AN APPROACH TO REDUCING PARAMETER UNCERTAINTY FOR ROBOTIC SURFACE ASSEMBLY TASKS

by

Amar Sarić

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2014

Approved by:

Dr. Jing Xiao

Dr. Srinivas Akella

Dr. Zbigniew W. Raś

Dr. Jianping Fan

Dr. Jaya Bishwal

ABSTRACT

AMAR SARIĆ. An Approach to Reducing Parameter Uncertainty for Robotic Surface Assembly Tasks. (Under the direction of DR. JING XIAO)

In contrast to hard automation, which relies on the precise knowledge of all parameters and special-purpose machinery, the goal of flexible assembly is to overcome the inherent uncertainty in the location of parts. The main result of this dissertation is that, for rigid, non-deformable objects, more accurate estimates of parameters, which describe their position and orientation in Cartesian space, can be obtained via active part interaction and estimation using numerical methods. If the objects have large polyhedral or convex features, the parameter estimation problem can be recast in terms of fitting the collected empirical data to a suitable geometrical model. The planning and execution steps are treated as conceptually separate from the estimation. Additionally, an algorithm for automatic conversion of a compliant path from Cartesian to the joint space of a general-purpose 7 DOF robotic arm is described. This allows for the assembly strategies to be planned in terms of objects' topological features in the task frame. A 'back-drivable' Barrett WAM robotic arm without a force sensor was used in all experiments, and approximate compliant motion was achieved by relying on torque limits and impedance. Consequently, the primary focus is on planning, control, and assembly without force sensing. The underlying concepts, however, are more general and could be extended to incorporate force feedback. The problem of initially grasping an object with the gripper is outside the scope of this work, and it is assumed that one of the parts is rigidly attached to the end-effector.

ACKNOWLEDGEMENTS

First of all, I would like to sincerely thank my advisor Prof. Jing Xiao for her guidance, encouragement and support, and to express my gratitude to Prof. Srinivas Akella, Prof. Zbigniew Raś, Prof. Jianping Fan, and Prof. Jaya Bishwal for serving on the committee. Others, as well, provided invaluable help while I was working on this dissertation. I am indebted to the General Motors Global R&D Center and Dr. Jane Shi for the Barrett WAM robotic arm, Bill Lindsey from the William States Lee College of Engineering for making the parts that were used in the experiments, and Doralyn Bradley who helped me with the administrative side of things. Also, it would not have been possible without my friends Vincent Saelzler, Sivagamasundari Veerappan, Greg Chunn, Barbara and Wolfgang Helbig, and Dr. Irmgard Bock who, in one way or another, have all played an essential role. Finally, I am truly grateful to my mother Aida, my sister Amra Popržanović, and my late father Elmedin.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Up to the present day, successful application of robotics to assembly tasks in the industry has mainly relied on specific strategies and tools for a given task in a controlled environment. However, moving away from hard automation holds the promise of being able to cost-effectively handle smaller batches and to accommodate for more difficult components, as well as to reduce the cycle times needed to respond to product changes. Flexible assembly is still an open research topic, see for instance Hamner et al. [21]. At the crux of the problem are the initial positions and orientations of the parts, and the additional uncertainty resulting from previous operations such as grasping. Errors in the sensory data and any motion uncertainty introduced by the manipulator mechanics or the control law can for the most part be neglected. As a result, it becomes necessary to align features of typically rigid, non-deformable parts with only approximate knowledge of their locations prior to the final assembly.

1.1 Problem Motivation

Since the parts have presumably been manufactured according to specifications, their geometric features will be known for all practical purposes. In fact, the relative error in positioning of the objects typically dominates other sources of uncertainty. In order to overcome it, compliant motion [48, 49], i.e. motion constrained by contacts between the parts, has been widely used in the literature. It is an intuitive and

effective concept that leverages the geometry of the parts to guide the assembly process and reduce the error. General approaches based on active compliant motion have also been proposed. However, there are still difficulties and open questions.

First of all, strategies based on compliant motion often ignore the constraints of the manipulator in the planning stage, while focusing solely on the parts. Note that these two simultaneous sets of constraints will be in conflict in the general case.

Furthermore, compliant motion control is often based on force sensing and formulated in the task space. It is common in the literature – see for instance Tang and Xiao [70], Pan and Schimmel [56], Hirai [23] et al. – to use higher-level properties and states to represent the contact information. Therefore, different contact states typically require separate controllers to realize motion compliant to each contact state. This in turn requires accurate detection of contact state changes, which due to uncertainty is a non-trivial problem.

Finally, parameter estimation is important for any strategy that aims at reducing uncertainty. From the point of view of implementation, simultaneous parameter estimation and identification of contact states is a complex process involving on-line exploration and optimization, which is difficult to complete quickly enough for realistic assembly in real time. Most researchers, e.g. Lefebvre et al. [44, 45], use Bayesian theory and probabilistic filters [71] to tackle this problem, which allows for the estimation of parameters based on the sensor data. Subsequently, the improved values are passed on to the motion planner. The downside is that on-line replanning is required *during* the task execution, and that the problem formulation can become intractable even for moderately sized problems.

1.2 Previous Work

Tasks such as polishing, grinding or deburring have naturally led to the concept of compliant motion, with the movement of the end-effector being constrained in at least one dimension based on the interaction between parts. There exists a large body of research on force control in connection with compliant motion. This includes the hybrid control, where a direction normal to the contact surface is force controlled, while other directions are position controlled. In general, if a degree of freedom (DOF) is specified in terms of position, it cannot be specified in terms of the force at the same time. Here, it is essential to set a saturation threshold for the force exerted by the end-effector, so that a 6-D force/torque sensor is required. Without a force sensor, impedance control can be used – see Hogan [26], but also [7] and [63] – as well as to some extent passive compliance [15].

Impedance control, passive compliance devices, and hybrid force control all allow for expressing the task in terms of the coordinates of the end-effector, the tool or the held object. This is commonly referred to as the *task frame* [1, 11, 73]. Inaccuracies will either lead to the loss of contact or to an increase in the applied force, i.e. an attempt to push the objects into each other. Recently, another related approach for control without a force sensor was introduced [68] using an ABB FRIDA robot. It is based on estimating the force at the end-effector and reducing the integral gain of the PID controller empirically to achieve compliance. Here, the problem can be mapped to the joint space of the manipulator by means of the Jacobian transpose and Jacobian pseudo-inverse methods, see [26] and [78]. If the trajectory would nominally

result in an overlap between objects, the stiffness of the end-effector is used to limit the applied force.

Since compliant tasks require contact, they can fail even for small errors in positioning of the parts when the trajectory is generated solely based on nominal planning. In fact, two objects can only be brought into contact by limiting the applied force, e.g. using either impedance or force control. However, it is a common strategy to have the task frame expressing the position and direction of impedance/force control change with every contact state, e.g. Meeussen et al. [52]. As a consequence, for compliant motion going through a sequence of contact states, separate impedance controllers might be needed, leading to the difficulty of accurately detecting the contact state transitions in order to switch to the next controller. Moreover, since each joint of the manipulator is usually controlled by its own PID controller, separate coefficient values need to be determined empirically for each of them, so that stable behavior can be obtained based on the stiffness settings in the task space. Also, for instance, Meeussen et al. [52] introduce a general approach to convert a compliant path to a force based task specification. However, while the task frame formalism is intuitive, it requires the contact states to be known explicitly and thus motivates a framework to facilitate task frame specifications for complex contact states [12]. In general, multiple contact state transitions have to be traversed before even a relatively simple assembly task, e.g. peg-in-hole, can be completed successfully.

In order to mate two arbitrary, non-flat surfaces, it is even more important to take advantage of contact state transitions in planning the assembly motion to reduce uncertainty. Xiao and Ji [74] describe a general method for creating a contact state

graph between two parts based on the topological and geometric information. Subsequently, the assembly motion can be planned using a two-step approach by first searching the contact state graph to automatically find a suitable sequence of contact state transitions, and then constructing a concrete path of configurations in the Cartesian space for motion compliant to a given contact state and the transition to the next one, see Ji and Xiao [74]. This work was later extended to deal with parts that have general curved surfaces [70]. Other researchers have also designed planners for compliant motion in terms of contact state transitions between two parts, see for example [43], [51], [24] and [61].

Assembly strategies that rely on compliant motion have typically focused on paths in the task space. Little addressed is the issue of ensuring that a compliant path not only satisfies contact constraints but also the manipulator constraints when converted to a joint space path for the manipulator to execute. Here, Stilman [66, 67] deserves to be mentioned, since he considers constraints in the joint space. Likewise, the previous work on compliant control is often based on force sensing with the underlying assumption that a compliant path described in the task space is executable by a robotic manipulator. However, a compliant path planner focused on the relative contact motion between parts is not guaranteed to produce a compliant path that will have inverse kinematic solutions for a particular robotic manipulator [53] or result in a singularity free trajectory. In the literature, researchers have proposed ways of avoiding singularities. Arguably, the most important strategy for practical purposes has been to design control schemes that maximize manipulability by taking advantage of the null space of the manipulator [10, 37]. Also, it is of course possible to search

for paths that do not contain singularities [5]. This relies on discretizing a path and using a minimum threshold for manipulability to classify singular vs. non-singular configurations. Unfortunately, such a threshold is difficult to decide beforehand. Furthermore, work addressing singularities is usually not concerned with compliant motion and the singularity-free solution might deviate from the compliant path.

The traversal of the contact state graph becomes more difficult with increasingly complex geometric features of the objects, so that online replanning plays an important role under uncertainty. Researchers, e.g. Lefebvre et al. [44, 45], Debus et al. [13] and Gadeyne et al. [16], have commonly used Bayesian reasoning in order to infer the most likely current contact state. Consequently, priors and conditional probabilities are required for every contact state transition. However, one of the main disadvantages of Bayesian filters is that the priors are in general difficult to know or estimate beforehand. Essentially, this leaves the model of the task only partially parametrized. Moreover, assembly tasks are inherently different from position tracking and localization for which Bayesian filters, e.g. the Kalman filter [34], are an invaluable tool. Furthermore, it could be argued that, in a sense, Bayesian reasoning always results in only probabilistic convergence under a set of assumptions. Finally, although the adaptive nature of the Bayesian approach makes it preferable to off-line planning and replanning strategies with fixed assumptions, it is incremental in the sense that the reduction of uncertainty relies on the previous estimation at time t_n for the new estimate at time t_{n+1}. On the one hand, this allows for using the new data as soon as it becomes available. However, other than time constraints imposed by continuous updates, there is little reason not to use all of the available empirical contact data

collected at $t_1, ..., t_n$ to calculate the new estimates.

Also, fuzzy rules, neural networks, and other types of classifiers are sometimes trained to recognize the different contact states of an assembly sequence, see for instance Asada [2], and Hara and Yokogawa [22].

Another possible approach is to guarantee the convergence for a specific task under some fixed set of assumptions based on a carefully constructed order of moves without updating the parameter estimates, e.g. Lozano-Perez, Mason and Taylor [49] and Xiao and Volz [75].

1.3 The Scope of This Dissertation

Our goal will be to address the following questions: How can manipulator constraints be incorporated to obtain executable compliant paths for assembly tasks? Can compliant control be successful even without accurate contact state detection and force sensing? How should the estimation of uncertain parameters be handled to result in successful assembly? Can we harness the existing optimization methods to improve the initial parameter estimates? Here, part interaction itself provides additional information on the relative position and orientation of the parts, which can be used for error correction during execution, see for instance Peshkin [57]. A suitable geometrical model of the task is required to provide the high-level topological information. However, the positional and force data collected during execution can only provide enough additional information for the occurrence of contact to be inferred or detected – the precise point of contact between the objects is not known in general. Therefore, methods and techniques are needed that estimate the correct locations of

the parts based on the contact information.

In this work, one of the parts will always be assumed to be static and the other one rigidly attached to the end-effector of the robot, which implies that it has already been successfully grasped, with at least one of the two objects subject to initial uncertainties in position and orientation. The objective will be to obtain additional information by performing exploratory compliant motion, see [7], [46] and [76], between the two objects along a set of predetermined features. From the data that is collected during exploratory moves, parameters can be estimated more accurately and, as a result, the uncertainty will be reduced. We will further assume that an initial, imperfect estimate exists. It can be known beforehand or obtained using other means, for example, computer vision.

Here, the focus will be on objects with a set of suitable features, such as planes and convex regions, which we will call *regular*. To explore the position and orientation of objects with respect to each other, we will assume that a parameterized geometrical model of such a feature can be obtained in closed form. Other, more general features, such as curved surfaces, are accessible numerically through discretization and simulation. Note, that this does not pose restrictions on the shape of the remaining sections of the parts. Finally, an implicit assumption will be that the original parameter estimates are accurate enough for the features to be brought into contact reliably in terms of the type of contact that is formed between the objects.

Description of the objects using a set of constraints as well as the resulting minimization problem are task dependent. Also, the planning of exploratory moves will be done in an ad hoc manner, since it is difficult to devise an algorithm to reason about

objects in 3D space. Similarly, searching the contact state graph can be used to find a valid path but a heuristic function that takes into account which degrees of freedom in the location of parts can be determined using a specific contact state transition, is not readily available in general. It is left for future research. However, this problem is accessible to our intuition, if the planning is performed in the Cartesian space.

The presented material is organized as follows: Chapter 2 establishes the notation and reviews some well-known results related to coordinate transformations, manipulator kinematics, and the control strategies for achieving contact without force sensors. Also, the minimal translational distance is discussed, as it pertains to the distance and penetration depth and, therefore, also the contact formation between two objects. Chapter 3 discusses the planning of compliant paths using contact states and a 7 degrees of freedom (DOF) robotic arm without a force sensor. It further addresses the issue of resolving simultaneous manipulator and contact constraints in order to obtain executable compliant paths. Chapter 4 deals with the parameter uncertainty for poses[1] of objects and features, in a manner which decouples parameter estimation from the execution. First, it is shown how certain cases can be reduced to the problem of fitting points in Cartesian space to the model. Later, a numerical approach for convex polygonal meshes is proposed. Chapter 5 further illustrates and validates the approach based on experimental results. It is demonstrated that surface assembly tasks can be implemented in a robust way by using the presented techniques. Chapter 6 concludes the dissertation. The more tedious kinematics formulas for the 7-DOF Barrett WAM that was used in the experiments are in Appendix A.

[1] A pose includes both position and orientation.

1.4 Contributions

This work's primary objective lies in enabling more flexible assembly operations that are subject to uncertainty – a long-standing problem still lacking a general solution. The proposed approach takes advantage of a sequence of deliberately planned, exploratory, compliant moves to align the parts. Effectively, a feedback loop is introduced at the end of such a move to update the position and orientation of the parts by relying on the collected contact points. No force sensor is required; however, if it is available, force sensing is helpful for achieving compliant motion.

The estimation of errors in the positioning of the parts is recast as a data fitting problem, which obviates the need for modeling the priors and conditional probabilities that the existing approaches based on Bayesian reasoning require. For large polyhedral features, parameter estimates are obtained by expressing the contact in terms of a system of linear equations and using linear least squares. Moreover, a numerical estimation algorithm is given for contacts that involve convex features represented as polygonal meshes. Although the features used for estimation need to be regular, i.e. flat or convex, no additional assumptions are made about the rest of the objects' geometry – in particular, the contacting features in the assembly goal state.

Finally, in order to express compliant paths in the task space, a novel method for automatic conversion of such paths to joint trajectories is introduced that can reliably detect singularities along a path described by the end-effector of a 7-DOF robotic arm. It is necessitated by the fact that the existing literature considers singularities only in terms of manipulator geometry, not the validation of compliant paths.

CHAPTER 2: PRELIMINARIES

Before reviewing the relevant theory, it is important to decide on the notation. In the literature, there are two standard choices for specifying the position and orientation of objects in the three dimensional Cartesian space: quaternions, which go back to Hamilton, and homogeneous transformation matrices. The latter will be used exclusively in the remainder of the text. Such matrices consist of a rotational and a translational part and operate on vectors or, rather, column matrices.

Next, we will give an overview of the Denavit-Hartenberg notation for the kinematic chains of robotic manipulators and the forward and inverse kinematics of the Barrett WAM redundant 7 degrees of freedom (DOF) robotic arm [69, 60]. Subsequently, a control law that allows for limiting the computed joint torques of the WAM is discussed. It can be used to realize compliant motion and contact state transitions in the Cartesian space. Without a force sensor one can take advantage of the 'back-drivability' of the WAM. In addition to being simple, with this strategy there is no need to switch controllers whenever a new contact state is encountered, and no need to use separate controllers for guarded and compliant motion. An alternative way to achieve approximate compliance without a force sensor is the impedance control [26]. It operates in Cartesian space and is known to be asymptotically stable. Both control schemes are well suited for experimenting with robotic tasks that involve contact.

Finally, the fact that two objects are in contact is equivalent to both the distance and the overlap, i.e. penetration depth, between them being zero. This can be combined into a single value known as the minimal translational distance, see Cameron [8]. We will discuss how this value can be calculated for convex objects and features, which will provide us with a means to treat the general contacts between convex meshes later in the text.

2.1 General Translations and Rotations in the Cartesian Space

The translation of a vector $\vec{u}$ is accomplished simply by adding another vector to it, e.g. $\vec{u}+\vec{p}$. The formula for the rotation of a vector in 3 dimensions is, however, a little more involved. Consider the vector $\vec{u}$ to be rotated about the vector $\vec{r}$ in the positive direction – determining both the axis and the orientation – as depicted in Fig. 1 a). The problem is simplified considerably by noting that $\vec{u}_{proj}$, the component of $\vec{u}$ that is parallel to $\vec{r}$, does not get altered by this transformation. Hence, the problem is reduced to a rotation in the plane as depicted in Fig. 1 b). It is now immediately clear that

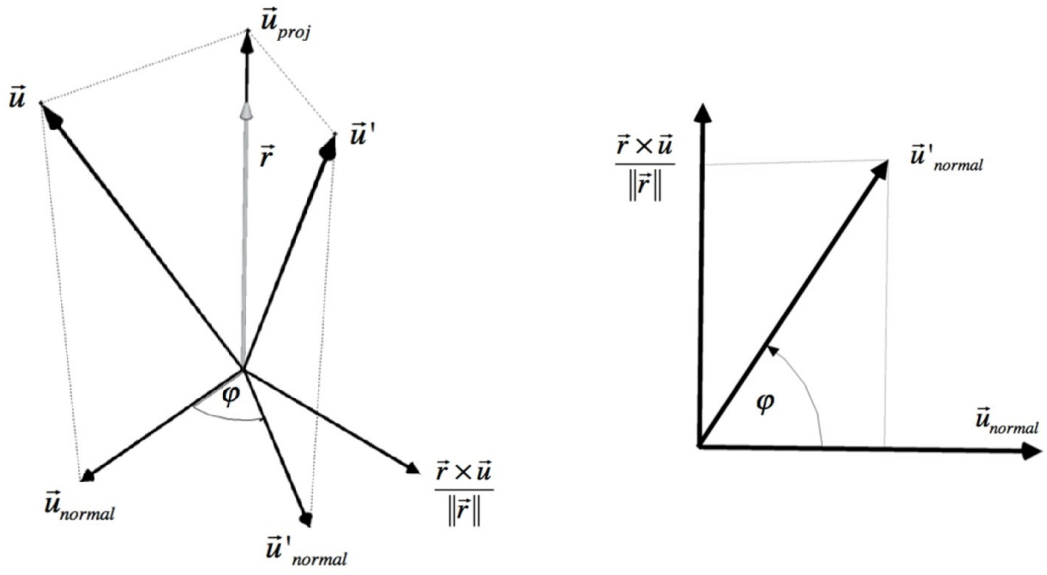
$$\vec{u}'_{normal} = \cos(\varphi)\vec{u}_{normal} + \sin(\varphi)\frac{\vec{r}\times\vec{u}}{\|\vec{r}\|} \tag{1}$$

must be true for any vectors $\vec{u}$ and $\vec{r}$. The components of the vector $\vec{u}$ are given by

$$\vec{u}_{proj} = \frac{\vec{u}\circ\vec{r}}{\|\vec{r}\|^2}\vec{r} \qquad \text{and} \qquad \vec{u}_{normal} = \vec{u} - \frac{\vec{u}\circ\vec{r}}{\|\vec{r}\|^2}\vec{r}. \tag{2}$$

From here we get

$$\vec{u}'_{normal} = \cos(\varphi)\left(\vec{u} - \frac{\vec{u}\circ\vec{r}}{\|\vec{r}\|^2}\vec{r}\right) + \sin(\varphi)\frac{\vec{r}\times\vec{u}}{\|\vec{r}\|} \tag{3}$$



(a) (b)

Figure 1: a) General rotation of a vector $\vec{u}$ about the axis $\vec{r}$ and onto the vector $\vec{u}'$. Only the component perpendicular to the vector $\vec{u}$ is affected. b) Projection to the plane that is normal to the rotation axis $\vec{r}$, with the rotation of the vector $\vec{u}_{normal}$ onto $\vec{u}'_{normal}$.

and, since the projected component of $\vec{u}$ is not affected by the rotation, we can calculate the final vector as

$$\vec{u}' = \vec{u}_{proj} + \vec{u}'_{normal} = \cos(\varphi)\vec{u} + (1-\cos(\varphi))\,\frac{\vec{u}\circ\vec{r}}{\|\vec{r}\|^2}\vec{r} + \sin(\varphi)\frac{\vec{r}\times\vec{u}}{\|\vec{r}\|}. \tag{4}$$

We will rewrite this result in terms of matrix operations by using the matrices u and r, instead of $\vec{u}$ and $\frac{\vec{r}}{\|\vec{r}\|}$, defined as $u = [u_1, u_2, u_3]^T$ and $r = \frac{1}{\|\vec{r}\|}[r_1, r_2, r_3]^T$ respectively. As $r^T u$ is a scalar value, it can be reinterpreted as a 1×1 matrix that is trivially commutative. Therefore, by changing the order of multiplication in the product $r^T u r$ to $r r^T u$, it follows that

$$u' = \left[\cos(\varphi)I + (1-\cos(\varphi))\,A(r) + \sin(\varphi)B(r)\right]u, \tag{5}$$

for matrices

$$A(r) = \begin{bmatrix} {r_1}^2 & r_1 r_2 & r_1 r_3 \\ r_1 r_2 & {r_2}^2 & r_2 r_3 \\ r_1 r_3 & r_2 r_3 & {r_3}^2 \end{bmatrix} \quad B(r) = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_1 & r_2 & 0 \end{bmatrix}.$$

Consequently, we have $u' = R(r, \varphi)u$, where $R(r, \varphi)$ defines a linear operator in u but, obviously, not in r or φ. After substitution, we obtain

$$R(r, \varphi) = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix}, \tag{6}$$

where

$$R_{1,1} = r_1^2 (1 - \cos(\phi)) + \cos(\phi)$$

$$R_{1,2} = r_1 r_2 (1 - \cos(\phi)) - r_3 \sin(\phi)$$

$$R_{1,3} = r_1 r_3 (1 - \cos(\phi)) + r_2 \sin(\phi)$$

$$R_{2,1} = r_2 r_1 (1 - \cos(\phi)) + r_3 \sin(\phi)$$

$$R_{2,2} = r_2^2 (1 - \cos(\phi)) + \cos(\phi)$$

$$R_{2,3} = r_2 r_3 (1 - \cos(\phi)) - r_1 \sin(\phi)$$

$$R_{3,1} = r_3 r_1 (1 - \cos(\phi)) - r_2 \sin(\phi)$$

$$R_{3,2} = r_3 r_2 (1 - \cos(\phi)) + r_1 \sin(\phi)$$

$$R_{3,3} = r_3^2 (1 - \cos(\phi)) + \cos(\phi)$$

and $\|r\|_2 = 1$. We will assume that rotation vectors are of unit length in the remainder of the text. One can further prove that $R(\vec{r}, \varphi)$ is orthogonal, i.e. $R(\vec{r}, -\varphi) = R(\vec{r}, \varphi)^{-1} = R(\vec{r}, \varphi)^T$. This makes finding the inverse of a rotation matrix trivial.

Conversely, given a rotation matrix $R(\vec{r}, \varphi)$ for some unknown $\vec{r}$ and φ, a measure of the alignment of the original vector $\vec{u}$ and the vector obtained after rotation $\vec{u'}$, can be expressed in terms of the $\cos(\varphi)$. For this purpose, one can use the sum of the elements on the diagonal that is known as the trace of the matrix R, i.e.

$$tr(R) = \sum_{i=0}^{3} R_{i,i}. \tag{7}$$

It holds that $tr(\lambda M) = \lambda tr(M)$ and $tr(M + N) = tr(M) + tr(N)$. Since in this case we have $tr(A) = 1$, $tr(B) = 0$, and $tr(I) = 3$, it follows that

$$tr(R) = 3\cos(\varphi) + (1 - \cos(\varphi)) = (1 + 2\cos(\varphi)), \tag{8}$$

which yields the formula

$$\cos(\varphi) = \frac{(tr(R) - 1)}{2} = \frac{\left(\sum_{i=0}^{3} R_{i,i} - 1\right)}{2}, \tag{9}$$

where the summation is carried out over the diagonal elements of the matrix R.

Homogeneous transformations, i.e. affine transformations in homogeneous coordinates, allow for the familiarity of linear algebra. The matrix notation is well-suited for expressing coordinate changes and the relative position and orientation of objects. To simplify the notation, we will follow a similar convention for writing vectors as column matrices and omitting the 'vector arrows' in the remainder of the text. The vector u, represented by the column matrix $[u_1, u_2, u_3]^T$, is mapped onto a new vector according to the formula

$$u' = R(r, \varphi)u + p, \tag{10}$$

where p is an offset vector. A homogeneous transformation matrix combines both

rotation and translation into a single matrix. It consists of a rotation matrix $R(r, \varphi)$, an offset vector or rather a column matrix $[p_1, p_2, p_3]^T$, and a constant last row.

$$ {}^{U}T_V = \begin{bmatrix} & & & p_1 \\ & R(r,\varphi) & & p_2 \\ & & & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11} $$

${}^{U}T_V$ operates on $u = [u_1, u_2, u_3, 1]^T$. Here, U and V are respectively the coordinate frames before and after the multiplication. By attaching a coordinate system, or the frame of reference to an object, its position and orientation, called the *pose*, can be expressed as a homogeneous transformation matrix with respect to the world frame. Moreover, since the rotational matrices are orthogonal, the inverse is given by

$$ {}^{V}T_U = \left({}^{U}T_V\right)^{-1} = \begin{bmatrix} & & \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \\ R^T(r,\varphi) & -R^T(r,\varphi) & \\ & & \\ 0 \quad 0 \quad 0 & 1 & \end{bmatrix}. \tag{12} $$

Using block multiplication, it is straightforward to verify the relationship

$$ {}^{U}T_V {}^{V}T_U = {}^{U}T_V \left({}^{U}T_V\right)^{-1} = I. \tag{13} $$

Note that we can further write ${}^{U}T_V = {}^{U}T_W {}^{W}T_V = \left({}^{W}T_U\right)^{-1} {}^{W}T_V$, where W is the world frame.

Evidently, the matrix notation is not the most compact way of specifying the 6 degrees of freedom needed to represent the position and rotation, i.e. the pose,

of an object in the Cartesian space. However, it simplifies the discussion of both the relative location of objects and manipulator kinematics. The change of basis is straightforward, since the identity ${}^{W}T_{U} = {}^{W}T_{V}{}^{V}T_{U}$ is true for any reference frames U, V and W.

2.2 Manipulator Kinematics

The pose of the end-effector of a robotic arm can be calculated in terms of joint parameters via chain multiplication as ${}^{O}T_{EE} = {}^{O}T_{Q_1}{}^{Q_1}T_{Q_2}...{}^{Q_i}T_{Q_{i+1}}...{}^{Q_{n-1}}T_{Q_n}{}^{Q_n}T_{EE}$, where $i \in \{1, ..., n\}$, n is the total number of joints, and Q_i denote the individual joints of the manipulator. Here, the matrices ${}^{Q_{i-1}}T_{Q_i}$ can be expressed using the Denavit-Hartenberg notation, see Asada and Slotine [3], or Spong et al. [65]. The leftmost coordinate system in the product is the robot's base frame, and the rightmost that of the end-effector. The intermediate multiplication steps refer to the ends of links that comprise the robotic arm. Each homogeneous transformation can be expressed as a function of joint position and link parameters. It can be written as

$$ {}^{Q_{i-1}}T_{Q_i} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (14) $$

which is the result of a multiplication of a pure rotation, $p = 0$, and a pure translation matrix, $R = I$. The corresponding transformation matrices for the 7 degrees of freedom Barrett WAM are listed in the Appendix A.

If only the pose of the end-effector is of interest, the robotic manipulator can be abstracted out as

$$\begin{bmatrix} u \\ \varphi \end{bmatrix} = H(q), \tag{15}$$

where q denotes the joint positions, u is the position of the end-effector expressed as a three dimensional column vector, and φ is the orientation of the end-effector in some suitable notation, e.g. a quaternion, a rotation matrix, or Euler angles. The 6×1 vector containing the translational and angular velocities, $v = \dot{u}$ and $\omega = \dot{\varphi}$ receptively, which is sometimes also referred to as *twist*, can be obtained as

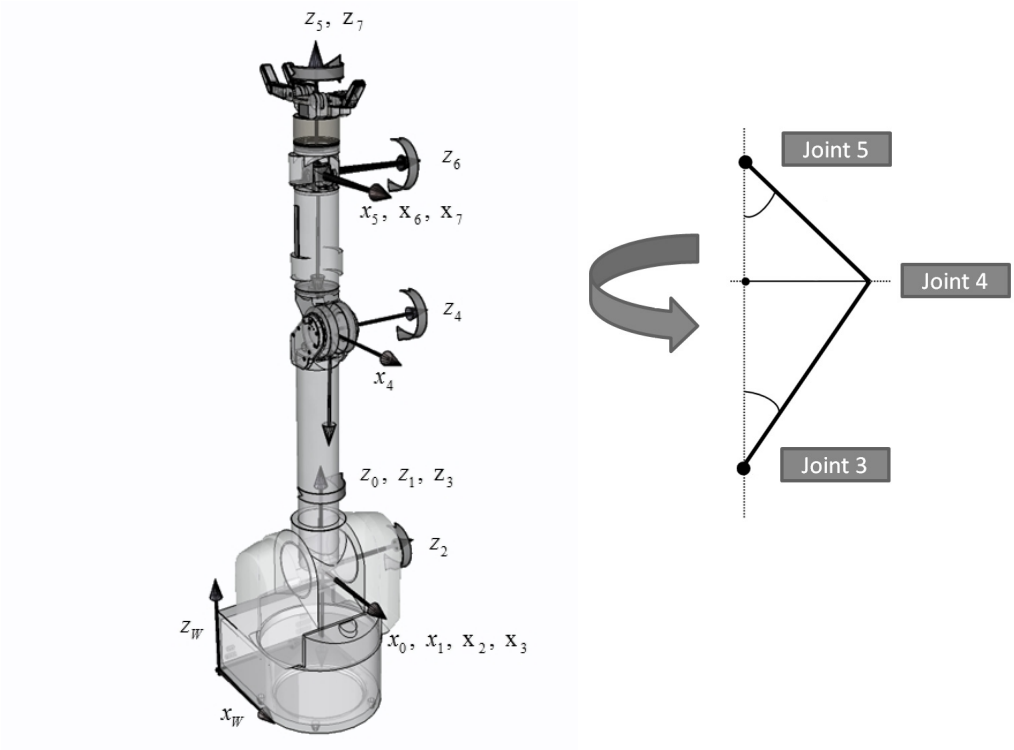
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{\partial H(q)}{\partial q}\dot{q} = J(q)\dot{q}. \tag{16}$$

$J(q)$ is called the Jacobian of the manipulator. It can be computed directly without the need of explicitly calculating the derivative of $H(q)$. For more details see any of the standard textbooks on robotics, for instance Spong et al. [65]. If Euler angles are used to describe the orientation, then the resulting Jacobian is not identical to the 'geometric' Jacobian that is obtained for homogeneous matrices and quaternions.

In a neighborhood of a *singularity point*, large changes in the joint positions q lead to only small changes in both u and φ. This reflects the physical constraints on the movement of the manipulator. The classical approach to singularity detection is to check whether the Jacobian is of full rank. This can be done for redundant manipulators based on the determinant of $J(q)J(q)^T$, which in this case should not be equal to or close to zero, see Yoshikawa [77].

2.3 Barrett WAM 7-DOF Robotic Arm

A 7 degrees of freedom (DOF) redundant robotic manipulator, the Barrett WAM, was used for all planning algorithms and experiments. This robot is 'back-drivable' in the sense that the joints can move based on the interaction with the environment. The joint positions are determined by the superposition of the commanded torques and the torques resulting from outside forces applied to the robot, see Rooks [60].



(a) (b)

Figure 2: a) Barrett WAM 7-DOF robotic arm fitted with BarrettHand BH8-262. b) For the same pose of the end-effector there can be infinitely many inverse kinematics solutions that correspond to the 4^{th} joint's (elbow) position on a circle around the axis defined by the 3^{rd} (shoulder) and the 5^{th} joint (wrist).

The inverse kinematics problem consists of finding the joint positions for a given pose of the end-effector. The complete set of forward and inverse kinematics formulas for the 7-DOF WAM can be found in the Appendix A.

```
1: procedure InverseKinematicsCloseExhaustive(T, Θ_ref)
2:     ▷ Θ_ref is the vector of reference joint positions
3:     Θ_3 ← lower_bound(θ_3)
4:     bestθ_3 ← NaN
5:     minDist ← inf
6:     while θ_3 ≤ upper_bound(θ_3) do
7:         ▷ Loop over the range of θ_3
8:         θ_3 ← θ_3 + 0.01
9:         Θ ← InverseKinematics(T, Θ_ref, θ_3)
10:        if SolutionExists(Θ) & ‖Θ − Θ_ref‖_1 < minDist then
11:            bestθ_3 ← θ_3
12:            minDist ← ‖Θ − Θ_ref‖_1
13:        end if
14:    end while
15:    return bestθ_3                    ▷ Return θ_3 for which Θ − Θ_ref is minimal.
16: end procedure
```

Figure 3: Exhaustive search over the range of the 3rd joint can be used in order to find the closest joint configuration vector for a given end-effector pose.

2.3.1 Optimization over the Range of the Shoulder Joint

In case of a 7-DOF redundant manipulator, the inverse kinematics solution is not unique. Instead, the elbow joint, as depicted in Fig. 2, will be constrained to the orbit around the axis formed by the line between the wrist and the shoulder, see also Singh and Claassens [64]. There exists an analytic solution based on a pre-set angle for the 3rd (shoulder) joint, for details see the Appendix A. For practical purposes, one can rely on a routine written by Huber, which is a part of the Robot Operating System (ROS) [58] project code. In order to find the nearest solution to a given joint space configuration vector, ROS uses the closed form formula and an exhaustive search with fixed increments over the discretized range of the 3rd joint angle, see Fig. 3. The reason behind choosing this strategy is that the function to be minimized – the Manhattan distance between two joint space configurations – is highly irregular.

2.3.2 A Control Scheme for Guarded and Compliant Motion

Not only compliant motion itself is important, but also achieving the initial contact between the objects. This is commonly done via a *guarded move*, in which the motion is terminated based on a force limit. Without a force sensor, one can instead rely on torque thresholds and the 'back-drivability' of the WAM to obtain both types of motion, and terminate the movement when necessary based on the positional error at the end-effector. The libbarrett library implements torque limits as a part of the joint space PID controller by calculating the torques according to the control law

$$\tau = \hat{G}(\Theta) + sat\left(K_P\left(\Theta(t) - \Theta_d\right) + K_D\,\dot{\Theta}(t) + sat\left(K_I \int_0^T (\Theta(t) - \Theta_d)dt, v\right), w\right), \tag{17}$$

where τ is the vector of torques to be applied to the joints, $\hat{G}(\Theta)$ is the output of the gravity compensation model, the first term in the parentheses determines the stiffness, the second one the damping, and the last one is the integral part introduced in order to cancel out any steady-state errors. The corresponding matrices K_S, K_D and K_I, are all positive and diagonal. The function $sat\,(x, v)$ is defined as

$$sat\,(x, v) = \begin{bmatrix} \mathrm{s}\,(x_1, v_1) \\ \vdots \\ \mathrm{s}\,(x_n, v_n) \end{bmatrix}, \tag{18}$$

where

$$\mathrm{s}\,(a, k) = \begin{cases} |k| \text{ if } a > |k| \\ -|k| \text{ if } a < -|k|\,. \\ a \text{ otherwise} \end{cases} \tag{19}$$

It provides a means to set the torque saturation values in the control law.

2.3.3 Impedance Control and Jacobian Transpose Methods

Let us consider the work done at the end-effector by a generalized force F, consisting of both forces and torques. It can also be expressed using the torques τ and displacements $\delta\Theta$ at the joints. Since the two must be equal, we obtain $\delta x^T F = \delta\Theta^T \tau$ in matrix notation. Now, using the definition of the Jacobian, $\delta x = J\delta\Theta$ yields $\delta\Theta^T J^T F = \delta\Theta^T \tau$, which must hold for all $\delta\Theta$. Hence, we obtain

$$J^T F = \tau. \tag{20}$$

Since for $F = -K\delta x$, it also follows that $-J^T K\delta x = \tau$. This result can be extended to make the end-effector of a robotic manipulator exhibit some desired stiffness properties. Hogan [26] describes an active impedance control, that uses the control law

$$\tau = \hat{G}(\Theta) - J^T(\Theta)\left(K_S\left(x - x_d\right) + K_D\dot{x}\right), \tag{21}$$

where $\hat{G}(\Theta)$ is the gravity compensation model, the first term in the parentheses determines the stiffness, and the second one the damping. The matrices K_S and K_D are positive and diagonal. Eq. 21 takes into account not only positional, but also the rotational errors in pose of the end-effector.[2]

Despite its indirect nature that relies on the forces and torques at the end-effector in combination with the Jacobian transpose to control the joint torques, impedance control is known to be asymptotically stable if the gravity model is accurate enough, i.e. $\hat{G}(\Theta) \approx G(\Theta)$, see Hogan [25, 26], and also for example Asada and Slotine [3]. If the contact occurs, then the actual position will be determined by the stiffness constants K_E and K_S of the environment and the manipulator respectively.

[2]Note that we are using generalized forces and $x = \begin{bmatrix} u & \varphi \end{bmatrix}^T$, see the Eq. 15. In fact, we have $\tau = \hat{G}(\Theta) - {J_p}^T(\Theta)\left(K_{S_p}\left(p - p_d\right) + K_{D_p}\dot{p}\right) - {J_\varphi}^T(\Theta)\left(K_{S\varphi}\left(\varphi - \varphi_d\right) + K_{D\varphi}\dot{\varphi}\right)$, where $J = \begin{bmatrix} J_p & J_\varphi \end{bmatrix}^T$ and $\varphi - \varphi_d$ can be expressed as a rotation vector scaled by the correction angle.

2.3.4 Estimated Force at the End-Effector

Without a force sensor, the approximate values for the force and torque at the end-effector can be obtained by solving the system of linear equations given by

$$J(\Theta)^T F = \tau_{TOTAL} - \hat{G}(\Theta) \tag{22}$$

in the least-square sense. This can, for example, be done using the QR decomposition. $J(\Theta)$ is here the tool Jacobian of the manipulator, the vector $\hat{G}(\Theta)$ contains the gravity compensation values, and τ_{TOTAL} denotes the applied joint torques. Although the results will, of course, be affected by inaccuracies of the gravity compensation model, they nevertheless provide good estimates. Since Barrett WAM does not have torque sensors, we will use the commanded torques, i.e. $\tau_{PID} = \tau_{TOTAL} - \hat{G}(\Theta)$.

2.4 Minimal Translational Distance

The contact between objects can be described in terms of the minimal translational distance. This is closely related to the concept of penetration depth and distance between objects. Following Cameron [8], we first define the unsigned minimal translational distance as

$$MTD^+(X, Y) = \inf_t \{|t| : X + t \text{ is in contact with } Y\}. \tag{23}$$

The minimal translational distance MTD is then equal to MTD^+, if the objects do not overlap, and the negative of the value of MTD^+, if there is penetration between the objects. The fact that the objects are in contact is equivalent to MTD^+, and hence also MTD, being zero. The translational configuration space obstacle

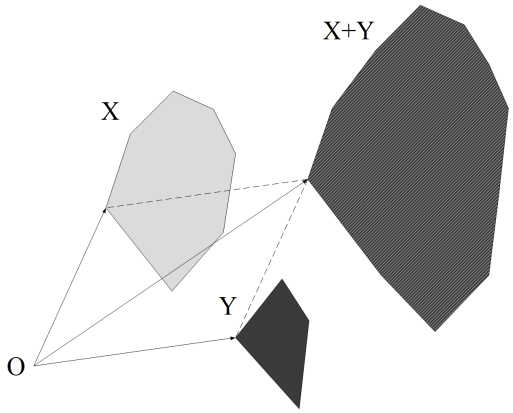


Figure 4: The Minkowski sum between two objects.

$$TSCO\,(X,Y) = \{y - x : x \in X \text{ and } y \in Y\} \tag{24}$$

is computed as the Minkowski difference between the sets X and Y. Here, the Minkowski sum between X and Y, $\{x + y : x \in X \text{ and } y \in Y\}$, is the set that is obtained by pairwise adding up points in X and Y, as shown for two polygons in Fig. 4. A detailed introduction can be found in Latombe [42]. One can think of it as adding the whole set Y to a fixed element of X and then forming the union over all elements of X.[3] The difference $X - Y$ is calculated as $X + (-Y)$ by first projecting Y through the origin. An important property of the Minkowski sum is that it is convex, if both X and Y are convex, implying that $TSCO\,(X,Y)$ will be convex as well. If the $TSCO$ is known, we can obtain MTD^+ as

$$MTD^+\,(X,Y) = MTD^+\,(O, boundary\,(TSCO\,(X,Y)))\,, \tag{25}$$

where O is the origin and $boundary\,(S)$ specifies the points on the boundary of the set S. This is true regardless whether the objects overlap or not. Moreover, the origin O being inside, on the boundary or outside of $TSCO\,(X,Y)$ is equivalent respectively to the cases of overlap, contact and no overlap between the objects X and Y.

[3] One will still obtain the same result, if the roles of X and Y are exchanged.

2.5 Distance and Penetration Depth between Convex Polygonal Meshes

For convex polygonal meshes, MTD^+ can be calculated by finding the point on the boundary of the translational configuration space obstacle with the shortest distance to the origin. This can be done explicitly by using two nested loops to obtain the difference between all pairs of vertices (x, y) with $x \in X$ and $y \in Y$, and then computing the convex hull of the result – see for example [35], Sec. IV C. The expected time complexity for the Quickhull algorithm [4] is $O(nmlog(nm))$ and $O(n^2m^2)$ in the worst case, where n and m are the number of vertices in X and Y. Alternatively, one could compute the convex hull using Chan's algorithm [9] and improve this bound to $O(nmlog(nm))$. However, the CGAL [20] library for example, see also Hachenberger [19] and [18], implements an algorithm that computes the vertices of a Minkowski sum of convex polyhedra in $O(nm)$ steps.[4] Once $TSCO$ has been obtained, we find the vertex that is closest to the origin. Then, one of the neighboring edges and faces of the $TSCO$ contains a point m such that $m = \min_p \|p\|_2$ for any $p \in TSCO$. $MTD^+ = \|m\|_2$ is equal to the penetration depth if the origin is inside $TSCO$, and the distance between the objects otherwise. We will rely on this value to construct the cost function for fitting convex features in Sec. 4.3.2.

In order to decide whether penetration between objects occurs, additional criteria are needed. One might use collision detection, see for instance [17] and [41], and, if there is no collision, check whether the convex hull of $X \cup Y$ is fully contained in that of either X or Y, in which case one of the objects must be inside the other.

[4] The $O(n + m)$ time complexity can only be achieved in 2 dimensions.

CHAPTER 3: NOMINAL PLANNING OF ASSEMBLY SEQUENCES

Motion compliant to contact between parts is useful for reducing relative pose uncertainties in assembly operations, and it is a common approach to plan an assembly task in terms of the compliant motion of a moving part against a fixed part. For this purpose, a sequence of contact state transitions in the task, i.e. Cartesian, space is constructed [46, 33, 32, 59, 40, 24]. However, an important issue that is less studied is how to convert such a motion from task space to an executable motion of the robot in the joint space without violating the contact constraints.

Sections of the current chapter correspond to the three stages in which this process will be carried out. First, we discuss planning contact state transitions using topological information. Next, a method is introduced to convert compliant paths from Cartesian to the joint space, while taking into account the physical constraints of the robotic arm including one extra degree of freedom of a 7-DOF manipulator. This allows us, in the later chapters, to focus on assembly planning in the task space and on estimation strategies. Finally, we will look at how approximate compliance without a force sensor can be implemented. Here, a force limit is achieved indirectly by specifying positional error thresholds. In connection with a 'back-drivable' robot, it allows for seamless transition from a guarded move to compliant motion, as the held part moves from no contact to being constrained by the contact.

3.1 High-Level Topological Planning via Contact State Transitions

Contact states are a high-level topological concept that serves a twofold function. In terms of planning, a complex task can be split into computationally more manageable subproblems. Moreover, robust strategies, which allow for successful assembly despite some limited amount of misalignment between the parts or other types of uncertainty during the execution, can be constructed in this manner. The main idea here is to subdivide the surface of an object into suitable topological features: vertices, edges and faces. Intuitively, it is easier to bring two convex features into contact with each other, for example, than to directly align and then put together two parts in a specific way. The latter is, in fact, practically impossible under uncertainty even for trivial tasks. This further suggests gradually aligning the parts via a sequence of deliberate exploratory moves.

More precisely, a contact state is defined as a set of contact configurations for which exactly the same features on both objects are in contact [74]. It is also possible to define principal contacts as those contact states that are not implied by other contact states [31]. For instance, if two faces are in contact, we ignore the contacts made by the edges and vertices, which form the boundary of those faces. The contact state graph, as shown in Fig. 5, can be constructed in an automatic manner [56, 74, 70, 39, 38] and there also exists some previous work, see Ji [31, 32], that considers the generation of compliant path trajectories in Cartesian space based on the contact state graph.

In previous research, compliant assembly sequences have not been constructed with parameter estimation in mind. However, for our purposes, the nominal plans will

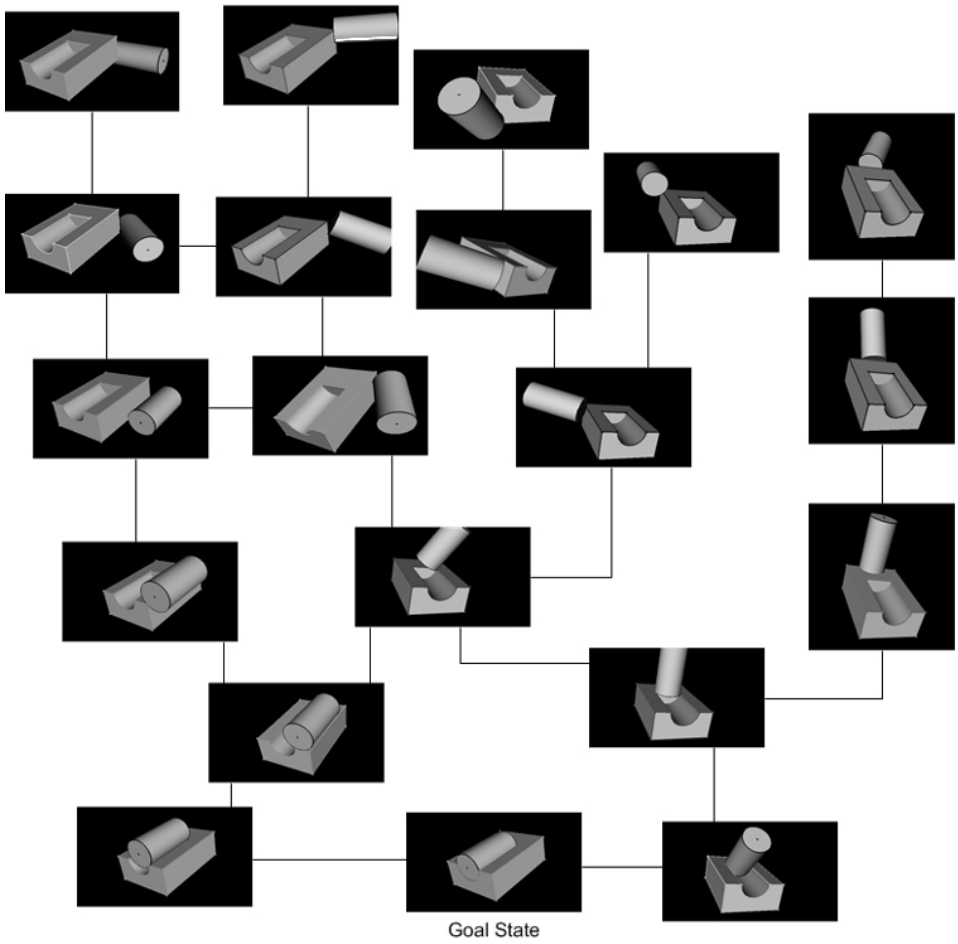


Figure 5: A subset of the contact state graph for an assembly task between two objects.

need to incorporate *estimation steps*, which complicates the matters further in terms of planning the high-level contact states and transitions in the task space. Since the main focus of this dissertation is the reduction in uncertainty of parameter values that describe the position and orientation of the parts, we will limit ourselves to planning both the exploratory and assembly steps in an ad hoc manner. Once a high-level plan is constructed, however, it is converted automatically to a joint space plan, and a trajectory needs to be generated in terms of positions of the individual joints of the robotic arm.

3.2 Converting a Compliant Path to a Joint Space Trajectory

Compliant motion, as depicted in Fig. 5, can be planned as a sequence of contact state transitions and the corresponding path of contact configurations [74, 32]. Such a trajectory of the end-effector in the Cartesian space can be specified using knot points, where each pose is represented by a 4×4 homogeneous matrix T consisting of a 3×3 rotational matrix R and an offset vector. Given a Cartesian space path in terms of homogeneous transformation matrices T_0 , T_1,..., T_G, where T_G is the goal pose for the end-effector that completes the assembly task, we need to determine a corresponding feasible sequence of joint-space configurations Θ_0, Θ_1, ..., Θ_G that is collision and singularity free. Merely calculating the inverse kinematics solution for discrete points is not enough. In this section, we present a novel method that bridges the gap between a compliant path in terms of the relative motion of the parts in contact and an executable trajectory in the joint space, see Sarić et al. [62].

3.2.1 Trajectory Generation via Knot Points

Starting from the goal state T_G, the corresponding joint configurations $\Theta_G \rightarrow$, ... $\Theta_m \rightarrow$, $\Theta_{m-1} \rightarrow$, ... Θ_0 are determined in the reverse order: The joint configuration immediately preceding Θ_G is calculated by minimizing the distance to Θ_G, and then its previous joint configuration is obtained, and so on. To fix ideas, given Θ_m, the joint configuration Θ_{m-1} will have the value of θ_3 set in such a way that the Manhattan distance between Θ_{m-1} and Θ_m, i.e. the sum of the absolute values of the differences between the angles of each pair of corresponding joints $\|\Theta_{m-1} - \Theta_m\| = \sum_{n=1}^{7} |\theta_{n,m-1} - \theta_{n,m}|$, is minimized. We proceed in the decreas-

ing order of m until Θ_0 is reached. If the performance is not an issue, one can use exhaustive search over the full range of the 3rd joint with fixed increments, see the ROS [58] implementation as described in Sec. 2.3.1. In order to obtain Θ_{m-1}, we would call the procedure InverseKinematicsCloseExhaustive from Fig. 3 with $\Theta_{ref} = \Theta_m$. However, for all but the final pose the solution will be close to the previously obtained one, allowing for good results when using a local search.

As outlined by the pseudocode in Fig. 6, we can search the neighborhood of $\theta_{3,ref}$, which is the value of the 3rd joint of a given reference vector Θ_{ref}, starting with the closest points first, and terminate when the distance between the solution Θ and Θ_{ref} begins to increase.[5] The result can be a significant speed up, since for trajectory planning we need to interpolate via points in Cartesian space that are relatively close to each other in order to obtain a smooth joint space trajectory. This makes it possible to use faster local optimization techniques, see Fig. 6, and it also enables us to recalculate the trajectory in response to additional information that was collected based on the physical interaction between the objects. The reference joint space configuration vector is in this case always equal to the joint space configuration vector for the closest point along the trajectory. Since our objective is to achieve a specific assembly, which is characterized by the goal configuration of the part held by the gripper and the corresponding end-effector pose, we give more importance to the end of the path, closest to the goal pose, and start by finding a goal joint space configuration vector Θ_G.

[5]Note that InverseKinematicsCloseLocal and InverseKinematicsCloseExhaustive both require only two parameters, T and Θ_{ref}. An extra pre-set value of the 3rd joint is not needed for a 7-DOF robotic manipulator.

```
1: procedure InverseKinematicsCloseLocal(T, Θ_ref)
2:     ▷ Θ_ref - reference joint positions
3:     flag_L ← true, flag_R ← true
4:     δθ_3 ← 0.01, θ_3,best ← NaN, minDist ← ∞
5:     Θ ← InverseKinematics(T, Θ_ref, θ_3,ref)
6:     if SolutionExists(Θ) then
7:         θ_3,best ← θ_3,ref
8:         minDist ← ‖Θ − Θ_ref‖_1
9:     end if
10:    while (t_R ← (flag_R AND θ_3,ref + δθ_3 ≤ up(θ_3)))
11:    OR (t_L ← (flag_L AND θ_3,ref − δθ_3 ≥ low(θ_3)))
12:    do
13:        if t_R then
14:            Θ ← InverseKinematics(T, Θ_ref, θ_3,ref + δθ_3)
15:            if SolutionExists(Θ) then
16:                if ‖Θ − Θ_ref‖_1 < minDist then
17:                    θ_3,best ← θ_3,ref + δθ_3
18:                    minDist ← ‖Θ − Θ_ref‖_1
19:                else
20:                    if ‖Θ − Θ_ref‖_1 > minDist + ϵ then
21:                        flag_R ← false
22:                    end if
23:                end if
24:            end if
25:        end if
26:        if t_L then
27:            Θ ← InverseKinematics(T, Θ_ref, θ_3,ref − δθ_3)
28:            if SolutionExists(Θ) then
29:                if ‖Θ − Θ_ref‖_1 < minDist then
30:                    θ_3,best ← θ_3,ref − δθ_3
31:                    minDist ← ‖Θ − Θ_ref‖_1
32:                else
33:                    if ‖Θ − Θ_ref‖_1 > minDist + ϵ then
34:                        flag_L ← false
35:                    end if
36:                end if
37:            end if
38:        end if
39:        δθ_3 ← δθ_3 + 0.01
40:    end while
41:    return θ_3,best
42: end procedure
```

Figure 6: Local search over the 3rd joint to find the closest joint space configuration vector. $\text{LOW}(\theta_3)$ and $\text{UP}(\theta_3)$ denote the lower and upper bound of the range of the 3rd joint respectively. $flag_L$ and $flag_R$ are used to stop the search.

The effectiveness of the algorithm hinges upon the step size for both the exhaustive and the local search. The latter further depends on the reference joint positions vector being close to the solution in order for it to be in the neighborhood of the minimum.[6] This is the case for a sufficiently densely sampled smooth Cartesian trajectory, which provides locality both in Cartesian and, if there are no singularities, also in joint space. The value of 0.01 rad for the $\delta\theta_3$ step size was chosen experimentally for the Barrett 7-DOF WAM. The range and step size are both dependent on how the end-effector poses in the task space are sampled. Because of rounding errors, it makes sense to additionally search a limited range after the first local minimum. Searching locally reduced the computational time by up to a factor of 4 compared to the exhaustive search with the same step size and allowed for a conversion of full paths on the Barrett WAM CPU without noticeable delay in the experiments. If a workstation is used, then the effect is lower, but still the conversion time was consistently at least twice as fast in the test. In order to obtain a more accurate solution, the final value θ_3 can be further refined using local gradient descent with contracting step size on the best interval, see Fig. 8, although the practical value of doing so is limited.

The 3$^{\text{rd}}$ joint can also be used to optimize over different criteria, and in general one has to resort to exhaustively searching over this dimension for a sufficiently small step size. Optimizing for grasping by positioning the rest of the arm away from the object requires an exhaustive search, since no initial estimate is readily available. Also, alternative optimization criteria can be used. For instance, one can maximize the distance to other objects or the manipulability measure, see Yoshikawa [77].

[6] As shown in the Fig. 7, there will be a range in which the function is still convex.

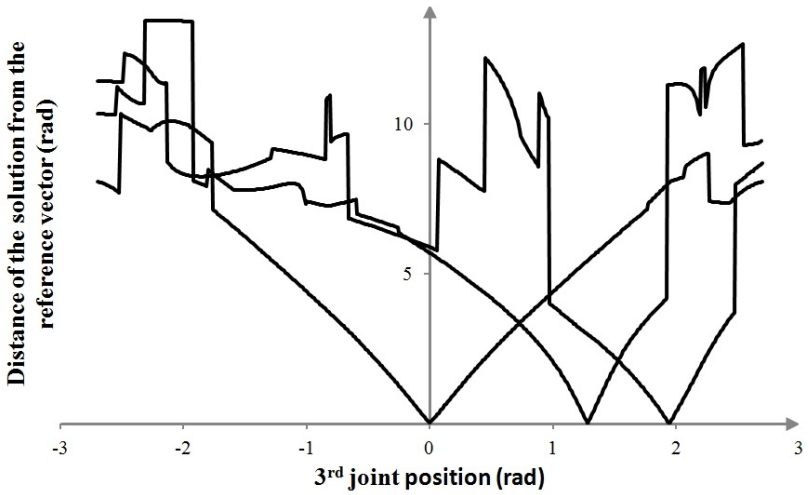


Figure 7: Examples of the position of the 3rd joint vs. the Manhattan distance from a reference joint space configuration vector, when computing the inverse kinematics of the 7-DOF WAM.

A simple yet effective strategy to obtain the grasping pose consists of searching the range of θ_3 to find the value that maximizes the projection of the manipulator links onto the z axis of the end-effector plate, which will be orthogonal to the surface of the held, i.e. moving object. Constrained by a fixed base and the desired end-effector pose, this heuristic pushes the intermediate links of the robot as far as possible from the held object. The result is a natural-looking arm posture that keeps the links of the robotic arm away from the objects.[7] The inverse kinematics solution obtained analytically for this value of θ_3 based on T_G yields a suitable Θ_G. We then work backward from the final pose to find joint space configurations for the previous poses along the Cartesian path by choosing the nearest joint space configuration vector for every pose as discussed above. Furthermore, one can incorporate other path validation strategies, such as collision detection and avoidance, into this conversion.

[7] Alternatively, a suitable surface normal could also be used instead of the z axis, or the manipulability measure could be maximized subject to constraints on the distance of the WAM joints from the objects, etc.

1: **procedure** RefineLocally($T, \Theta_{ref}, \theta_3, minDist, numIter$)
2: $flag \leftarrow true$
3: $stepSize \leftarrow 0.01$
4: **for** $numIter$ **do** ▷ Contract the $stepSize$ $numIter$ times
5: $stepSize \leftarrow \frac{stepSize}{2}$
6: $\theta \leftarrow$ InverseKinematics($T, \Theta_{ref}, \theta_3 + stepSize$)
7: ▷ Check if solution vector θ is closer to Θ_{ref}
8: **if** SolutionExists(θ) & $\|\theta - \Theta_{ref}\|_1 < minDist$ **then**
9: $\theta_3 \leftarrow \theta_3 + stepSize$
10: $minDist \leftarrow \|\theta - \Theta_{ref}\|_1$
11: $flag \leftarrow false$ ▷ i.e. descent direction found
12: **end if**
13: **if** $flag$ **then**
14: $\theta \leftarrow$ InverseKinematics($T, \Theta_{ref}, \theta_3 - stepSize$)
15: ▷ Check if solution vector θ is closer to Θ_{ref}
16: **if** SolutionExists(θ) & $\|\theta - \Theta_{ref}\|_1 < minDist$ **then**
17: $\theta_3 \leftarrow \theta_3 - stepSize$
18: $minDist \leftarrow \|\theta - \Theta_{ref}\|_1$
19: **end if**
20: **end if**
21: $flag \leftarrow true$
22: **end for**
23: **return** θ_3 ▷ Return θ_3 for which $\theta - \Theta_{ref}$ is minimal.
24: **end procedure**

Figure 8: Once the solution is found by searching over the range of the 3rd joint, it may be further refined locally by using gradient descent with a contracting step.

3.2.2 Singularity Detection

After obtaining the sequence of joint space configurations Θ_0, Θ_1, ..., Θ_m, ..., Θ_G, we need to check if there are singularities between Θ_{i-1} and Θ_i in order to know whether we have a feasible compliant path in the joint space that realizes the compliant Cartesian space path from T_{i-1} to T_i. Here, a singularity is considered to occur whenever the end-effector cannot be moved in the desired direction from T_{i-1} to T_i. Note that, if two poses T_{i-1} and T_i are close and have corresponding solutions Θ_{i-1} and Θ_i in the joint space, Θ_{i-1} and Θ_i need not necessarily be close at all, which may

result in a loss of contact and the compliant Cartesian path from T_{i-1} and T_i is not feasible due to the physical constraints of the robot. To illustrate this, consider that the two closest existing solutions Θ_{i-1} and Θ_i may for instance be mirror-inverted, in which case moving from Θ_{i-1} to Θ_i requires rotating the end-effector away from the contact state and performing a significant detour in the air – this will obviously disrupt the compliant motion and also unintended collisions might occur.

Therefore, to avoid a significant deviation from the desired contact state, we could require that Θ_{i-1} be sufficiently close to Θ_i when T_{i-1} is close to T_i. Otherwise, we say that a singularity is encountered. However, it is difficult to determine 'sufficient closeness' between Θ_{i-1} and Θ_i, since there may not be a uniform threshold on the distance. Thus, our strategy will be to check for behaviors associated with a singularity. We first interpolate linearly between Θ_i and Θ_{i-1} in the joint space as

$$\Theta(j) = \Theta_i + \frac{j}{n}(\Theta_{i-1} - \Theta_i), \tag{26}$$

where $n > 0$ is an integer and $0 < j < n$. The corresponding $T(j)$ is calculated using forward kinematics based on $\Theta(j)$. Now, if there is no singularity, then as j increases $P(j)$, the offset vector of $T(j)$, should move in the direction of P_{i-1}, the offset vector of T_{i-1}. In other words, the distance between the offset vectors of the matrices $T(j)$ and T_{i-1} should be smaller than that between the offset vectors of $T(j-1)$ and T_{i-1}. Therefore, we see that if the following condition holds, a singularity is detected:

$$\|P(j) - P_{i-1}\|_2 > \|P(j-1) - P_{i-1}\|_2 + \epsilon. \tag{27}$$

Here ϵ is a small positive constant introduced in order to take rounding errors into account (e.g. $\epsilon = 10^{-8}$ for double precision floating point numbers).

Next, let $\phi(j)$ be the angle expressing the difference in orientation in the rotational parts of the matrices $T(j)$ and T_{i-1}, denoted as $R(j)$ and R_{i-1} respectively. The cosine of this angle can be calculated as

$$\cos\left(\phi(j)\right) = \frac{tr\left(R(j)R_{i-1}^{T}\right) - 1}{2}, \tag{28}$$

where $tr(*)$ is the trace of a matrix. Now, $|\phi(j)|$ should decrease as j increases. Thus, we require $\cos\left(\phi(j)\right)$ to be a monotonically increasing function of j and, consequently, if the following condition is satisfied a singularity is detected:

$$\cos\left(\phi(j)\right) < \cos\left(\phi(j-1)\right) - \epsilon. \tag{29}$$

In order to check for singularities using the two conditions from Eq. 27 and Eq. 29, we now simply loop over j. This works well in practice, since a singularity would lead to an erratic turn of the end-effector, even when the offset vector and the rotation matrix do not change much between $T(j)$ and T_{i-1}. Note, that the technique avoids checking for a singular Jacobian, where it is difficult to determine a uniform threshold.

To validate the entire path, we begin with the goal joint space configuration Θ_G and backtrack one knot configuration at a time, until a singularity is encountered or Θ_0 is reached. In this way, we can at least obtain a partial path of joint configurations Θ_s, Θ_{s+1},..., Θ_G that is free of singularities and corresponds to the compliant Cartesian space path T_s, T_{s+1}, ..., T_G, where s denotes the index of the first configuration after the singularity or T_0. For $s \neq 0$, the path between Θ_0 and Θ_s is invalid due to a singularity, and it is discarded. Finally, we fit the path Θ_s, Θ_{s+1},..., Θ_G using splines to obtain a smooth, compliant Cartesian-space path passing through T_s, T_{s+1}, ..., T_G.

Whenever Θ_s can be reached, the above approach offers a viable way to find a singularity-free compliant path ending in the goal pose of the assembly sequence. In contrast to for example [54] and [72], we are not attempting to find an approximate solution near the singularities, but to detect the singularities prior to the execution in order to ensure valid compliant motion.[8] It was found empirically, that in connection with a 7-DOF redundant manipulator this is not too restrictive, and leads to good solutions in practice. Note that the evaluation of forward kinematics is efficient and can be used repeatedly along the interpolated path without much overhead.

3.2.3 Velocity and Time Limits

Next, we set proper thresholds for the joint velocities. First of all, the values of the linear and angular velocities at the end-effector need to be limited. We consider the translational and rotational components of the pair of homogeneous transformation matrices T_{i-1} and T_i to find $\Delta P_i = P_i - P_{i-1}$ and the angle ϕ_i such that

$$\cos\phi_i = \frac{tr(R_i R_{i-1}^T) - 1}{2}. \tag{30}$$

Given the desired maximum end-effector linear and angular velocities v and ω, we can obtain the time interval to move the manipulator from T_{i-1} to T_i as

$$\Delta t_i = \max(\frac{\Delta P_i}{v}, \frac{\phi_i}{\omega}).$$

This increment ensures that the linear and angular velocities of the end-effector will not exceed the maximum limits v and ω. Since we are merely rescaling the time increments between the knot points, the synchronized translational and rotational

[8] If one's goal is to solve the general case, then it makes more sense to allow the robot to move.

motion in the task space is maintained.

Finally, we need to ensure that the physical limitations of the joints are taken into account. To this end, we limit the joint speeds by multiplying all Δt_i's by the ratio

$$c = \frac{\max_k(\|\Theta_k - \Theta_{k-1}\|/\Delta t_k)}{\text{max\_joint\_speed}}, \tag{31}$$

where the indices go over all the knot point configurations.

3.3 Approximate Compliant Motion without Force Sensing

In order to realize the compliant motion without a force sensor, the contact between objects is maintained by choosing a suitable path in the task frame for which the objects nominally overlap. If the Barrett WAM or a similar robotic arm is used, the end-effector will be moved in the direction of the sum of torques applied to the joints by the environment and the motors. Additionally, the saturation threshold for the force at the end-effector can be approximated by positional error and joint torque thresholds. Due to the difference between the nominal and the actual path caused by inherently imperfect information, we cannot guarantee the validity of the new path. However, since a 'back-drivable' manipulator moves in the direction of the least resistance, it will also tend to avoid singularities whenever it is physically possible. Moreover, based on the relationship expressed by the Eq. 20, we know that at a singularity point the static forces are mapped onto the null space of the J^T. Hence, the robotic arm might get stuck, but the force moving it toward the singularity will not make it deviate erratically from the trajectory.

For the control law from Sec. 2.3.2, the following technique, as illustrated in Fig. 9,

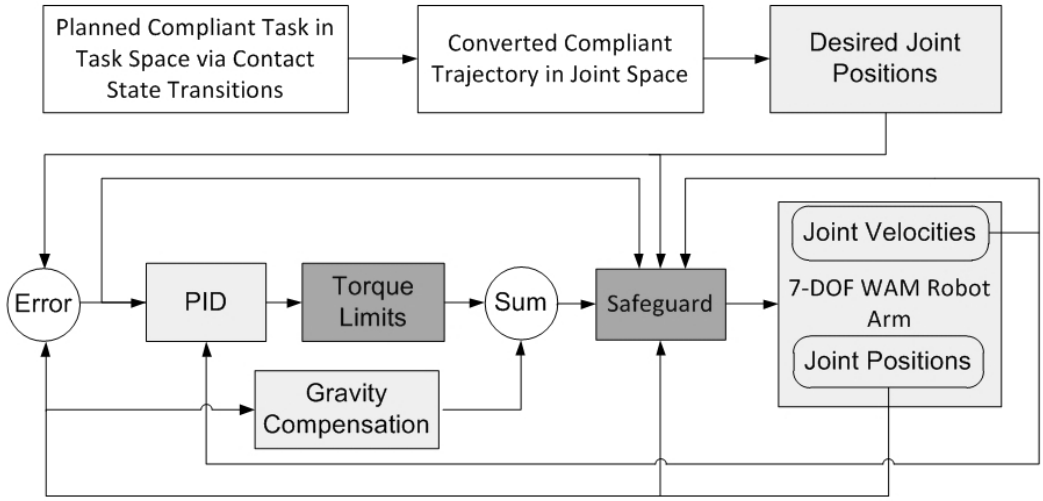


Figure 9: Schematics for the joint-space control scheme based on limiting computed joint torques of a 'back-drivable' robot that was used in the experiments.

is proposed in order to realize simple compliant motion. The PID controller of the Barrett WAM takes positional errors as input and outputs joint torques, and the logic that allows for the contact with the environment to occur is represented by the dark shaded boxes. After a contact occurred, there would be a steady increase in joint torques unless the control signal limits are set in the PID controller for each joint. The torque limits must be small enough to limit the force exerted during the contact (i.e., on the parts in the assembly), but large enough to cancel the difference between the gravity compensation model and the torques that are required to move the arm to the desired joint positions when no contact with the environment occurs. To take into account positional uncertainty and possible unplanned contacts, we also set an upper limit for the positional errors in the Cartesian space. The reason for this is that the PID controller tracks the error between the control signal and the current position. Without a torque limit, not only would the control signal keep increasing once an

obstacle has been encountered and the current positional error become larger; if the obstacle were then removed, the robotic arm would start to accelerate. Since the torques are proportional to the errors in the joint space, an abrupt jump could occur for large errors. Also, even when the contact is maintained, the positional error normal to the surface defined by the contacting features should not be allowed to increase disproportionately. As a stopping criterion for the safeguard[9] in the experiments, the position error, which is the difference between the measured position and the desired end-effector position P_i for the current time step i, was required not to exceed a set threshold value in the experiments. The inequality

$$\|P_i - P_{measured}\| > error_{max} \tag{32}$$

can therefore be used to detect when the motion should be stopped, i.e. the current position should be held. Obviously, there are some shortcomings, the most notable of which is not being able to effectively limit the contact force based on an arbitrary low prespecified threshold: The force limit, which is achieved indirectly through the joint torque limits, cannot be made arbitrarily small, because the torque limits must of course take into account moving the robotic arm itself. Nevertheless, this strategy performed well in the experiments. Also, note that this reasoning might further be extended to a fuzzy controller using relationships such as 'error is large' or 'velocity is small'.

As will be seen later when the parameter estimation is discussed, maintaining the orientation of the end-effector during an exploratory move can simplify the estimation

[9]See the corresponding shaded box in Fig. 9.

step. Exact inverse kinematics in real time are impractical, due to the optimization over θ_3. The joints θ_5, θ_6 and θ_7, which control only the orientation of the hand, can be used to maintain it during execution, by using the relationships from Eq. 107 in Appendix A, and picking the solution that is closest to the current joint positions. The values of $\theta_1, ..., \theta_4$ for this calculation can be obtained directly from the sensors. This means that the first 4 joints are following the preplanned spline trajectory, while the last 3 are not. This strategy will counteract the force resulting from the contact between the objects to some extent. The actual orientation of the end-effector will depend on the chosen gain constants and the saturation values for the torques of each joint, see Eq. 17 and Eq. 18. However, this also motivates additional research on a control scheme that can maintain the orientation precisely, while adjusting the position based on a force or impedance based scheme.

The standard impedance control does not follow the trajectory in joint space, and it is therefore more likely to result in an infeasible path. Also, we need to introduce the stopping criteria based on the positional error as in Eq. 32. An additional drawback is that finding the right gains for a particular task can be difficult, but at the same time, since the contributions from the positional and errors in orientation are superimposed in Eq. 21, different gains can be used to increase the relative importance of the orientation error. For example, the libbarrett library provides a `systems::ToolOrientationController` class, through which a desired orientation of the end-effector can be specified separately from the position.

CHAPTER 4: PARAMETER ESTIMATION

Uncertainty in the initial position and orientation of the parts with respect to each other can to some degree be handled by passive compliance, the 'back-drivability' of the robot, or a force-controlled move. However, this essentially constitutes an off-line, open-loop planning strategy. In this chapter, we will go one step further and use the data collected during previous compliant moves to update the parameterization of the part location, thereby introducing a high-level feedback. A geometrical model of the task is used in order to, for deliberately chosen contact states, infer where the contact occurs during the assembly sequence. The parameters of the model, which are known only approximately at the onset of the task, are estimated based on the error between the commanded positions of the end-effector and those observed during a surface compliant exploratory move. As a result, the assembly task can be accomplished more reliably based on the updated model. This is different from constructing a sequence that converges probabilisticly, as is the case for Bayesian methods, and represents the main contribution of this dissertation.

4.1 Overview of the Proposed Strategy

The strategy consists of two separate stages. First a compliant move between a pair of suitable features is performed. Then, the data points collected during the execution are utilized to effectively reduce positioning uncertainties. Thus, the estimation is

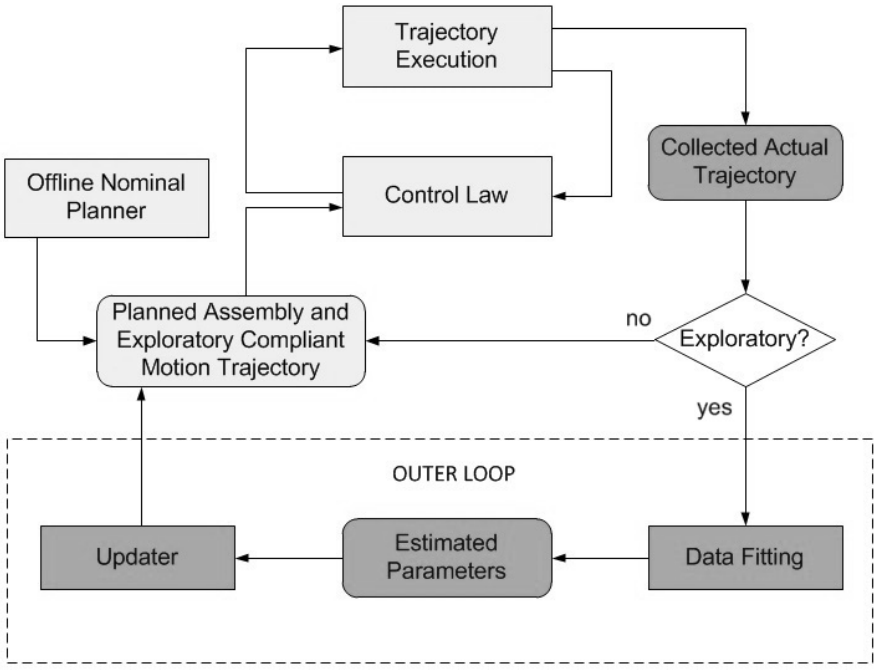


Figure 10: A schematic representation of the parameter update. The dark shaded elements show the operations that have been added to the off-line planner. The feedback in the outer loop does not occur at fixed intervals, but rather only at the end of an exploratory move.

separated from the execution, while a feedback, in terms of updating parameters of the geometrical model that represents the assembly task, is introduced at the end of a compliant move. This is depicted by an additional 'outer' loop in Fig. 10. It is unrelated to the control law used to position the robotic arm. Rather, it builds on top of it by utilizing the additional information that was gained from the previous exploratory moves. If the initial values of the parameters describing the contact formation are accurate enough to ensure that at least the correct contact state is reached, then better estimates can be obtained based on the collected data.

From a practical point of view, we cannot realistically expect to be able to update the parameters more often than every few seconds. The reasons for this are three-fold. First of all, the sampled compliant path must be long enough to contain new

information about the relative poses of the parts. Secondly, a rather large number of collected data points is needed for fitting the model. Thirdly, if the estimation is running without interruption, then it cannot accommodate for more complex models because of high computational requirements that are difficult to handle in real-time. Hence, the outer loop as outlined here can only used in an 'event-based' fashion – with the parameter estimation carried out only after the completion of a deliberately designed exploratory move.

With regard to the estimation of position and orientation of the parts, we will limit our discussion to linear and convex features, such as convex vertices, straight-line edges, and flat or convex faces. As previously stated, such features will be considered to be *regular*. The rationale here is that they can be aligned with each other relatively easily under uncertainty and that it is often even possible to find a closed form solution for the contact points. This does not, however, preclude other features, e.g. those that need to be aligned in the final assembly state, from having more general geometry. Rather, we require that some of the features can be used for achieving active compliant motion and, in addition to that, also the non-deformability of the parts which is necessary for the estimation step. Moreover, for linear convex features the contact will occur in exactly one point or in an invariant set of points w.r.t. one of the objects, and in some cases there is a single contact point even if one of the features is non-linear, see for example Fig. 15 later in this chapter.

In Fig. 11 the contacting features of the parts A and B are denoted as F and G respectively. It is straightforward to express the relationship in terms of homogeneous transformations. From the diagram we readily obtain two matrix equations:

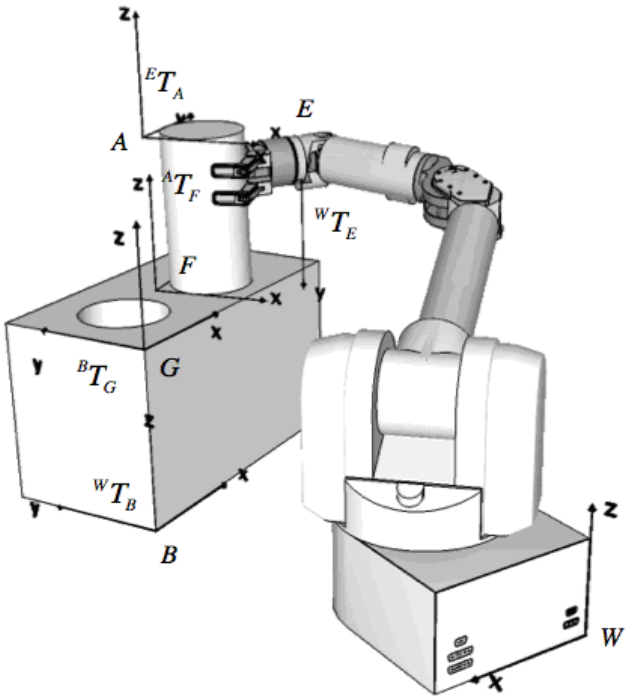


Figure 11: An assembly task expressed in terms of different coordinate frames: W – world, E – end-effector, the parts A and B, and a feature F.

$$ {}^{W}T_{F} = {}^{W}T_{E}\,{}^{E}T_{A}\,{}^{A}T_{F} \tag{33} $$

and

$$ {}^{W}T_{G} = {}^{W}T_{B}\,{}^{B}T_{G}. \tag{34} $$

In general, only the parameters that determine the homogeneous transformation matrices ${}^{E}T_{A}$ and ${}^{W}T_{B}$ reflect the uncertainty in the position and orientation of the objects in the world frame.

The error in ${}^{E}T_{A}$ will be limited by the presumably successful grasping operation before the assembly, see for instance Hsiao et al. [29] and [28]. Therefore, unless otherwise stated, we will assume that the uncertainty in the pose of the part A can be neglected. The one notable exception here will be the cases, in which the orientation of the end-effector can be maintained throughout the exploratory move.

As we will see, being able to do that is useful for aligning linear and convex features under rigid attachment despite errors in ${}^{E}T_{A}$.

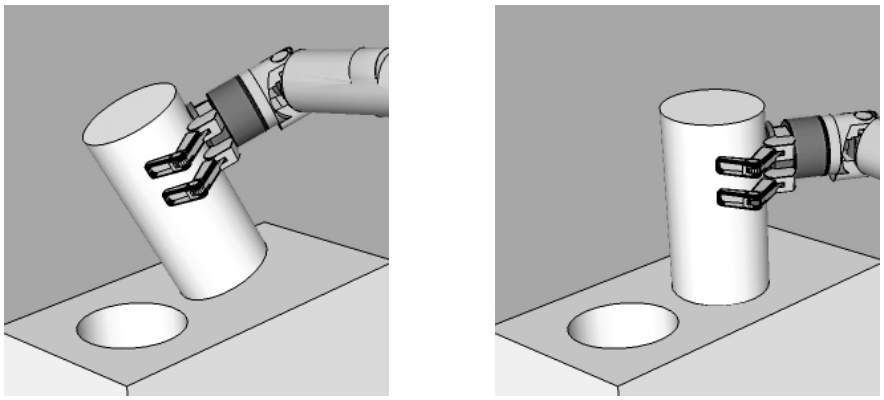
Here, we shall consider only rigid, non-deformable objects. Additionally, we will require that once an object has been grasped and the gripper locks onto the object properly, it also stays fixed with respect to the end-effector, or the gripper frame. Therefore, the part A is assumed to be *rigidly attached to the end-effector*. The part B, on the other hand, is required to be *static*, e.g. clamped down or held by a different robotic arm. Finally, errors due to uncertainty must allow for two suitable, regular features to be reliably brought into contact. Note that, by using the sensed joint positions and instantaneous forward kinematics of the manipulator to infer the corresponding end-effector pose, the rigid attachment of the held object A allows us to calculate the position of the contact point p by applying the homogeneous transformation ${}^{E}T_{A}$.

Also, one might in some situations be able to relax the constraint on ${}^{E}T_{A}$ by fitting the contact data for different assumed values of its parameters and then selecting the values resulting in the lowest least square error. In this case one should ensure that the improvements are large enough not to merely be a result of a rounding error. Here, an appropriate threshold must be set depending on the task. However, varying ${}^{E}T_{A}$ in this manner is computationally intensive and left for future research.

The results on conversion of compliant paths from Sec. 3.2 now allow for all reasoning to be done in the Cartesian space. Moreover, the end-effector position can be obtained in real-time via forward kinematics formulas. In particular, the nominal assembly sequence planning can be done in the task frame, while only assuming the

sensors for the joint positions to be present. The position and orientation of any two objects in the Cartesian space can be described by no more than 12 parameters – corresponding to the 3 rotational and 3 translational DOFs per object. Moreover, contact formations reduce the number of free dimensions further. One can therefore expect to be able to successfully apply numerical optimization methods to the parameter estimation. The necessary assembly steps can, based on the task, be interleaved with exploratory moves that are used for estimating the free parameters of the geometrical model of the task.

It is intuitively clear that contact and compliant motion between rigid objects reduce relative positioning errors under uncertainty. For instance, bringing a face of one object into contact with another object's vertex is clearly easier than aligning the features in the goal assembly state directly. The fact that the features in contact simplify the estimation by reducing the number of free DOFs results in a trade-off, since the more constrained contact states that are difficult to align at the same time provide more information on the relative position and orientation of the parts. In fact, in the goal state of an assembly task, several DOFs need to be constrained, but accomplishing this is precisely at the heart of the problem – if we knew how to do it, there would be nothing left to solve. Clearly, the only way out of this apparent dilemma is for the objects to be aligned gradually, starting with the least constrained contact formation first, as shown in Fig. 12. In the experiments that were carried out, the estimation problem was kept as simple as possible at every step by focusing only on a few of the DOFs in the objects' position and orientation. Depending on the geometry of the features, one might ideally be able to devise an ad hoc assembly



(a) (b)

Figure 12: The less constrained states are generally easier to align. Therefore, gradual alignment of objects is used to assemble the parts. Similarly, if exploratory moves are preformed, then more accurate estimates of the objects' pose can be obtained while focusing only on simpler contact formations and a subset of parameters at each step.

sequence so that the uncertain parameters are determined one at a time.

If the contacting features of the static object B are convex, or sufficiently flat[10], we can recast the parameter estimation as a data fitting problem based on the geometrical model of the task using a two-step approach: First, we realize a compliant translation of a single convex point on one of the objects along a trajectory on a suitable surface of the other object. This constitutes an exploratory compliant move. Then, we estimate the position and orientation of the surface based on the data that was collected during the move using least-squares.

Let us consider the trajectory $traj(x, y, z, t)$ described by the contact point p, which is invariant in the frame of reference of A. If the part A is moved compliantly, the contact point is always going to satisfy the equation of the feature of the part B that it is in contact with. By applying the transformations ${}^{W}T_{E}$ and ${}^{E}T_{A}$, see also Eq. 33, this makes it possible to estimate the parameters of the feature by fitting the model

[10]In terms of the curvature of the surface.

to a set of observed points that lie on $traj(x, y, z, t)$. In case of the point-to-plane contact, we obviously get an infinite number of end-effector configurations for the same contact state, e.g. the point may touch the plane in different spots.

Note that for general features, when the contact point is not invariant w.r.t. the frame of one part, the estimation step is much more involved. If the contact point or set of points – that we will refer to as *contact set*– changes on both objects during the exploratory move, then the pose estimation cannot be recast as a curve fitting problem. In order to handle these more general cases, in Sec. 4.3.2 we will consider convex surfaces represented by polygonal meshes and discuss how a *cost function* measuring the misalignment of features can be calculated.

4.2 Estimation via Curve Fitting

Next we discuss the fitting of parameters for a given geometrical model of the assembly task. The first class that we will consider comprises those estimation problems that can be recast as simple curve fitting. This is always feasible for contact formations in which the contact set consists of a single point on one of the objects that is invariant w.r.t. that object's frame of reference during the execution of the exploratory move. In particular, for features described in terms of linear equations and constraints, we will be able to solve the problem directly using linear least squares by keeping the contact point on the object held by the gripper constant during the move. Non-linear surfaces can be treated in a similar manner by relying on a suitable numerical non-linear least squares algorithm, and assuming that the initial parameter values are close enough so that the issue of local minima does not arise.

4.2.1 Point-to-Plane Contact

When the surface in contact is a plane, i.e. for a point-to-face contact, the curve fitting problem can be solved directly using linear algebra. Let the contact points be given by $p^{(1)}, \ldots, p^{(k)}$, where k is the total number of collected points. For a contact point $p^{(i)} = \left[p_1^{(i)}, p_2^{(i)}, p_3^{(i)}\right]$ with $1 \leq i \leq k$, expressed in the world coordinates, the plane equation is given by $ap_1^{(i)} + bp_2^{(i)} + cp_3^{(i)} + 1 = 0$, where the last constant is set to 1 in order to fix one unknown.[11] The least squares estimate is obtained by solving

$$\left(\sum_i {p^{(i)}}^T p^{(i)}\right) \begin{bmatrix} a \\ b \\ c \end{bmatrix} = -\sum_i {p^{(i)}}^T. \tag{35}$$

Eq. 35 allows us to calculate the plane parameters efficiently, as we only need to solve a linear system with 3 unknowns. We can further express this using matrices as

$$\left(P^T P\right) \begin{bmatrix} a \\ b \\ c \end{bmatrix} = -P^T \mathbf{1}. \tag{36}$$

Here, $\mathbf{1}$ denotes a vector of length k with all entries equal to 1. P is the matrix containing the collected contact points p_i as rows

$$P = \begin{bmatrix} p_1^{(1)} & p_2^{(1)} & p_3^{(1)} \\ \vdots & \vdots & \vdots \\ p_1^{(k)} & p_2^{(k)} & p_3^{(k)} \end{bmatrix}. \tag{37}$$

If the collected contact points are collinear, then the plane normal $[\, a \;\; b \;\; c \,]^T$ can be estimated based on the end effector position, provided that we specify an additional

[11] Alternatively, one could normalize the equation but this requires an unnecessary division.

orthogonality constraint. This allows us to reduce a 3 dimensional problem to a 2 dimensional one, assuming that the object is constrained in one direction by, for example, a flat surface on which the static object rests. To this end, let the column vector $n = [\, n_1 \;\; n_2 \;\; n_3 \,]^T$ denote the normal of the plane to which the collected points are constrained. Now, the error function to be minimized is given by

$$E = \sum_i \left[\left(\left(1 + p^{(i)} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \right)^2 + \left(n^T \begin{bmatrix} a \\ b \\ c \end{bmatrix} \right)^2 \right]. \tag{38}$$

Differentiating w.r.t. the vector $[\, a \;\; b \;\; c \,]^T$ and setting the derivative to zero yields

$$\sum_i \left({p^{(i)}}^T p^{(i)} + n n^T \right) \begin{bmatrix} a \\ b \\ c \end{bmatrix} = -\sum_i {p^{(i)}}^T. \tag{39}$$

Then, by rewriting this formula in matrix notation, we obtain

$$\hat{P}^T \hat{P} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = -\hat{P}^T \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix}, \tag{40}$$

where

$$\hat{P} = \begin{bmatrix} p_1^{(1)} & p_2^{(1)} & p_3^{(1)} \\ \vdots & \vdots & \vdots \\ p_1^{(k)} & p_2^{(k)} & p_3^{(k)} \\ \sqrt{k} n_1 & \sqrt{k} n_2 & \sqrt{k} n_3 \end{bmatrix}, \tag{41}$$

k is the total number of collected points, and $\mathbf{1}$ denotes a column vector containing all 1s that is of length k.

A useful fact from a practical point of view is that, if ${}^{E}T_{A}$ is not known accurately, it will not affect the estimation of the normal *as long as the orientation of the end-effector is maintained during the exploratory move*, i.e. as long as the rotational part of ${}^{W}T_{E}$ can be achieved with relatively high accuracy. In this case, the trajectory of the contact point, or in fact any other point on the part A, will be parallel to the trajectory described by the end-effector as can be seen from Fig. 13. Hence, the offset of the plane equation corresponding to a flat face of the part B will be affected but not its normal vector. Once ${}^{W}T_{B}$ has been updated, we can align the objects one more time in order to determine the orientation of a feature of the object A and then update ${}^{E}T_{A}$ as well.

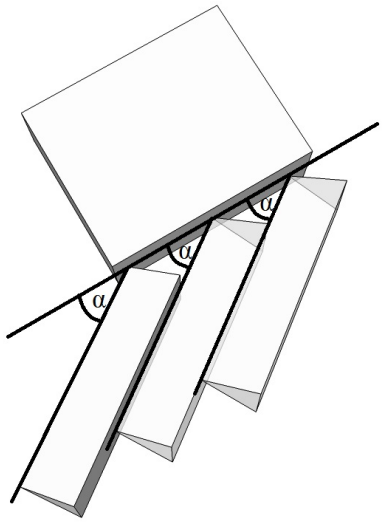


Figure 13: Maintaining orientation during the exploratory move results in parallel trajectories of the contact point and the end-effector under ridged attachment. This makes the estimation of the plane normal on the part B independent of errors in ${}^{E}T_{A}$.

4.2.2 Line-to-Plane Contact

For other more constrained types of contacts, the contacting features must fit together. When a straight edge on the part A is aligned with a flat surface on the part B, the direction vector of this edge, denoted as l, can be used as an orthogonality

constraint. In a completely analogous manner as in Eq. 39 and Eq. 40, we can derive the system of equations

$$\sum_i \left({p^{(i)}}^T p^{(i)} + ll^T \right) \begin{bmatrix} a \\ b \\ c \end{bmatrix} = -\sum_i {p^{(i)}}^T. \tag{42}$$

In order to collect contact points $p^{(1)}, \ldots, p^{(k)}$, we now follow a direction d along the flat face on the part B, and estimate the normal as $d \times l \neq 0$. In other words, since the term ll^T is ensuring that the system of linear equations will be full rank, the sampled contact points can be collinear. Therefore, for a line-to-face contact the data can be collected during an exploratory move along a straight line, i.e. in a single dimension, in terms of the trajectory of the end-effector.

On the other hand, if one of the contacting features is not a straight edge but rather a convex surface then different types of line-to-face contacts will in general lead to a change in the contact set w.r.t. both objects' frames of reference. This is a direct consequence of the difference between the nominal position and orientation of the plane used in planning, and the actual position and orientation encountered during the execution, see Fig. 14 c) and d), and compare to a) and b). Even if the ${}^E T_A$ is known, we cannot expect to always be able to infer the contact point and the direction vector l of the contact line. In some special cases, the direction vector can be known under uncertainty in position and orientation of the part B. For instance, if the part A is a cylinder and its side is contacting a plane, then although the position of the contact changes based on the orientation of the part B, the contact line is always parallel to the axis of the cylinder, see Fig. 14 e) and f). The direction

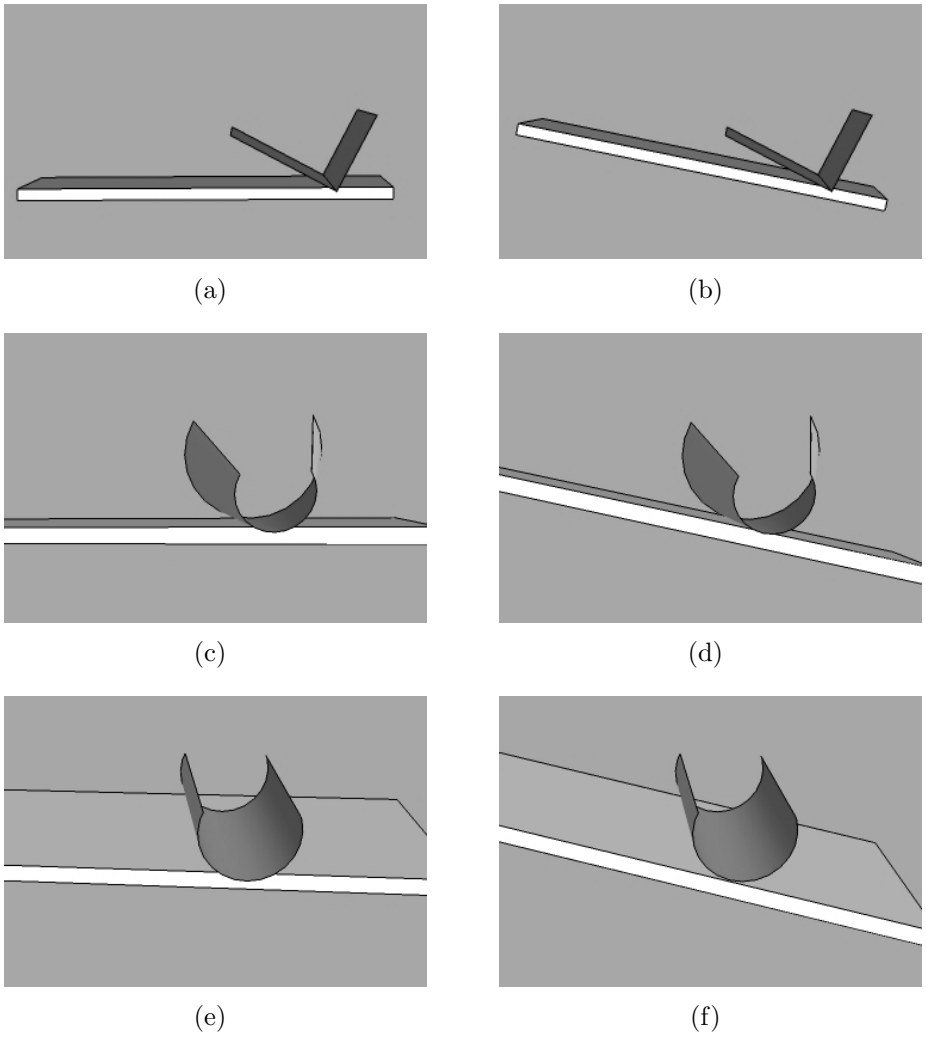


Figure 14: For a straight edge and a plane, as shown in a) and b), the set of contact points is preserved under uncertainty as opposed to c) and d), or e) and f). Nevertheless, the plane normal can be obtained in cases e) and f).

d of the compliant translation of the end-effector is here sufficient to calculate the plane normal using the cross product $d \times l$. The orientation can be inferred from the approach direction that was used to make the initial contact by using the fact that the dot product between the normal and the approach direction vector must be negative. In order to also obtain a contact point, we can use the fact that it is always at the distance r in the direction opposite to n from the central axis of the cylinder, which is known based on ${}^{E}T_{A}$ and the size of the cylinder.

4.2.3 Plane-to-Plane Contacts

Bringing two flat surfaces into contact results in their normal vectors being aligned but opposite. Consequently, the normal of the face of the part B is always known based on that of the part A and vice versa. Clearly, contacts with multiple constrained DOFs tell us more in terms of the relative position of the objects. The difficulty, of course, lies in achieving such alignment in the first place. In order for a face-to-face and edge-to-face contacts to even be possible between linear features, they either need to fit into each other or to be completely flat. Under the condition that there are no additional contacts between the objects, for flat features the only possibility is a plane-to-plane contact with two translational and one rotational degree of freedom. Therefore, assuming that ${}^{E}T_{A}$ is known, the position of the object B can be estimated based on the offset vector of the contacting plane of the object A by averaging over the observed end-effector poses.

4.2.4 Point-to-Face Contact Involving Non-Linear Surfaces

We will limit our discussion to contacts that involve a single convex point – that is approximately constant w.r.t. the frame of the object A during the exploratory move – and a surface, as depicted in Fig. 15. In such cases, one can again relatively easily fit the sampled points to the contacting surface for deliberately planned contact states and estimate the position and orientation of the object. Additionally, a point-to-face contact can sometimes be used as an approximation for contacts involving convex or curved, e.g. round, features. Note, however, that the main difficulty with non-linear features is maintaining the contact, and that the estimation strategy depends on it.

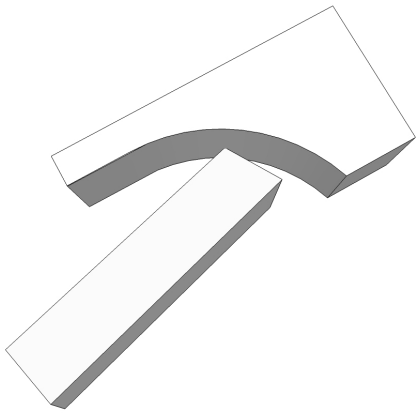


Figure 15: Non-linear least squares curve fitting can be used to estimate the parameters based on the contact between a vertex and a parametric surface. Note that in this example the surface of the object B is not convex, but the curvature of the objects makes it possible to ensure that it is in contact with a vertex of the object A.

The fact that the vertex of the object A is physically in contact, means that calculated contact points should satisfy the equation of the surface of the object B. For the set of contact points ${p_s}^{(1)}, \ldots, {p_s}^{(k)}$, obtained at different times during the exploratory move and the equation of a surface on the part B given by $S(\alpha_1, \ldots, \alpha_m, x) = 0$ for $x \in S$, we define

$$E\left(r, \phi, q\right) = \sum_{i=1}^{k} S(R\left(r, \phi\right) p^{(i)} + q)^2, \tag{43}$$

and solve the minimization problem $\min_{r,\phi.q} E\left(r, \phi, q\right)$. Here, $R\left(r, \phi\right)$ is a rotational matrix, q is the offset in the Cartesian space, and r and ϕ are the rotational axis and angle respectively. Note that $\alpha_1, \ldots, \alpha_m$ are now treated as constants and can therefore be omitted, since optimizing the cost function does not change the shape of the surface. Additionally, in order to simplify the notation, we set

$$\tilde{S}(r, \phi, q, p) := S(R\left(r, \phi\right) p + q). \tag{44}$$

This is a non-linear optimization problem that can be tackled using numerical meth-

ods. One of the most effective non-linear fitting algorithms is due to Marquardt [50] and Levenberg [47]. It is related to the Newton's method in one dimension, as it uses a linear approximation of the function in the neighborhood of the current solution. In higher dimensions, a system of linear equations needs to be solved using least squares at every iteration. Applied to our problem we have

$$\min_{\Delta\phi,\Delta q} \sum_i \left[\tilde{S}(r,\phi,q,p^{(i)}) + \left(\frac{\partial \tilde{S}(r,\phi,q,p^{(i)})}{\partial \phi} \right) \Delta\phi + \left(\frac{\partial \tilde{S}(r,\phi,q,p^{(i)})}{\partial q} \right)^T \Delta q \right]^2 \quad (45)$$

for a fixed rotational normal. In Eq. 45 the last partial derivative is taken w.r.t. the vector q, and the rotational vector is treated as a constant. This is now a linear least squares problem. Using matrix notation, we have

$$P^T P \begin{bmatrix} \Delta\phi \\ \Delta q \end{bmatrix} = P^T r, \quad (46)$$

where

$$P(r,\phi,q) = \begin{bmatrix} \frac{\partial \tilde{S}(r,\phi,q,p^{(1)})}{\partial \phi} & \frac{\partial \tilde{S}(r,\phi,q,p^{(1)})}{\partial q_1} & \frac{\partial \tilde{S}(r,\phi,q,p^{(1)})}{\partial q_2} & \frac{\partial \tilde{S}(r,\phi,q,p^{(1)})}{\partial q_3} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \tilde{S}(r,\phi,q,p^{(k)})}{\partial \phi} & \frac{\partial \tilde{S}(r,\phi,q,p^{(k)})}{\partial q_1} & \frac{\partial \tilde{S}(r,\phi,q,p^{(k)})}{\partial q_2} & \frac{\partial \tilde{S}(r,\phi,q,p^{(k)})}{\partial q_3} \end{bmatrix} \quad (47)$$

and

$$r(r,\phi,q) = - \begin{bmatrix} \tilde{S}(r,\phi,q,p^{(1)}) \\ \vdots \\ \tilde{S}(r,\phi,q,p^{(k)}) \end{bmatrix}, \quad (48)$$

which constitutes the Gauss-Newton step. A damping factor is introduced in order to find the right step size for which the linear approximation is valid. This strikes a balance between the Gauss-Newton algorithm and gradient descent. Marquardt [50] showed that performing the above minimization subject to the constraint $\| [\, \Delta\phi \quad \Delta q \,]^T \|_2 = C$ for some constant C, is equivalent to solving the system of linear equations

$$\left[\; P^T P + \lambda I \; \right] \left[\begin{array}{c} \Delta\phi \\ \\ \Delta q \end{array} \right] = P^T r, \tag{49}$$

where λ now must be non-negative in order for the matrix to be semi-positive definite, i.e. corresponding to a minimum. Clearly, we do not know a priori what the right values of λ or C should be. However, for every λ the solution is the best update for a given radius. Moreover, $\| [\, \Delta\phi \quad \Delta q \,]^T \|_2$ is a monotonically decreasing function of λ. Hence, one needs to find the smallest λ for which there is an improvement, either because $\| [\, \Delta\phi \quad \Delta q \,]^T \|_2$ is close to the unconstrained value, or because of the proper choice for the boundary of the area on which the linear approximation to $\tilde{S}(r, \phi, q, x)$ is valid. A good estimate for λ can be found by decreasing its value if there was an improvement resulting from the last update and retracting after unsuccessful steps. Since $P^T r$ is, in fact, the negative gradient of $E\,(r, \phi, q)$, the method resorts to gradient descent for large λ as the parameter update vector $[\, \Delta\phi \quad \Delta q \,]^T$ approaches the direction of $P^T r$.

It remains to calculate the partial derivatives of the cost function w.r.t. the parameters describing the pose of the object B. Let us assume that the partial derivatives are known for a given surface, then that means that the derivatives w.r.t. q are also

known, since

$$\frac{\partial \tilde{S}}{\partial q_i} = \frac{\partial S}{\partial x_i}. \tag{50}$$

Furthermore, the chain rule yields

$$\frac{\partial \tilde{S}}{\partial \phi} = \sum_{i=1}^{3} \frac{\partial S}{\partial x_i} \frac{\partial}{\partial \phi} \left(\sum_{j=1}^{3} R_{i,j} p_j + q_i \right) = \sum_{i=1}^{3} \frac{\partial S}{\partial x_i} \sum_{j=1}^{3} \frac{\partial R_{i,j}}{\partial \phi} p_j, \tag{51}$$

where R_{ij} is the corresponding entry of the matrix $R(r, \phi)$ and r is normalized. $\dfrac{\partial R(r, \phi)}{\partial \phi}$ can be written as

$$\begin{bmatrix} {r_1}^2 \sin(\phi) - \sin(\phi) & r_1 r_2 \sin(\phi) - r_3 \cos(\phi) & r_1 r_3 \sin(\phi) + r_2 \cos(\phi) \\ r_1 r_2 \sin(\phi) + r_3 \cos(\phi) & {r_2}^2 \sin(\phi) - \sin(\phi) & r_2 r_3 \sin(\phi) - r_1 \cos(\phi) \\ r_1 r_3 \sin(\phi) - r_2 \cos(\phi) & r_2 r_3 \sin(\phi) + r_1 \cos(\phi) & {r_3}^2 \sin(\phi) - \sin(\phi) \end{bmatrix} \tag{52}$$

for r of unit length. These formulas can be used to update the position and orientation for a known axis of rotation. It is possible, in principle, to try to estimate the rotation axis as well, but it is better to construct other exploratory moves for which it remains fixed.

4.3 Estimation Using Convex Features

For our purposes, convex features are the most important class. There are two reasons for this – one is practical and the other computational. They are easier to reliably bring into contact under uncertainty, and the contact point calculation can be performed numerically using local optimization methods. So far we have assumed that the set of contact points, as expressed in ${}^{E}T_{A}$, will stay at least approximately

the same for the duration of the exploratory move. However, when two surfaces or an edge and a surface, as opposed to a vertex and a surface, are in contact, the contact set is generally going to change even for regular, convex features. Hence, we now allow the contact point between two selected features to occur anywhere as long as the contact state is maintained. Also, note that due to uncertainty, meshes cannot be treated by simply reducing the size of the features as that would simply not work with our approach.

4.3.1 Contact Formations Involving Convex Features

Let us consider what happens when the base of a cylinder is moved along a plane while forming an edge-to-face type of contact. Assume, however, that the cylinder also deviates form the nominal trajectory. If the orientation of the cylinder is not maintained during the exploratory move, i.e. if it turns due to the contact and uncertainty in the positioning of the objects, the cylinder can 'roll' off the surface and consequently the contact point on the edge of the cylinder changes although the contact type remains the same. Our next step will be to try to consider possible solutions to the resulting estimation problem.

One conceivable approach consists of finding the intersection points based on the equations of the circle and the plane explicitly, and selecting parameters so that, at least for a convex edge and a plane, there is ideally only one such point for every collected end-effector pose. Let the vector $\vec{q}$ denote the center of the circle shown in the Fig. 16, another vector $\vec{n}$ its normal, and let $\vec{q}+\vec{v}$ be a point on the circumference. We further obtain an orthonormal 3D basis by calculating the vector $\vec{u} = \vec{n} \times \vec{v}$, where

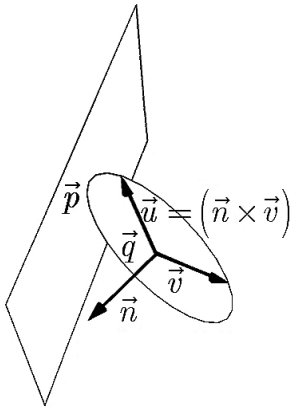


Figure 16: Contact between a circle (edge) and a plane in 3D.

$\vec{q}+\vec{v}$ is another point on the circumference. The equation of the circle can be expressed as $\vec{q}+\cos(\phi)\vec{u}+\sin(\phi)\vec{v}$, where $-\pi < \phi \leq \pi$. Also, let the plane equation be given by $ax + by + cz + 1 = 0$, and let the contact point between the two be specified by the vector $\vec{p}$ on the circumference of the circle. Our goal is to find the parametrization of the plane, so that there is exactly one solution for $\vec{p}$. Substituting, we obtain

$$\begin{gathered} a\left[\mathrm{q}_x + \mathrm{u}_x \cos(\phi) + \mathrm{v}_x \sin(\phi)\right] + b\left[\mathrm{q}_y + \mathrm{u}_y \cos(\phi) + \mathrm{v}_y \sin(\phi)\right] + \\ c\left[\mathrm{q}_z + \mathrm{u}_z \cos(\phi) + \mathrm{v}_z \sin(\phi)\right] + 1 = 0. \end{gathered} \tag{53}$$

Now, applying the identities $\sin(\phi) = \dfrac{2\tan\left(\frac{\phi}{2}\right)}{1+\tan\left(\frac{\phi}{2}\right)^2}$ and $\cos(\phi) = \dfrac{1-\tan\left(\frac{\phi}{2}\right)^2}{1+\tan\left(\frac{\phi}{2}\right)^2}$ yields

$$\tan\left(\frac{\phi}{2}\right) \rightarrow \frac{-K \pm \sqrt{K^2 + M^2 - (1+L)^2}}{1+L-M}, \tag{54}$$

where $K = av_x + bv_y + cv_z$, $L = aq_x + bq_y + cq_z$, and $M = au_x + bu_y + cu_z$. There are only three possible cases: one, two or no intersection points between a circle and a plane in 3 dimensions. So, the following equality

$$(av_x + bv_y + cv_z)^2 + (au_x + bu_y + cu_z)^2 = (1 + aq_x + bq_y + cq_z)^2 \tag{55}$$

must be satisfied when the edge and the plane are in contact. All expressions in Eq. 55 can be calculated based on the vectors $\vec{q}$, and $\vec{q}+\vec{v}$ in world coordinates for each end-effector pose. A suitable cost function can be obtained by summing over the squares of the error for individual end-effector positions (Eq. 55). Here, the discriminant of the expression in Eq. 54 is used to indirectly measure the distance between the cylinder and the plane. The resulting function is

$$\begin{aligned}\min_{a,b,c} E(a,b,c) = \min_{a,b,c} \frac{1}{4} \sum_i \Big[\left(a{v_x}^{(i)} + b{v_y}^{(i)} + c{v_z}^{(i)}\right)^2 + \left(a{u_x}^{(i)} + b{u_y}^{(i)} + c{u_z}^{(i)}\right)^2 \\ - \left(1 + a{q_x}^{(i)} + b{q_y}^{(i)} + c{q_z}^{(i)}\right)^2 \Big]^2,\end{aligned} \tag{56}$$

where i goes over all recorded end-effector positions, which are obtained from the joint configurations and forward kinematics of the manipulator. This is a non-linear minimization problem, which can be solved, for example, with the Leveberg-Marquardt method. The partial derivatives for a *single* end-effector position are

$$\begin{aligned}\frac{\partial E^{(i)}}{\partial a} &= \mu\left(\alpha a + \beta b + \gamma c - q_x\right) \\ \frac{\partial E^{(i)}}{\partial b} &= \mu\left(\beta a + \delta b + \epsilon c - q_y\right) \\ \frac{\partial E^{(i)}}{\partial c} &= \mu\left(\gamma a + \epsilon b + \zeta c - q_z\right),\end{aligned} \tag{57}$$

with $\mu = (av_x + bv_y + cv_z)^2 + (au_x + bu_y + cu_z)^2 - (1 + aq_x + bq_y + cq_z)^2$, $\alpha = -{q_x}^2 + {u_x}^2 + {v_x}^2$, $\beta = -q_xq_y + u_xu_y + v_xv_y$, $\gamma = -q_xq_z + u_xu_z + v_xv_z$, $\delta = -{q_y}^2 + {u_y}^2 + {v_y}^2$, $\epsilon = -q_yq_z + u_yu_z + v_yv_z$ and $\zeta = -{q_z}^2 + {u_z}^2 + {v_z}^2$. However, for more complicated parts, calculating the set of all possible contact points and selecting the parameter values that correspond to some desired contact is simply not feasible.

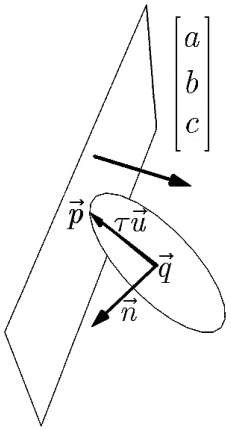


Figure 17: Calculation of the distance and penetration depth between a circle (edge) and a plane in 3D.

A better way to perform the estimation step is to look at the penetration depth whenever the features overlap in the model, and the distance between the features otherwise. This directly addresses the fact that the correct parametrization is characterized by the contact between features. Hence, a – possibly weighted – sum of the distance and penetration depth values between the objects over all observed instances should be minimized. In our example, as depicted in Fig. 17, we now consider the signed distance between the vector $\vec{p}$, which corresponds to the point on the circle that is the farthest in the direction of the negative normal of the plane $[\,-a\ -b\ -c\,]^T$, and the plane itself. The signed distance between the features is negative in cases when penetration occurs. On the other hand, if the signed distance between the features is positive, then it will be equal to the distance between the circle and the plane. The vector $\tau\vec{u}$ points from the center of the circle $\vec{q}$ in the direction of the point represented by $\vec{p}$. One can calculate $\vec{u}$ by projecting the negative normal of the plane onto another plane defined by the circle, where the normal of the circle is denoted as $\vec{n}$. Using the matrix notation, this relationship can be expressed as

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = - \begin{bmatrix} a \\ b \\ c \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \begin{bmatrix} n_x & n_y & n_z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{58}$$

for $\sqrt{{n_x}^2 + {n_y}^2 + {n_z}^2} = 1$. Now, rescaling and adding $q = [q_x, q_y, q_z]^T$, we obtain $p = [p_x, p_y, p_z]^T$ and the signed distance D as

$$p = q + \frac{R}{\sqrt{{u_x}^2 + {u_y}^2 + {u_z}^2}} u, \tag{59}$$

and

$$D = \frac{ap_x + bp_y + cp_z + 1}{\sqrt{a^2 + b^2 + c^2}} \tag{60}$$

respectively. Finally, dropping the factor $\sqrt{a^2 + b^2 + c^2}$ and simplifying the expression we arrive at

$$\hat{D}\left(a, b, c, q, n\right) = aq_x + bq_y + cq_z + 1 - R\sqrt{a^2 + b^2 + c^2 - \left(n_x a + n_y b + n_z c\right)^2}, \tag{61}$$

which is the signed error for the vectors n and q based on a single end-effector position. In order to form a suitable error function, we sum the squares of these values over all end-effector positions denoted by the index i, resulting in

$$E\left(a, b, c\right) = \frac{1}{2} \sum_i \hat{D}\left(a, b, c, q^{(i)}, n^{(i)}\right)^2, \tag{62}$$

where $n^{(i)}$ and $q^{(i)}$ can be obtained based on the sensed joint positions, the matrix ${}^E T_A$, and the object's geometry. We need to compute $\min\limits_{a,b,c} E\left(a, b, c\right)$. The partial derivatives for a *single* observed data point

$$\frac{\partial E^{(i)}}{\partial a} = \left(q_x + R\frac{{n_x}^2 a + n_x n_y b + n_x n_z c - a}{\sqrt{a^2 + b^2 + c^2 - (n_x a + n_y b + n_z c)^2}} \right) \hat{D}\,(a, b, c, q, n)$$

$$\frac{\partial E^{(i)}}{\partial b} = \left(q_y + R\frac{n_x n_y a + {n_y}^2 b + n_y n_z c - b}{\sqrt{a^2 + b^2 + c^2 - (n_x a + n_y b + n_z c)^2}} \right) \hat{D}\,(a, b, c, q, n) \tag{63}$$

$$\frac{\partial E^{(i)}}{\partial c} = \left(q_z + R\frac{n_x n_z a + n_y n_z b + n_z^2 c - c}{\sqrt{a^2 + b^2 + c^2 - (n_x a + n_y b + n_z c)^2}} \right) \hat{D}\,(a, b, c, q, n).$$

can now be used to estimate the parameters for this contact using, for example, the Levenberg-Marquardt algorithm.

Moreover, we see that it is possible to generalize this approach: As long as we are able to numerically calculate the distance and the penetration depth for the features, both values can be minimized over a set of valid parameters that describe the pose of the objects, i.e. free DOFs, in order to obtain better estimates.

4.3.2 Error Calculation for Polygonal Meshes

Describing the contact formations based on a set of equations can be cumbersome for more complicated contact features and in many cases a closed solution does not exist. Although a solution can be found in an ad hoc way by relying on numerical methods to find the solution of a set of equations in certain cases, it would be advantageous to be able to optimize the parameters of ${}^{E}T_{A}$ and ${}^{W}T_{B}$ for features that are specified using polygonal meshes. In this section, we will give an algorithm that calculates the cost function for contacts between two convex features approximated by sufficiently densely sampled polygonal meshes. To this end, we define the energy

```
1: procedure CalculateError(A, B, ^E T_A, ^W T_B, collected end-effector poses )
2:     sum ← 0
3:     count ← 0
4:     ▷ Transform the coordinates of vertices of object B based on ^W T_B
5:     ^W B ← GetMeshGlobal(B, ^W T_B)
6:     ▷ Calculate the average MTD+
7:     for all ^W T_E ∈ collected end-effector poses do
8:         ▷ Transform the coordinates of vertices of the object A via ^E T_A and ^W T_E
9:         ^W A ← GetMeshGlobal(A, ^W T_E ^E T_A)
10:        ▷ MTD+ is used as a measure of 'displacement'
11:        sum ← sum + CalculateMTD+ (^W A, ^W B)
12:        count ← count + 1
13:    end for
14:    if count > 0 then
15:        return sum / count
16:    else
17:        return +∞
18:    end if
19: end procedure
```

Figure 18: The computation of the cost function for a contact between two convex polygonal meshes can be done using MTD^+ by transforming the meshes via ${}^E T_A$ and ${}^W T_B$ based on the current parameter estimates.

function as the sum of MTD^+ values[12] for the two meshes over all sampled end-effector poses. Since there are at most 12 free parameters for two objects, with additional physical constraints for a particular contact state and features reducing this number further, the parameter optimization can be carried out using direct methods, such as Hooke-Jeeves [27] or Nelder-Mead [55]. Unfortunately, the expected computational complexity is $O(knmlog(nm))$, where n and m are the number of vertices that comprise the meshes, and k is the number of sampled end-effector poses. This precludes the algorithm from running in real-time, but reasonable responsiveness could be obtained by using parallelism on suitable hardware.

[12] As defined in Sec. 2.4.

The pseudocode for computing the energy function is shown in Fig. 18. The input parameters are the meshes A and B, the poses ${}^{E}T_{A}$ and ${}^{W}T_{B}$ obtained for the current parameter estimates, and the end-effector poses that were sampled during an exploratory move. The algorithm loops over all collected end-effector poses, see lines 6-12. The calls *GetMeshGlobal* $\left(B, {}^{W}T_{B}\right)$ and *GetMeshGlobal* $\left(A, {}^{W}T_{E}{}^{E}T_{A}\right)$ change the frame of reference by applying the necessary transforms to every vertex individually. In order to obtain the pose of A we multiply ${}^{W}T_{E}$ and ${}^{E}T_{A}$. Note that ${}^{W}T_{E}$ is calculated using the collected, i.e. measured, joint positions. In line 11, meshes ${}^{W}A$ and ${}^{W}B$ expressed in the world frame are passed to the function that computes MTD^{+}. The return value of *CalculateError* is the average over the MTD^{+} values.

To compute MTD^{+}, see Fig. 19, we first form the Minkowski difference between the objects in order to obtain the translational configuration space obstacle $A-B$. Then, in the next step we calculate the distance of the closest feature to the origin. Since $A-B$ is convex, see for example Latombe [42], it can be identified by finding the closest vertex, then the closest edge, and finally the closest polygon in the neighborhood of the vertex. The crux of the problem is finding $A-B$. This was, however, discussed in Sec. 2.4, and can for instance be calculated in $O(nmlog(nm))$ steps using the Quickhull algorithm. Note that regardless of what algorithm is used to calculate the convex hull, only the *surface* points, edges and faces of $A-B$ will be needed for the subsequent operations.

However, the solution might not be unique and the algorithm itself does not take care of spurious solutions that might occur based on the parameters of ${}^{E}T_{A}$ and ${}^{W}T_{B}$, i.e. their chosen range. Hence, parameter constraints based on the physical model

```
1: procedure CalculateMTD+(A, B)
2:     ▷ Create the translational configuration space obstacle for meshes A and B.
3:     tsco ← MinkowskiSum3D (A, −B)
4:     d ← inf
5:     v_min ← any vertex v ∈ tsco
6:     for all vertices v ∈ tsco do      ▷ Find the vertex that is closest to the origin.
7:         if ‖v‖_2 < d then
8:             d ← ‖v‖_2
9:             v_min ← v
10:        end if
11:    end for
12:    e_min ← any edge e ∈ tsco connected to v_min
13:    for all edges e ∈ tsco connected to v_min do      ▷ Find the closest edge.
14:        if distanceEdgePoint(e, O) < d then
15:            d ← distanceEdgePoint(e, O)
16:            e_min ← e
17:        end if
18:    end for
19:    ▷ Find the distance to the closest polygon.
20:    for all polygons p ∈ tsco connected to e_min do
21:        if distancePolygonPoint(p, O) < d then
22:            d ← distancePolygonPoint(p, O)
23:        end if
24:    end for
25:    return d
26: end procedure
```

Figure 19: By relying on the Minkowski sum, see Sec. 2.4, an algorithm for computing the minimal translational distance between two convex meshes can be implemented.

are needed, as the objects' poses might otherwise be parametrized so that according to the model one of the parts is, for instance, inside of the other, on the wrong side of a feature, etc. The algorithm does not use the normals, nor does it have any other means to tell the inside of a mesh from the outside, or the correct side of a plane, etc. Obviously, since the minimization problem does not always have a unique solution, it will not be not convex in general – although the features themselves are. Therefore, we also require the initial guess to be in the neighborhood of the solution. Examining this optimization problem in more detail, however, is left for future research. Also,

there is the challenge of possibly devising a more efficient algorithm to perform these calculations and to provide a robust software library implementation.

4.4 Parameter Update and Path Adjustment

Using feature pairs for estimation implies that not all of the DOFs of an object will be known after a single exploratory move. This might at first seem as a disadvantage, but it actually comes down to devising a strategy of finding the right sequence of moves across different contact states in order to estimate all relevant parameters related to the pose of B. Every new exploratory move and the subsequent estimation of the parameters based on the features in contact provides new information that can be used to update the relative position and orientation of the objects in the model. We treat the estimation of homogeneous transformations of the held object A and the static object B separately, and proceed one DOF at a time whenever possible. This allows us to replace the original problem with a series of simpler ones. As previously discussed, the estimation process can be interleaved with an assembly sequence by alternating exploratory moves and assembly steps. Also, for the purposes of estimating the orientation of the object B, the uncertainty in ${}^{E}T_{A}$ can be handled by maintaining the orientation of the end-effector throughout the exploratory move and using a fixed contact point w.r.t. A's frame of reference.

Note that we are making a clear distinction between the contact, or the lack of it, according to the model, and the fact that the actual parts are in contact during a compliant move. The former is conceptual and pertains only to the model, whereas the latter can only be observed during execution. The computations in the model

itself can of course be performed according to any parametrization. Changing the pose of B in one dimension, be it rotational or translational, while at the same time maintaining the contact with the object A at point p, will in general require correcting other parameters as well, so that the objects stay in contact at p according to the model that is based on the new parametrization. The simplest way to do this is by transforming p from the frame of the object A to the world frame based on ${}^{E}T_{A}$, choosing the closest point to p on the face of the object B, and then applying a translation to the estimated pose of the object B to make the points coincide.

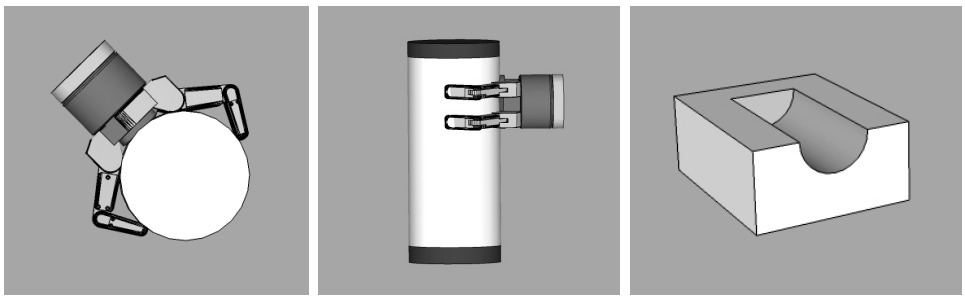
The high-level assembly sequence in Cartesian space does not necessarily have to be discarded after the estimation. Instead, we can use the same task-based assembly sequence to replan. However, the updated parameter values will result in a new trajectory for the remaining portion of the path, and we will not obtain a continuous trajectory at the end of the previous exploratory move. Obviously, if the parameterization of the model stayed the same, then the last configuration of the exploratory move would result in a continuous joint space trajectory, but this is not the case. Therefore, after the pose of the object B has been updated in the model, we need to re-compute the portion of the path of the object A between the current pose and the goal state. The remainder of the planned path could become infeasible due to the static object B now being estimated as outside of the dexterous space of the robotic manipulator according to the new parameterization. In that case, the assembly cannot be accomplished – at least not without moving the robotic arm.

If the path calculated by the conversion algorithm is feasible, then the path segments need to be patched at the end of the exploratory move. Let C_p denote the last

joint space configuration of the exploratory move that was used to collect the data for the estimation. Since the conversion algorithm works its way backwards form the goal pose, C_p will not seamlessly connect with the joint space configuration corresponding to the first knot point of the remainder of the path. From the practical point of view, if the difference between the two paths is not large, it will suffice to reconnect them using the shortest path in the joint space, i.e. by interpolating directly in the joint space for all joint trajectories between the configuration C_p and the configuration C_S that corresponds to the new starting end-effector pose for the remaining path segment in the joint space. One can also search the remaining joint space path for the configuration C, which is the closest one to C_p in terms of the Manhattan distance, i.e. $C_{min} = \min_{C} \|C - C_p\|_1$. The configuration C_{min} may not be at the beginning of the remaining joint space, i.e. $C_{min} \neq C_S$, and we may therefore obtain a more direct trajectory by skipping a portion of it. Path segments will typically have to be patched at contact state transitions. In such cases, in order to avoid unintended collisions due to the update under uncertainty, one will want to ensure that there is sufficient clearance between the objects by lifting the held object away from the static one in order to move it around an edge or a feature.

4.5 An Assembly Example with Estimation

In order to illustrate the concepts that have been introduced, let us now use a concrete example. Denote the cylinder and the cradle-shaped concave object in Fig. 20 as part A and part B respectively. The gripper is holding the free end of the part A, which is therefore rigidly attached to the robotic arm for all practical purposes.



(a) (b) (c)

Figure 20: The subfigures a) and b) show the object A, and c) the static object B.

The part B is secured and will not move if pressure is applied. It is placed on a flat surface of known height and we further assume that the surface normal is given.

We note that the cylinder, i.e. part A, once it has been properly grasped as shown in Fig. 20 a) and b), will have very little uncertainty in directions orthogonal to the axis of the cylinder. Similarly, the position along the axis, at which the gripper is holding the object, is determined by the height of the surface from where it was picked up. In Sec. 3.2, it was shown how to convert assembly sequences from the task space to the joint space. All planning can therefore be done in the task space, which considerably simplifies the reasoning. Note that the coordinate system of the part B is updated immediately after the estimation at the end of the exploratory moves.

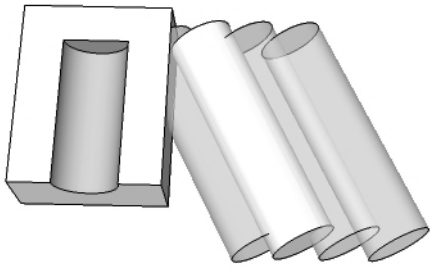


Figure 21: Guarded motion is used to bring the objects into contact by nominally planning for penetration. This also enables us to realize approximate compliant motion without a force sensor by relying on 'back-drivability'.

As the first estimation step, consider the contact formation shown in Fig. 21. Due to the fact that the edge is a circle, there is only one contact point. To ensure compliance, we extend the guarded move when the part A is approaching the part B, so that they would nominally overlap according to the model. During the execution one can rely on the torque limits and 'back-drivability' of the robot or, better yet, set the force limit if a force sensor is available to terminate the move.

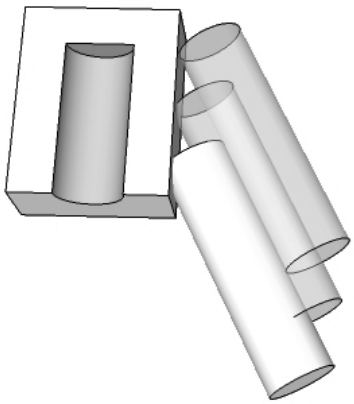


Figure 22: An exploratory move to determine the orientation of the object B.

In order to estimate the orientation, we probe the alignment of the object B by sliding the cylinder along its surface. Without a force sensor, the path of the cylinder must nominally penetrate the object B as depicted in Fig. 22. The actual executed trajectory will be constrained along the surface of the object B as a point to plane contact. Based on ${}^{E}T_{A}$, we can calculate the contact points in Cartesian space by solving the system of linear equations

$$\begin{aligned} a{p_x}^{(i)} + b{p_y}^{(i)} + c{p_z}^{(i)} + 1 = 0 \\ \lambda a n_x + \lambda b n_y + \lambda c n_z = 0 \end{aligned} \tag{64}$$

in the least-squares sense, where the contact points and the table normal are given

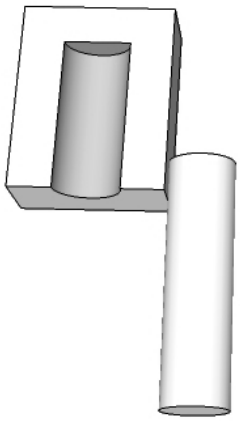


Figure 23: Aligning features based on the updated model parameters.

by $\left({p_x}^{(i)}, {p_y}^{(i)}, {p_z}^{(i)}\right)$ and (n_x, n_y, n_z) respectively, and i goes over all collected points.

In order to normalize, the last constant in the plane equation is set to 1 for all data points and λ to the square root of the number of collected points, see Eq. 41. The formulas are, strictly speaking, applicable only to a point-to-plane contact. However, if we consider the 2D projection of both objects onto the flat surface on which the part B is secured, we have a contact between a rectangle and a line. Therefore, the cylinder is kept parallel to the, e.g. table, surface and any change in the orientation during execution is neglected. To approximately calculate ${}^{W}T_A$ for every sampled joint configuration vector, we use ${}^{E}T_A$ and the forward kinematics of the manipulator. After the estimation is complete, we can already align the edges of the two objects, see Fig. 23), since this is the same alignment that we need for the final assembly.

Next, we lift the object A above the convex edge of the object B in order to preclude an unintended collision from happening, and then put the object A down onto the object B, followed by inserting the object A into the concave opening of the part B, as shown in Fig. 24.

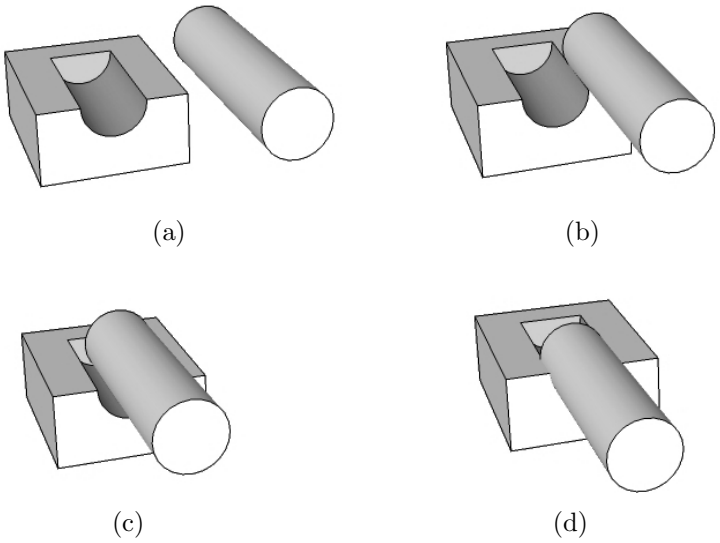


Figure 24: A simplified assembly problem based on the previous alignment.

We do not yet know how far the cylinder should be slid along the concave slot of the part B, as shown in Fig. 25. Nevertheless, as long as we leave a large enough margin in the remaining direction, we can place the part A into the concave 'cradle', see a), and then move the part A as far as necessary. To ensure contact, we once again nominally plan for overlap and terminate based on a force or torque threshold. The last uncertain parameter can be determined in this manner, as shown in b).

Experimental results for this example can be found in Chapter 5.

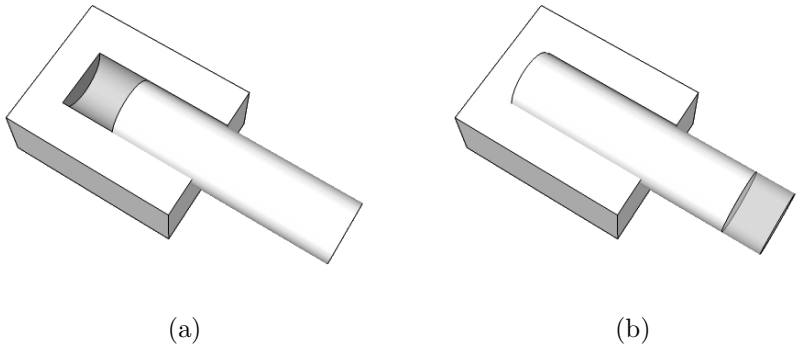
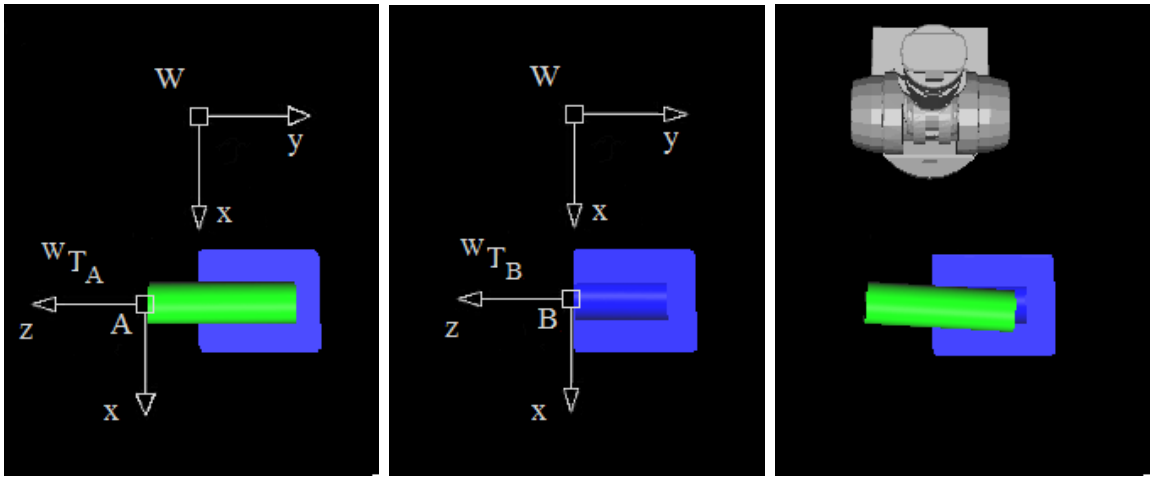


Figure 25: The last uncertain parameter can be determined by sliding the object A into position while nominally planning for penetration at the back of the slot.

CHAPTER 5: EXPERIMENTAL RESULTS

5.1 Implementation

A Barrett 7-DOF WAM robotic arm and a BarrettHand BH8-262 gripper were used in all experiments. The libbarrett software library was relied upon for low-level control, and both the path conversion algorithm and the individual examples were implemented on top of it. The initial uncertainty of the part A was engineered away, and a firm grasp was ensured in order to obtain approximately rigid attachment to the end-effector. The assembly sequences were planned in terms of a Cartesian compliant path of the held part A going through a number of contact states with the clamped down part B. The resulting contact configurations were automatically converted to a singularity-free joint space trajectory via the algorithm from Sec. 3.2. Guarded and compliant moves use the control scheme from Sec. 2.3.2, with the torque limits for the joints 1 through 7 set to `control_signal_limit = (10 Nm, 15 Nm, 5 Nm, 5 Nm, 1 Nm, 2 Nm, 1 Nm)` respectively. The control loop frequency was 500Hz. The threshold for the positional error of the end-effector, see Eq. 32, was 2cm. For the part A a cylinder with $3\frac{1}{4}$" in diameter was used that fits exactly into the concave slot of the part B. After every parameter update, the remainder of the sequence was reconverted based on the same high-level topological plan, while using the new parameters determining the pose of the object B. The path between the sensed joint



(a) (b) (c)

Figure 26: The origin of the world coordinate system is at the shoulder joint of the robot. The local coordinate systems are shown in a) for the held part A and in b) for the static part B. c) shows the deviation in the final pose of part A after assembly.

positions at the end of the exploratory move and the closest joint space configuration from the remainder of the path was patched using a straight line in the joint space, see Sec. 4.4.

5.2 Nominal Path Planning and Conversion

Fig. 27 shows an assembly sequence that relies on compliant motion, and Fig. 28 the corresponding forces at the end-effector that were calculated according to Eq. 22 based on the sensed values of the joint torques. This is also true for all other force plots later in this section. The time axis units are in seconds, and the forces are measured in newtons. The axes used in the plot are those of the world reference frame, depicted in Fig. 26. It was chosen so as to be identical to that of the base frame of the manipulator and is located at the shoulder joint, see Fig. 2 and Fig. 42. In Fig. 26 c) we see the difference in the nominal pose of part B and actual pose of part A in the goal state. By the actual pose we mean the pose that is inferred based

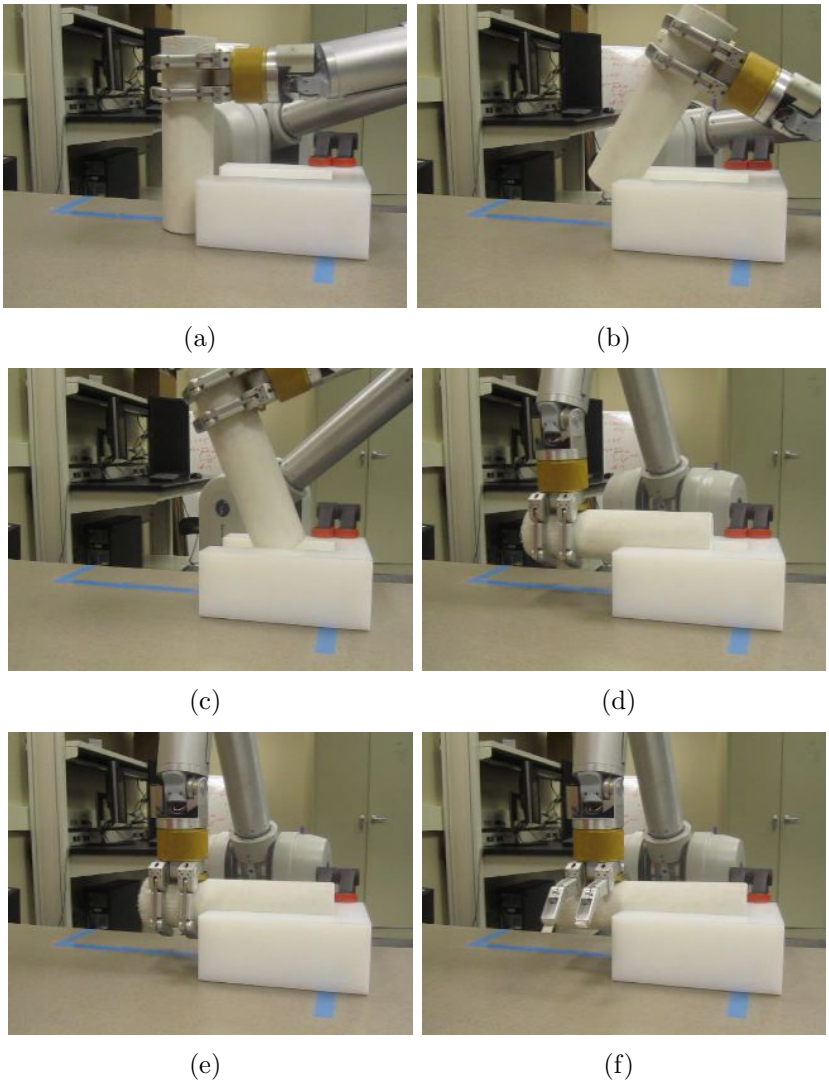


Figure 27: Execution of compliant motion, consisting of a sequence of contact state transitions, to accomplish an assembly.

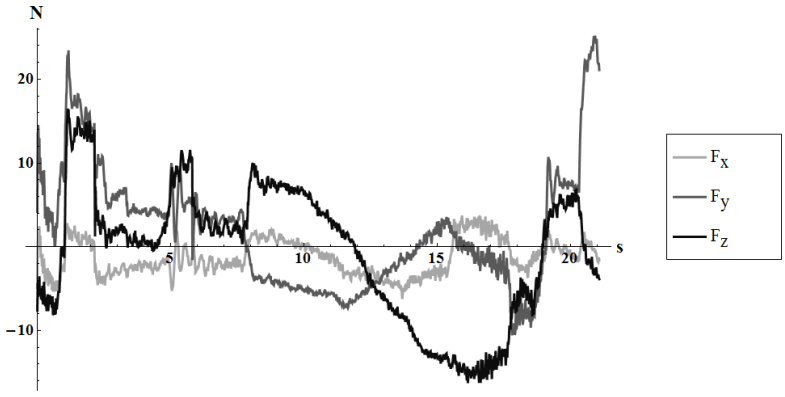


Figure 28: The end-effector force was calculated based on sensed joint torque values and Eq. 22, Also, the applied force limits were achieved via joint torque thresholds.

on the joint positions, the forward kinematics, and ${}^{E}T_{A}$. The pose of the part A in the goal state deviates from what was nominally planned, since the part B was clamped down and at the end of the sequence the slot and cylinder are aligned. The error in ${}^{B}T_{A}$ was canceled due to compliant motion. In particular, the part A was first constrained by placing it on the part B, as shown in Fig. 27 c), followed by being rotated into the concave opening d), and then slid horizontally e). The first peak in the y force component reflects the initial contact between the parts. As the part A is lifted and then moved while constrained by the edges of part B, the z component of the force increases and becomes more or less constant with an extra 'bump' due to the unintentional contact with the edge. Then the part A is rotated, and the force in the negative z direction increases in a roughly linear fashion. The bottom of the cylinder reaching the back of the slot is reflected in the plot by a sudden increase in force along the y axis, which is along the normal of the part B. The small increase of the force magnitude along the x axis is due to the part A being deflected and ending up slightly tilted in its final state as indicated in Fig. 26.

The fact that the last move aligns the bottom of the cylinder with the back of the concave part is due to serendipity: If the error in the position of part B happened to be in the other direction, the slide would have been too short to ensure contact, since it was planned based on the nominal estimate of ${}^{W}T_{B}$. The examples with estimation in the next section will take that into account by nominally planning for penetration.

This example demonstrates that the 'back-drivablity' of the WAM in connection with torque limits and compliance can make the assembly under uncertainty possible for small errors in locations of the parts. We have also successfully tested alternative

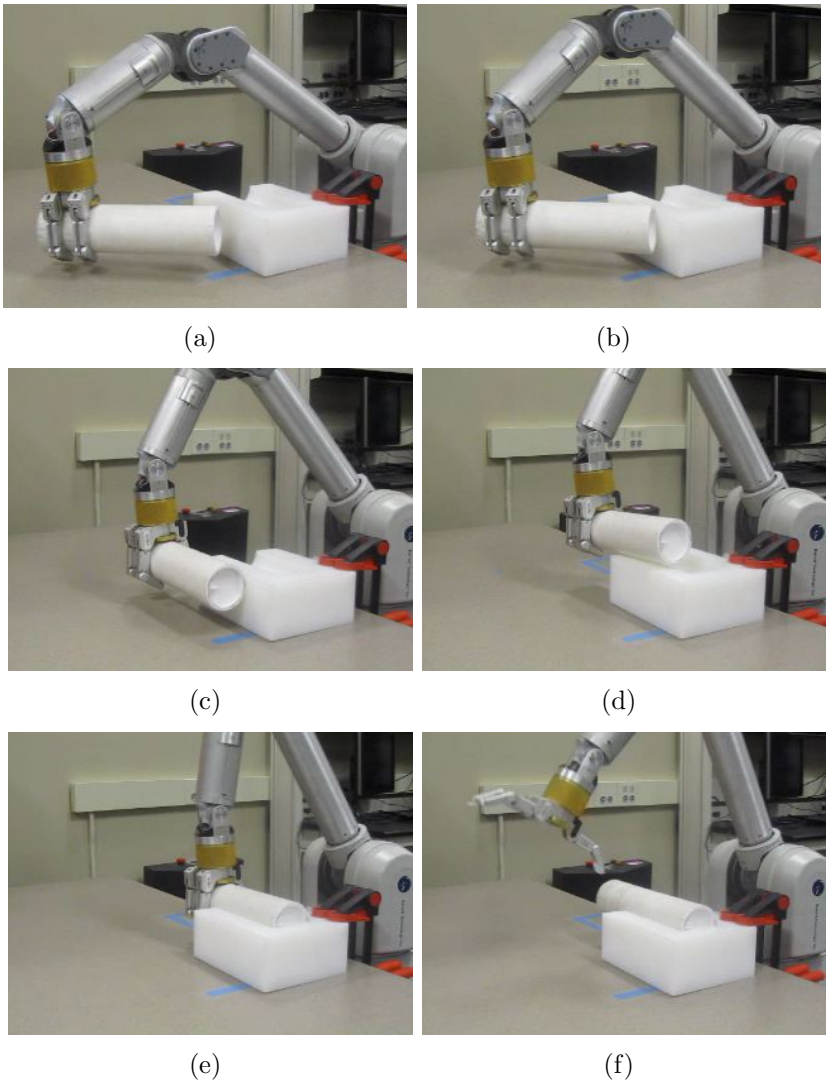


Figure 29: Execution of a nominally planned assembly sequence in a case when the error between the nominal and the actual pose of the static object B is small.

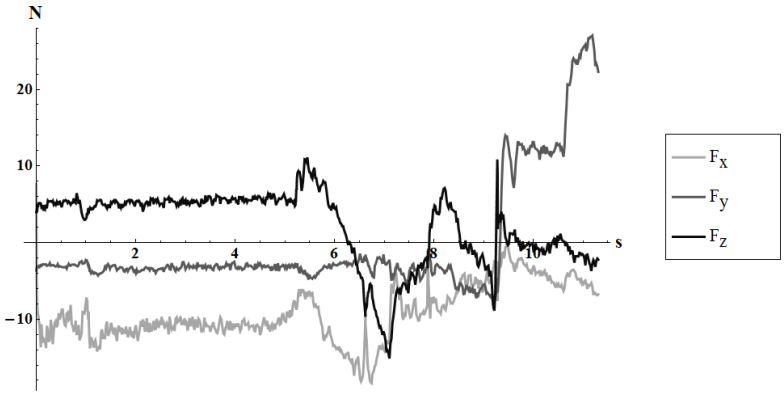
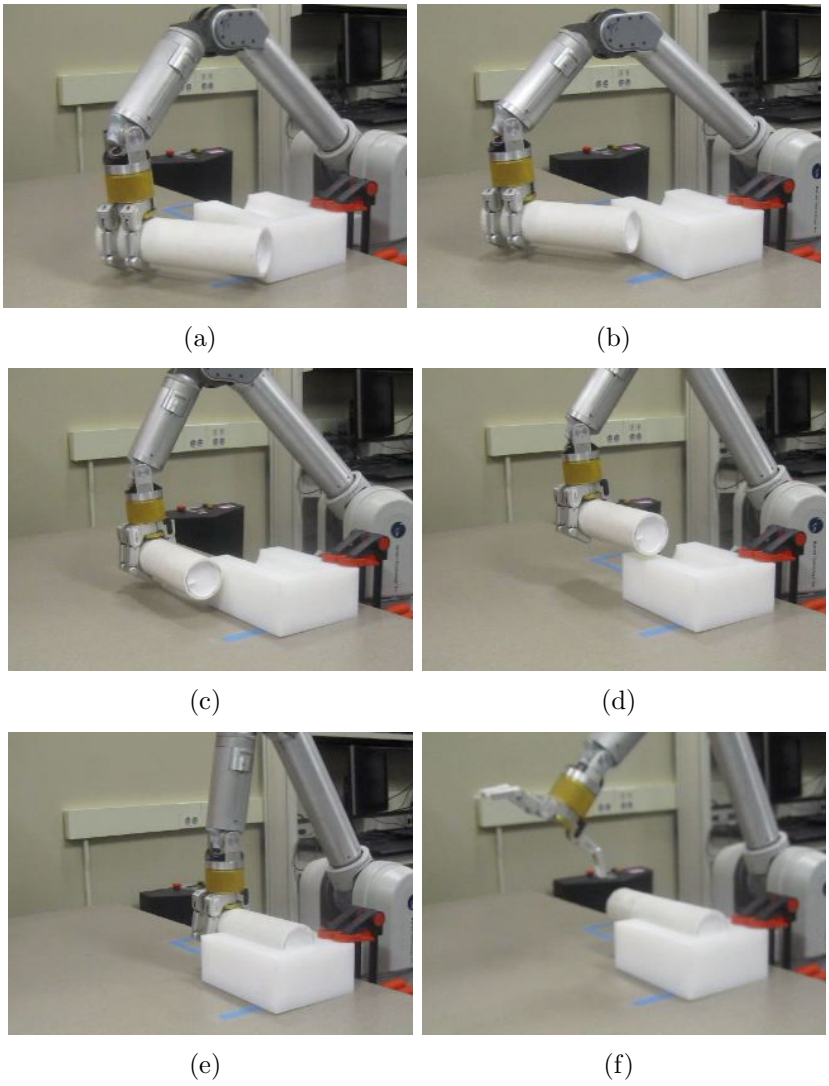


Figure 30: The force at the end-effector for a small error in pose of the object B. This plot will be used for comparison with cases where the error in alignment is larger.

compliant paths through different sequences of contact states, confirming the effectiveness of the algorithm for automatically converting compliant paths from Cartesian to the joint space, see Sec. 3.2. As a case in point, let us consider the execution of the sequence in Fig. 29. The cylinder is first brought into contact with the static object and rotated until aligned, as shown in subfigures a) through c). Then, it is hoisted over the edge and placed into the concave slot. The contacts in d) through f) enable us to accomplish the assembly despite uncertainty. Fig. 30 shows the corresponding force plot. We will later use this sequence for comparison with examples in which compliance and estimation are used together to overcome errors in pose of the part B, when the deviations from the nominal values are more significant. Since the force at the end-effector is estimated based on the commanded torques, it reflects the correction applied by the PID controllers to the joints. Hence, it also contains the force component used to correct the imperfections of the gravity compensation model, e.g. the increase in force in the z direction at the beginning of the assembly sequence.

5.3 Parameter Estimation under Uncertainty

Both sequences in the previous section relied exclusively on nominal planning. Although the same contact states were used in planning the experiment in Fig. 30 as in the example from Sec. 4.5, we did not take advantage of the active part interaction. In this section, we will compare the results to a faithful implementation of the sequence described in Sec. 4.5, which nominally results in overlaps between objects in order to ensure active compliance under uncertainty. As a result, we can subsequently estimate the uncertain parameters based on compliance between the parts.



(a) (b) (c) (d) (e) (f)

Figure 31: This example shows a task space plan that takes into account possible uncertainty and results in successful assembly for a small error in pose of the part B.

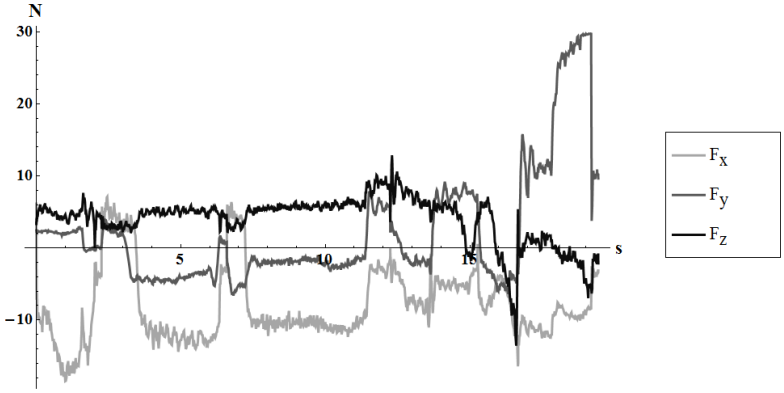


Figure 32: The only notable change in the force plot comes from the extra pressure applied in the x direction during the slide along the face of the part B.

The sequence is shown in Fig. 31. The purpose of the moves a) trough c) is to estimate the orientation of the fixed part B. The edge of the bottom of the cylinder is kept in contact with the flat side of B and moved in a straight line. The initial homogeneous transformation matrix of the object B w.r.t. the world frame is

$$ {}^{W}T_{B}^{(1)} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.4 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{65} $$

Now, the updated pose of B, ${}^{W}T_{B}^{(2)}$, is obtained based on the first contact. As one can see, the position vector has changed, i.e. a translational degree of freedom (DOF) was estimated based on the difference between the contact point as predicted by the geometrical model using ${}^{W}T_{B}^{(1)}$ and the position at which the contact actually occurred according to joint positions as measured by the sensors.

$$ {}^{W}T_{B}^{(2)} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.4084 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{66} $$

Next, sliding the part A, while maintaining the contact between the edge of the cylinder and the flat face of the part B, see Fig. 31 b), allows for estimating the orientation of B in the plane based on the collected contact points, as inferred by the joint positions, forward kinematics, and ${}^{E}T_{A}$. Then a rotation about the axis defined by the table normal is applied to ${}^{W}T_{B}^{(2)}$, yielding ${}^{W}T_{B}^{*}$. In order to maintain the

contact between the objects *in the model*, the offset in the plane needs to be updated as well. This is accomplished by a translation so that the contact point now coincides with the point on the plane describing the face of B, that was closest to it according to ${}^{W}T_{B}^{*}$. Hence, both rotation and offset are updated, yielding

$$ {}^{W}T_{B}^{(3)} = \begin{bmatrix} 0.9963 & 0.0 & -0.0863 & 0.3987 \\ -0.0863 & 0.0 & -0.9963 & 0.0091 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{67} $$

At the end of the move, the final DOF along the x axis of the world coordinate system is determined using a translational compliant move. The resulting homogeneous transformation matrix is given by

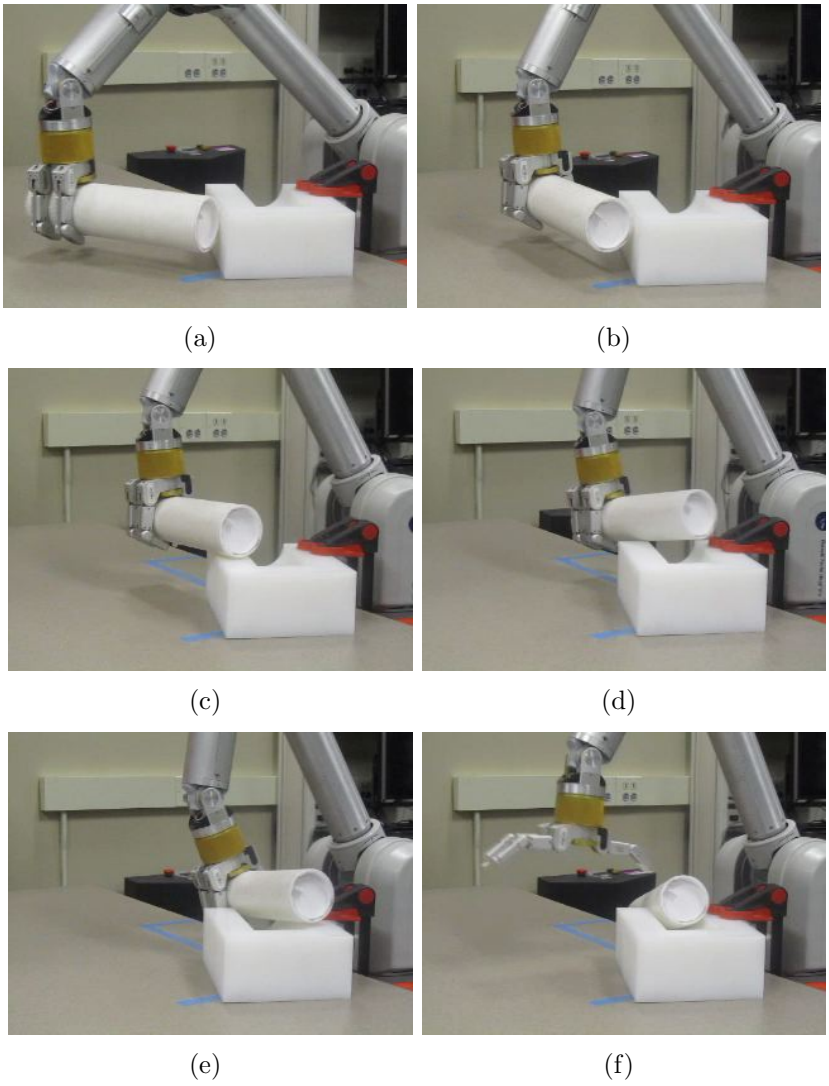
$$ {}^{W}T_{B}^{(4)} = \begin{bmatrix} 0.9963 & 0.0 & -0.0863 & 0.396 \\ -0.0863 & 0.0 & -0.9963 & -0.0216 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{68} $$

This is the final estimated pose of B in world coordinates. We see that the error between the initially assumed nominal pose ${}^{W}T_{B}^{(1)}$ and the pose estimated during the execution ${}^{W}T_{B}^{(4)}$ is not large indeed. The error in offset can be calculated to be 2.2 cm, and the rotational error is less than 5 degrees. In such a case, we can rely on passive compliance to correct the trajectory.

The robustness of the compliant surface assembly subject to small errors in positioning of the parts is important for the overall strategy. For larger errors, however,

the sequence without estimation outright fails. The problem occurs when attempting to place the cylinder into the concave slot, as shown in Fig. 33. The side of the cylinder is not aligned with the edge of the part B. Even with the increasing downward force, as indicated by the z component of the calculated end-effector force in Fig. 34, the manipulator is not able to insert the part A into the correct place. On the other hand, in Fig. 35 the alignment of the parts can be achieved by using the collected data to estimate the pose of B. At the end of the exploratory move, we are able to determine its orientation. Afterwards, we only need to see how far to slide the cylinder for it to become aligned with the back of the slot, at which point the parts will be properly assembled. Since this is done by nominally planning for penetration, we get the same peak in the y component of the force at the end-effector as we did previously. Incidentally, we also get a small increase in the x direction presumably due to a rotational error in the pose of the part B.

If we turn our attention to the force plot in Fig. 36, we see that the end of the sequence is similar to what was previously obtained for the nominal path and small error example shown in Fig. 30. There is little extra pressure applied in this case. This is consistent with the alignment of features of the part A as indicated by Fig. 35 c) and d). Note that the last peak in the x direction is the result of sliding in the part A into the concave slot at the end of the assembly sequence, which effectively removes the last bit of uncertainty in the pose of B. There are two additional points in time when the force in the x direction increases. The first spike corresponds to the end of the guarded move and the first contact between the objects, see Fig. 35 a), and the second one to the slide used for estimating the orientation in Fig. 35 b).



(a) (b)

(c) (d)

(e) (f)

Figure 33: The error in pose of the part B is larger in this example resulting in a failure of a nominally planned trajectory without estimation.

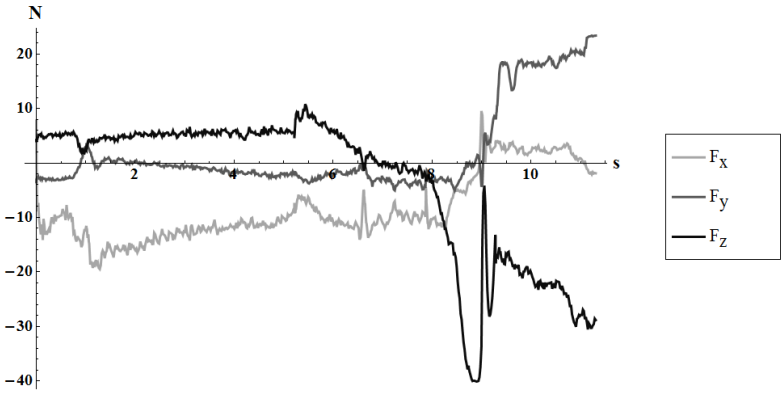
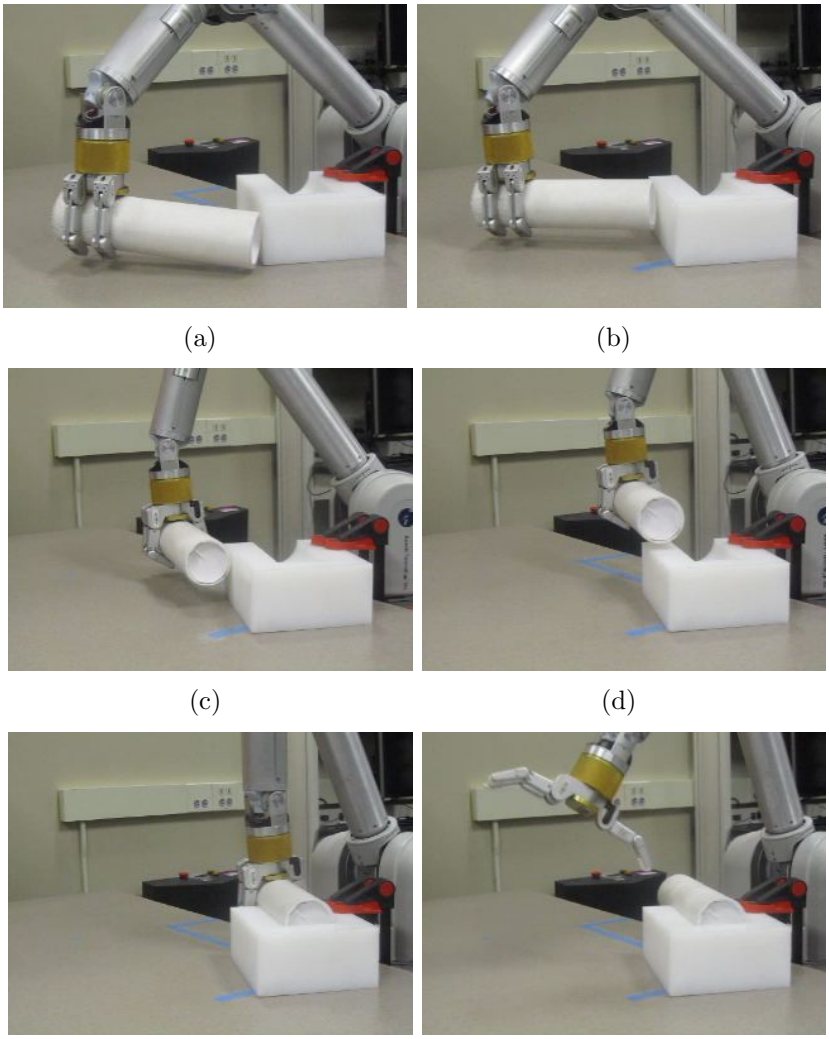


Figure 34: Without the estimation, the fact that the edge of the part B is not aligned with the side of the cylinder leads to an increase in force in the negative z direction.



(a) (b) (c) (d) (e) (f)

Figure 35: It is possible to estimate the pose of B for large positioning errors, provided that the initial ${}^{W}T_{B}$ is accurate enough to bring the correct features into contact.

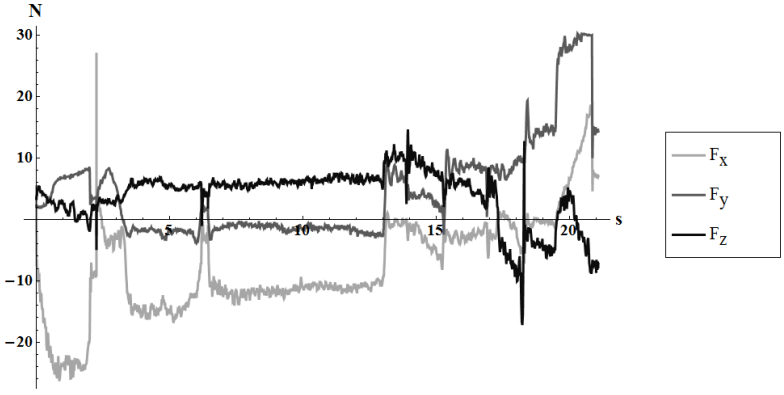


Figure 36: Since the parts are better aligned after estimation, the rightmost part of the force plot now looks almost identical to the case with the small positioning error.

Let us now examine the estimated homogeneous transformation matrices for the part B at different steps of the assembly sequence. The initial, i.e. nominal, ${}^{W}T_{B}^{(1)}$ is the same as before and will not be repeated. The guarded move yields

$$ {}^{W}T_{B}^{(2)} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.4026 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{69} $$

as an improved estimate based on a single translational DOF. At the end of the slide, see Fig. 35 b), we get

$$ {}^{W}T_{B}^{(3)} = \begin{bmatrix} 0.8959 & 0.0 & -0.4443 & 0.3552 \\ -0.4443 & 0.0 & -0.8959 & 0.0491 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{70} $$

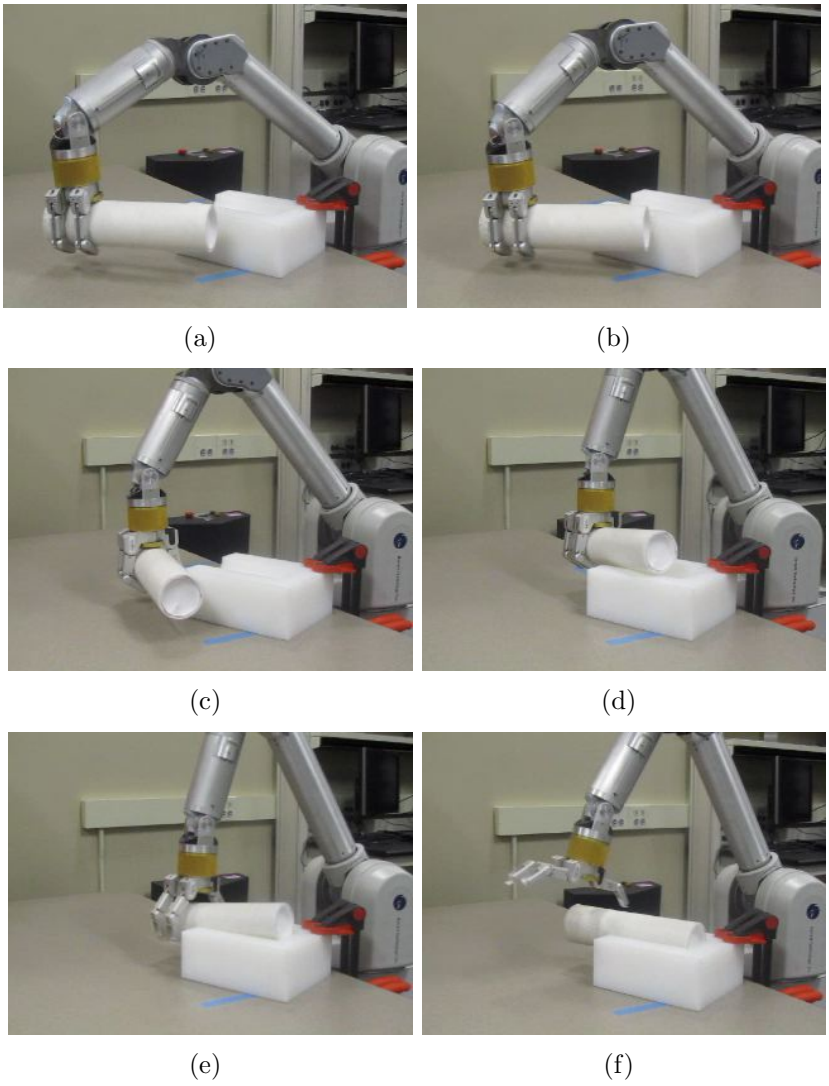
Finally, after aligning the cylinder and the slot, the final estimate for the position and orientation of B is obtained as

$$ {}^{W}T_{B}^{(4)} = \begin{bmatrix} 0.8959 & 0.0 & -0.4443 & 0.344 \\ -0.4443 & 0.0 & -0.8959 & 0.0265 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{71} $$

This corresponds to an offset of 6.2 cm and a correction in angle of 26.4 degrees. As one can see, there is a significant error in the pose of the object B compared to the initial nominal guess ${}^{W}T_{B}^{(1)}$. ${}^{W}T_{B}^{(4)}$ is depicted in Fig. 41 a).

Similar results are obtained, if the direction of the translational and rotational error of the part B is opposite to the previous case. As can be seen from Fig. 37, the assembly without estimation fails. The lack of alignment in d) is a direct consequence of the large error in the orientation of B in the plane. Consequently, the cylinder is pressed against the edges of the concave slot. In subfigure e), we can see that the object A is released before reaching the goal state of the assembly sequence. In addition, the force in the negative z direction in Fig. 38 increases, and then stays approximately constant. We conclude that at this point, the object A is jammed. Although the cylinder somehow eventually falls into place, its bottom is not in contact with the back of the slot.

On the other hand, the assembly sequence in Fig. 39 and the corresponding force plot in Fig. 40 show that the assembly was completed successfully after estimation. The obtained force plot does not show any significant deviations from the case with small positioning errors for the final assembly state, indicating that the object was not forced into the slot. The two increases in F_x are due to the guarded move that is needed to bring the objects into contact and the exploratory compliant move used to determine the orientation of the object B in the plane. The latter is more pronounced compared to the previous example due to the fact that the deeper nominal penetration based on the model results in a larger actual error in position, and therefore the PID controller generates higher torques. In the goal state, which is in fact a more constrained contact state involving less regular features, the response that we see in the force plots is again almost identical to the case where the error between the nominal and correct poses of the parts is small.



(a) (b)

(c) (d)

(e) (f)

Figure 37: In this nominally planned example without parameter estimation the cylinder is not properly fitted into the fixed slot before being released.

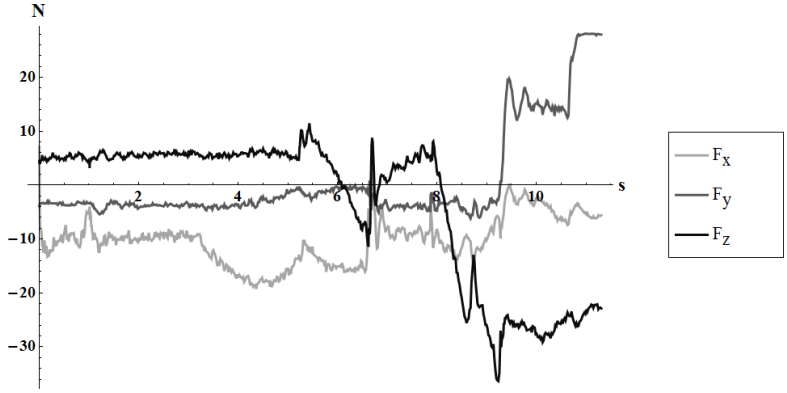


Figure 38: The force in the negative z direction increases at the end of the nominally planned trajectory reflecting the fact that the part A got jammed during execution.

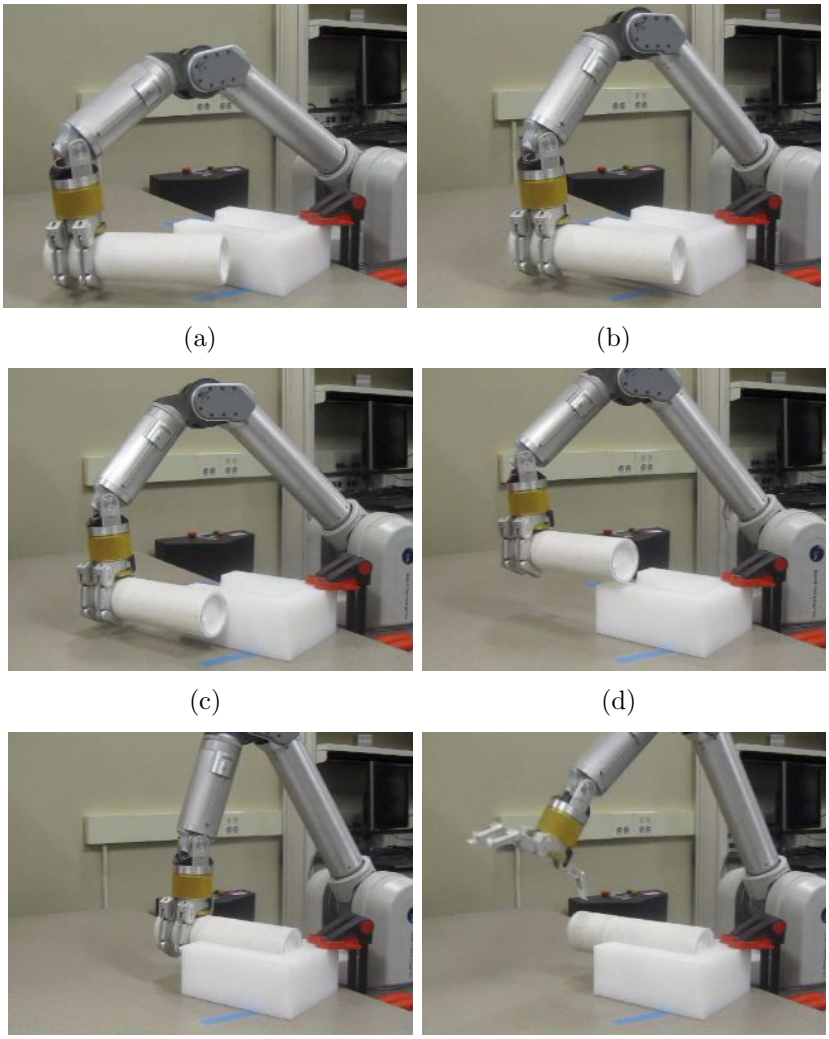


Figure 39: Sliding the part A along the side of the part B allows for estimating the orientation. As a result, the parts can be assembled in the presence of a large error.

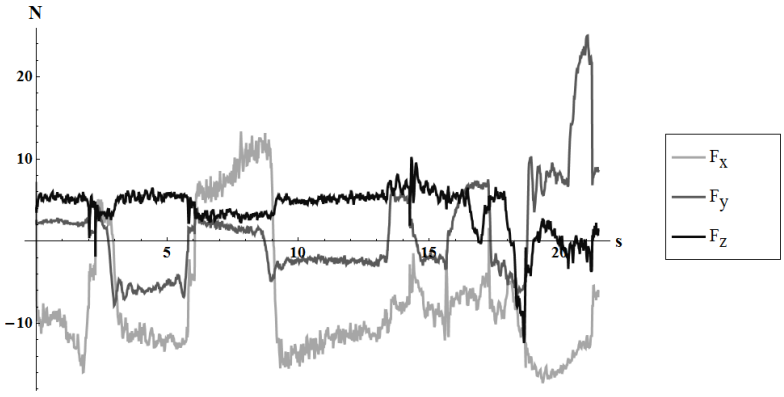


Figure 40: The estimation phase can be identified by an increase in force in the x direction due the contact that is maintained between the objects.

After the individual exploratory moves, the homogeneous transformation matrices for the pose of the object B are updated as follows.[13] First,

$$ {}^{W}T_{B}^{(2)} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.3731 \\ 0.0 & 0.0 & -1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{72} $$

is obtained after the guarded move. Next, the orientation is updated based on the compliant move along the side of the object B. This results in

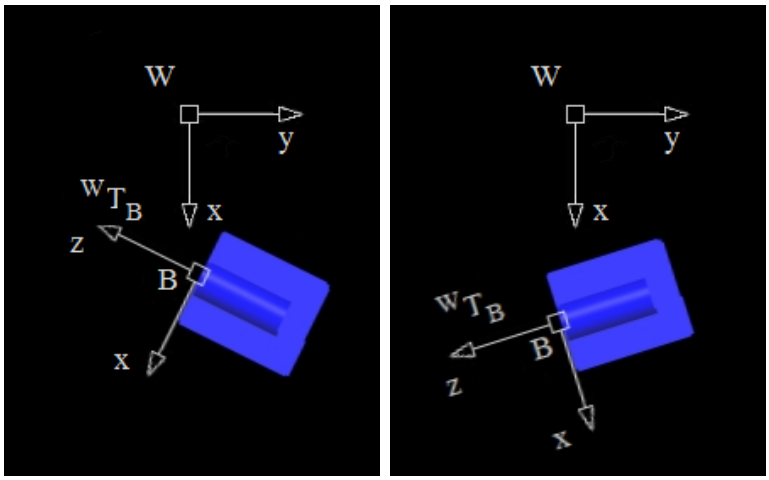
$$ {}^{W}T_{B}^{(3)} = \begin{bmatrix} 0.9536 & 0.0 & 0.3012 & 0.4404 \\ 0.3012 & 0.0 & -0.9536 & -0.0257 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}, \tag{73} $$

which is the pose of B that allows for aligning the features of the objects in Fig. 39 d) and e). The last DOF for the object B can be calculated after the assembly goal state has been achieved. The final estimated pose is given by

$$ {}^{W}T_{B}^{(4)} = \begin{bmatrix} 0.9536 & 0.0 & 0.3012 & 0.4434 \\ 0.3012 & 0.0 & -0.9536 & -0.035 \\ 0.0 & 1.0 & 0.0 & -0.01 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}. \tag{74} $$

In this case, the offset and orientation were updated by 5.5 cm and 17.52 degrees respectively. ${}^{W}T_{B}^{(4)}$ is displayed in Fig. 41 b).

[13] The initial matrix ${}^{W}T_{B}^{(1)}$ remains unchanged and is therefore omitted.



(a) (b)

Figure 41: The estimated final poses ${}^{W}T_{B}^{(4)}$ for the sequences from Fig. 35 and Fig. 39 are shown in a) and b) respectively. Comparison with Fig. 26 b) illustrates the large initial parameter uncertainty. Moreover, the results are *robust under perturbations – the assembly sequence performs similarly for errors of the same or smaller magnitude.*

A compliant move, which is too short, will fail to provide enough information. It is therefore necessary to collect enough points for the estimation step. In the above examples, the orientation of the object B was estimated by sliding the object A over a distance of 8 cm in length – of course, this is the nominally planed length. The total execution time varied between 3.5 and 4 seconds. This is due to the fact that the actual trajectory is different due to uncertainty. Likewise, the number of collected points varied and was in the range between 1800 and 2200.

Lastly, let us address the issue of not using rotations in exploratory moves. As it turns out, the proper angle is difficult to determine in this manner, because the rotational axis can shift even during a continuous rotation. It happens whenever a contact with an edge occurs, see for example Fig. 38 c). Note that this does not imply that a rotation is ill-suited for aligning the parts in general – in fact, in Fig. 27 we have encountered an example to the contrary.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

A method for correcting pose uncertainties in robotic assembly tasks has been presented that takes advantage of exploratory compliant motion involving easy-to-make contacts between regular features of the parts. Since this does not introduce any restrictions on other features, e.g. those in contact in the final assembly state, the parts as a whole can have non-trivial geometry. The estimation procedure can be added to or even seamlessly embedded in the nominal assembly sequence. The proposed approach is also simpler to use than Bayesian filters, because it focuses on a few parameters at a time. The conditional probabilities are not required. Instead, contact point data collected during specifically crafted exploratory moves is used to estimate the uncertain parameters that describe the pose of the object in Cartesian space. The experiments illustrate that compliant moves between simple features and the subsequent estimation can significantly reduce uncertainty in the pose of the static object and therefore make the assembly tasks more stable under variation in its position and orientation.

There are, however, several open issues that deserve future attention. First of all, a general criterion for detecting the success or failure of the assembly in the goal state is needed. Of course, one can always use a set of inequality constraints for the parameters describing the final pose of the end-effector on a case-by-case basis, see

for example Desai and Volz [14]. Moreover, if the parts are assembled properly the movement of the held object will be constrained. This allows for additional criteria based on force thresholds, if the force sensor is available.

Secondly, a force sensor would allow for various extensions of the current strategy, such as terminating exploratory moves based on force criteria or, more importantly, a control scheme that relies on force feedback. Not only would this limit the applied force much more accurately, but it could also help in dealing with parameter uncertainties, see Bruyninckx et al. [6], and possibly allow for schemes to identify the contact states, or rather to verify that the parts are in the desired contact state, which is currently being assumed.

For practical applications, computer vision should provide the first approximation of the pose of the objects. Then the remaining uncertainty could be eliminated by exploratory moves using the presented techniques for final alignment. This would have to be experimentally verified, but one would expect it to work well in practice. The question arises as to which information should be used when – scheduling and planning would in that case be a challenging task. However, the idea itself seems very solid and is most certainly worth being explored further.

A software library implementing the numerical algorithm for convex polygonal meshes from Sec. 4.3.2 is a logical next step. Being able to perform this type of optimization for any given convex mesh would simplify the application of the presented approach to new problems, which is important in manufacturing. This would be the first step toward testing the approach on a large number of examples and applying it to real world problems.

There are also other questions that are raised by this research. One of them is being able to handle, not only flat or convex, but also general features using meshes, which leads to a difficult optimization problem. Although the distance calculation between two meshes has been thoroughly studied, see for instance [17] and [41], the calculation of penetration depth still poses a considerable challenge. The applicability of local techniques [36] is limited, but other approaches, e.g. [30], might be more promising. Also, the value of penetration depth is not always uniquely defined. However, for $E = |\text{distance}| + |\text{penetration depth}|$ or $E = \text{distance}^2 + \text{penetration depth}^2$, it suffices to minimize a quantity $\eta(E)$, such that $\eta(E) \to 0$ implies $E \to 0$. That would, for example, be the case if $\eta(E) \geq \lambda E$ for some positive λ. Finding derivatives in this case is out of question. Instead, since the number of parameters is small – there are at most 12 DOFs for two objects – one could use direct methods, or even genetic algorithms and evolutionary computation.

Also, uncertainty in the pose of the grasped object was largely ignored. It can, however, be handled by aligning the held object with a suitable static object. For instance, one could 'measure' a dimension of the held part by pushing[14] it against another object for which the dimensions and pose are known.

The high-level planning was done in an ad hoc manner in all examples. A more systematic way of constructing exploratory moves would be highly beneficial. Still, automatic generation of a particular trajectory based on the contact state graphs is a difficult task by itself, see Ji and Xiao [32] and [74], and using the low-level feedback to improve the high-level contact state based plan is even more intricate. It

[14] Using a guarded move.

is not immediately clear which criteria should be used in order to search the contact state graph to generate an assembly sequence suitable for estimation. Moreover, the trajectory of the held object, the exploratory moves and the points in time at which the parameter estimation occurs, would all need to be generated by such an algorithm. Nevertheless, even a semi-automatic method to assist in planning assembly sequences would be a valuable contribution from the practical point of view.

Finally, a control scheme could address the issue of maintaining the orientation of the end-effector, which can help in estimation. The Jacobian transpose methods could, in theory, give higher importance to the error in orientation of the end-effector. More precisely, let $\Delta x = [\, \Delta p \quad \Delta\phi \,]^T$ denote the error in pose of the end-effector where Δp and $\Delta\phi$ are the errors in position and orientation respectively. The pseudo-inverses of the Jacobians in terms of the position and orientation of the end-effector are given by ${J_p}^\dagger = {J_p}^T \left(J_p {J_p}^T\right)^{-1}$ and ${J_\phi}^\dagger = {J_\phi}^T \left(J_\phi {J_\phi}^T\right)^{-1}$. Here, the matrix $N\left(J_\phi\right)$ that projects onto the nullspace of J_ϕ could be used. It is a well-known fact that it can be represented as $N\left(J_\phi\right) = I - {J_\phi}^\dagger J_\phi = I - {J_\phi}^T \left(J_\phi {J_\phi}^T\right)^{-1} J_\phi$, see for example Yoshikawa [77]. One could therefore try to devise a control law that maps any contribution to the position of the end-effector – regardless whether it is a result of force feedback or a position error – onto the nullspace of J_ϕ. Note that this would not be merely a superposition of two types of error as opposed to Eq. 21, but rather that it would give higher importance to the orientation error which would be set independently of J_p with the positional error only affecting the null space of J_ϕ. The question, of course, is whether such a control scheme would be stable.

REFERENCES

[1] Anderson, R. J., and Spong, M. W. Hybrid impedance control of robotic manipulators. *IEEE Journal on Robotics and Automation 4* (1988), 549–556.

[2] Asada, H. Representation and learning of nonlinear compliance using neural nets. *IEEE Transactions on Robotics and Automation 9* (1993), 863–867.

[3] Asada, H., and Slotine, J.-J. *Robot Analysis and Control*. John Wiley and Sons, New York, 1986.

[4] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software 22* (1996), 469–483.

[5] Bertram, D., Kuffner, J., Dillmann, R., and Asfour, T. An integrated approach to inverse kinematics and path planning for redundant manipulators. *IEEE International Conference on Robotics and Automation* (2006), 1874–1879.

[6] Bruyninckx, H., Demey, S., Dutr, S., and De Schutter, J. Kinematic models for model-based compliant motion in the presence of uncertainty. *The International Journal of Robotics Research 14* (1995), 465–482.

[7] Buckley, S. Planning compliant motion strategies. *IEEE International Symposium on Intelligent Control* (1988), 338–343.

[8] Cameron, S. Enhancing GJK.: Computing minimum and penetration distances between convex polyhedra. *Proceedings of International Conference on Robotics and Automation* (1997), 3112–3117.

[9] Chan, T. M. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry 16* (1996), 361–368.

[10] Cheng, F., Chen, J., and Kung, F. Study and resolution of singularities for a 7-dof redundant manipulator. *IEEE Transactions on Industrial Electronics 45* (1998), 469–480.

[11] Craig, J. *Introduction to Robotics: Mechanics and Control*, 2 ed. Addison-Wesley Longman Publishing Co., Inc. Boston, 1989.

[12] De Schutter, J., Rutgeerts, J., Aertbeliën, E., De Groote, F., De Laet, T., Lefebvre, T., Verdonck, W., and Bruyninckx, H. Unified constraint-based task specification for complex sensor-based robot systems. *IEEE International Conference on Robotics and Automation* (2005), 3618–3623.

[13] Debus, T., Dupont, P., and Howe, R. Contact state estimation using multiple model estimation and hidden Markov models. *The International Journal of Robotics Research 23* (2004), 399–413.

[14] Desai, R., and Volz, R. Identification and verification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties. *IEEE International Conference on Robotics and Automation 2* (1989), 800–807.

[15] Drake, S. H. *Using Compliance in Lieu of Sensory Feedback for Automatic Assembly.* PhD thesis, MIT, 1978.

[16] Gadeyne, K., Lefebvre, T., and Bruyninckx, H. Bayesian hybrid model-state estimation applied to simultaneous contact formation detection and geometrical parameter estimation. *International Journal of Robotics Research 24* (2005), 615–630.

[17] Gottschalk, S., Lin, M. C., and Manocha, D. OBBTree: a hierarchical structure for rapid interference detection. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), 171–180.

[18] Hachenberger, P. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. *"Computational Geometry 38* (2007), 64 – 99.

[19] Hachenberger, P. Exact Minkowski sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces. *Algorithmica 55* (2009), 2.

[20] Hachenberger, P. 3D Minkowski sum of polyhedra. In *CGAL User and Reference Manual.* CGAL Editorial Board, 2013.

[21] Hamner, B., Koterba, S., Shi, J., Simmons, R. G., and Singh, S. An autonomous mobile manipulator for assembly tasks. *Autonomous Robots 28* (2010), 131–149.

[22] Hara, K., and Yokogawa, R. Recognition of state in peg-in-hole by fuzzy schema. *Journal of Advanced Automation Technology 4* (1992), 134–139.

[23] Hirai, S., Asada, H., and Tokumaru, H. Kinematics of manipulation using the theory of polyhedral convex cones and its applications to grasping and assembly operations. *IEEE International Workshop on Intelligent Robots* (1988), 451–456.

[24] Hirukawa, H., Papegay, Y., and Matsui, T. A motion planning algorithm for convex polyhedra in contact under translation and rotation. *IEEE International Conference on Robotics and Automation 4* (1994), 3020–3027.

[25] Hogan, N. Impedance control: An approach to manipulation. *American Control Conference* (1984), 304–313.

[26] Hogan, N. Stable execution of contact tasks using impedance control. *IEEE International Conference on Robotics and Automation 4* (1987), 1047–1054.

[27] Hooke, R., and Jeeves, T. A. "Direct search" solution of numerical and statistical problems. *Journal of the Association for Computing Machinery 8* (1961), 212–229.

[28] Hsiao, K. *Relatively robust grasping*. PhD thesis, Massachusetts Institute of Technology, 2009.

[29] Hsiao, K., Kaelbling, L. P., and Lozano-Pérez, T. Robust grasping under object pose uncertainty. *Autonomous Robots Journal 31* (2011), 253–268.

[30] Je, C., Tang, M., Lee, Y., Lee, M., and Kim, Y. J. Polydepth: Real-time penetration depth computation using iterative contact-space projection. *ACM Transactions on Graphics 31* (2012), 5:1–5:14.

[31] Ji, X. *On Contact State Space Generation and Contact Motion Planning*. PhD thesis, University of North Carolina at Charlotte, 2000.

[32] Ji, X., and Xiao, J. Planning motion compliant to complex contact states. *International Journal of Robotics Research 20* (2001), 446–465.

[33] Jiménez, P. Survey on assembly sequencing: A combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing 24* (2013), 235–250.

[34] Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering 82* (1960), 35–45.

[35] Kim, Y., Lin, M., and Manocha, D. Deep: Dual-space expansion for estimating penetration depth between convex polytopes. *In IEEE Conference on Robotics and Automation* (2002), 921–926.

[36] Kim, Y., Lin, M., and Manocha, D. Incremental penetration depth estimation between convex polytopes using dual-space expansion. *IEEE Transactions on Visualization and Computer Graphics 10* (2004), 152–163.

[37] Kreutz-Delgado, K., Long, M., and Seraji, H. Study kinematic analysis of 7-dof manipulators. *International Journal of Robotics Research 11* (1992), 469–481.

[38] Kwak, S. J., Hasegawa, T., and Chung, S. Y. A framework for automatic generation of a contact state graph for robotic assembly. *Advanced Robotics 25* (2011), 1603–1625.

[39] Kwak, S. J., Hasegawa, T., Mozos, O. M., and Chung, S. Y. Elimination of unnecessary contact states in contact state graphs for robotic assembly tasks. *The International Journal of Advanced Manufacturing Technology* (2013), 1–15.

[40] Lai, H. Y., and Huang, C. T. A systematic approach for automatic assembly sequence plan generation. *The International Journal of Advanced Manufacturing Technology 24* (2004), 752–763.

[41] Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. Fast proximity queries with swept sphere volumes. *IEEE International Conference on Robotics and Automation* (2000), 3719–3726.

[42] Latombe, J. C. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[43] Laugier, C. Planning fine motion strategies by reasoning in the contact space. *IEEE International Conference on Robotics and Automation 2* (1989), 653–659.

[44] Lefebvre, T., Bruyninckx, H., and De Schutter, J. Exact non-linear Bayesian parameter estimation for autonomous compliant motion. *Advanced Robotics 18* (2004), 787–799.

[45] Lefebvre, T., Bruyninckx, H., and De Schutter, J. Polyhedral contact formation identification for autonomous compliant motion: exact nonlinear Bayesian filtering. *IEEE Transactions on Robotics 21* (2005), 124–129.

[46] Lefebvre, T., Xiao, J., Bruyninckx, H., and De Gersem, G. Active compliant motion: a survey. *Advanced Robotics 19* (2005), 479–499.

[47] Levenberg, K. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics 2* (1944), 164–168.

[48] Lozano-Perez, T. Spatial planning: A configuration space approach. *IEEE Transactions on Computers C-32* (1983), 108–120.

[49] Lozano-Perez, T., Mason, M. T., and Taylor, R. H. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research 3* (1984), 3–24.

[50] Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics 11* (1963), 324.

[51] McCarragher, B. J., and Asada, H. A discrete event approach to the control of robotic assembly tasks. *IEEE International Conference onRobotics and Automation 1* (1993), 331–336.

[52] Meeussen, W., Staffetti, E., Bruyninckx, H., Xiao, J., and De Schutter, J. Integration of planning and execution in force controlled compliant motion. *Journal of Robotics and Autonomous Systems 56* (2008), 437–350.

[53] Meeussen, W., Xiao, J., De Schutter, J., Bruyninckx, H., and Staffetti, E. Automatic verification of contact states taking into account manipulator constraints. *IEEE International Conference on Robotics and Automation 4* (2004), 3583–3588.

[54] Nakamura, Y., and Hanafusa, H. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control 108* (1986), 163–171.

[55] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The computer journal 7* (1965), 308–313.

[56] Pan, F., and Schimmels, J. Efficient contact state graph generation for assembly applications. *IEEE International Conference on Robotics and Automation, 2003. 2* (2003), 2592–2598.

[57] Peshkin, M. A. Programmed compliance for error corrective assembly. *IEEE Transactions on Robotics and Automation 6* (1990), 473–482.

[58] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. ROS: an open-source robot operating system. *IEEE ICRA Workshop on Open Source Software* (2010).

[59] Remde, A., Henrich, D., and Wom, H. Manipulating deformable linear objects-contact state transitions and transition conditions. *IEEE/RSJ International Conference on Intelligent Robots and Systems 3* (1999), 1450–1455.

[60] Rooks, B. The harmonious robot. *Industrial Robot: An International Journal 33* (2006), 125–130.

[61] Rosell, J., Basañez, L., and Suárez, R. Determining compliant motions for planar assembly tasks in the presence of friction. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (1997), 946–951.

[62] Sarić, A., Xiao, J., and Shi, J. Robotic surface assembly via contact state transitions. *IEEE International Conference on Automation Science and Engineering* (2013), 966–971.

[63] Shen, Y., and Huper, K. A joint space formulation for compliant motion control of robot manipulators. *IEEE International Conference on Mechatronics and Automation 1* (2005), 362–369.

[64] Singh, G. K., and Claassens, J. An analytical solution for the inverse kinematics of a redundant 7dof manipulator with link offsets. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), 2976–2982.

[65] Spong, M. W., Hutchinson, S., and Vidyasagar, M. *Robot Modeling and Control*. John Wiley and Sons, 2006.

[66] Stilman, M. Task constrained motion planning in robot joint space. Tech. rep., Robotics Institute, Carnegie Mellon University, CMU-RI-TR-06-43., 2006.

[67] Stilman, M. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics 26* (2010), 576–584.

[68] Stolt, A., Linderoth, M., Robertsson, A., and Johansson, R. Force controlled robotic assembly without a force sensor. *IEEE International Conference on Robotics and Automation* (2012), 1538–1543.

[69] Stückler, J., and Behnke, S. Compliant task-space control with back-drivable servo actuators. *International RoboCup Symposium* (2011), pp. 78–89.

[70] Tang, P., and Xiao, J. Automatic generation of high-level contact state space between 3d curved objects. *International Journal of Robotics Research 27* (2008), 832–854.

[71] Thrun, S. Probabilistic algorithms in robotics. *AI Magazine 21* (2000), 93–109.

[72] Wampler, C. W. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics 16* (1986), 93–101.

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

[74] Xiao, J., and Ji, X. On automatic generation of high-level contact state space. *International Journal of Robotics Research (and its first multi-media extension issue eIJRR) 20* (2001), 584–606.

[75] Xiao, J., and Volz, R. On replanning for assembly tasks using robots in the presence of uncertainties. *IEEE International Conference on Robotics and Automation 2* (1989), 638–645.

[76] Xiao, J., and Zhang, L. Contact constraint analysis and determination of geometrically valid contact formations from possible contact primitives. *IEEE Transactions on Robotics and Automation 13* (1997), 456–466.

[77] Yoshikawa, T. Dynamic manipulability of robot manipulators. *Journal of Robotic Systems 2* (1987), 113–124.

[78] Yoshikawa, T. *Foundations of Robotics: Analysis and Control*. MIT Press, 1990.

APPENDIX A: KINEMATICS OF THE 7-DOF BARRETT WAM

Since the Barrett WAM robotic arm was used for all experiments and planning algorithms, the following description is provided for completeness. The Denavit-Hartenberg (D-H) parameters, forward kinematics formulas and joint ranges are all based on the manual. The inverse kinematics formulas were derived independently. They are not provided by Barrett, and the ROS [58] procedure is not well-documented.

A.1 Forward Kinematics

There are several possible ways to define the transformation matrices that describe the frames of reference for the individual joints. The conventions that are used here are based on Eq. 14, see Spong et al. [65], and Asada and Slotine [3]. The parameters for the 7-DOF WAM are listed in the Table 1, where $l_1 = 55$cm, $l_2 = 30$cm, $l_3 = 6$cm, and the offset in the x-direction at the 4^{th} joint is given by $k = 4.5$cm.

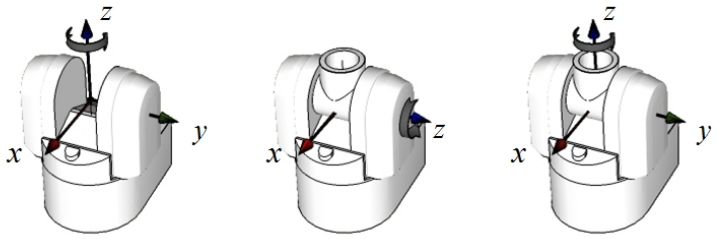


Figure 42: The coordinate systems attached to the first 3 joints θ_1, θ_2 and θ_3 respectively, shown left-to-right.

By convention, the rotation axes are denoted as z. We obtain the homogeneous transformation matrices by substituting the values from Table 1 into Eq. 14. The base coordinate system is located at the shoulder joint. This point is static, as can be seen from Fig. 42. The reference frames for the first 3 joints are all located here.

Joint	a	α	d
1^{st}	0	$-\frac{\pi}{2}$	0
2^{nd}	0	$\frac{\pi}{2}$	0
3^{rd}	k	$-\frac{\pi}{2}$	l_1
4^{th}	$-k$	$\frac{\pi}{2}$	0
5^{th}	0	$-\frac{\pi}{2}$	l_2
6^{th}	0	$\frac{\pi}{2}$	0
7^{th}	0	0	l_3

Table 1: Joint parameters and the corresponding links of the Barrett 7-DOF WAM in the Denavit-Hartenberg notation.

$$
{}^{0}T_{1} = \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{75}
$$

$$
{}^{1}T_{2} = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0 \\ \sin(\theta_2) & 0 & -\cos(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{76}
$$

$$
{}^{2}T_{3} = \begin{bmatrix} \cos(\theta_3) & 0 & -\sin(\theta_3) & k\cos(\theta_3) \\ \sin(\theta_3) & 0 & \cos(\theta_3) & k\sin(\theta_3) \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{77}
$$

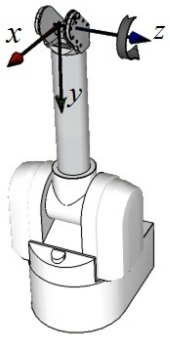


Figure 43: The coordinate systems for the elbow joint θ_4.

$$ {}^3T_4 = \begin{bmatrix} \cos(\theta_4) & 0 & \sin(\theta_4) & -k\cos(\theta_4) \\ \sin(\theta_4) & 0 & -\cos(\theta_4) & -k\sin(\theta_4) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{78} $$

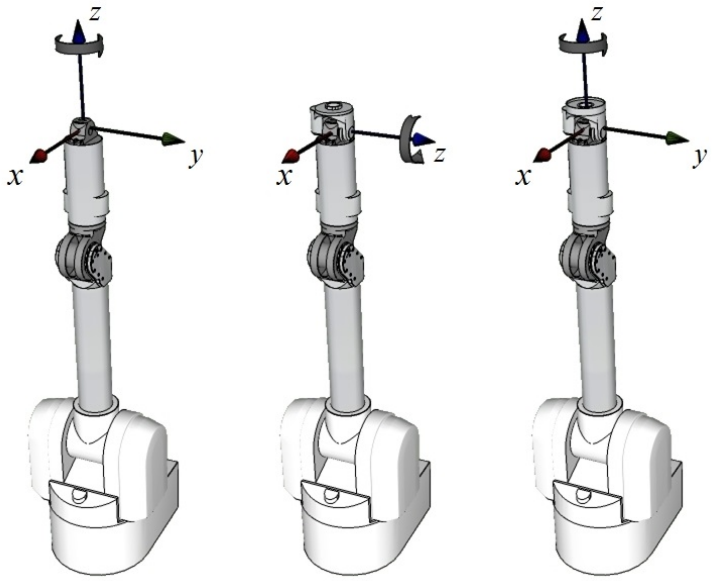


Figure 44: The coordinate systems for the joints θ_5 , θ_6 and θ_7 (from left to right).

$$ {}^4T_5 = \begin{bmatrix} \cos(\theta_5) & 0 & -\sin(\theta_5) & 0 \\ \sin(\theta_5) & 0 & \cos(\theta_5) & 0 \\ 0 & -1 & 0 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{79} $$

$$ {}^{5}T_{6} = \begin{bmatrix} \cos(\theta_6) & 0 & \sin(\theta_6) & 0 \\ \sin(\theta_6) & 0 & -\cos(\theta_6) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{80} $$

$$ {}^{6}T_{7} = \begin{bmatrix} \cos(\theta_7) & -\sin(\theta_7) & 0 & 0 \\ \sin(\theta_7) & \cos(\theta_7) & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{81} $$

Similarly, we get the corresponding matrices for the elbow and wrist joints, as depicted in Fig. 43 and Fig. 44. Finally, the dexterous space of the manipulator is defined in terms of the range for the individual joints as listed in Table 2.

Joint	Positive Joint Limit in Rad (Deg)	Negative Joint Limit in Rad (Deg)
1^{st}	2.6 (150°)	-2.6 (-150°)
2^{nd}	2.0 (113°)	-2.0 (-113°)
3^{rd}	2.8 (157°)	-2.8 (-157°)
4^{th}	3.1 (180°)	-0.9 (-50°)
5^{th}	1.3 (75°)	-4.8 (-275°)
6^{th}	1.6 (90°)	-1.6 (-90°)
7^{th}	2.2 (128°)	-2.2 (-128°)

Table 2: Joint limits of the 7-DOF Barrett WAM.

A.2 Inverse Kinematics

In this section, an inverse kinematics solution for the 7-DOF Barrett WAM robotic arm is presented. It is equivalent to the one used by Manfred Huber for the ROS [58] implementation. In order to find the joint positions of the first 4 joints, one can assume one joint value, i.e. make it an input to the procedure that computes the other joint positions. Here, one can consider the 3^{rd} joint value as an extra degree of freedom when compared to the 6-DOF manipulators. It is chosen as a free parameter, i.e. the 7^{th} DOF. In order to get the wrist position $w = [w_1, w_2, w_3]^T$, described by the offset vector of ${}^0T_5$, we note that in the non-degenerate case the 4^{th} joint position determines the distance to the origin and the 1^{st} and the 2^{nd} joint values determine two intersecting half-planes, i.e. the suitable direction vector. As for the last three joints, if the positions of all the other four joints are known, then their values can be calculated based on the rotational part of the homogeneous transformation matrix that specifies the desired end-effector pose. The calculation is analogous to computing the Z-Y-Z Euler angles and can be found, for instance, in Craig [11]. Thus, it is possible to subdivide the original problem into two simpler subproblems.

In order to make our considerations precise, let T specify the desired position and orientation of the end-effector without any tooling or gripper attached. It is equal to the matrix describing the local coordinate system of the tool plate ${}^0T_7$. The wrist position vector w can be calculated as

$$w = T_p - l_3 T_z \tag{82}$$

where T_p is the offset vector of T and T_z its z component. The same point can be calculated also as

$$w = \left({}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_5\right)_p = \left({}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\right)_p + l_2\left({}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\right)_z. \tag{83}$$

However, if we are only interested in the length of w, we can simplify the calculation further. Here, we need only the vector

$$v = \left({}^2T_3\,{}^3T_4\right)_p + l_2\left({}^2T_3\,{}^3T_4\right)_z \tag{84}$$

and its squared Euclidean norm

$$\|w\|_2{}^2 = \|v\|_2{}^2 = v^\tau v. \tag{85}$$

This last line is sufficient to calculate the position of θ_4. The multiplication by ${}^2T_3$ and ${}^3T_4$ leads to the following expression for v

$$v = \begin{bmatrix} \cos(\theta_3)\,(k - k\cos(\theta_4) + l_2\sin(\theta_4)) \\ \sin(\theta_3)\,(k - k\cos(\theta_4) + l_2\sin(\theta_4)) \\ l_1 + l_2\cos(\theta_4) + k\sin(\theta_4) \end{bmatrix} \tag{86}$$

and

$$l_1{}^2 + l_2{}^2 + 2k^2 + 2(l_1 l_2 - k^2)\cos(\theta_4) + 2(l_1 + l_2)k\sin(\theta_4) = \sum_{i=1}^{3} w_i{}^2. \tag{87}$$

Substituting the tangent half-angle identities

$$\sin(\theta_4) = \frac{2t_4}{1 + t_4{}^2} \tag{88}$$

and

$$\cos(\theta_4) = \frac{1 - t_4{}^2}{1 + t_4{}^2} \tag{89}$$

into Eq. 87, we get

$$ {l_1}^2 + {l_2}^2 + 2k^2 + \frac{4(l_1 + l_2)kt_4}{1 + {t_4}^2} + \frac{2(l_1 l_2 - k^2)(1 - {t_4}^2)}{1 + {t_4}^2} = \sum_{i=1}^{3} {w_i}^2. \tag{90} $$

The solution of Eq. 90 is given by

$$ t_4 = -\frac{b \pm \sqrt{b^2 - 4ac}}{2a}, \tag{91} $$

where

$$ a = (l_1 - l_2)^2 + 4k^2 - {w_1}^2 - {w_2}^2 - {w_3}^2) \tag{92} $$

$$ b = 4(l_1 k + l_2 k) \tag{93} $$

$$ c = ((l_1 + l_2)^2 - {w_1}^2 - {w_2}^2 - {w_3}^2). \tag{94} $$

Since $a = 0$ yields $\theta_4 = -2.68054$, which is out of range, we obtain two solutions based on $\theta_4 = 2\arctan(t_4)$.

As stated previously, the value of θ_3 is to be treated as a free parameter. Hence, our next step will be to calculate the positions of the 1st and the 2nd joint in terms of the 3rd and the 4th joint. To this end, we express the first two joints' positions using the vector v. Fig. 45 suggests that the y component of v is unaffected by the position of the 2nd joint, since it is aligned with its rotation axis. In fact, premultiplying the vector $[v, 1]^\tau$ with the product of ${}^0T_1$ for $\theta_1 = 0$ and ${}^1T_2$, which changes the coordinate system to that of the base frame, yields

$$ \begin{bmatrix} \sin(\theta_2)(...) + \cos(\theta_2)(...) \\ \sin(\theta_3)(k - k\cos(\theta_4) + l_2 \sin(\theta_4)) \\ \cos(\theta_2)(...) - \sin(\theta_2)(...) \end{bmatrix}, \tag{95} $$

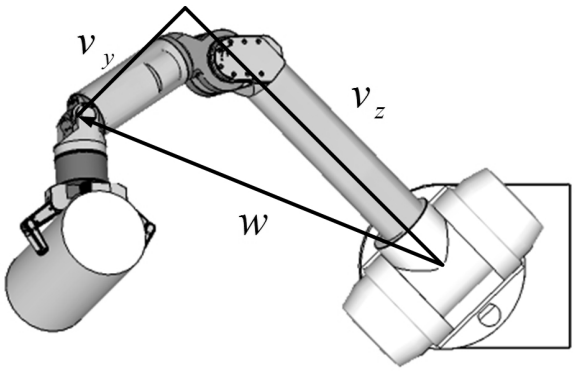


Figure 45: The v_y axis is parallel to the z axis of the second joint and, therefore, unaffected by the position of that joint. It is mapped onto the y-axis of the first joint for $\theta_1 = 0$ and any value of θ_2. v and w are in fact both the same vector expressed in different coordinate systems.

where the additional dimension that is always 1 has been omitted. This extra row is a vestige of the matrix formalism for homogeneous transformations, as defined by Eq. 11. Since the y component of the vector v remains unchanged, we conclude that for $\theta_1 = 0$ the angle between the projection of w onto the x-y plane and the x axis is given by

$$\delta_1 = \operatorname{asin}\left(\frac{\sin(\theta_3)\,(k - k\cos(\theta_4) + l_2\sin(\theta_4))}{{w_1}^2 + {w_2}^2}\right). \tag{96}$$

Now, if $\theta_3 = 0$, then the correct value for the rotation of the 1^{st} joint is $\phi_1 = \operatorname{atan2}(w_2, w_1)$. In the general case, we have to correct for the angle δ_1 based on the positions of θ_3 and θ_4, which amounts to rotating the coordiante frames and the projection. Consequently, the correct formula is $\phi_1 - \delta_1$, or

$$\theta_1 = \operatorname{atan2}(w_2, w_1) - \operatorname{asin}\left(\frac{\sin(\theta_3)\,(k - k\cos(\theta_4) + l_2\sin(\theta_4))}{{w_1}^2 + {w_2}^2}\right). \tag{97}$$

after substitution. Another solution for θ_1 is obtained by rotating the 2^{nd} joint and

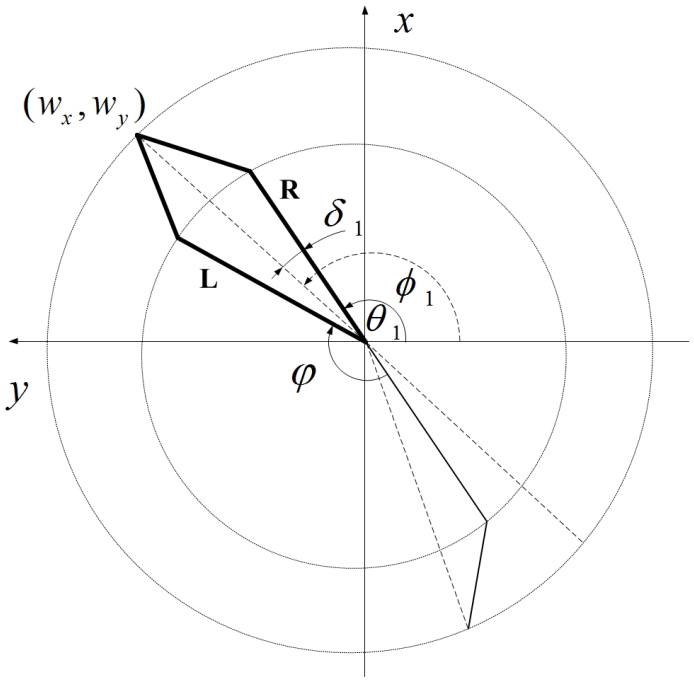


Figure 46: The relationship between the alternative solutions for θ_1, the uncorrected angle ϕ_1, the difference between them ψ, and the correction angle δ_1 for one solution. Note that the poses denoted by L and R are mirror inverted.

changing the direction in which the 1st joint is to be corrected, as shown in Fig. 46. When the correction angle $\delta_1 = \phi_1 - \theta_1$ is equal to 0, one can obtain the second solution for θ_1 by adding π to the one that is already known. In general, since its direction changes between the two solutions, the value needs to be subtracted out twice resulting in $\psi = \pi - 2\delta_1$. Additionally, one needs to ensure that the θ_1 values with $180 < \theta_1 \leq 360$ are properly mapped onto the negative angles $-180 < \theta_1 \leq 0$. Finally, if ${w_1}^2 + {w_2}^2 = 0$, then the wrist position must be on the z axis and the position of the 1st joint can be freely chosen. One can either set it to 0 or, since it will be aligned with the 5th joint, distribute the total rotation between the two joints.

To find the rotation angle for the 2nd joint, we consider the z column of the product of homogeneous transformation matrices of the first two joints $\left({}^0T_1 {}^1T_2\right)_z$. This vector,

let us denote it with u, points in the same direction as the vector w if the value of v_x is zero. Otherwise, a correction angle δ_2 needs to be subtracted from the angle ϕ_2 that is calculated by aligning the vectors u and w. We obtain

$$u = \begin{bmatrix} \cos(\theta_1)\sin(\theta_2) \\ \sin(\theta_1)\sin(\theta_2) \\ \cos(\theta_2) \end{bmatrix}. \tag{98}$$

Setting $\cos(\theta_1)u_1 + \sin(\theta_1)u_2 = \cos(\theta_1)kw_1 + \sin(\theta_1)kw_2$ and $u_3 = kw_3$ for some positive constant k, we get $\sin(\theta_2) = kw_1\cos(\theta_1) + kw_2\sin(\theta_1)$ and $\cos(\theta_2) = kw_3$, which uniquely determines the position of the 2^{nd} joint. It follows that

$$\phi_2 = \text{atan2}\left(w_1\cos(\theta_1) + w_2\sin(\theta_1), w_3\right). \tag{99}$$

From Fig. 47 we see that δ_2 has to be equal to $\text{atan2}(v_x, v_z)$. Substituting, we get

$$\delta_2 = \text{atan2}(\cos(\theta_3)\left(k - k\cos(\theta_4) + l_2\sin(\theta_4)\right), l_1 + l_2\cos(\theta_4) + k\sin(\theta_4)]), \tag{100}$$

and the value of the second joint can be calculated as

$$\begin{aligned} \theta_2 = \text{atan2}\left(w_1\cos(\theta_1) + w_2\sin(\theta_1), w_3\right) \\ - \text{atan2}(\cos(\theta_3)\left(k - k\cos(\theta_4) + l_2\sin(\theta_4)\right), \\ l_1 + l_2\cos(\theta_4) + k\sin(\theta_4)). \end{aligned} \tag{101}$$

We now turn our attention to determining the positions of the last 3 joints. If we multiply the rotational parts of the last 3 homogeneous transformation matrices, see for example Craig [11], we get the matrix equation

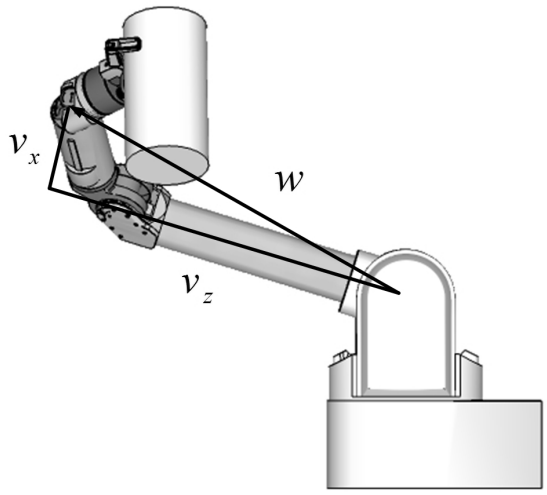


Figure 47: The x and z components of the vector v enable us to compute the correction angle for the position of the second joint.

$$ {}^4R_5{}^5R_6{}^6R_7 = \begin{bmatrix} S_{1,1} & S_{1,2} & \cos(\theta_5)\sin(\theta_6) \\ S_{2,1} & S_{2,2} & \sin(\theta_5)\sin(\theta_6) \\ -\cos(\theta_7)\sin(\theta_6) & \sin(\theta_6)\sin(\theta_7) & \cos(\theta_6) \end{bmatrix}, \tag{102} $$

where

$$ S_{1,1} = \cos(\theta_5)\cos(\theta_6)\cos(\theta_7) - \sin(\theta_5)\sin(\theta_7) \tag{103} $$

$$ S_{1,2} = -\cos(\theta_7)\sin(\theta_5) - \cos(\theta_5)\cos(\theta_6)\sin(\theta_7) \tag{104} $$

$$ S_{2,1} = \cos(\theta_6)\cos(\theta_7)\sin(\theta_5) + \cos(\theta_5)\sin(\theta_7) \tag{105} $$

$$ S_{2,2} = \cos(\theta_5)\cos(\theta_7) - \cos(\theta_6)\sin(\theta_5)\sin(\theta_7)\,. \tag{106} $$

Based on the given homogeneous transformation matrix T that specifies the desired pose of the tool plate and its rotational part R, we get the following equation

$$ {}^4R_5{}^5R_6{}^6R_7 = \left({}^0R_1{}^1R_2{}^2R_3{}^3R_4\right)^{-1} R =: Q. \tag{107} $$

Note that all the matrices on the right-hand side, denoted as Q, are now known.

Hence, we obtain the values of the 5^{th}, 6^{thd} and 7^{th} joint as

$$\theta_6 = \pm \text{acos}\left(Q_{3,3}\right) \tag{108}$$

$$\theta_5 = \text{atan2}\left(\frac{Q_{2,3}}{\sin\left(\theta_6\right)}, \frac{Q_{1,3}}{\sin\left(\theta_6\right)}\right) \tag{109}$$

$$\theta_7 = \text{atan2}\left(\frac{Q_{3,2}}{\sin\left(\theta_6\right)}, -\frac{Q_{3,1}}{\sin\left(\theta_6\right)}\right). \tag{110}$$

Both values of θ_6 are valid, because the entries of the submatrix S do not change the sign regardless of which of the two possible solutions for θ_6 is used. Indeed, if $\sin(\theta_6)$ changes the sign, so will the values of both the sine and cosine of θ_5 and θ_7. Finally, if $\sin(\theta_6)$ is zero, then the z axes of the 5^{th} and the 7^{th} joint are aligned, and the position of the 5^{th} joint can be freely chosen. It can either be set to 0 or the total rotation can be split between the two joints. This is in accordance with the relationships $Q_{1,1} = S_{1,1} = \cos\left(\theta_5 - \theta_7\right)$ and $Q_{1,2} = S_{1,2} = -\sin\left(\theta_5 + \theta_7\right)$, which can be obtained for $\cos\left(\theta_6\right) = 1$ using the angle sum formulas for sine and cosine. Note that $\cos\left(\theta_6\right) = -1$ would require $\theta_6 = \pi$, which is out of range. Setting $\theta_5 = 0$ yields $\theta_7 = \text{atan2}\left(-Q_{1,2}, Q_{1,1}\right)$.

Subject to range constraints, this leads to 8 possible solutions in total. When converting a trajectory from the Cartesian to the joint space, one is typically interested in the solution, which is closest to a particular, known joint configuration $(\theta_1^*, \theta_2^*, \theta_3^*, \theta_4^*, \theta_5^*, \theta_6^*, \theta_7^*)$, as is the case with the local search algorithm described in Fig. 6 that takes the parameter Θ_{ref} .