining configurations falls within assembly clearance distance of the actual hole position more frequently. A possible reason for this result can be that, after 20 or more iterations, the points are highly localized w.r.t. the current observation, and if the observation error is large, then, although the mapped point closest to the actual *is* contained in the belief set, the majority of the points are clustered according to the observation. Since we compute the estimate as the mean of all the points in the belief set, a heavily populated cluster corresponding to a flawed observation could drag the estimate out of the goal region. On the other hand, after 10 iterations, the points are still scattered in a comparatively larger area, and the mean might fall within the goal region more frequently.

Discussion

In testing our localization strategy on the circular peg-in-hole assembly, we realized that satisfying assembly-sufficiency conditions requires ultra-fine discretization, resulting in huge maps. The overall map size depends on the range of relative peg-hole configurations to be mapped. Hence, when the initial uncertainty in hole configuration is restricted to a small range of configurations, the map size is tractable. But, with larger ranges of hole-configuration uncertainty the map size becomes prohibitively big. Below, we give a short analysis of the increase in map size relative to the uncertainty region.

For the circular peg-in-hole assembly, the hole-uncertainty region is defined as a circular area of radius R in the (x, y) -plane.³ We fix the peg motion region to be equal to the hole-uncertainty region, and hence a circular area of radius R . Then the total area of relative peg-hole configurations to be mapped is given by a circle of radius $2R$. The number of points to be

³Even for three-dimensional uncertainty, only the free-space region (x, y) is bounded, uncertainty along the contact-axis (z) can be unbounded.

mapped is roughly proportional to the mapped area, which is equal to $4\pi R^2$. If we increase the radius of the initial hole-uncertainty region by r , then the area to be mapped increases to $4\pi(R+r)^2$, which corresponds to an increase of $4\pi r^2 + 8\pi Rr$.

5.7.3 Square Peg-in-Hole Assembly

Set Up

We simulated a restricted version of the Square Peg-in-Hole II assembly problem (described in Section 2.3.5). The general square peg-in-hole is a six-dimensional localization problem. However, we restricted the uncertainty in the position of the hole to the (x, y) -plane, and in the orientation of the hole to rotations about the vertical (θ_z). That is, the orientation of the hole surface in the (θ_x, θ_y) -space and its vertical placement (z) are known to the controller. Hence, the contact hyper-surface map was sampled in the (x, y, z, θ_z) dimensions.

This simulation was used to test the general localization strategy using sampled maps given in Algorithm 5.3 (see Figure 5.9). For the Square Peg-in-Hole I assembly, described in Section 2.3.2, with hole- and peg-edges of 70 mm and 69.5 mm, respectively, a contact hyper-surface map that satisfies Sampling Condition 5.3 would contain more than 10^8 configurations for a hole-uncertainty region bounded within a 25 mm radius circle and $[0, \frac{\pi}{2})$ orientation range. We make the problem tractable by increasing the size of the goal region and reducing the hole-uncertainty region, thus reducing the size of the map. The goal region is made bigger by decreasing the peg-size (to an edge-length of 65 mm), while the size of the hole is kept the same. The hole-position uncertainty ranges over a circle of radius 10 mm. Although the hole-orientation uncertainty range is $[0, \frac{\pi}{2})$, only half that range, $[0, \frac{\pi}{4}]$, needs to be mapped. This is made possible by exploiting the fact that every

relative peg-hole (x, y, θ_z) -configuration, where $\theta_z \geq \frac{\pi}{4}$, has the same contact height as the configuration $(-x, y, -\theta_z + \frac{\pi}{2})$, which is mapped because $-\theta_z + \frac{\pi}{2} \in [0, \frac{\pi}{4}]$. The two configurations can be considered mirror images of each other as described in Section 3.2.2. The peg motion region in (x, y) , which also contributes to the map size, is limited to a circle of radius 5 mm. The peg orientation range is bounded within $[0, \frac{\pi}{2})$, however, since the relative peg-hole orientation range of $[0, \frac{\pi}{4}]$ is already being mapped, limiting the peg-orientation range does not affect the map size. Besides using these new parameters, we set a conservative (large) z-bound of 0.5 mm, which brings the final map size down to 332932 configurations.

Localization Termination Conditions

Localization was terminated if either of the following two conditions were met:

1. All the configurations in the belief set are contained within the goal region.
2. The number of iterations has exceeded a pre-determined limit.

For the square peg-in-hole assembly, the goal region in (x, y, θ_z) is diamond-shaped, as shown in Figure 5.21, and computing whether all the configurations in the belief set are contained within this volume is complicated. We solve the problem in stages as follows. From the extents of the points in the belief set along x , y , and θ_z , we find the smallest rectangular prism (aligned with x , y , and θ_z) that can contain all the points. This is a simple computation. Now, we check if this prism can be contained within the goal region. If yes, then the belief set is adequately localized. Otherwise, we check if the prism can contain the goal region. If so, then we know that the points in the belief set cannot be contained within the goal region. Otherwise, we can undertake more complicated computations to determine if the points are

contained within the goal region. If this convergence condition is satisfied, then localization is terminated, and the hole-configuration estimate is set equal to the center of the enclosing volume.

If the above condition has not been met, then we check if the number of iterations has exceeded the pre-determined limit of 25. If this is true, then localization is terminated, and an estimate of the hole position is computed from the mean of the configurations in the belief set.

In either case, whether localization terminates successfully or is forcibly terminated, the estimated hole configuration is compared against the actual. If this estimate is within assembly clearance distance of the actual, then assembly is declared successful. Otherwise, the trial is considered a failure.

Results

For the square peg-in-hole assembly, 100 localization trials were simulated. Estimation error greater than the goal region leads to assembly failure. There were a total of 27 failures out of 100 trials. Figure 5.22(a) and (b) show the final estimated hole-position and orientation w.r.t. the actual, respectively.

For the square peg-in-hole assembly, with the conservative sampling tolerance and z -bound used, all the trials had to be forcibly terminated after 25 iterations. However, for every trial, even after 25 iterations, the actual relative peg-hole configuration was contained within the sampling tolerance of a mapped point in the belief set. This shows that the localization algorithm is consistent.

We also recorded the unresolved uncertainty volume⁴ after the 5th, 10th, 15th, 20th, and 25th iterations for comparison. Figure 5.23 shows the volumes for the selected iterations, averaged over 100 trials. There is a sharp reduction in uncertainty between the 5th and 10th iterations, however, after

⁴The uncertainty volume is computed as the volume of a rectangular prism (in x, y, θ_z) containing all the configurations in the belief set.

Figure 5.21: The goal region in (x, y, θ_z) for the square peg-in-hole assembly is diamond-shaped. The smallest rectangular prism containing the goal region is shown. On the other hand, a prism contained inside the goal region can be of various different sizes.

the 10th iteration, the uncertainty varies little.

We also computed an estimate of the relative peg-hole configuration from the mean of the configurations in the belief set after each of the selected iterations. If localization had been terminated after the 5th, 10th, 15th, and 20th iterations, 75, 17, 21, and 20 trials would have failed to localize sufficiently (within the goal region), respectively. These numbers corroborate the analysis above to show that, on average, after the 10th iteration, sufficient localization has been achieved, and more iterations will not help in localizing further. One important observation was that the first few iterations are computation intensive because of the global initial uncertainty. The belief set, $\mathcal{B}_0$, is initialized to include all the mapped configurations corresponding to the hole uncertainty region, and in the next iteration, all these configurations, after being updated, have to be matched with the map. After a few iterations, as the belief set is localized, the computational load eases, and the final few iterations require little computation.

Figure 5.22: Localization results from the square peg-in-hole assembly trials: 73% trials successful, where the estimated hole-configuration is within the goal region of the actual.

Discussion

In testing our localization strategy on the square peg-in-hole assembly, we again realized that satisfying assembly-sufficiency conditions requires ultra-fine discretization, resulting in huge maps. Here, we give a short analysis of the increase in map size relative to the uncertainty region for a square peg-in-hole assembly. For the restricted problem, the contact surface map is four-dimensional in (x, y, z, θ_z) . The uncertainty in hole-position in (x, y) is defined as a circular area of radius R . The uncertainty in hole-orientation (rotation about θ_z) is within a range of $[0, \theta]$, $\theta < \pi/2$. As the peg motion region is set equal to the hole-uncertainty region, the total relative peg-hole configuration region to be mapped is given by a circle of radius $2R$, and a range of $[0, \min(2\theta, \pi/2)]$. With the added dimension of orientation, the number of mapped points for a given mapped area in (x, y) for the square peg-in-hole is N -times as large as the number of mapped points for the circular peg-in-hole, where N follows from the discretization of the orientation range. The total number of points to be mapped is proportional to the three-dimensional volume of $4\pi R^2 \times 2\theta$. If we increase the radius of the ini-

Figure 5.23: The residual uncertainty volume (averaged over 100 trials) after the 5th, 10th, 15th, 20th, and 25th iterations for the square peg-in-hole assembly. The uncertainty volume is computed as the volume of the rectangular prism containing all the configurations in the belief set.

tial hole-uncertainty region by r , then the volume to be mapped increases to $4\pi(R + r)^2 \times 2\theta$, which corresponds to an increase of $(4\pi r^2 + 8\pi Rr) \times 2\theta$.

5.8 Conclusions

In general, for an arbitrary peg-in-hole assembly, we can carry out localization using sampled maps as described above. The goal region and the sampling conditions can be derived from the assembly parameters. Once the sampling conditions have been determined the contact hyper-surface map can be sampled using either a robot or a CAD model. The sampled map can then be used to implement the localization algorithm described above, and estimate the hole configuration relative to the peg. The actual hole-configuration is then guaranteed to be within the goal region of the estimate. If assembly is attempted by aligning the peg with the estimated hole-configuration, assembly will be successful.

Each iteration in the localization algorithm can be divided into two

phases, the motion (or, updating) phase and the observation (or, matching) phase. In general, after the update phase, there is an increase in uncertainty, and after the matching phase, there is a reduction in uncertainty. This can be explained as follows. Suppose we keep moving the peg around and updating the belief set without matching the observations. For each configuration, T_k , in the belief set, $\mathcal{B}_k$, we include all the mapped configurations that are within the sampling tolerance of T_k , in the new belief set, $\mathcal{B}_{k+1}$. Generally, there will be more than one such configuration surrounding T_k . Thus, after an update, the uncertainty increases. However, in the observation-matching phase, all the configurations that do not agree with the observation are pruned out, and the uncertainty is lowered. Usually, at least in the initial iterations, the uncertainty resolution in the observation phase exceeds the uncertainty addition in the update phase, and there is a net lowering of uncertainty. After a certain degree of localization resolution has been achieved (depending on the sampling tolerance), the addition and resolution of the uncertainty in the two phases balance each other out and localization is stalled. However, it is important to note that after this equilibrium condition is achieved, the uncertainty can oscillate about its equilibrium point depending on the region of contact hyper-surface encountered in the iteration.

As evidenced by the circular and square peg-in-hole assembly implementations, the localization algorithm based on sampled maps is plagued with the curse of dimensionality. With increasing dimensions, the fine-grained maps needed to satisfy assembly sufficiency conditions grow intractably large. This affects not only the memory requirements to store the map, but also the computation time required to match the large number of initial configurations to the map. Therefore, in the next chapter, we introduce the cell approach algorithm to implement the general localization strategy in stages of increasing resolution, thus distributing the computational load more evenly over the localization period.

Chapter 6

Assembly Localization Using the Cell Approach

In the previous chapter, we described localization using sampled maps. Starting with global uncertainty, the initial belief set includes all the configurations in the map corresponding to the uncertainty region. As the amount of computation required to match a peg move to the map is proportional to the size of the belief set, the first few iterations are computationally much more expensive than later iterations as the belief set is pruned down. These computations are carried out in real-time, and ideally we would like to limit the computation to the time required to sample a peg-move, which remains constant over all iterations. Hence, our goal is to find an algorithm that implements our localization strategy with the computation equally distributed (approximately) over all iterations. In this chapter, we introduce the **cell approach**, which implements our localization strategy in stages of increasing resolution, thus distributing computational load more evenly over all iterations.

The sampling conditions, described in the previous chapter, were uniform over the entire map. That is, all configurations on the map had to abide by the same contact-axis bound, regardless of the different features in various

regions of the map. With the cell approach, on the other hand, the contact axis bound is unique for every cell (which is a collection of configurations, as explained later). Hence, cells in flat regions of the map with little variation can have small contact-axis bounds, while in regions with large variations along the contact-axis, the cells have larger bounds, eliminating the need for very fine sampling to satisfy the sampling conditions.

We use the lock-key example (introduced in Chapter 2) to fix ideas throughout this chapter. We *do* assume that the lock-surface is perfectly horizontal and the key held by the robot is vertical. Thus, uncertainty is limited to the (x, y, z) position of the lock and its orientation about the z -axis. Using the key to probe the lock-surface, we realize that the pointed tip of the key that contacts the lock can be *approximated* as a point probe. Now, we can construct a four-dimensional contact hyper-surface map in (x, y, z, θ_z) , as we did for the square peg-in-hole assembly. This four-dimensional map can be considered as a collection of three-dimensional (x, y, z) maps for each relative probe-surface orientation (θ_z) possible. However, a crucial difference in the four-dimensional contact surface maps of the lock-key and the square peg-in-hole is that for the lock-key all the (x, y, z) maps are identical to each other if aligned properly in θ_z , while for the square peg-in-hole, the (x, y, z) maps are not identical. Actually, within a $[0, \frac{\pi}{2}]$ interval of θ_z , none of the maps is identical to another. Figure 6.1 shows a sampled map of the key's position while contacting the lock surface, recorded w.r.t. the lock's coordinate frame.

6.1 The Cell Approach

In the previous chapter, the resolution of the map was fixed by the sampling tolerance, and was held constant throughout a localization trial. Here, we localize with a coarse resolution in the initial stages with global uncertainty,

Figure 6.1: Key-lock contact surface map sampled using the ParaDex robot, and plotted using the `mesh` command in Matlab [31]. Each sample is the position of the key w.r.t. the lock coordinate frame.

and then increase resolution as localization proceeds. The concept of a **cell** is introduced to substitute for the sampling tolerance, hence, the implementation is called the **cell approach**. An example four-dimensional localization problem is used for illustration.

Consider the problem of localizing a mobile probe on a three-dimensional surface, shown in Figure 6.2. The controller knows neither the starting position of the probe on the surface, nor the orientation of the surface (rotation about the vertical) w.r.t. the probe. Thus, it is a four-dimensional localization problem. As the probe moves over the surface, the controller samples contact point observations in (x, y, z) . Our goal is to localize the probe on the surface by matching the observations to a map of the surface. However,

Figure 6.2: Three-dimensional surface with two probe moves shown; we sample and store a map of the probe-surface contact transformations.

while we need to localize in four dimensions (x, y, z, θ_z) , we employ a three-dimensional map of key-lock contact positions in (x, y, z) . Hence, besides matching the observations to the three-dimensional map, we also compute the possible transformations from $\mathcal{F}_s$ (coordinate frame attached to the surface) to $\mathcal{F}_p$ (coordinate frame attached to the probe). This helps us localize both the position of the probe on the surface and the relative orientation between the probe and the surface.

The cell approach is explained below. We start with a simple two-dimensional example of a sine-curve. The curve, as shown in Figure 6.3(a), is sampled with a fine-grid resolution, leading to a large number of sampled points in the map. Localization on the curve starting with global uncertainty and using the sampled map requires heavy computation in the first few iterations. The computational load tapers off as localization proceeds. Using the cell approach, the mapped configurations on the sine curve are grouped into larger cells anchored at regularly-spaced grid points along x , as shown in Figure 6.3(b). Each cell represents all the points on the curve that are

closest to its center (grid-point) along x . The cell also represents the height extents of the curve along z . Hence, the cell-size in z can be found from the minimum and maximum z -values of all the points on the curve within the x -range of the cell. Localization then proceeds using these cells instead of the individual configurations. As the number of cells is smaller than the total number of mapped configurations, the initial computational load is reduced.

The matching of a probe move (in (x, z)) to the cells can be carried out in two parts: the **x -match** and the **z -match**. For the x -match, we check whether the x -length of the move can be accommodated within a pair of cells. If yes, then the z -match can be carried out. Then for the z -match, we check whether the **z -difference** (i.e., the difference in heights of the two contact points) of the move can be accommodated within the cell-pair. Thus, we can figure out if the probe *could* have moved from one cell to the other.

As localization proceeds and uncertainty is resolved, the number of cells representing current uncertainty is reduced. For instance, in Figure 6.4(a) the uncertainty is resolved to within four cells. Hence, at this time, the computational load is much lower than it was initially. Then the four large cells can be decomposed into smaller cells, as shown in Figure 6.4(b), thus increasing the resolution. The computational load is again raised to the original level. Thus, the cell approach allows us to localize in stages of increasing resolution while distributing computational load evenly over the entire period.

Similarly, for our four-dimensional localization problem, we overlay the surface-map with a two-dimensional grid of cells as shown in Figure 6.5. Each cell represents all the mapped locations contained within its (x, y) range. We add a third dimension to each cell—a z -range, which is determined individually for each cell depending on the z -values of the points contained within it. Suppose $\mathcal{T}$ is the set of all the mapped configurations contained within a cell. Then the z -range for that cell is given by the minimum and maximum z -

(a) Localizing on a sine curve using a sampled map.

(b) Sampled points grouped into larger cells.

Figure 6.3: (a) A sine curve sampled with fine-grid discretization. Localization on the curve starting with global uncertainty and using the sampled map requires heavy computation in the first few iterations. The computational load tapers off as localization proceeds. (b) Using the cell approach, the mapped configurations on the sine curve are grouped into larger cells, and localization is implemented on the cells instead of the individual configurations. As the number of cells is smaller, the computational load is reduced.

values among all the configurations in $\mathcal{T}$. This z -range can also be “inflated” (i.e., expanded/dilated) to account for incomplete mapping or anticipated errors in observation and actuation. Thus, the cell-approach can accommodate the worst-case scenario. Now, we have a collection of square prisms, each representing a cell, covering the entire surface. However, each cell has a square cross-section in (x, y) , which makes it difficult, when matching a probe move to the map, to check whether the (x, y) -component of a move fits between two cells. To facilitate this checking, we circumscribe each cell

(a) Localization progress after a few iterations.

(b) Large cells decomposed into smaller cells.

Figure 6.4: (a) After a few iterations of localization using the large cells, the uncertainty is resolved to within four cells. Hence, the computational load is much reduced. (b) Then the four large cells can be decomposed into smaller cells, thus increasing the resolution in stages. The computational load is again raised to the original level.

with a circle in (x, y) , thus changing the cells from square prisms to cylinders. This is shown in Figure 6.6 for an example set of four square prisms. Figure 6.6(b) shows the cylinders circumscribing the square prisms. Each cylindrical cell now represents the surface contained within the grid-square, i.e., the surface has been expanded into the three-dimensional cylindrical volume. Figure 6.7 shows an example of a portion of a surface being represented by a cylinder. It can be seen that the cylinder covers the entire z -range of the surface (contained in the grid-square).

Figure 6.5: The surface is divided into a square grid of points that form the centers of the cells. The cells represent all the points that fall within the x, y -squares occupied by them.

6.1.1 Algorithm for Localization Using Cell Approach

We represent a cell with two parameters, its location and its volume. The location of the center of the cell w.r.t. the world coordinate frame is denoted by the vector l .¹ The cylindrical volume of the cell is denoted $\mathcal{V}$, where $\mathcal{V}$ is the set of all the points (described w.r.t. l) within the whole volume (not just on the surface). So all the points in some cell l_i would be given by $\mathcal{V}_{l_i}$. Even though each point in the volume can be represented simply as a vector w.r.t. l , we set up $\mathcal{V}$ as a set of coordinate transformations, so that we can carry out the regular transformation operations on it. Then the volume for a cell, l_i , w.r.t. the world coordinate frame can be given as $T_{l_i} \mathcal{V}_{l_i}$.

Below we present the incremental algorithm for the cell approach with fixed resolution. The belief set, $\mathcal{B}_0$, representing all possible current locations of the probe on the surface, is initialized to include all the cells corresponding to the uncertainty region. However, for the cell approach,

¹In the following description, l is also used to refer to the cell itself.

Figure 6.6: An example of how the square prisms are circumscribed with cylinders to facilitate the matching computations.

a belief set for iteration k , $\mathcal{B}_k$, contains not just the candidate cells for the current location, but their entire history in terms of *sequences of cells*. Hence, if $|\mathcal{B}_k| = n$, then it contains n *sequences* of cells, where each sequence $S_i^k = [l^0, l^1, \dots, l^k]_i, i = 1, \dots, n$. After the next iteration, $k + 1$, each of these sequences can produce several more sequences, e.g., S_i^k can produce j sequences: $S_{ij}^{k+1} = [l^0, l^1, \dots, l_j^{k+1}]_i$, by adding a cell l_j^{k+1} to each new sequence. Or, S_i^k can terminate without producing any S_{ij}^{k+1} . The reason for maintaining a record of an entire sequence instead of only the currently occupied cell is that it helps resolve orientation uncertainty. Knowing the sequence of cells that the probe has been through narrows the range of possible orientations it can be in w.r.t. the surface. This is explained in more detail in Example 6.1.

As mentioned before, the controller samples contact point observations in (x, y, z) as the probe moves over the surface. Each observation, T_p^w is a record of the probe configuration w.r.t. some robot coordinate frame. The probe move between two consecutive observations is then matched to the cells to localize the position and orientation of the probe on the surface.

As an example, consider the first probe move, α , recorded in (x, y, z) . There is global initial uncertainty in both position and orientation. Hence,

Figure 6.7: Cylinder representing a portion of the surface shown. The entire z -range of the surface is covered by the cylinder.

for each cell, we have to compute what other cells can be reached with move α . The matching can be divided into two parts: an (x, y) -match and a z -match. Figure 6.8 shows an illustration of the (x, y) -matching process for a cell, l^0 . To account for global orientation uncertainty, all the cells that intersect the marked peripheral region around l^0 , which can be reached from l^0 with move α in (x, y) , are selected. Then, for the z -match, we check whether the z -difference (i.e., the difference in heights of the two contact points) of the move can be accommodated within the cell-pair. If both the (x, y) - and z -match are good, then the cell pair is included in the belief set as a valid sequence. As shown in Figure 6.8, l^1 passes the (x, y) -match for move α from l^0 , and also the z -match (not shown) and hence, is included in the belief set.

After a second probe move, β , each sequence in the belief set is advanced

Figure 6.8: Matching of the first move, α , for one of the cells, l^0 , in the (x, y) -plane is shown. There is also a z -match, which is not shown here. There is global orientation uncertainty. Hence, the entire peripheral region that can be reached from l^0 with the length of move α is checked. This region intersects cell l^1 . If a z -match between l^0 and l^1 based on move α is also satisfied, then sequence $[l^0, l^1]$ is included in the belief set.

and matched to the probe move. At this time, for each sequence, both the (x, y) - and z -match are recursively carried out between the new cell and all the old cells in the sequence. For example, to match sequence $[l^0, l^1]$ with a new cell, l^2 , move β is matched between l^1 and l^2 , and the combined move $\alpha + \beta$ is matched between l^0 and l^2 .

The complete algorithm is presented in Figure 6.9, where $\mathcal{L}$ refers to the collection of all possible cells. Each iteration involves making an observation, matching it to the belief set, and updating the belief set. A detailed explanation of the algorithm follows.

Algorithm 6.1

- Step 0: • Set $k = 0$
- Record observation: $T_{p_0}^w$
 - Initialize belief set, $\mathcal{B}_0$
- Step 1: • Update $k = k + 1$
- Record observation: $T_{p_k}^w$
 - For each sequence, $S_i^{k-1} = [l^0, \dots, l^{k-1}]_i \in \mathcal{B}_{k-1}$,
 - For each cell $l_j \in \mathcal{L}$,
 - * Generate sequence $S_{ij}^k = [l^0, \dots, l^{k-1}, l_j^k]_i$
 - * For $m = 0$ to $k - 1$,
 - match move $T_{p_k}^{p_m}$ from cell l^m to l_j^k :
 - perform (x, y) -match
 - perform z -match
 - * If all matches are good, then select S_{ij}^k to $\mathcal{B}_k$
- Step 2: • Check if $\mathcal{B}_k$ is satisfactory $\Rightarrow$ goto Step 3.
- Else check if too many iterations $\Rightarrow$ goto Step 4.
 - Else goto Step 1.
- Step 3: • (Localization successful) Assemble
- Step 4: • (Localization failed) Stop

Figure 6.9: Algorithm for implementing the cell approach.

Since our problem deals with global initial uncertainty, the belief set, $\mathcal{B}_0$, is initialized to include all the cells in set $\mathcal{L}$. After each observation, $T_{p_k}^w$, we compute the probe move $T_{p_k}^{p_{k-1}} = (T_{p_{k-1}}^w)^{-1} T_{p_k}^w$. This $(\Delta x, \Delta y, \Delta z)$ probe move is w.r.t. the probe coordinate frame. The belief set, $\mathcal{B}_{k-1}$, contains sequences $S_i^{k-1}, i = 1, \dots, n$. For each sequence, we generate p new sequences, by adding each cell $l_j \in \mathcal{L}$ to the end of the original sequence. Here, $|\mathcal{L}| =$

p . The end-cell, l_j , represents the cell currently occupied by the probe. Now, for each new sequence, S_{ij}^k , we check whether each of the probe moves, $T_{p_k}^{p_0}, T_{p_k}^{p_1}, \dots, T_{p_k}^{p_{k-1}}$, can be accommodated between cells, $l_i^0, l_i^1, \dots, l_i^{k-1}$, and l_j^k , respectively. The matches are carried out taking into consideration the lengths of the probe moves and the orientation bounds, as explained below. If all the matches are successful, then sequence S_{ij}^k is selected to the new belief set, $\mathcal{B}_k$; else, it is pruned.

Thus, after every probe move, the belief set is updated to include all the possible cell sequences observed by the probe. Figure 6.10 shows some of the possible cell sequences for the probe moves shown in Figure 6.2.

Figure 6.10: Some of the possible cell sequences for the probe moves shown in Figure 6.2. The actual cell sequence is included.

Example 6.1 *This example illustrates the advantage of recording an entire sequence of cells. Knowing the sequence of cells that the probe has been through narrows the range of possible relative orientations between the probe and surface. Moreover, matching each cell in the sequence to the new cell helps prune out errant candidates. For simplicity, we restrict the probe to straight line motion. Hence, the orientation of the probe w.r.t. $\mathcal{F}_w$, and $\mathcal{F}_s$, is fixed for the duration of the localization. Moving the probe in a straight line makes it easier to match moves to the cells. Also, computing and pruning the orientation uncertainty associated with each sequence is much easier for straight line motions. Figure 6.11 shows the computation involved in the (x,y) -matching of the second probe move, β , for the cell sequence, $[l^0, l^1]$. Because the moves are made along a straight line, the region where the probe can currently be (in (x,y)) is easily computed, both in terms of distance along the travel-direction and the orientation range, as the intersection of three regions. As shown in Figure 6.11, the three intersecting regions are:*

1. *Probe position based on the length of move β from l^1 .*
2. *Probe position based on the length of the combined move $\alpha + \beta$ from l^0 .*
3. *Probe position based on the orientation bounds enforced by the fact that the probe has to have moved through cells, l^0 and l^1 .*

Once the region of possible current occupation in (x,y) is determined, all the cells that intersect this region are considered for a z -match. Again, for the z -match, both β and $\alpha + \beta$, are matched to the z -differences between the current candidate cells and l^1 and l^0 , respectively.

6.1.2 Localization in Stages of Increasing Resolution

One of our goals in implementing the cell approach was to distribute computation more evenly over the localization period. Ideally, the computation

Figure 6.11: After the second move, β , both move β and the combined move, $\alpha + \beta$, are matched to advance sequence $[l^0, l^1]$. From the two cells, l^0 and l^1 , we compute the possible region of current occupation of the probe. The matching process in the (x, y) -plane is shown here. The computation is made easier because we have chosen to restrict the probe moves to be along a straight line.

for each iteration should take no longer than the time taken by the probe to make a move, which is constant if the probe moves are of fixed length. However, starting with global initial uncertainty, the first few iterations are computation intensive, while the computational load reduces considerably over the later iterations as uncertainty is resolved. A comparison of the computation time required for localization using different cell-sizes, described below, reveals an inversely proportional relation between cell-size and the required computation time. For a cell-size of 1.0 mm, the computation time is much smaller than for a cell-size of 0.2 mm, however, the larger cell-size is also more susceptible to localization failure. To combine the respective advantages of different cell-sizes, we propose to start the localization with a large cell-size, benefiting from a lowered computation time, and after a certain degree of localization has been achieved, shift to a smaller cell-size for more accurate localization and a higher success rate. In our implementation, the cells approximating the surface were pre-computed for each cell-size that was employed in the localization. It is also possible to pre-compute for each

large cell the smaller cells that it will decompose into. The decomposition can be easily generalized to higher dimensions, where it resembles the notion of **octrees** [23, 22].

When shifting between different cell-sizes, the biggest disadvantage is that the cell-sequences have to be discarded. The end-cells from the sequences are retained in the belief set with their corresponding orientation bounds. Now, many sequences will have the same end-cells, but with different orientation bounds. For the sequences with the same end-cells but different orientation bounds that do not overlap, we have to add duplicate end-cells to the belief set with their respective orientation bounds. However, in cases where the same end-cell has two or more overlapping orientation bounds, we combine the bounds into a **super-bound**, which is the union of the individual overlapping bounds, on the same cell. This is shown in Figure 6.12. Figure 6.12(a) shows an end-cell with two non-overlapping orientation bounds, while Figure 6.12(b) shows an example of how overlapping orientation bounds can be combined into a super-bound.

6.2 Simulation Results for Probe-on-Surface

6.2.1 Implementation Details

The four-dimensional localization problem described above, was simulated using Matlab. Figure 6.13 shows a block diagram of the simulation code. The simulation consists of an **environment** module and a **controller** module, invoked alternately by a **main** program. Uncertainty is introduced in the problem as the **environment** initializes the surface to be traversed by the probe at a random position and orientation, unknown to the **controller**. The **environment** updates the position of the probe on the surface based on motion commands issued by the **controller**. It also returns contact height *observations* to the **controller**. The **controller** uses the observations to

Figure 6.12: Different cell-sequences can have the same end-cell with different orientation bounds. (a) For non-overlapping orientation bounds, two copies of the same end-cell are added to the belief set with the individual bounds. (b) For overlapping orientation bounds, a super-bound, given by the union of the overlapping bounds is assigned to the end-cell (duplicates of the end-cell are not required).

implement the cell approach, and localize the probe on the surface.

One important addition was made to the approach described above. Moving the probe along a straight line helps simplify the matching computation. However, with a finite surface to move on, the probe will hit the boundary at some point of its trajectory, and further linear motion will not be possible. In this case, we turn the probe by -90° about θ_z (i.e., right turn). If the new probe heading is still into the boundary, then we turn the probe by 180° , which most often leads it back onto the surface.² The probe does make an observation every time it hits the boundary, because there is an immediate resolution of some uncertainty as the probe is bound to be in a cell on the boundary. One could carry this idea further and devise a wall-following algo-

²In the rare event that the probe gets stuck in a corner at the boundary, it will take four direction changes before it comes out of the corner.

Figure 6.13: Block diagram of simulation code: the environment module gets peg motion commands from the controller, updates the position of the probe on the surface and returns the probe height-*observation* to the controller. The controller uses the observations to localize the probe position on the surface.

rithm for localization. However, we avoid exploiting the boundary to carry out localization, and concentrate on our localization strategy using position measurements over the surface. This can be justified taking into consideration that for a very large surface, reaching the boundary and then following it might take a longer localization period than our strategy. After making the probe turn away from the boundary, we discard the sequence of cells preceding the boundary-cell, because computation required to match the length of moves including a right-angle is very complicated. However, we do preserve the orientation bounds inherited from the preceding sequence.

A trial terminates with success when the probe is localized within a pre-determined threshold. For the results reported here, success was declared when the probe position was localized within one or two contiguous cells and the orientation bounds on all the sequences were within pre-determined limits. The localization error was computed as the residual uncertainty associated with the area of the remaining cells. This area depends both on the number of cells remaining and the size of the cell. The localization strategy

Figure 6.14: When the probe runs into the boundary of the surface, it is first turned to its right (turned by -90° about θ_z). If the probe is still heading into the boundary, then it is turned by 180° , and now the probe will most probably head out onto the surface.

was not always successful. The trial was aborted if the localization was not resolved within a fixed number of iterations (set at 20 iterations for the trials here), or if the number of candidate sequences exceeded a pre-determined threshold (set at 10000 sequences).

6.2.2 Simulation Results with Fixed Cell Size

Figures 6.15–6.20 show some of the salient moments from a trial localization run. Figure 6.15 shows the initial position of the probe on the surface—at this time there is global uncertainty in the position and orientation of the probe w.r.t. the surface. After the first probe move, the cells of possible current occupation are highlighted in Figure 6.16. These are the end-cells for the sequences in the belief set and represent the location uncertainty of the probe. The number of sequences in the belief set at this point is 3862. The sequences represent not only the location uncertainty but also orientation uncertainty. The second probe move takes the probe to a high point on the

surface leading to a rapid reduction in uncertainty. The resulting end cells are shown in Figure 6.17. After the third and fourth probe moves, as the probe returns to level surface, the number of end-cells is higher than before, signifying greater uncertainty in the location of the probe. However, the overall uncertainty, in terms of position and orientation, has been reduced. (This is better explained in Example 6.2 below.) Once the probe hits the boundary, its overall uncertainty is quickly resolved to within three end-cells. Localization would thus proceed until sufficient uncertainty resolution has been achieved, and localization can be terminated.

Example 6.2 *Suppose starting with global uncertainty within a region, a person climbs to the top of the highest mountain in the region. At this time, although the location of the person is known accurately, his or her orientation is uncertain. Hence, when the person climbs down from the mountain, not knowing which way he or she was facing, the person could be anywhere on the circumference at the foot of the mountain. Thus, after moving from an exactly known location, the person ended up with location uncertainty owing to unresolved orientation uncertainty.*

Localization was implemented using the cell approach with three different cell-sizes for comparison, and results were averaged over 100 trials for each cell-size. The results are plotted in Figure 6.21. As the cell-size is increased, the number of failures increases. For cell-sizes of 0.2 mm, 0.6 mm, and 1.0 mm, the number of failures were 1, 3, and 8, respectively. This is because with larger cells the height range within each cell generally becomes larger—reducing cell-variation. With increase in resolution (smaller cell-size), the average estimation error reduces drastically, but the average time consumed per trial increases. This can be seen in Figure 6.21. This is because with smaller cells the number of cell-combinations to be considered is much larger—taking more computation time. However, it should be

Figure 6.15: Initial position of the probe on surface for a trial localization run. There is global uncertainty in the position and orientation of the probe w.r.t. the surface.

Figure 6.16: After the first probe move, cell sequences that accommodate the move are computed and selected to the belief set. The possible cells of current occupation are highlighted here.

Figure 6.17: The second probe move results in a steep climb up to a high point on the surface leading to a rapid reduction in uncertainty.

noted that, beyond a certain limit, increasing the cell-size does not lead to a proportionate reduction in CPU time. As can be seen from Figure 6.21(a), the difference in CPU time consumed for cell-sizes of 1.0 mm and 0.6 mm is negligible. This is because, while each iteration for the larger cell-size takes a shorter time, many more iterations are required to localize sufficiently compared to the smaller cell-size.

The cell-size can be decided taking several factors into account:

1. Amount of computational power available: the smaller the cell-size, the more computation is required to check all the possible cell-pairs.
2. Size of surface: with a bigger surface, a larger cell-size will reduce computational requirements.
3. Type of surface (waviness): the cell-size should also be adjusted to get the greatest variation (in height ranges) among the cells. That is, the height range for each cell should be as small as possible.

Figure 6.18: The third probe move and the resulting end-cells.

Figure 6.19: After the fourth probe move, the number of end-cells is higher than before, meaning greater position uncertainty. However, the overall uncertainty, in terms of position and orientation, has been reduced.

Figure 6.20: Once the probe hits the boundary, the uncertainty is quickly resolved to within three end-cells. The localization would thus proceed until sufficient resolution of uncertainty is achieved.

6.2.3 Simulation Results with Changing Cell Size

We also conducted localization trials where the cell-size was changed during the trial to distribute computational load and increase the resolution of localization. As explained above, the trial starts with a larger cell-size, and at some point in the trial, the larger *remaining* cells are decomposed into smaller cells. The cell-sizes are selected in such a way that the larger cell-size is an exact multiple of the smaller cell-size. This ensures that during the shift the smaller cells fit within the bigger cell.

The results from these experiments showed the feasibility of the approach. A crucial insight gained concerns the juncture in the trial where the shift is made from larger cells to smaller cells. It was found beneficial to make the switch when the probe hits the boundary, because then, when a large boundary cell is decomposed into smaller cells, only the cells that lie on the boundary need be selected. This results in a large reduction in the number of cells. However, the implementation needs to be pursued further to make it as successful as the implementation using fixed cell sizes.

Figure 6.21: Change in CPU time, and average estimation error with varying cell-size.

6.3 Conclusions

Our primary goal in introducing the **cell approach** was to implement the general localization strategy in stages of increasing resolution, thus distributing computational load more evenly over the localization period. We showed how mapped configurations can be grouped into cells, and localization can be carried out using these cells instead of the individual configurations. Because the number of cells is smaller than the total number of mapped configurations, the initial computational load is reduced. As localization proceeds and the uncertainty is reduced to a small number of cells, the computational requirement decreases. At this time, the remaining cells can be decomposed into smaller cells, thus increasing the resolution of localization while maintaining a level computational load.

The cell approach was successfully implemented on a four-dimensional localization problem. The simulated problem required the localization of a probe on a two-dimensional surface, with uncertainty in the location (x, y, z) of the probe on the surface and its orientation (θ_z) . Results from localization experiments using the cell approach with fixed cell sizes showed that with

decreasing cell size we obtain a higher success rate and better localization resolution. However, the computational requirement is also much higher. We also discussed several factors that should be taken into account when deciding on the right cell size.

The results from localization experiments with changing cell size showed the feasibility of the approach. However, the implementation needs to be pursued further to make it as successful as the fixed cell size implementation.

Chapter 7

Assembly Localization Using Particle Filtering

7.1 Introduction

The implementation of our localization strategy requires the matching of each observation to the map to progressively reduce the uncertainty. In an error-free, simulated assembly, deterministic matching using either an analytical map or a sampled map can guarantee uncertainty resolution up to any required degree of accuracy. However, to make the algorithm robust in the presence of errors in measurement and actuation associated with an actual robotic assembly, we had to implement the Modified Algorithm given in Section 5.7.2 and the cell approach given in Algorithm 6.1 (see Figure 6.9). These algorithms increase their matching-tolerance bounds based on the worst-case error to accommodate erroneous actuation and measurement. However, the conservative matching tolerance leads to lower rates of convergence and success.

In the mobile robot localization field, widespread success has been reported with the use of particle filters for map-matching even in the presence of large errors [45, 16]. Particle filtering is a probabilistic technique that can

accommodate inaccurate models and measurements using probabilistic laws such as Bayes' rule [14]. With particle filters a probability distribution can be maintained over a set of sampled states (or particles) and the number of particles can be adjusted depending on the computational resources available. In our problem, even coarsely-sampled maps of the contact surface contain close to 10^5 configurations. Hence, maintaining a probability distribution over the entire map would be computationally expensive. Particle filters allow us to concentrate resources on promising solutions without completely discarding less promising ones, which can turn out to be the real solutions camouflaged by error. This approach also allows us to track multiple possible solutions, which occur frequently in our particular problem. (E.g., for the lock-key problem, if the key touches the elevated portion of the lock, all the points on that surface become possible solutions to the observation.) Thus, particle filters provide a tractable means for implementing our localization strategy.

Below, we give a brief introduction to the probabilistic framework for state estimation. We also introduce particle filters and explain the implementation of our localization strategy using particle filters with a simple two-dimensional example.

7.1.1 The Probabilistic Framework

As explained in Chapter 5, under a deterministic framework, we maintain a set of candidate solutions called the belief set. At the start of localization, the belief set is initialized to contain all the transformations falling under the uncertainty region, and is subsequently pruned with each observation. Hence, with an incorrect observation it is possible that the correct solution might be eliminated from the belief set. On the other hand, under a probabilistic framework, we maintain a probability distribution over the discrete transformation space to represent our best estimate. This distribution is

updated after every action and observation. Hence, with an incorrect observation, the probability (confidence) of the correct solution might be reduced, but it is not eliminated. Thus, the probabilistic framework is better able to handle errors associated with measurement and actuation inherent in actual robotic assemblies.

Most probabilistic techniques use Bayes filters to update the probability distribution. Bayes filters recursively estimate the posterior using probability density functions over the state space representing actions (also known as the motion model) and observations (also known as the measurement model), as described below. Let x denote the state of a dynamical system— x_k is the state at iteration k , and o_k and a_k denote the observation and action at iteration k , respectively. Then the localization task is to estimate the probability distribution, $p(x_k)$, given the prior distribution, $p(x_{k-1})$, action a_k , and observation o_k . The recursive update equation based on Bayes' Theorem can be given as follows:

$$p(x_k|o_k, a_k) = C \, p(o_k|x_k) \sum_{i=0}^{k-1} p(x_k|a_k, x_{k-1})p(x_{k-1}|o_{k-1}, a_{k-1}).$$

Here, the initial distribution $p(x_0)$ is determined from initial conditions. The **motion model**, $p(x_k|a_k, x_{k-1})$, predicts the likelihood of reaching state x_k from state x_{k-1} , given action a_k , for all possible states and actions, respectively. The **measurement model**, $p(o_k|x_k)$, predicts the likelihood of making observation o_k , given current state x_k , for all possible states and observations, respectively. The **normalization constant**, C , ensures that the distribution sums up to 1.

In the assembly localization problem, the state of the dynamical system is the transformation from the hole coordinate frame to the peg coordinate frame. Thus, the k th state is $T_h^{p_k}$. Then the motion model, $p(x_k|a_k, x_{k-1})$, can be defined as $p(T_h^{p_k}|T_{p_{k-1}}^{p_k}, T_h^{p_{k-1}})$, where $T_{p_{k-1}}^{p_k}$ is the latest peg move.

The observation model becomes $p(o_k|T_h^{p_k})$, where the observations are peg contact configurations w.r.t. the robot coordinate frame, $T_{p_k}^w$. Because of the high resolution of the ParaDex robot, actions and observations do not introduce any significant uncertainty into our problem. Hence, our motion model has a uniform distribution over those $T_h^{p_k}$ and $T_h^{p_{k-1}}$ satisfying $T_h^{p_k} = T_{p_{k-1}}^{p_k} T_h^{p_{k-1}}$, and it is equal to zero otherwise. The observation model has a similar distribution for each state $T_h^{p_k}$. The global initial uncertainty in our problem leads to a uniform initial distribution, $p(T_h^{p_0}) = 1/\text{volume}(U)$, where U is the initial uncertainty region.

7.1.2 Particle Filters

In the grid-based approach, a probability distribution is maintained over the entire transformation space (discretized into a grid). Instead, with particle filters the distribution is maintained over a set of sampled states, $\{x_k^{(i)}\}, i = 1, \dots, N$. This adds the advantage that the number of particles, N , can be adjusted to accommodate the available computational resources. Below, we give an outline of the particle filtering algorithm that we use to implement our localization strategy.

We maintain a **belief set**, representing the best estimate of the current peg-hole coordinate transformation, given by $Bel_k = \{T_h^{p_k(i)}, w_k^{(i)}\}$. Here, each $T_h^{p_k(i)}$ is an estimate of the current (k th iteration) peg-hole coordinate transformation, and is called a **particle**. The **importance weight**, $w_k^{(i)}$, represents the likelihood of the particle state $T_h^{p_k(i)}$, given the observation (in our case, peg move). There is global initial uncertainty in the configuration of the hole. Hence, we initially distribute the particles uniformly throughout the uncertainty region. Also, the weights are set equal. In the algorithm presented below, the belief set is updated after every peg move (which can be represented as a coordinate transformation, $T_{p_{k-1}}^{p_k}$) by advancing each par-

ticle by $T_{p_{k-1}}^{p_k}$, and matching it to the map. The map, denoted by M , is a collection of *contact* coordinate transformations. With each iteration, the belief set (ideally) gives an improved estimate of the actual peg-hole coordinate transform. The algorithm exits with success when the peg-hole transform has been estimated with sufficient accuracy, and assembly is attempted. If the algorithm is not successful within a fixed number of iterations, then localization is terminated and assembly is attempted using an estimate based on a weighted average of the particles.

During resampling, duplicate particles are produced in the belief set. In other particle filtering implementations where there is large uncertainty in the motion model, the duplicate particles generally get scattered after the motion update phase. However, in our problem, with negligible uncertainty in motion, the duplicate particles do not scatter. Hence, to encourage variation and better space coverage in the updated particles, we add a small random offset to each particle. The above algorithm is explained in detail with the help of a two-dimensional assembly example below.

Two-dimensional Assembly Example

The implementation of our localization strategy using particle filters can be suitably illustrated with the following two-dimensional example. Consider the peg-in-hole assembly shown in Figure 7.2, where the uncertainty in hole location along x (horizontal) far exceeds the assembly clearance. Moreover, we assume that the hole location along y (vertical) is also unknown. An exhaustive search for peg-hole alignment would involve laying the peg flat on the hole surface and searching through the uncertainty region until the assembly is successful. To implement our localization strategy, we tilt the peg by a small amount and build a contact C-space map. This map is a collection of coordinate transformations from the peg-frame to the hole-frame

Algorithm 7.1

Step 0: • Set $k = 0$

- Initialize Bel_0 , $w_0^{(i)} = 1/N$, for all $i = 1 \dots N$

Step 1: • Move peg, update $k = k + 1$

- Update particles:

$$T_h^{p_k(i)} = T_{p_{k-1}}^{p_k} T_h^{p_{k-1}(i)}, \text{ for all } i = 1 \dots N$$

- Match each particle to map:

$$Err^{(i)} = \min_j \{ \|T_h^{p_k(i)} - T_j\| \}, T_j \in M$$

- Assign relative weights:

$$w_k^{(i)} = N(Err^{(i)}; 0, \sigma^{(i)}), \text{ for all } i = 1 \dots N$$

- Resample particles, with replacement, according to these

$$\text{weights, then reset weights: } w_k^{(i)} = 1/N, \text{ for all } i = 1 \dots N$$

Step 2: • Check Bel_k : if satisfactory $\Rightarrow$ goto Step 3

- Check Localization Time $\Rightarrow$ goto Step 4

- Goto Step 1

Step 3: • (Localization successful) Attempt assembly

Step 4: • (Localization terminated) Attempt assembly

Figure 7.1: Algorithm for implementing the general localization strategy using particle filtering.

Figure 7.2: Two dimensional peg-in-hole localization example: tilted peg in contact with hole surface; initial relative peg-hole configuration is $T_h^{p_0}$.

with the tilted peg contacting the hole as shown in Figure 7.3.¹ Figure 7.3 also highlights the coordinate transform $T_h^{p_0}$, which forms a point on the map. Once the contact C-space map has been acquired, localization can be implemented.

The peg is moved to a random point² in (x, y) , and is brought into contact with the hole. At this time, there is global uncertainty, and hence the particles are distributed uniformly throughout the uncertainty region, as shown in Figure 7.3. The weights are all set equal and normalized. This forms our initial belief set, $\mathcal{B}_0$. Now the peg is moved to another random contact point² and the peg move between the two contact points is noted. Let the peg-move be $T_{p_1}^{p_0}$ as shown in Figure 7.4. According to Step 1(b) of the algorithm given in Section 7.1.2, we update the particles by translating each by $T_{p_1}^{p_0}$, i.e., the old peg position w.r.t. the new one. The updated particles are shown in Figure 7.5. Now each updated particle is matched to the map (Step 1(c)) and the weights are updated (Step 1(d)). The matching error can be computed in two steps: first, we interpolate the mapped configurations to find the point that matches the particle in x . Then, we find the error in y

¹For this simplified problem, the coordinate transformation is simply a translation.

²The moves can also be to fixed points (Section 7.2.2) or chosen actively (Section 7.2.3).

Figure 7.3: The map is a collection of hole positions relative to the peg with the peg in contact with the hole.

between the interpolated point and the particle. From Figure 7.5, it is clear that the green/light-gray triangle and the red/medium-gray circle are still close to the map, while the other particles fall off the map. Hence, these two particles generate much smaller errors compared to the others. We update the weight associated with each particle based on its error. The weight is computed from a Gaussian distribution on the error with a mean of zero and a variance, σ^2 , that is lowered with each iteration. As in Step 1(e), we resample the particles, according to the updated weights. As we resample with replacement, the particles with smaller errors will generally be resampled more often and hence the particles bunch up around the green/light-gray triangle and the red/medium-gray circle. At this time we may have multiple copies of several particles, and we can add a small random disturbance to each particle to increase variation. The weights of the resampled particles

Figure 7.4: First peg move: represented as the new peg position w.r.t. the old, $T_{p_1}^{p_0}$.

Figure 7.5: Particles updated according to first move, $T_{p_0}^{p_1}$.

are reset to equal. The resampled particles are shown in Figure 7.6.

After another peg move, we again update the particles, compute the corresponding error, and resample. Over successive iterations, the particles generally localize around the actual transformation. Now, for every iteration, the peg move can be selected in several ways: in a fixed pattern, at random, or actively.

In **active localization**, actions (or peg moves) are selected actively, i.e., to optimize the uncertainty resolution of the resulting observation. This is explained in more detail in Section 7.2.3. Here, we show the advantage of active localization over random localization. With random localization it is

Figure 7.6: After resampling, the particles bunch up around the original particles that generated low errors.

possible that the randomly selected peg move will result in all the updated particles falling close to the map as shown in Figure 7.7, and hence, uncertainty is unchanged. However, active localization will pick a peg move that leads to maximum discrimination between the particles, and hence a large reduction in uncertainty, as shown in Figures 7.8 and 7.9. Regardless of whether one or the other set of particles represents the actual configuration, the same peg move³ will produce the same clear discrimination between the two groups. This can be seen from a comparison of Figures 7.8 and 7.9.

³Note that the controller simply selects the peg move along x . The change in z depends on the relative peg-hole position (not known to the controller).

Figure 7.7: Random localization: Peg move is selected randomly.

Figure 7.8: Active localization: Peg move is selected so as to optimally resolve the current uncertainty.

Figure 7.9: Even if the other set of particles had represented the actual configuration, the same actively selected peg move would produce maximum discrimination between the two groups of particles.

7.2 Localization Strategy Implementation Using Particle Filtering

Our aim in implementing the localization strategy using particle filters is to reduce large configuration uncertainties to smaller values in a relatively short time. As described in Chapter 5, it is possible to achieve any required degree of localization-resolution depending upon the resolution of the sampled map. However, a coarsely sampled map is advantageous in terms of storage size and sampling time. There is also a drop in time-efficiency in localizing beyond a certain degree of accuracy [9]. Moreover, from an *assembly line* point of view, to accommodate critical cycle-time requirements [19], we need to comply with a preset maximum localization time. Hence, after a pre-determined number of iterations we switch from the localization strategy to either an exhaustive search strategy (as explained in Chapter 3), or a compliant form of assembly [2] that can handle the reduced uncertainty.

7.2.1 Computer Simulations

The localization strategy implementation using particle filters was first tested in computer simulations. The simulations also helped to statistically evaluate the effect of significant parameters on success rate (percentage of localization trials ending in success) and rate of convergence, or average localization time, which we evaluate from the number of iterations. The two significant parameters are the number of particles used, N , and σ . As described before, when the updated particle is matched to the map and the error is computed, we use a Gaussian distribution on the error with variance σ^2 to compute the corresponding weight. Here, σ is a measure of the expected observation error. With global initial uncertainty we expect large observation errors, and as the uncertainty is resolved and the belief set is refined, we expect smaller observation errors. Hence, σ is lowered over successive iterations.

We ran simulation trials with varying values for both factors—the results are evaluated below.

Figure 7.10(a) shows the average estimation error as a function of both the number of particles used and the σ -value. The estimation error was averaged over 100 trials for each combination. We can see that for average values of σ , increasing the number of particles leads to a reduction in error up to a certain number of particles. Figure 7.10(b) shows the average number of iterations as a function of the number of particles used and the σ -value. Clearly, with increasing σ , the rate of convergence drops and more iterations are required for localization. From these results we can estimate good values for the number of particles and σ for the actual peg-in-hole assemblies.

Figure 7.10: Comparison of variation in average error and number of iterations with change in the number of particles and σ .

Finally, we tested the localization strategy implementation on actual robotic assemblies as described below.

7.2.2 Robotic Assemblies

Actual assemblies of the circular, square, and pentagonal peg-in-holes, and the lock-key assembly (all described in Section 2.3) were carried out using the ParaDex robot. We recorded 30 trials each for the circular and square peg-

in-hole assemblies, and 50 trials for the lock-key assembly. In every assembly trial, the localization phase was followed by a previously-devised compliant assembly phase [2]. The localization phase provided an estimate of the hole configuration and the compliant phase attempted to assemble the peg (or, the key) into the estimated hole configuration.

Localization was terminated if all the particles were contained within the goal area of the respective assembly. The estimate of the relative peg-hole configuration was computed from the localized particles, and assembly was attempted after aligning the peg with the estimated hole-configuration. Note, however, that this did not guarantee successful assembly, because, particle filtering being a probabilistic scheme, there was a small probability that the resulting configuration estimate was not within assembly clearance of the actual configuration. Actually, localization was rarely terminated on fulfillment of the above condition—most trials were terminated after a fixed number of iterations. In this case, although the uncertainty was not resolved to within the goal region,⁴ we still computed the relative peg-hole configuration estimate from the probability-weighted average of the particles. This estimate was then used to attempt assembly. Only if the assembly failed, was the trial considered a failure.

The selection of the peg move at each iteration can be achieved in one of three ways: from a fixed set, at random, or actively. We initially experimented with fixed moves and random moves within the peg-motion bounds. A combination of the two choices yielded good results. For the circular peg-in-hole and the lock-key problem, we move the peg to the south, west, north, and east boundaries of the peg-motion region before switching to selecting the peg moves randomly. These fixed peg moves were intended to force maximum coverage of the map, and thus produce maximum diversity in the peg-hole contact configurations. For the square and pentagonal peg-in-hole

⁴That is, all the particles were not contained within the goal region.

assemblies, only the first peg move was fixed, while the rest were selected randomly. Choosing the peg move actively involved computing the best move possible (the one that leads to the largest reduction in current uncertainty) and is called **active localization** [15]. Results from our implementation of active localization are reported in Section 7.2.3.

We used coarsely-sampled maps for all the assemblies. We also experimented with several different values for the number of particles employed for each assembly. The results from the particle filtering implementation on actual robotic assemblies are presented below.

A video of the ParaDex performing the key-lock plus the circular, square, and pentagonal peg-in-hole assemblies using particle filtering for localization can be seen in [10].

Circular Peg-in-Hole II Assembly Results

For the circular peg-in-hole assembly, the hole-uncertainty region ranged over a circle of radius 25 mm in (x, y) . Even though the assembly clearance was quite small (0.5 mm), a compliant assembly strategy following the localization phase could accommodate position errors within a 4 mm radius. Hence, our goal for the localization phase was to estimate the hole location within 4 mm of the actual. The number of iterations after which localization was forcibly terminated was set to 5. The contact surface map for the circular peg-in-hole contained 68630 configurations. The algorithm used 400 particles. For the circular peg-in-hole, we conducted 30 assembly trials, all of which ended successfully. Figure 7.11 shows the estimated hole locations w.r.t. the actual produced by the localization phase for the 30 assemblies. A reduction in uncertainty from a radius of 25 mm to 4 mm represents a reduction of more than 97% in terms of uncertainty area, which is the significant parameter for an exhaustive search strategy.

Figure 7.11: Results from 30 assembly trials for the circular peg-in-hole assembly.

Square Peg-in-Hole I Assembly Results

For the square peg-in-hole assembly, the compliant phase was quite sensitive to orientation errors. It could only accommodate errors up to 0.18 rad. The hole orientation uncertainty (rotation about z) was unbounded—effectively, the uncertainty range was $[0, \frac{\pi}{2}]$ owing to the $\frac{\pi}{2}$ -symmetry for rotations about z . In (x, y) , the uncertainty ranged over a circle of radius 20 mm, while errors only within a radius of 4 mm could be accommodated by the compliant phase. The number of iterations after which localization was forcibly terminated was set to 6. The contact surface map for the square peg-in-hole contained 159640 configurations. The algorithm employed 1000 particles. For the square peg-in-hole assembly, 27 assembly trials were successful out of the 30 attempted. Figure 7.12 shows the error in the estimated hole locations produced by the localization phase for 29 assemblies (error from one trial is off the plot). As can be seen, 25 estimates fall within the 4 mm radius. More importantly,

Figure 7.13 shows the error in the orientation estimate. All but two of the estimates fall within the bound of 0.18 rad of the actual hole orientation. On successful localization, there is a reduction of 96% in the (x, y) -uncertainty, and a reduction of more than 88% in the orientation uncertainty.

Figure 7.12: Results from 29 assembly trials for the square peg-in-hole assembly: Estimated hole positions w.r.t. the actual in (x, y) .

Figure 7.13: Results from 29 assembly trials for the square peg-in-hole assembly: Estimated hole orientations w.r.t. the actual (θ_z).

Lock-Key Assembly Results

For the lock-key problem, the initial lock-position can be anywhere within a circle of radius 10 mm, and our goal is to localize the lock-position to within a circle of radius 2 mm—which is required for assembly (for further details see Section 2.3). There is also uncertainty in the vertical (z) position of the lock. After the uncertainty is resolved to within a 2 mm region in (x, y) , we can attempt insertion of the key into the lock using a compliant strategy that simply attempts to press the key into the lock, and permits the key to be moved horizontally by the contact forces generated. This strategy is successful because around the key-hole there is a spherical depression (of approximately 2 mm radius) designed to guide the key into the key-hole. The number of iterations after which localization was forcibly terminated was set to 8. The contact surface map for the lock-key assembly only had 5256 configurations recorded. The algorithm employed 2000 particles. We conducted 50 assembly trials for the lock-key problem, out of which 45 assemblies were *successful*, meaning the key was able to insert into the lock. Figure 7.14 shows the error in the estimated position of the lock for 47 trials (estimates from 3 trials are off the plot). Resolving the position uncertainty from a circle of radius 10 mm to a circle of radius 2 mm corresponds to an uncertainty resolution of 96%.

Figures 7.15 and 7.16 shows the salient moments of a trial run of the lock-key assembly.

Pentagonal Peg-in-Hole Assembly Results

For the pentagonal peg-in-hole assembly problem, the initial hole-position can be anywhere within a circle of radius 15 mm, and the uncertainty in hole-orientation is unbounded (effectively, in the range $[0, \frac{2\pi}{5}]$ rad considering the symmetry of the pentagon). Our goal is to localize the hole-position to within

Figure 7.14: Estimated positions of the lock w.r.t. the actual for 47 trials of the lock-key assembly.

a circle of radius 2 mm of the actual hole-position, and the hole-orientation to within 0.126 rad of the actual hole-orientation.

The initial trials conducted on the pentagonal peg-in-hole assembly were moderately successful. All the trials had to be forcibly terminated after 7 iterations, and an estimate of the hole configuration was computed from the weighted average of the particles. Compliant assembly was attempted using this estimate, resulting in successful assemblies about half the time. However, the assembly was not tuned and the number of trials conducted were not statistically significant to report results. Further tuning of the parameters used for the particle filter employed in the assembly trials should improve the results.

(a)

(b)

(c)

(d)

Figure 7.15: The updated belief sets after the first and fourth observations for the lock-key assembly. (a,c) Key contacting lock-surface. (b,d) Particles in the belief set.

(a)

(b)

(c)

Figure 7.16: (a) Key contacting the lock-surface. (b) The updated belief set after the eighth observation for the lock-key assembly. Successive observations shrink the belief set. After eight iterations, lock-position is estimated from the weighted average of the particles. (c) Assembly accomplished by compliant phase using estimated lock-position.

7.2.3 Active Localization

Active localization involves finding the action (in our case, peg move) that most efficiently resolves the current uncertainty. In general, we would consider the uncertainty-reducing potential of all possible actions, and select one. However, in our problem, at every iteration we can choose a peg move to any point within the peg motion bounds, resulting in a potentially prohibitive number of possible actions. We can either discretize this continuous space sufficiently to produce a small set of possible actions, or choose a fixed number of random actions. We implemented active localization in computer simulations and on the actual circular and square peg-in-hole assemblies using the latter approach.

Figure 7.17 shows the cumulative distribution of the number of trials that successfully end after k iterations for 1000 simulated trials each using random, fixed,⁵ and active localization on the circular peg-in-hole assembly. Comparing the three plots, it is evident that with active localization the average number of iterations required for success is much smaller than with random and fixed localization. Computer simulation trials for the square peg-in-hole assembly yield similar results—with active localization there is a 20% drop in the number of iterations required for successful localization. Turning to the actual assemblies, for the circular peg-in-hole, the hole-position is sufficiently localized for compliant assembly after just 3 iterations using either fixed or active strategies. However, using active localization, the residual uncertainty (area in (x, y)) after three iterations is almost half the residual uncertainty with fixed localization, which is a significant improvement. For the square peg-in-hole assembly, localization is terminated after 6 iterations. Figure 7.18 shows the configuration uncertainty volume (in (x, y, θ)) after each iteration (averaged over 10 trials) using fixed and active localization, respectively, for

⁵Four fixed moves followed by random moves.

the square peg-in-hole assembly. Here, the first move for active localization is pre-determined since the initial global uncertainty is fixed. From the two plots we can see that actively choosing the peg move for the second and third iterations results in a large advantage in terms of uncertainty-reduction compared to fixed moves. We can also see from the plots that the rate of localization drops with increasing localization resolution.

Figure 7.17: Cumulative distribution of the number of trials ending successfully after k iterations of the circular peg-in-hole assembly.

7.3 Conclusions

Our primary goal in implementing the particle filtering algorithm was to execute our localization strategy using coarsely-sampled maps, and to make it robust to errors in observation and actuation. Among the many probabilistic formulations available [45, 27] (described in Section 1.3.3), the particle

Figure 7.18: Average configuration uncertainty after each iteration, averaged over 10 assembly trials for the square peg-in-hole assembly.

filtering algorithm specifically allows us to track multiple possible solutions, which abound in our problem.

We first tested the particle filter implementation on computer simulations of both circular and square peg-in-hole assemblies to evaluate the effect of significant parameters on the rate of convergence and success rate. These parameter values were then used to implement actual robotic assemblies of the circular and square peg-in-holes. For robotic assemblies of the circular peg-in-hole, we had 100% success in 30 assembly trials. The uncertainty in the area of hole-location was reduced by more than 97%. Even with the three-dimensional uncertainty (x, y, θ_z) of the square peg-in-hole assembly, the particle filtering implementation had good success. 90% of the assembly trials were successful, with an uncertainty reduction of 96% in the (x, y) -uncertainty and more than 88% in orientation uncertainty. The implementa-

tion was also tested on the lock-key assembly, with three-dimensional (x, y, z) uncertainty in the lock-position. This assembly was especially challenging, because the variation in the contact surface along the contact axis (z) is minimal (within 2 mm) for a major portion of the map. Moreover, a coarsely-sampled map, with only 5256 configurations, was employed. Despite these challenges, 90% of the assembly trials conducted were successful—reducing the uncertainty by 96%. Finally, the pentagonal peg-in-hole assembly was implemented. The initial trials were moderately successful with half the trials ending in successful assembly. Further tuning of the particle filtering parameters should improve these results.

Chapter 8

Conclusions and Future Work

8.1 Summary

In this dissertation, we have dealt with the problem of robotic peg-in-hole assemblies with position uncertainty. The research described here was motivated by real-world transmission assemblies described in Chapter 1. These assemblies belong to a larger class of robotic assemblies, where the relative position uncertainty of the mating parts exceeds assembly clearance. Hence, the automated assembly has to rely on search strategies, and the assembly task becomes a problem of motion planning under uncertainty, with or without the use of feedback. Our solution is to *localize the misalignment of mating parts by exploring the peg-hole contact configuration space and matching observations to a pre-acquired map of the contact C-space*.

The ideas presented in this dissertation were validated with mathematical analyses, computer simulations, and actual robotic assemblies. Among the various assemblies successfully implemented were the key-lock plus the circular, square, and pentagonal peg-in-hole assemblies.

8.2 Conclusions

In Chapter 3, we started by transforming the search space for peg-hole alignment to configuration space, which facilitated the subsequent analysis for **exhaustive coverage** of the search space. We established conditions for exhaustive coverage of the continuous search space with a finite set of points, based on the assembly clearance. We presented various search paths that fulfill these conditions, and discussed their merits in terms of implementation using a robot. Moving away from these *exhaustive* search strategies, we also introduced *intelligent* localization strategies based on peg-hole contact position measurements, including the **precession** and **sweep strategies**. The precession strategy involves precessing a tilted peg while maintaining contact with the hole. The height of the peg is recorded during the precession, and this record is then used to compute the hole position relative to the peg. For the sweep strategy, the hole-configuration relative to the sweep can be computed from the recorded peg-heights as the peg is tilted and swept across the hole while maintaining contact with it. Drawing inspiration from these preliminary strategies, we developed the **general localization strategy** based on matching individual peg-hole contact measurements to a map of the contact C-space. The strategy can work with both analytical and sampled maps of the contact hyper-surface.

In Chapter 4, we gave a detailed description of *localization using analytical maps* of the contact hyper-surface. For assemblies with simple geometrical shapes, e.g., circular and square peg-in-hole assemblies, it is relatively easy to generate analytical maps of the contact hyper-surface with restricted uncertainty dimensions. For the circular peg-in-hole, the hole-configuration uncertainty was restricted to the (x, y) plane, and hence only the space of (x, y, z) configurations of the peg w.r.t. the hole needed to be mapped. For the square peg-in-hole, the hole-configuration uncertainty was restricted to the (x, y)

plane and rotations about z . This reduced the contact hyper-surface to four dimensions in (x, y, z, θ_z) . For both the circular and square peg-in-holes, tilting the peg w.r.t. a horizontal axis produces a much more informative contact hyper-surface than for matching relative peg-hole orientations. Once the analytical maps were generated, they were used to implement the general localization strategy given in Algorithm 3.1 (see Figure 3.16). Successful applications on circular and square peg-in-hole assemblies were presented to validate the strategy.

Creating an analytical map for the peg-hole contact hyper-surface requires painstaking effort to generate equations of three-dimensional volumetric intersections, which we would like to avoid. On the other hand, the contact hyper-surface map can be easily sampled using either a robot or a CAD model. This sampling procedure was described in detail in Chapter 5. However, a sampled map does not provide a complete representation of the contact C-space. To overcome this problem, we introduced the concepts of **assembly sufficiency**, **goal region**, and **approximate localization**. A new algorithm to implement *approximate localization* using error bounds was presented in Algorithm 5.3 (see Figure 5.9). The new algorithm was validated with mathematical analysis and computer simulations of the circular and square peg-in-hole assemblies. Later, this algorithm was modified to make it more robust to observation errors inherent in physical assemblies. This modified algorithm was successfully tested on simulations mimicking real assemblies.

However, with increasing dimensionality of the assembly uncertainty and small assembly clearances, the computational load becomes large and uneven over the localization period. We introduced the **cell approach** algorithm in Chapter 6, which implements the localization strategy in stages of increasing resolution to distribute the computational load more evenly. In the cell approach, given in Algorithm 6.1 (see Figure 6.9), mapped configurations

are grouped into cells, and localization is carried out using these cells instead of the individual configurations. Because the number of cells is smaller than the total number of mapped configurations, the initial computational load is reduced. As localization proceeds and the uncertainty is reduced to a small number of cells, the computational requirement decreases. At this time, the remaining cells are decomposed into smaller cells, thus increasing the resolution of localization while maintaining a level computational load. We experimented with several different cell sizes and compared their respective advantages and shortcomings. In general, with larger cells, computation time per iteration is reduced, but the failure rate is higher, and localization resolution is poor. On the other hand, with smaller cells, each iteration has a higher computational load, especially in the early stages with global initial uncertainty. However, as the smaller cells approximate the surface better, the failure rate is lower and the resolution of localization is improved.

The implementation of our localization strategy requires the matching of each observation to the map to progressively reduce the uncertainty. In an error-free, simulated assembly, deterministic matching using either an analytical map or a sampled map can guarantee uncertainty resolution up to any required degree of accuracy. However, to make the algorithm robust in the presence of errors in measurement and actuation associated with an actual robotic assembly, we had to implement the Modified Algorithm given in Section 5.7.2 and the cell approach given in Algorithm 6.1 (see Figure 6.9). These algorithms increase their matching-tolerance bounds based on the worst-case error to accommodate erroneous actuation and measurement. However, the conservative matching tolerance leads to lower rates of convergence and success.

To make the localization strategy more robust, the application of **particle filtering for robotic assemblies** with position uncertainty was pioneered in this dissertation. With particle filters, instead of including all the possi-

ble configuration estimates in the belief set, we maintain a fixed number of the best estimates (called particles). The biggest advantage of using particle filters is that the number of particles can be adjusted depending on the computational resources available. Obviously, localization cannot be guaranteed using particle filtering. However, we were able to obtain excellent results with the actual physical assemblies of the circular and square peg-in-hole assemblies, and the lock-key assembly. We had 100% success with all 30 assembly trials of the circular peg-in-hole with a reduction in uncertainty of more than 97%. For the square peg-in-hole, even with three-dimensional uncertainty in (x, y, θ_z) , we had 97% success in 30 assembly trials. For the lock-key assembly, 43 out of the 50 assembly trials were successful, with an average reduction in uncertainty of 96% for the successful trials. We also implemented the pentagonal peg-in-hole assembly with promising preliminary results. Furthermore, we implemented active localization, which involves computing the best move possible at every iteration, i.e., the one that leads to the largest reduction in current uncertainty. The results show that active localization provides a distinct advantage in terms of uncertainty-reduction compared to localization with fixed or random moves.

8.3 Recommendations

There were four major algorithms described in this dissertation: Analytical Map (Algorithm 3.1), Sampled Map (Algorithm 5.3), Cell Approach (Algorithm 6.1), and Particle Filtering (Algorithm 7.1). Below, we provide recommendations for selecting the right algorithm among these, based on the type of assembly implementation.

1. If an assembly with small actuation and observation errors needs to be implemented, then the original algorithms using analytical or sampled maps can be selected. However, for both these algorithms, a lot of

computational power and memory will be required. Further:

- (a) If the assembly has a simple geometrical shape, then use the Analytical Map algorithm.
 - (b) Otherwise, a robot or a CAD model should be used to sample a map of the contact hyper-surface, and the Sampled Map algorithm should be implemented.
2. If assembly success is critical, then localize using the Cell Approach algorithm. The cell approach, being a worst-case algorithm, can accommodate errors from actuation, observation, and incomplete maps.
 3. If assembly time is the critical parameter and some assembly failures can be tolerated, then the Particle Filtering algorithm is the right choice. Also, particle filters can be adjusted to work with the available computational resources, and they are easy to implement.

The recommendations given above are tabulated in Table 8.1 below for quick reference. The merits of three algorithms: Sampled Map, Cell Approach, and Particle Filtering are compared w.r.t. computation and memory available for implementation, the inherent errors in observation and actuation for the assembly, and whether the critical parameter for the assembly is success or time efficiency. The symbols used in the table signify the following: the symbol “√” signifies a good fit, the symbol “—” signifies moderately good fit, and the symbol “×” signifies a bad fit.

8.4 Future Work

In this section, we describe new ideas and extensions for future work related to the research presented in this dissertation.

<i>Algorithm</i>	<i>Computation Available</i>		<i>Errors</i>		<i>Critical Parameter</i>	
	Large	Small	Large	Small	Success	Time
Sampled Map (5.3)	✓	×	×	✓	✓	×
Cell Approach (6.1)	✓	–	✓	✓	✓	–
Particle Filtering (7.1)	✓	✓	✓	✓	×	✓

Table 8.1: Recommendations for the localization algorithm to be used based on the requirements and parameters of the assembly to be implemented.

If analytical maps are available, the general localization strategy can be implemented for arbitrary peg-in-hole assemblies using Algorithm 3.1. This was accomplished for the circular and square assemblies, which have simple geometrical shapes. However, generating analytical maps of the contact C-space for more complicated shapes may require painstaking effort. One area of future work is the use of CAD models to produce the equations for the contact C-space, and perhaps even the translations and intersections of the belief sets during matching. However, in actual assemblies, actuation and measurement errors are introduced. To overcome these errors, an algorithm using analytical maps would have to shift from exact matching to approximate matching using tolerance bounds. One possible solution would be to dilate the contact hyper-surface into a hyper-volume to account for the maximum possible error.

The implementation of the precession and sweep strategies (described in Sections 3.2.1 and 3.2.2, respectively), could be extended to arbitrary polygonal peg-in-hole assemblies. The strategies could also be tested on the circular, square, and pentagonal peg-in-hole assemblies with higher-dimensional uncertainty, e.g., allowing uncertainty in the orientation of the hole-surface in the (θ_x, θ_y) -space.

In the cell approach, described in Chapter 6, we computed and recorded cell-sequences to localize both the location and orientation of the probe. In future work, it would be beneficial to find a way to easily compute the relative

probe-surface *transformation sets* corresponding to each cell of possible probe occupation. This would make the process of matching a probe move to cell-pairs much easier than our current matching technique.

In the initial implementation of the cell approach, the length of the probe-moves was fixed, i.e., after moving a certain distance the controller would sample an exploration point and obtain an observation. For most surfaces, there is a range of $(\Delta x, \Delta y, \Delta z)$ observations that occurs infrequently, and which can be pre-computed. These are **profitable observations**, because they would efficiently resolve uncertainty. Knowing this range of observations, the controller can keep sampling exploration points but not make a comparison (compute the matching cell-pairs) until it recognizes a profitable observation. This is similar to the concept of *active localization*, which was discussed in detail in Section 7.2.3.

In the future, extending the implementation of the cell approach to localize on contact hyper-surfaces for actual robotic assemblies would pose an interesting challenge. Accommodating the observation and actuation errors inherent in robotic assemblies using the cell approach algorithm would require dilation of cell bounds along z , based on the maximum error possible. Although, the convergence rate for localization would suffer because of the conservative cell bounds, the algorithm would be robust to errors encountered during assembly.

Another source of error in actual assemblies arises from the tolerancing of manufactured parts. This error can lead to a reduction in the assembly clearance for some parts. Within the cell approach, we can overcome these errors by forcing neighboring cells to overlap.

The particle filtering algorithm, described in Chapter 7, was implemented on the circular, square, and pentagonal peg-in-hole assemblies and the lock-key assembly with restricted uncertainty regions. In future work, it would be interesting to test the implementation by extending the uncertainty to

the full six-dimensional space. A much coarser sampling would be required to maintain (approximately) the current size of the map. Moreover, the number of solutions possible for any given sequence of observations would also be much greater. This should prove to be an interesting challenge for the particle filtering algorithm.

Particle filtering is a probabilistic scheme, where the particles are sampled in a random fashion. In future work, it would be interesting to implement a deterministic version of particle filtering, where the sampling is carried out using some deterministic strategy [3, 29].

Bibliography

- [1] H. Asada and J.-J.E. Slotine. *Robot Analysis and Control*. John Wiley & Sons, Inc., New York, NY, 1992.
- [2] C.E. Birkhimer, M.E. Dohring, R. Hebbar, and W.S. Newman. Human inspiration for blind robotic assemblies using force feedback. In *Proceedings of the Twelfth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, May 2003.
- [3] E. Bolviken and G. Storvik. Deterministic and stochastic particle filters in state space models. In A. Doucet, J.F.G. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2000.
- [4] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence*, Portland, OR, August 1996.
- [5] M.E. Caine, T. Lozano-Perez, and W.P. Seering. Assembly strategies for chamferless parts. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, May 1989.
- [6] S. Chhatpar. *Experiments in Force-Guided Robotic Assembly*. Master's thesis, Case Western Reserve University, 1999.
- [7] S.R. Chhatpar and M.S. Branicky. A hybrid systems approach to force-guided robotic assemblies. In *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, Porto, Portugal, July 1999.
- [8] S.R. Chhatpar and M.S. Branicky. Search strategies for peg-in-hole assemblies with position uncertainty. In *Proceedings of the IEEE/RSJ*

International Conference on Intelligent Robots and Systems, Maui, HI, October 2001.

- [9] S.R. Chhatpar and M.S. Branicky. Localization for robotic assemblies with position uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, October 2003.
- [10] S.R. Chhatpar and M.S. Branicky. Localization for robotic assemblies with position uncertainty. WWW Link. <http://dora.case.edu/msb/pubs/icra06submit.mp4>, October 2005.
- [11] S.R. Chhatpar and M.S. Branicky. Particle filtering for localization in robotic assemblies with position uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Canada, August 2005.
- [12] G. Chrystal. On the problem to construct the minimum circle enclosing n given points in the plane. *Proceedings of the Edinburgh Mathematical Society*, 3:30–33, 1885.
- [13] Y.X. Chu, J.B. Gou, B. Kang, K.T. Woo, and Z.X. Li. Performance analysis of localization algorithms. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.
- [14] A. Doucet, J.F.G. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2000.
- [15] D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.

- [16] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In A. Doucet, J.F.G. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2000.
- [17] M. Gavrilov, P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric pattern matching: A performance study. In *Proceedings of the Fifteenth Annual ACM Symposium on Computational Geometry*, Miami Beach, FL, June 1999.
- [18] K.Y. Goldberg, M.T. Mason, and A. Requisha. Geometric uncertainty in motion planning: Summary report and bibliography. Technical Report IRIS TR 297, Institute for Robotics and Intelligent Systems, 1992.
- [19] D. Gravel. Forced assembly. *Assembly Magazine*, January 2002.
- [20] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, British Columbia, Canada, October 1998.
- [21] J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.
- [22] K. Hamada and Y. Hori. Octree based approach to real-time collision-free path planning for robot manipulator. In *Proceedings of the IEEE 4th International Workshop on Advanced Motion Control*, pages 705–710, Tsu-City, Japan, March 1996.
- [23] V. Hayward. Fast collision detection scheme by recursive decomposition of a manipulator workspace. In *Proceedings of the IEEE International*

Conference on Robotics and Automation, pages 1044–1049, Minneapolis, MN, April 1996.

- [24] R. Hebbar, S.R. Chhatpar, C. Birkhimer, M.S. Branicky, W.S. Newman, S. Charles, M. Stuart, R. Stoughton, A. Reiman, A. Korzhenyak, and D. Kozlowski. Flexible robotic assembly for power-train components (FRAPA). In *Video Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, D.C., May 2002.
- [25] R. Hebbar, D.M. Morris, S.R. Chhatpar, M.S. Branicky, and W.S. Newman. Force-responsive robotic assembly of powertrain components. In *Video Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, South Korea, May 2001.
- [26] H. Inoue. Force feedback in precise assembly tasks. *Artificial Intelligence: An MIT Perspective*, 2:219–241, 1981.
- [27] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation*, Stockholm, Sweden, August 1999.
- [28] Y.-B. Jia and M. Erdmann. The complexity of sensing by point sampling. In *Workshop on the Algorithmic Foundations of Robotics*, San Francisco, CA, February 1994.
- [29] S.M. LaValle, M.S. Branicky, and Lin. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8):673–692, 2004.
- [30] T. Lozano-Perez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.

- [31] The MathWorks, Inc. Matlab version 7.0.1. Natick, Massachusetts: The MathWorks Inc., 2004.
- [32] N. Megiddo. Linear-time algorithms for linear programming in r^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.
- [33] W.S. Newman. Stability and performance limits of interaction controllers. *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, 114(4):563–570, 1992.
- [34] W.S. Newman, M.S. Branicky, S.R. Chhatpar, L. Huang, and H. Zhang. Impedance based assembly. In *Video Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, May 1999.
- [35] W.S. Newman, M.S. Branicky, H.A. Podgurski, S.R. Chhatpar, L. Huang, J. Swaminathan, and H. Zhang. Force-responsive robotic assembly of transmission components. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, May 1999.
- [36] M.A. Peshkin. Programmed compliance for error corrective assembly. *IEEE Transactions on Robotics and Automation*, 6(4):473–482, 1990.
- [37] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, 2nd Edition*. Cambridge University Press, 1992.
- [38] J.E. Schimmels. Method and apparatus for assembling rigid parts. *U.S. Patent No. 6,408,531*, 2002.
- [39] J.M. Schimmels and M.A. Peshkin. Admittance matrix design for force-guided assembly. *IEEE Transactions on Robotics and Automation*, 8(2):213–227, 1992.

- [40] S. Simunovic. *An Information Approach to Parts Mating*. PhD thesis, M.I.T., 1979.
- [41] R.S. Stoughton and T. Arai. A modified Stewart platform manipulator with improved dexterity. *IEEE Transactions on Robotics and Automation*, 9(2):166–172, 1993.
- [42] D.R. Strip. Insertions using geometric analysis and hybrid force-position control: Method and analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April 1988.
- [43] D.R. Strip. A passive mechanism for insertion of convex pegs. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, May 1989.
- [44] J. Swaminathan. *An Object-Oriented Application Framework for Peg-in-Maze Assemblies*. Master’s thesis, Case Western Reserve University, 2000.
- [45] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.
- [46] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI*, Seattle, WA, August 2001.
- [47] Y. Wang, W.S. Newman, and R.S. Stoughton. Workspace analysis of the ParaDex robot—a novel, closed-chain, kinematically-redundant manipulator. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.
- [48] D.E. Whitney. Force feedback control of manipulator fine motions. *ASME Journal of Dynamic Systems, Measurement, and Control*, 99(2):91–97, 1977.

- [49] D.E. Whitney. Quasi-static assembly of compliantly supported rigid parts. *ASME Journal of Dynamic Systems, Measurement, and Control*, 104:65–77, 1982.