ABSTRACTION OF MANIPULATION TASKS TO AUTOMATICALLY GENERATE ROBOT MOTION FROM OBSERVATION

ロボット動作の自動生成のための観察による手作業の抽象化

by

Jun Takamatsu

高松 淳

A Doctoral Dissertation

博士論文

Submitted to
the Graduate School of
Information Science and Technology
the University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of Doctor of
Information Science and Technology

Abstract

A framework for automatically generating robot motion to execute various tasks has been actively studied in the field of industrial robotics; in fact, it has been a primary research topic in the field of artificial intelligence. However, it is difficult to automatically generate appropriate robot motion from only information on the purpose of a task; thus, the *Learning from Observation* (LFO) paradigm has recently been proposed.

In this paradigm, a robot system observes and recognizes a demonstration of a task, builds an abstract representation of the task, and finally generates robot motion which consists of repetitions of movement primitives. Movement primitives are essential operations to realize the situation corresponding to the abstract representation. Here, the key to successful implementation of the paradigm depends on the representation of the target tasks and the definition of sufficient movement primitives.

The target tasks of this thesis are everyday manipulation tasks including an assembly task, manipulation of linkages which are connected by a joint, and a knot-tying task.

First, we propose a robot system to automatically generate robot motion to execute assembly tasks using two rigid polyhedral objects. In the tasks, a contact relation between the two objects is employed as the abstract representation. However, it is very difficult to generate robot motion from only the representation because numerous kinds of contact relations exist. To overcome this difficulty, we define a *Motion Degree of Freedom* (DOF), which is an index that represents the quality of legal local displacement of an object in each contact relation. In actuality, we propose a method to calculate motion DOFs using 3D CAD models of the objects and the contact relations. Using the concept of practical transitions of motion DOFs, we can define sufficient movement primitives in advance and propose a method to convert the task into a sequence of the movement primitives.

Furthermore, we try to improve calculation of the legal local displacement using the second order approximation of the displacement. Although motion DOFs are calculated using the first order approximation of the displacement in the system, the approximation sometimes introduces erroneous solutions and cannot deal with the curvature information because of the truncated errors. Therefore, we propose a method to resolve these problems using the second order approximation.

Next, we propose a method to deal with manipulation of linkages which are connected by a joint. In this case, a type of a joint corresponds to the abstract

representation. Consider the task of rotating a doorknob, which is connected to a door by a revolute joint. In contrast to assembly tasks, it is very difficult to decide how to rotate a doorknob from only its 3D CAD model. Fortunately, it is possible to rotate the knob, if one knows: that the knob can rotate about some axis (a kind of the joint); the axis direction; and the center of rotation (parameters of the joint). Therefore, we propose a method to estimate the parameters from the result of 3D object tracking. Our proposed method is robust to vision errors.

Finally, we propose a method to recognize a knot-tying task using one rope. We employ the P-data representation as the abstract representation. The representation is reversible to the topological information of the knot. We define sufficient movement primitives to execute any knot-tying tasks in advance and propose a method to select the corresponding movement primitives from P-data transitions. The theoretical background comes from the *knot theory* and the *graph theory*.

論文要旨

様々な作業を行うロボット動作を自動生成することは, 産業的応用として大きな可能性があるだけでなく, 人工知能の観点からも大きなテーマであり, 現在まで様々な研究が行われてきた. しかし作業の目的のみを手がかりとして, その作業を達成する動作を自動生成することは非常に困難であり, その困難さを解決する方法として「観察による行動獲得」手法が近年注目されている.

「観察による行動獲得」手法では, まず教示者の作業を観察, 認識することにより, 作業を抽象化して表現する. それを用いて, 動作プリミティブの繰り返しとして, 目的の作業を再現することができる. 動作プリミティブとは, 抽象化して表現された作業中の状況を再現するために必要不可欠となる動作群である. つまり, この手法の適用の際には, 作業表現の方法と作業を再現するために十分な動作プリミティブの定義が非常に重要となる.

本論文では, 日常の手作業を対象とし, 組み立て作業, ジョイントにより接続されたリンク物体の操作, 紐結び作業を扱う.

まず, 2 物体の組み立て作業を再現するロボットシステムを提案する. 組み立て作業では 2 物体間の接触状態により作業を抽象化して表現することができる. しかしながら, 無数の接触状態を扱う困難さにより, この表現のみからロボット動作を生成することは非常に困難である. そこで, 各接触状態における物体の局所可動範囲の質を表す指標である運動自由度を定義する. 実際に物体の 3 次元 CAD モデルと接触状態から運動自由度を計算する方法を示す. 作業中に出現する運動自由度の変化を調べ, 作業の再現に十分な動作プリミティブをあらかじめ定義し, 実際に作業を動作プリミティブの列に変換する方法について述べる.

さらに, 局所可動範囲の式の計算を, その 2 次近似を用いて改善することを試みる. 前述のシステムでは, 運動自由度を物体の局所可動範囲の 1 次近似を用いて計算しているが, 1 次近似では時々間違った結果を導き出してしまったり, 曲率の情報を扱うことができないという問題があった. そこで 2 次近似を用いて, それらの問題を解消する方法を提案する.

次に, ジョイントにより接続されたリンク物体の操作作業を扱う方法を提案する. この場合, ジョイントの種類が作業の抽象表現に対応する. 回転ジョイントでドアとつながっているドアノブを回す作業を考える. 組み立て作業の場合と異なり, ノブの内部構造だけからドアノブの回し方を決定することは非常に困難である. しかし, ノブがある軸回りに回転すること (ジョイントの種類), さらにはその軸の向きや中心の位置 (ジョイントパラメータ) を知っていれば, ノブを回すことができる. そこで, 実際に作業の様子を観察することにより, 視覚誤差に対してロバストに, ジョイントパラメータを推定する方法を提案する.

最後に, 一本の紐を結ぶ作業を再現するロボットシステムを提案する. まず P-data

表現を用いて作業を抽象化して表現する．この表現は, 紐の位相的な特徴を完全に保持している．次に紐結び作業の再現に十分な動作プリミティブを定義し, P-data の変化から対応する動作プリミティブを選択する方法を提案する．その理論的な背景は「結び目理論」や「グラフ理論」から来ている．

Acknowledgments

I would first like to thank Professor Katsushi Ikeuchi for being my advisor and mentor. He gave me the opportunity to conduct research on the *Learning from Observation* paradigm, which is the central theme of this thesis. The research theme is so esoteric that I would like to spend my all life in pursuit of the answer of the research. I would like to reiterate my thanks to him for giving me the opportunity to conduct such research.

I would also like to express my gratitude to Professor Hiroshi Kimura and Dr. Koichi Ogawara. They kindly instructed me in robotics when I was a beginner with respect to the field. Especially, I would like to thank Koichi Ogawara for offering me the various software necessary for experiment in using our test bed.

I would also like to thank Yoshihiro Sato, who gave me the opportunity to research tasks for manipulating linkages connected by a joint, and Takuma Morita, who gave me the opportunity to research knot-tying tasks. Thanks to them, I was enabled to propose attractive research in Chapters 4 and 5.

I would also like to thank Marie Elm for spending her time to proofread and edit all of my publications, including this thesis. That I, who do not have sufficient writing skills, have achieved writing this thesis in English definitely depended on her.

I am happy to have been able share enjoyable space and time with the other members of our laboratory, especially the members of the robotics group and the Kakuri-room, not only as a member, but also as a friend. I gratefully appreciate all secretaries of our laboratory for reducing my burden of official work.

I would also like to thank my friends outside the lab. Yoshiyasu Yasutomi, Satoshi Tasumi, and Tsukiashi Koji have kept me company during my student days, – more than ten years. I am lucky for spending together the first two years important to build my character in Komaba dormitory. I gratefully appreciate Chigusa Mizutani, who gave me a lot of encouragement, particularly when I most needed it.

Finally, I would like to thank my family for their support and encouragement throughout my long-term student life.

Contents

Chapter 1

Introduction

1.1 Background

Although various kinds of robots have recently appeared in our everyday environment, almost all of them can be utilized only for a special purpose, for example, entertainment[1], cleaning, surveillance, and so on. Of course, there are multipurpose robots like humanoid robots[2, 3, 4], but they are less available to us at the present time. Even if one could possess them, one might not be able to utilize them for various purposes because of the difficulty of operating them as one would like, i.e., generating robot motion to execute a desired task.

Much research on the automatic generation using only information about the purpose of the desired task has been proposed[5]. However, the generation is very difficult, sometimes even impossible, because the necessity to solve several NP-complete problems for the generation requires much computational time.

In order to overcome this difficulty, the *Learning from Observation* (LFO) paradigm has been proposed[6, 7, 8, 9]. In this paradigm, a robot system generates robot motion not only from information about the task's purpose but also from observation of the task. Using information obtained from the observation, the system can resolve the difficulty. Furthermore, the system may be able to acquire techniques which the performer unconsciously utilizes. Therefore, a framework of the LFO paradigm is focused on the field of artificial intelligence and neuroscience[10].

Concretely speaking, in the paradigm, the system generates robot motion through the following steps[8]: First, it observes and recognizes a demonstration of a task. Next, it builds an abstract representation of the task. Finally, it generates robot motion which consists of repetitions of movement primitives. Movement primitives are essential operations to realize the situation corresponding to the abstract representation. Here, the key to successful implementation of the paradigm depends on the representation of the target tasks and the definition of sufficient movement primitives.

1.2 Thesis Contribution

In this thesis, we propose novel methods that enable a robot to recognize tasks for manipulating various kinds of objects from observation by a real-time stereo vision system, which can obtain the distance to a target object. In this case, recognition means to build an abstract representation of the tasks and to generate robot motion which consists of repetitions of movement primitives.

We first overview such manipulation tasks with respect to the type of object. With regard to the types of manipulated objects, there are the following two types:

- Rigid objects
- Deformable objects

Further, there are the following three types of deformable objects:

- String-like objects (strings, ropes, etc.)
- Plate-like objects (clothes, papers, etc.)
- Other types (clay, etc.)

In this thesis, our target objects include rigid objects and string-like deformable objects. Therefore, tasks such as paper folding, clay modeling, and so on are beyond the scope of this thesis. Because a string-like deformable object is more flexible than other deformable objects, its manipulation can be employed for various purposes, including making knots, binding together objects, and so on.

Next, we concretely consider manipulation tasks using these objects. We first consider tasks for manipulating a rigid object. The state of only one object is constant, because it is rigid. Therefore, a positional relationship among more than one object is very important for the abstract task representation.

Almost all of the manipulation involves the constraint of motion by contacts. Then, we focus on the constraint and propose a method for recognizing such manipulation by using the information of such a constrained motion. We think that there is a possibility that our proposed methods can deal with almost all of the manipulation.

With regard to how to extract the information, there are the following two methods: One method is to directly extract it from the contact relation, which just corresponds to the abstract task representation in this case. This method also requires us to know an object shape precisely. Concretely, we deal with assembly tasks as the application. As a result, we can deal with tasks to pick-and-place an object, to insert a key, a connector, etc, and so on. The key is to precisely formulate and calculate the constrained motion. Then, we also propose a novel and powerful mathematical tool which is available for that purpose.

The other method is to extract it from a primitive with respect to the constrained motion, which is defined in advance. Of course, its type corresponds to the abstract task representation. In this thesis, we focus on a so-called joint as such a primitive and concretely deal with tasks for manipulating linkages connected by a joint as the application. As a result, we can deal with tasks to open a door, a drawer, etc., to mill coffee beans using a coffee mill, to fix a screw using a screw driver, and so on. The key is to enable a method to deal with sufficient types of joints. Although we mainly deal with three types of joints in this thesis, our proposed method plays an important role in dealing with various types of primitives, including joints.

Although unconstrained motion offers very important information for recognizing the tasks, we do not explicitly deal with it in this thesis. As a result, in a task to rotate a crank, our proposed method can rotate it, but the rotation is not effective because the technique of effectively rotating should be extracted from unconstrained motion. However, we propose a method to extract as much information as possible about unconstrained motion from noise-contaminated observation.

Secondly we consider tasks for manipulating a string-like deformable object. In contrast to manipulation of a rigid object, because transitions of states of only one deformable object appear by deformation, we first propose a method to deal with the states and their transitions which correspond to the abstract task representation. In this thesis, we describe how we concentrate on designing such an abstract task representation. Although, as the result, our proposed method can deal only with making a knot, we believe that it plays an important role as the first step in manipulation for various other purposes. Concretely, we deal with knot-tying tasks. Such tasks for manipulating between a rigid and a string-like deformable object, for example, binding a package with a rope, are beyond the scope of the thesis.

As mentioned above, we deal with the following typical manipulation tasks:

Manipulation of rigid objects

- Assembly tasks using two polyhedral objects

- Tasks for manipulating linkages connected by a joint

Manipulation of deformable objects

- Knot-tying tasks using one rope

We deal with assembly tasks where constrained motion can be formulated from precise 3D CAD models of objects and contact relations between the objects. In such tasks, a contact relation corresponds to the abstract task representation. We define *Motion Degrees of Freedom* (DOF) to index the quality of the motion. Using them, we can efficiently define sufficient movement primitives for any assembly tasks. And we generate robot motion from transitions of motion DOFs.

Further, we try to improve calculation of the constrained motion using the second order approximation of the motion. Although in the previous method, motion DOFs are calculated using the first order approximation of the motion, the approximation sometimes introduces erroneous solutions and cannot deal with the curvature information because of truncated errors; thus, we propose a mathematical tool for resolving these problems using the second order approximation of the motion.

Next, we deal with tasks for manipulating linkages connected by some kind of a joint, for example, rotating a doorknob, pulling open a drawer, inserting a screw with a screw driver, etc. In this case, we determine the constrained motion from a kind of a joint. Thus, it corresponds to the abstract task representation. Manipulation of linkages also requires us to know the parameters of the joint (in the case of rotating a doorknob, we need to know the axis direction and the center of rotation). These parameters are essential for executing movement primitives. Then, we propose a method for obtaining the parameters from observation. Our proposed method is robust to vision errors.

Finally, we deal with knot-tying tasks using one rope. We employ the P-data representation as the abstract task representation. The P-data representation is reversible to the topological information of a knot. We define sufficient movement primitives to execute any knot-tying tasks and propose a method for selecting the corresponding movement primitives from P-data transitions. The theoretical background comes from the *knot theory*[11] and the *graph theory*.

1.3 Related Works

In this section, we first survey research on the Learning from Observation (LFO) paradigm based on the difference of the solution of the following two issues:

- How is a target task observed?

- How are movement primitives designed?

The implementation style deeply depends on these two issues. Next, we survey research on our target tasks mentioned above, which are not basically based on the LFO paradigm.

1.3.1 Learning from Observation Paradigm

First, we survey research on the LFO paradigm with respect to "How is a target task observed?" In some of the research methods, the task is observed through a haptic interface[12] or a robot arm[13]. These methods are similar to the so-called *teaching-pendant* method. The difference between the two is that these methods employ information which is obtained from a force/torque sensor attached to the device. However, meaningless noise motion that the performer unconsciously executes complicates the generation of robot motion. Thus, these methods require a post-process to manually remove such noise motion.

In other research methods, the task is observed through the performer executing the task in virtual reality (VR) space[14, 15]. In these methods, it is very easy for the system to observe the task; however, it is very difficult for the performer to execute the task in VR space if the system is not equipped with special haptic devices or a smart user interface.

In most research methods, the task is observed by vision systems[6, 7, 16]. The advantages of employing a vision system are as follows:

- The observation does not require a special device because a multi-purpose robot is usually equipped with a vision system for visual feedback. Note that such a vision system includes a real-time stereo-vision system which we employ in the work described in this thesis.

- The performer can perform the task naturally as always.

Some of the research methods additionally employ a sensor grove[17, 18], which is sometimes equipped with pressure sensors on the surface[19]. A sensor grove is very useful for observing motion and configurations of performer's hands, which helps in generating robot motion. However, because the grove prevents natural execution by the performer, we do not employ it.

Next, we survey research on the LFO paradigm with respect to "How movement primitives are designed?" When solving this question, one can select the following two solutions:

- Predict and implement sufficient movement primitives for the target tasks in advance
- Automatically extract and implement movement primitives through observation

The advantage of the former solution is that it is easy to implement the movement primitives, because they are known in advance. The disadvantage of the solution is that it is impossible to address the task which includes non-implemented movement primitives. Because insufficiency of the predicted movement primitives causes this disadvantage, we should fully take into account the sufficiency.

In contrast to the former solution, the advantage of the latter solution is that the system can address the task by extracting such a movement primitive from the observation and then dynamically implementing it on the system. Of course, the disadvantage is that it is very difficult to establish a mechanism to dynamically implement a movement primitive.

The reason why we select the former solution is that we believe that it is impossible to apply the latter solution to our target tasks which involve manipulation, even if all available techniques, hardwares, etc. could be employed. It is well-known that such tasks require not only position control but also force control. That means that we have to accommodate many parameters for position and force control. Such accommodation is very difficult, even if we can utilize the expert knowledge and are sufficiently familiar with the whole task in advance. Furthermore, the two assumptions are usually not satisfied.

Now we present examples of implementing movement primitives. Almost all of the research methods define the motion itself as a movement primitive, for example, trajectories of limbs[20, 21, 22, 23, 24, 25] and a variation of a joint angle[26] of the performer, a trajectory of a target object[27], and so on. Some of them edit the motion to optimize an estimation function which is defined by the user. In these methods, which we refer to as *motion-based* methods, a movement primitive is executed by going over the original motion. They can deal with a task that does not require manipulation to an external world (for example, reach motion of a hand, gesture, and so on); however, it is very difficult to deal with a manipulation task, even if it is a pick-and-place task, which is a very simple task, because movement primitives employed by the methods are based on position control.

Some of the research methods, e.g.,[28, 6, 7, 29, 15] define the motion for realizing a state, i.e., the abstract task representation, as a movement primitive. In these methods, which we refer to as *state-based* methods, in order to implement the movement primitives, we must first define the state and extract it from the observation. It is well-known that a contact relation can be utilized as such a state in assembly tasks. However, an appropriate definition of a state for other kinds of tasks has not yet been established. The movement primitives must realize any desired states. Of course, it is very difficult to implement the movement primitives which satisfy the condition. However, we believe that only the state-based method can realize the automatic generation of robot motion to execute the manipulation tasks.

1.3.2 Assembly Tasks

Assembly tasks have been actively studied from various viewpoints because such tasks are applicable not only for the field of industrial robotics but also for everyday life, and because they require intelligent control, i.e., not only position control but also force control[30].

Up to now, various research on assembly tasks has been proposed[31]. Generally speaking, as a result of this research, robot motion for executing an assembly task is generated through the following steps:

1. Represent contact relations and their transitions using a graph representation, of which a vertex represents each contact relation and an edge represents validity of a transition between two contact relations

2. Decide on a preferred sequence of transitions to realize the task by searching a path on the graph

3. Generate robot motion to realize the sequence

First we consider research on the first step, Hirai et al. proposed the method to effectively obtain such a graph representation by searching contact relations around *limit angles*, that represent an object orientation at the exact moment to translate another contact relation[32]. However, this method requires us to determine all limit angles. Xiao and Ji proposed the method of searching contact relations with a fewer contact-elements around contact relations with more contact-elements that the user manually assign[33]. However, this method requires contact relations with more contact-elements. More recently, the methods[34, 35] to automatically obtain the graph representation using the growth distance[36] and the non-linear optimization method were proposed. However, these methods require much computational time and sometimes fail to search some contact relations because of incomplete convergence of the non-linear optimization method.

Next, we consider research on the second step. Generally speaking, a process of an assembly task which corresponds to a path on the graph is realizable. However, there are usually various processes to execute the same assembly task; obviously, the process easiest to accomplish is preferred. The selection is easy, if the difficulty is defined in advance. Various methods to define the difficulty from kinematic properties have been proposed[37, 38]. Also, Uchiyama and Imahashi proposed a method for deciding the difficulty using a dynamics simulator[39]. Of course, these methods require us to obtain the graph representation in advance.

Then, we consider research on the third step, which may be somewhat remotely related to the thesis. McCarragher and Asada proposed the method for generating the motion to realize the transition and for selecting one of the optimal motions with a higher possibility to achieve the task, when such a motion cannot be uniquely determined[40]. However, the method can deal with only planer motion. Hirukawa[41] proposed the method to formulate the condition which the motion must satisfy and to solve by using the concept of the Gröbner basis. Generally speaking, to generate the motion requires us to solve non-linear equations with six variables. To solve them is so difficult that the method sometimes fail to generate the motion. Ji and Xiao[42] proposed the method to solve the condition using the Probabilistic Roadmap Method (PRM)[43], which is a kind of the randomized algorithm. They efficiently generate the motion by selecting an appropriate coordinate system. However, the method requires much calculation time and cannot generate the optimal motion to realize the transition, when the motion cannot be uniquely decided.

Applying the LFO paradigm, to solve the problems with respect to the first and the second steps is easier, because one can obtain the bare minimum of the graph representation and the preferred sequence of the transition from observation. Note that the observation includes some errors, which usually complicate obtaining the correct graph representation. We propose a method to correct the errors using roughly estimated contact relations and the validity of their transitions.

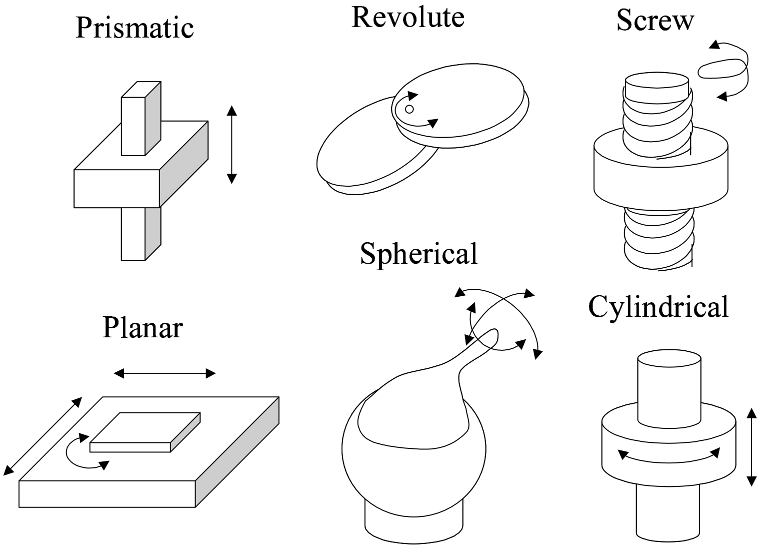


Figure 1.1: Lower pairs

Unfortunately, to solve the problem with respect to the third step is unavoidable and essential. If one does not mind the computational time and the quality of the motion, one can employ the past research. In Appendix A, we propose a method to calculate the optimal motion using only the linear solution, i.e., this method does not employ any non-linear optimization methods or randomized algorithms.

1.3.3 Manipulation of Linkages Connected by a Joint

Characteristics of joints have been actively studied in the field of the *mechanism*. The main purposes of the field are 1) to investigate kinematic and dynamic properties of various types of joints and 2) to more effectively use multiple joints. Piston, parallelogram linkages, etc. are good examples of effective use.

Research on manipulation of linkages which are connected by a joint has been conducted. For example, Mason investigated kinematic properties of six lower pairs as shown in Fig. 1.1 and illustrated a method for manipulating connected linkages based on position/force hybrid control[44].

However, few researchers have proposed a method to estimate parameters of a joint from observation. These parameters are essential for manipulating the connected linkages. Considering estimation of parameters of a revolute joint, almost all of them estimate only the center of rotation[45, 46] and cannot estimate the axis of rotation. We propose a method for estimating parameters of various kinds of joints from noise-contaminated observation.

1.3.4 Knot-Tying Tasks

A knot-tying task is a kind of manipulation of a deformable object. Almost all research concentrates on manipulation of a spring-like object, because it is well-known that the dynamic properties can be represented by the hook's law. However, there has been little research on manipulation of other deformable objects, for example, ropes, cloths, etc., because of the difficulty of modeling.

Wakamatsu and Wada[47] proposed the method to model dynamic properties of a rope using the *finite element method* (FEM). However, this method dealt only with the modelling.

As research on manipulating a rope, Inaba and Inoue proposed a hand-eye manipulation system to manipulate a rope[48]. Although they concentrated on visual feedback for success of manipulation of a rope, they did not consider how to generate the motion for executing the manipulation.

Hopcroft et al. proposed a higher language to execute a knot-tying task[49]. Using visual information, the compiler converts such a language into a robot command. However the method permits only the motion that one of two ends of a rope crosses over or under a part of a rope. Therefore, it must unnaturally tie a bow knot, which requires *Reidemeister move* II as mentioned below.

As research on how to represent a state of a knot, Wolter and Kroll proposed a method for representing a knotted rope using a graph representation[50]. This research is well suited to represent a location of a knot, but, not good enough to represent the process of tying a knot.

As mentioned above, there is the knot theory[11] for investigating characteristics of various tangled loops. As an application of such a theory, Yamada et al. proposed the method to define a state of a loop on a *Cat's Cradle*[51], which is a traditional Japanese play to design various figures from a simple loop. Each state on a Cat's Cradle is equivalent to a trivial knot, that is, it cannot be distinguished by the conventional knot theory. To classify it, they improved the so-called polynomial invariant.

As far as we know, sufficient movement primitives for knot-tying tasks have not yet been proposed. First, we define the state of a knot in knot-tying tasks. Next we introduce sufficient movement primitives for accomplishing the tasks. Then we propose a method for obtaining a sequence of the corresponding movement primitives from the state transitions of a knot-tying task.

1.4 Thesis Organization

This thesis is organized as follows: First, we describe a method to recognize assembly tasks using two polyhedral rigid objects from observation in Chapter 2. We define *Motion Degrees of Freedom* (DOF) to index the quality of the motion. Using them, we can efficiently define sufficient movement primitives for assembly tasks. In actuality, we propose a method for representing an assembly task as a sequence of corresponding sub-skills. In Appendix A we illustrate the actual implementation of sub-skills, which may be somewhat remotely related to this thesis.

Next, in Chapter 3, we describe how we try to improve calculation of the constrained motion by using the second order approximation of the motion. It is well-known that the first order approximation of the motion sometimes introduces erroneous solutions and cannot deal with the curvature information, because of truncated errors. Then, we propose a mathematical tool to resolve these problems using the second order approximation of the motion.

In Chapter 4, we propose a method for robustly estimating the parameters of joints from the observation under the existence of vision errors. The parameter is essential to manipulate linkages which are connected by a joint.

In Chapter 5, we describe a method to recognize knot-tying tasks. We describe a method to represent a knot state in the P-data representation and define sufficient movement primitives for any knot-tying tasks. In Chapter 6 we finally conclude this thesis.

Chapter 2

Recognizing Assembly Tasks Using Two Rigid Polyhedral Objects From Observation

In this chapter, we describe a method for recognizing assembly tasks using two rigid polyhedral objects from observation. First, we assume that sufficient data have been obtained from observation and concentrate on the recognition. The observed data consist of the followings:

- Transitions of contact relations
- One configuration of each object for realizing each contact relation

A contact relation consists of a set of contact primitives. Each contact primitive consists of two contacting object primitives. An object is composed of the object primitives, i.e., vertices, edges, and faces.

As mentioned above, we recognize assembly tasks using information about constrained motion, i.e., indices of the legal infinitesimal displacement of a target object. We describe a method for calculating indices from 3D CAD models of objects and contact relations. By investigating practical transitions of the indices, we define sufficient movement primitives for any assembly task. And we illustrate a method to select the corresponding movement primitives from transitions of the indices.

Next, we describe a method for extracting the observation data as mentioned above from observations which include some errors. The errors complicate extraction of the data. We propose methods for correcting the errors with respect to object configurations using a contact relation and the errors with respect to the contact relations using valid transitions of contact relations.

Note that we assume that one object is grasped by the performer and moves (referred to as a *grasping object*), while the other is fixed to the environment (referred to as an *environmental object*). Of course, the shape of each object is precisely obtained in advance.

Our proposed method in this chapter is an improvement of the method proposed by Ikeuchi and Suehiro[6]. Novelty of our method is that our method can deal with rotation and any contact relations. Note that their method can deal only with translation and face-contact relations. To realize such an improvement, we need to solve the following problems:

- Index for rotational displacement
- Deal with more kinds of transitions of contact relations

2.1 Formulating Legal Infinitesimal Displacement

Hirukawa et al. proposed a method to formulate the legal infinitesimal displacement from a contact relation and 3D CAD models of objects[52]. They showed that the displacement can be formulated by simultaneous linear inequalities as shown in Equation (2.1), where $\mathbf{F}_{ij}$ is a surface normal of a *separate plane* defined in [52] (For example, in a vertex-face-contact case, $\mathbf{F}_{ij}$ is equal to a surface normal of the face.), N is the number of contacting points, $M(i)$ is the number of separate planes in the i-th contacting point, $\Delta\mathbf{X}$ and $\mathbf{\Omega}$ are infinitesimal displacements with respect to location and orientation, and J_i is a Jacobian matrix which represents the relationship between the displacement of the i-th contacting point and the displacement of the object $[\Delta\mathbf{X}, \mathbf{\Omega}]$.

$$\bigcap_{i}^{N} \bigcup_{j}^{M(i)} \mathbf{F}_{ij}^{T} J_i \begin{pmatrix} \Delta\mathbf{X} \\ \mathbf{\Omega} \end{pmatrix} \geq 0 \tag{2.1}$$

In this thesis, we represent the displacement in the *screw* representation[53]. In the representation, the displacement can be represented as a pair of translation along a *screw* axis and rotation about the same axis. Concretely speaking, the displacement is represented as a six dimensional vector $[\mathbf{S}_0, \mathbf{S}_1]$ (referred to as a *screw vector*), where $\mathbf{S}_0$ $(\in R^3)$ is the direction of the screw axis, $\mathbf{S}_1$ is equal to $\mathbf{P} \times \mathbf{S}_0 + p\mathbf{S}_0$, $\mathbf{P}$ $(\in R^3)$ is a position of the axis, and p $(\in R)$ is a proportion of an amount of translation to rotation. When $p = 0$, the displacement represents pure rotation. And when $p = \infty$, that is, the screw vector is $[\mathbf{0}, \mathbf{S}_0]$, the displacement represents pure translation.

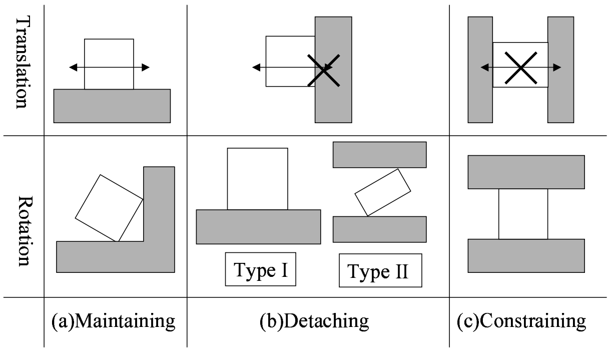


Figure 2.1: Maintaining, detaching, and constraining displacement

Using the screw representation, Equation (2.1) is converted into Equation (2.2), where $\mathbf{P}_i$ is a position of a contacting point[53]:

$$\bigcap_{i}^{N} \bigcup_{j}^{M(i)} \mathbf{F}_{ij} \cdot \mathbf{S}_1 + (\mathbf{P}_i \times \mathbf{F}_{ij}) \cdot \mathbf{S}_0 \geq 0 \tag{2.2}$$

2.2 Indexing Infinitesimal Displacement

First, we consider the case where $M(i) = 1$ for all i, that is, Equation (2.2) is one system of simultaneous linear inequalities (we refer to the case as a *non-singular* contact relation.). Next we consider another case (referred to as a *singular* contact relation).

2.2.1 Indexing in Non-Singular Contact Relations

In the case of a non-singular contact relation, the infinitesimal displacement can be classified into the following three types based on the transition of the contact relation caused by the displacement (See Fig. 2.1)[6]:

Maintaining displacement Displacement to maintain a contact relation

Detaching displacement Displacement to translate a contact relation

Constraining displacement Displacement to be constrained by contacts

DOFs of maintaining, detaching, and constraining displacement in translation m_t, d_t, and c_t can be formulated by Equation (2.3), where V_t is the solution space of Equation (2.2) when substituting $\mathbf{S}_0 = \mathbf{0}$, $d(V_t)$ is the maximum dimension of faces of the space V_t[54], $R_t = (\mathbf{F}_{11} \cdots \mathbf{F}_{N1})$, and $\mathrm{Rank}(R_t)$ returns the rank of a matrix R_t.

$$\begin{aligned} m_t &= 3 - \mathrm{Rank}(R_t) \\ d_t &= 3 - (m_t + c_t) \\ c_t &= 3 - d(V_t) \end{aligned} \tag{2.3}$$

The method for calculating $d(V_t)$ has already been proposed by Kuhn and Tucker[54], where they employ linear programming, and Hirukawa et al.[52], where they employ the singular value decomposition and convex hull. We refer to the three types of DOFs as *maintaining*, *detaching*, and *constraining* DOFs in translation. Now, we additionally define indices to deal with rotation.

First, we define DOFs of maintaining, detaching, and constraining displacement in all motion m_a, d_a, and c_a which includes not only translation but also rotation as Equation (2.4), where V_a is the solution space of Equation (2.2) and

$$R_a = \begin{pmatrix} \mathbf{F}_{11} & \cdots & \mathbf{F}_{N1} \\ \mathbf{P}_1 \times \mathbf{F}_{11} & \cdots & \mathbf{P}_N \times \mathbf{F}_{N1} \end{pmatrix}.$$

$$\begin{aligned} m_a &= 6 - \mathrm{Rank}(R_a) \\ d_a &= 6 - (m_a + c_a) \\ c_a &= 6 - d(V_a) \end{aligned} \tag{2.4}$$

We refer to the three types of DOFs as *maintaining*, *detaching*, and *constraining* DOFs in all motion.

For improvement of the recognition, we index the displacement with respect to an axis direction of rotation (See Fig. 2.1). We define the three types of DOFs as follows:

Maintaining DOF in rotation DOF of an axis direction in maintaining displacement.

Detaching DOF in rotation DOF of an axis direction in detaching displacement.

Constraining DOF in rotation DOF of an axis direction in constraining displacement.

Now, we consider a method for calculating such DOFs using Equation (2.1).

Proposition 1 *Let a subspace* $V_r = \{\mathbf{S}_0 | [\mathbf{S}_0, \mathbf{S_1}] \in V_a\}$*,*

$$c_r = 3 - d(V_r),$$

where c_r *is the constraining DOF in rotation.*

Proof This is trivial from the definition.

□

Proposition 2

$$d_r = 3 - (m_r + c_r),$$

where m_r *and* d_r *are the maintaining and the detaching DOFs in rotation, respectively.*

Proof The DOF of an axis direction is three and any displacement belongs to one of the three types (maintaining, detaching, and constraining displacement).

□

Proposition 3

$$m_r = m_a - m_t,$$

where m_r *is the maintaining DOF in rotation,*

Proof Let $\{[\mathbf{s}_{10}, \mathbf{s}_{11}], \ldots, [\mathbf{s}_{m_a 0}, \mathbf{s}_{m_a 1}]\}$ be a set of bases of Equation (2.5). Then, maintaining displacement can be represented as a linear combination of these bases.

$$\bigcap_i^N \mathbf{F}_i \cdot \mathbf{S}_1 + (\mathbf{P}_i \times \mathbf{F}_i) \cdot \mathbf{S}_0 = 0 \tag{2.5}$$

Next, by applying the algorithm as shown in Fig. 2.2 to the set of bases, we obtain the equivalent set $\{[\mathbf{0}, \mathbf{s}_{11}], \ldots, [\mathbf{0}, \mathbf{s}_{t1}], [\mathbf{s}_{t+1,0},\ \mathbf{s}_{t+1,1}], \ldots, [\mathbf{s}_{m_a 0}, \mathbf{s}_{m_a 1}]\}$. Because a linear combination of vectors in the set $\{[\mathbf{0}, \mathbf{s}_{11}], \ldots, [\mathbf{0}, \mathbf{s}_{t1}]\}$ represents translational displacement, t is equal to m_t. Because vectors in the set $\{\mathbf{s}_{t+1,0}, \ldots, \mathbf{s}_{m_a 0}\}$ are linearly independent and the linear combinations represent the range of axis directions of rotational displacement, m_r is equal to $m_a - t$. As the result, we can conclude that $m_r = m_a - m_t$.

□

Given a set of bases $B = \{[\mathbf{s}_{10}, \mathbf{s}_{11}], \dots, [\mathbf{s}_{m_a0}, \mathbf{s}_{m_a1}]\}$.

1. Search a linearly dependent minimum combination C of $\{\mathbf{s}_{10}, \dots, \mathbf{s}_{r0}\}$, where $\mathbf{s}_{i0} \neq \mathbf{0}$ for all $\mathbf{s}_{i0} \in C$. If such a combination does not exist, return B.

2. From the combination, the following equation is obtained:

$$\sum_{\mathbf{s}_{i0} \in C} a_i \mathbf{s}_{i0} = \mathbf{0} \ (a_i \neq 0)$$

3. Remove one of elements in C from B.

4. Append $[\mathbf{s}'_0, \mathbf{s}'_1]$ to B, where

$$[\mathbf{s}'_0, \mathbf{s}'_1] = \sum_{\mathbf{s}_{i0} \in C} a_i [\mathbf{s}_{i0}, \mathbf{s}_{i1}].$$

5. Go to 1.

Figure 2.2: Algorithm to calculate the maintaining DOF in rotation

In contrast, $d_r = d_a - d_t$ and $c_r = c_a - c_t$ are not always satisfied. From that, we additionally determine the facts to be as follows:

Proposition 4 *Considering some axis direction* $\mathbf{s}_0$ *in detaching displacement, there are the following two types (See Fig. 2.1):*

- *Both* $\mathbf{s}_0$ *and* $-\mathbf{s}_0$ *are included in the subspace* V_r*.*
- *Only* $\mathbf{s}_0$ *is included in the subspace.*

In the former type, a grasping object can infinitesimally rotate about the axis both clockwise and counter-clockwise. In the latter type, however, it can infinitesimally rotate either clockwise or counter-clockwise; it is obvious that the object can rotate both clockwise and counter-clockwise with respect to maintaining displacement, and cannot rotate with respect to constraining displacement.
We additionally define *Type I* and *Type II* detaching DOFs in rotation as follows:

Type I detaching DOF in rotation DOF of the former type of an axis direction in detaching displacement

Type II detaching DOF in rotation DOF of the latter type of an axis direction in detaching displacement

Proposition 5 *The type I detaching DOF in rotation d_{r1} can be calculated by performing the following steps:*

1. *Generate Equation (2.6) of which the solution is equal to the subspace V_a. Note that the method to generate it has already been proposed[54, 52].*

$$\bigcap_i G_i \cdot S_0 \geq 0 \tag{2.6}$$

2. *Calculate the rank of $G = (G_1 \cdots G_n)$. Then d_{r1} is equal to $3 - Rank(G) - m_r$.*

Proof The solution of Equation (2.6) represents any axis directions in the legal (i.e., maintaining and detaching) displacement. $3 - Rank(G)$ represents a DOF of the solution of Equation (2.7). The solution represents the range of axis directions about which a grasping object can rotate both clockwise and counterclockwise. Thus, $3 - Rank(G) = m_r + d_{r1}$ is always satisfied.

$$\bigcap_i G_i \cdot S_0 = 0 \tag{2.7}$$

□

Proposition 6

$$d_{r2} = d_r - d_{r1},$$

where d_{r2} is the Type II detaching DOF in rotation.

Proof Trivial.

□

2.2.2 Indexing in Singular Contact Relations

In [52], when a contact relation which includes some of two types of contact primitives as shown in Fig. 2.3 (referred to as *singular contact primitives*), multiple separate planes exist as some contacting point; therefore, $M(i) \neq 1$ for some i. In the contact relation, we cannot formulate the displacement as one system of simultaneous linear inequalities.

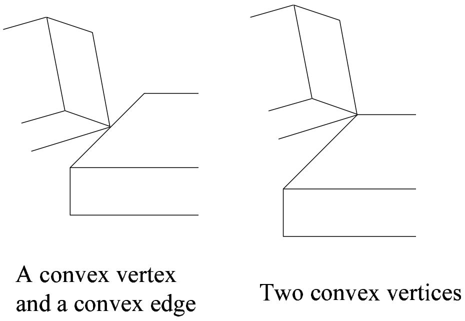


Figure 2.3: Two kinds of singular contact primitives

We define three additional indices as follows: singular maintaining, singular detaching, and singular constraining DOFs. These DOFs are equal to maintaining, detaching, constraining DOFs in the contact relation which is obtained by removing all singular contact primitives from the original contact relation. That is, we can calculate the indices by applying the method mentioned above to Equation (2.8).

$$\bigcap_{\{i|M(i)=1\}} \mathbf{F}_i \cdot \mathbf{S}_1 + (\mathbf{P}_i \times \mathbf{F}_i) \cdot \mathbf{S}_0 \geq 0 \tag{2.8}$$

Note that we define that maintaining, detaching, and constraining DOFs are zero in a singular contact relation. Is the same way, singular maintaining, singular detaching, and singular constraining DOFs are zero in a non-singular contact relation. The method proposed by Ikeuchi and Suehiro[6] does not explicitly deal with singular contact relations.

2.3 Analysis of Transitions of Contact Relations

In this section, we define sufficient movement primitives (referred to as *sub-skills*) and describe a method for obtaining a sequence of corresponding *sub-skills* from transitions of contact relations. For the purpose, Ikeuchi and Suehiro[6] classified all face-contact relations into ten kinds based on the difference of maintaining, detaching, and constraining DOFs in translation. Next, they proved that only 13 kinds of transitions existed between the ten kinds. Then, they concretely investigated the 13 kinds of transitions, defined sufficient movement primitives, and designed the method for selecting the corresponding movement primitive from the transition.

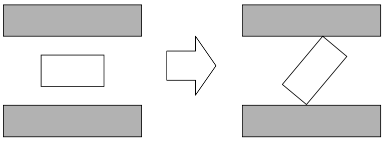


Figure 2.4: DOF-transition from a maintaining DOF to a constraining DOF

First, we tried to classify all non-singular contact relations based on the difference of maintaining, detaching, and constraining DOFs in translation and rotation. As a result, we found that 138 kinds of non-singular contact relations existed and we concluded that it was not practical to concretely investigate every possible transition. In this thesis, we propose a method for analyzing the transition based on increase and decrease of six types of indices (referred to as *motion DOFs*) defined above.

2.3.1 DOF-Transitions

From the definition, it is obvious that each sum of all motion DOFs in translation and rotation is always constant, i.e., three. Therefore, if one of the indices increases, another one decreases. Considering one pair of the increased and decreased indices, ${}_6\mathrm{P}_2 = 30$ kinds of pairs may exist. We refer to such an increase and a decrease as a DOF-transition. We define a DOF-transition from an A DOF to a B DOF as any transition between two contact relations which decreases an A DOF by one and increases a B DOF by one.

Theoretically speaking, all the 30 kinds of DOF-transitions actually appear. However, we focus on 20 kinds of DOF-transitions as shown in Fig. 2.5 among the 30 kinds, because these 20 kinds of DOF-transitions frequently appear, while the other 10 kinds of DOF-transitions seldom appear.

The 10 kinds of DOF-transitions are as follows:

- Six kinds of DOF-transitions which appear when a grasping object moves while maintaining a singular contact relation
- Four kinds of DOF-transitions between a maintaining DOF and a (singular) constraining DOF

The former seldom appears, because it is very difficult to move a grasping object while maintaining a singular contact relation. The latter also seldom appears, because such a DOF-transition appears only when directly translating between the

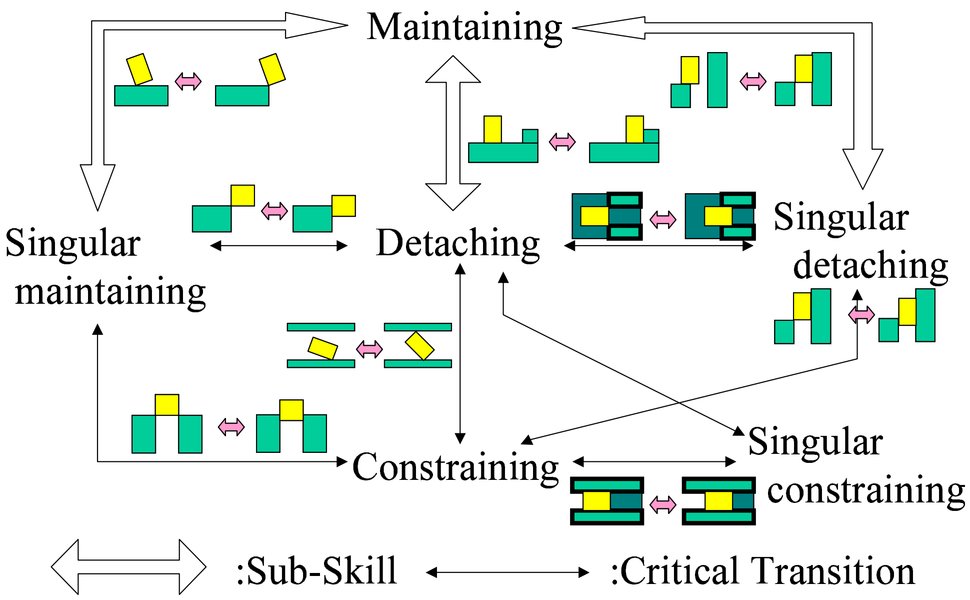


Figure 2.5: Practical transition which appears in practical assembly tasks. Transitions which are not shown in this figure are so impractical that they seldom appear in the tasks.

two contact relations as shown in Fig. 2.4. Of course it is also very difficult to realize such a transition.
The former corresponds to six kinds of DOF-transitions between two of singular maintaining, singular detaching, and singular constraining DOFs. The latter corresponds to four kinds of DOF-transitions between a maintaining DOF and a constraining DOF, and between a maintaining DOF and a singular constraining DOF. As a result, we have only to consider the 20 kinds of DOF-transitions as shown in Fig. 2.5.

2.3.2 Investigating the 20 Kinds of Practical DOF-Transitions

Based on the relationship between directions of displacement and of a basis corresponding to a DOF-transition, these 20 kinds of DOF-transitions are classified into two types as follows: The first type includes the following six kinds of DOF-transitions: maintaining DOF ↔ detaching DOF, maintaining DOF ↔ singular maintaining DOF, and maintaining DOF ↔ singular detaching DOF. In these cases, each direction is corresponding. We regard motion to realize these DOF-

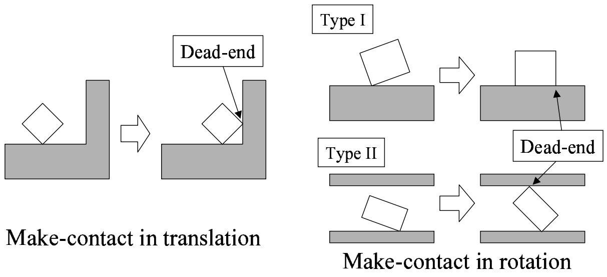


Figure 2.6: DOF-transition from a maintaining DOF to a detaching DOF

transitions as movement primitives and call them *sub-skills*.

The second type includes another 14 kinds of DOF-transitions. Of course, each direction is different in these cases. They are useful to determine how execution errors of a robot arm prevent the DOF-transition. We define the DOF-transition seriously prevented by the errors as a *critical transition*. First, we will introduce sub-skills. Next we will illustrate critical transitions.

2.4 Sub-skills

In all figures of the following three sections, let the white object and the gray object represent the grasping object and the environmental object, respectively.

2.4.1 From Maintaining DOF to Detaching DOF

The left in Fig. 2.6 shows an example of motion to lead to the DOF-transition from a maintaining DOF to a detaching DOF in translation. Before and after the transition, translational displacement along the horizontal direction, which corresponds to the direction of the motion, translates from maintaining displacement to detaching displacement. As a result, the motion leads to the DOF-transition in translation. We define motion to lead to the DOF-transition as a **make-contact sub-skill in translation**.

The top right and the bottom right in Fig. 2.6 show examples of motion to lead to the DOF-transition from a maintaining DOF to Type I and Type II detaching DOFs in rotation, respectively. Before and after the transition, rotational displacement about the perpendicular direction in this thesis translates from maintaining displacement to Type I or Type II detaching displacement. As a result, each motion leads to the DOF-transition from a maintaining DOF to a Type I or Type II

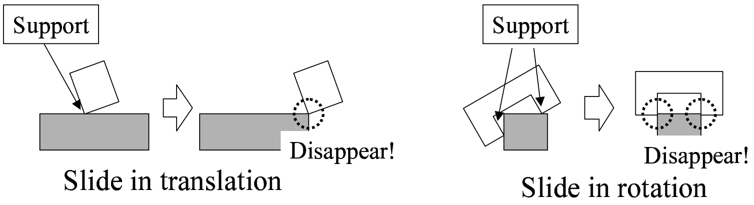


Figure 2.7: DOF-transition from a maintaining DOF to a singular maintaining DOF

detaching DOF in rotation. We define motion to lead the DOF-transitions as **Type I** and **Type II make-contact sub-skills in rotation**, respectively. Note that, in the example shown at the bottom right in the figure, a DOF-transition from a detaching DOF to a constraining DOF also appears along the vertical direction.

These sub-skills can be realized to move the grasping object while maintaining the contact relation until it contacts the *dead-end* contact primitive. This characteristic is useful for implementation.

2.4.2 From Maintaining DOF to Singular Maintaining DOF

The left in Fig. 2.7 shows an example of motion to lead to a DOF-transition from a maintaining DOF to a singular maintaining DOF in translation. In this example, the DOF-transition appears along the horizontal direction. We define motion to lead to the DOF-transition in translation as a **slide sub-skill in translation**. Note that, in this example, a DOF-transition from a detaching DOF to a singular maintaining DOF in translation also appears along the vertical direction.

The right in Fig. 2.7 shows an example of motion to lead to the DOF-transition in rotation. In this example, the DOF-transition appears about the perpendicular direction to this thesis. We define motion to lead the DOF-transition in rotation as a **slide sub-skill in rotation**. Note that, in this example, DOF-transitions from a detaching DOF to a singular maintaining DOF in translation also appear along the horizontal and vertical directions.

In contrast to a make-contact sub-skill, there is no dead-end contact primitive after the transition. The DOF-transition is always accompanied by a DOF-transition from a detaching DOF to a singular maintaining DOF in translation. That is, these sub-skills can be realized to move the grasping object while maintaining the contact relation until the *support* contact primitives disappear.

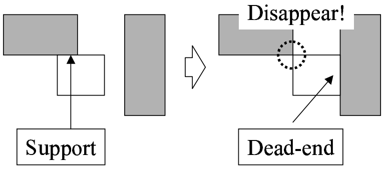


Figure 2.8: DOF-transition from a maintaining DOF to a singular detaching DOF

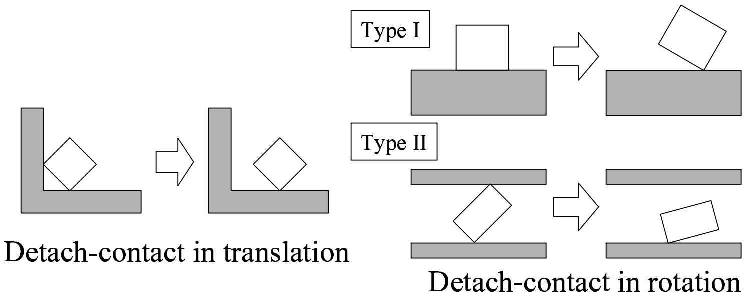


Figure 2.9: DOF-transition from a detaching DOF to a maintaining DOF

2.4.3 From Maintaining DOF to Singular Detaching DOF

Figure 2.8 shows an example of motion to lead to a DOF-transition from a maintaining DOF to a singular detaching DOF in translation. In this example, the DOF-transition appears along the horizontal direction. The motion resembles a make-contact sub-skill and a slide sub-skill in translation. Actually, the contact relation before the transition includes a support contact primitive and the contact relation after the transition includes a dead-end contact primitive. Motion to lead to the DOF-transition in translation can be substituted by a make-contact sub-skill in translation. Therefore, we do not additionally define a sub-skill corresponding to it.

In the case of rotation, that is the same. Therefore, motion to lead to the DOF-transition in rotation is substituted for by a make-contact sub-skill in rotation.

2.4.4 Another DOF-Transition

The left and the right in Fig. 2.9 show examples of motion to lead to DOF-transitions from a detaching DOF to a maintaining DOF in translation and rotation, respectively. Such motion is the inverse motion of a make-contact sub-skill

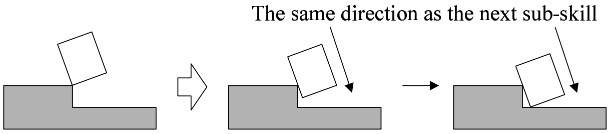


Figure 2.10: DOF-transition from a singular maintaining DOF to a detaching DOF

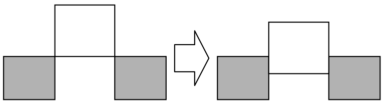


Figure 2.11: DOF-transition from a singular maintaining DOF to a constraining DOF

in translation or rotation. We define motion to lead to the DOF-transition in translation and/or rotation as **detach-contact sub-skills in translation** and/or **rotation**, respectively. However, the process of an assembly task usually decreases maintaining DOFs. Therefore, such sub-skills seldom appear. And motion to lead to a DOF-transition from a singular detaching DOF to a maintaining DOF can be substituted for by a detach-contact sub-skill, and seldom appears.
Figure 2.10 shows an example of motion to lead to a DOF-transition from a singular maintaining DOF to a maintaining DOF. Such motion is the inverse motion of a slide sub-skill and regarded as a part of the successive sub-skill. Therefore, we do not additionally define a sub-skill.

2.5 Critical Transitions

2.5.1 From Singular Maintaining DOF to Constraining DOF

Figure 2.11 shows an example of the transition to involve DOF-transitions from a singular maintaining DOF to a constraining DOF. In this example, the DOF-transitions appear along the horizontal direction in translation and about the perpendicular direction to this thesis in rotation.
The DOF-transition requires us to precisely control an object configuration with respect to the DOF-transition, because the grasping object must move through a *narrow entrance* for the transition. That is, the DOF-transition is a critical transition. This critical transition corresponds to the critical dimension defined by Miura et al. [55]. They controlled the configuration using visual feedback.

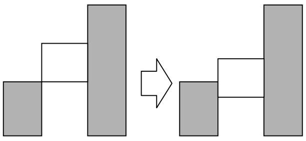


Figure 2.12: DOF-transition from a singular detaching DOF to a constraining DOF

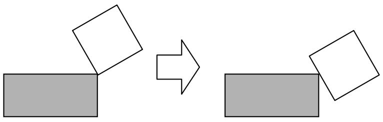


Figure 2.13: DOF-transition from a singular maintaining DOF to a detaching DOF

2.5.2 From Singular Detaching DOF to Constraining DOF

Figure 2.12 shows an example of the transition to involve DOF-transitions from a singular detaching DOF to a constraining DOF. In this example, the DOF-transitions appear along the horizontal direction in translation and about the perpendicular direction to this thesis in rotation.
Because the DOF-transition also requires us to precisely control an object configuration with respect to the DOF-transition, it is a critical transition. However, in this case, the grasping object before the transition can move along only one way with respect to the direction of the DOF-transition because of contacts. The constraint eases to precisely control the object configuration.

2.5.3 From Singular Maintaining DOF to Detaching DOF

Figure 2.13 shows an example of the transition to involve DOF-transitions from a singular maintaining DOF to a detaching DOF. Of course, it is difficult to realize the singular contact relation before the transition; however, the transition is not prevented by small execution errors. Therefore, this DOF-transition is not a critical transition.

2.5.4 Singular Detaching DOF ↔ Detaching DOF and Singular Constraining DOF ↔ Constraining DOF

Figure 2.14 shows an example of the transition to involve DOF-transitions between a singular detaching DOF and a detaching DOF. The legal displacement with re-

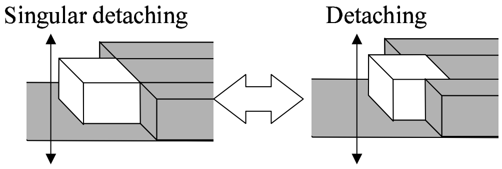


Figure 2.14: DOF-transitions between a singular detaching DOF and a detaching DOF

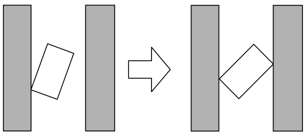


Figure 2.15: DOF-transition from a detaching DOF to a constraining DOF

spect to such DOF-transitions is one-sided before the transition, and the errors seldom occur in that direction. Therefore, the DOF-transitions are not critical transitions. As the same manner, the DOF-transitions between a singular constraining DOF and a constraining DOF are not critical transitions.

2.5.5 Another DOF-Transition

Figure 2.15 shows an example of the transition to involve a DOF-transition from a detaching DOF to a constraining DOF. The DOF-transition is accompanied by a Type II make-contact sub-skill in rotation. The legal displacement with respect to such DOF-transitions is one-sided before the transition and the errors seldom occur in the direction. Therefore, the DOF-transition is not a critical transition. Figure 2.16 shows examples of the transitions to involve the following DOF-transitions:

1. From a constraining DOF to a singular maintaining DOF

2. From a constraining DOF to a singular detaching DOF

3. From a detaching DOF to a singular maintaining DOF

These DOF-transitions are accompanied by a slide sub-skill. The displacement with respect to the first and second DOF-transitions is completely constrained, i.e., the errors cannot arise in that direction. The displacement with respect to the

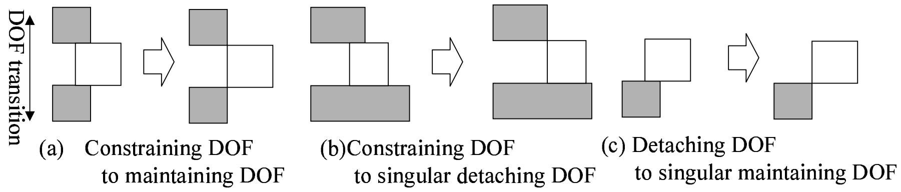

DOF transition
(a) Constraining DOF to maintaining DOF
(b)Constraining DOF to singular detaching DOF
(c) Detaching DOF to singular maintaining DOF

Figure 2.16: DOF-Transitions accompanied by a slide sub-skill

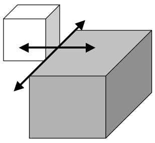


Figure 2.17: True singular maintaining DOF or not

third DOF-transition is one-sided before the transition and the errors seldom occur in the direction. Therefore, these three DOF-transitions are not critical transitions.

2.6 Selection of Corresponding Sub-Skills and Extraction of Critical Transitions from DOF-Transitions

From the definition of sub-skills, the selection rule can be estimated as follows:

- Select a make-contact sub-skill in translation and/or rotation, when a DOF-transition from a maintaining DOF to a detaching DOF in translation and/or rotation appears, respectively.
- Select a detach-contact sub-skill in translation and/or rotation, when a DOF-transition from a detaching or singular detaching DOF to a maintaining DOF in translation and/or rotation appears, respectively.
- Select a slide sub-skill in translation and/or rotation, when a DOF-transition from a maintaining DOF to a singular maintaining DOF in translation and/or rotation appears, respectively.

The first and the second rules are correct; however, the third one is not correct. Over-simplification in analysis of a singular contact relation causes this result.

For example, consider a parallel edge-edge contact as shown in Fig. 2.17. Of course, this is a singular contact relation. Because the singular maintaining DOF in translation is three, any translational displacement should have an equivalent characteristic. The grasping object maintains the contact relation while it translates along the edge. However it does not maintain the relation while it translates along the direction perpendicular to the edge. For selecting a slide sub-skill, we need to classify these two types of singular maintaining DOFs.

Now, we consider how to classify singular maintaining DOFs into these two types. The former singular maintaining DOF resembles a maintaining DOF, that is, the grasping object maintains a contact relation while it moves along the corresponding direction. We focus on such a characteristic, and additionally define restricted DOFs in translation, rotation, and all motion.

A restricted DOF is a DOF of restricted displacement. Restricted displacement is defined as the displacement which cannot maintain a contact relation despite a singular or not-singular contact relation[1].

When the legal infinitesimal displacement is represented by Equation (2.2), from the definition restricted DOFs in translation, rotation, and all motion are calculated as follows:

Restricted DOF in translation $r_t = \mathrm{Rank}(R_t)$

Restricted DOF in rotation $r_r = r_a - r_t$

Restricted DOF in all motion $r_a = \mathrm{Rank}(R_a)$

where,

$$\begin{aligned} R_t &= \left(\mathbf{F}_{11} \cdots \mathbf{F}_{NM(N)}\right) \\ R_a &= \begin{pmatrix} \mathbf{F}_{11} & \cdots & \mathbf{F}_{NM(N)} \\ \mathbf{P}_1 \times \mathbf{F}_{11} & \cdots & \mathbf{P}_N \times \mathbf{F}_{NM(N)} \end{pmatrix}. \end{aligned}$$

Using restricted DOFs, the correct rule to select the corresponding sub-skill is as follows:

- Select a make-contact sub-skill in translation and/or rotation, when a DOF-transition from a maintaining DOF to a detaching DOF in translation and/or rotation appears, respectively.

[1] From the definition, each restricted DOF is equal to the sum of detaching and constraining DOFs with respect to translation, rotation, and all motion in a not-singular contact relation, respectively. However, each restricted DOF is not equal to the sum of singular detaching and singular constraining DOFs in a singular contact relation.

- Select a detach-contact sub-skill in translation and/or rotation, when a DOF-transition from a detaching or singular detaching DOF to a maintaining DOF in translation and/or rotation appears, respectively.

- Select a slide sub-skill in translation and/or rotation, when a DOF-transition from a maintaining DOF to a singular maintaining DOF in translation and/or rotation appears and a **restricted DOF** increases in translation and/or rotation, respectively.

The rule for extracting critical transitions is as follows:

- The transitions to involve a DOF-transition from a singular maintaining DOF to a constraining DOF.

- The transitions to involve a DOF-transition from a singular detaching DOF to a constraining DOF.

2.7 Extracting Assembly Task Information from Observation

In this section, we describe the actual implementation to extract assembly task information from the observation. Unfortunately, data obtained from the observation usually include some errors. As a result, erroneous assembly task information, i.e., object configurations and contact relations, may be obtained.
Then, we propose two methods to correct the errors with respect to object configurations using a contact relation and a method to correct errors with respect to contact relations using the validity of the transition. Generally speaking, to decide the validity requires much computational time because the calculation is NP-complete[5]. Therefore, we propose a fast method to practically determine the validity.

2.7.1 Outline

We employ a multi-baseline real-time stereo vision system which can obtain aligned intensity and disparity images at 30 fps[56] as the observation device. The top left and the top center in Fig. 2.18 show examples of intensity and disparity images which were actually obtained from the vision system.
We extract assembly task information from the obtained images through performing the following steps:

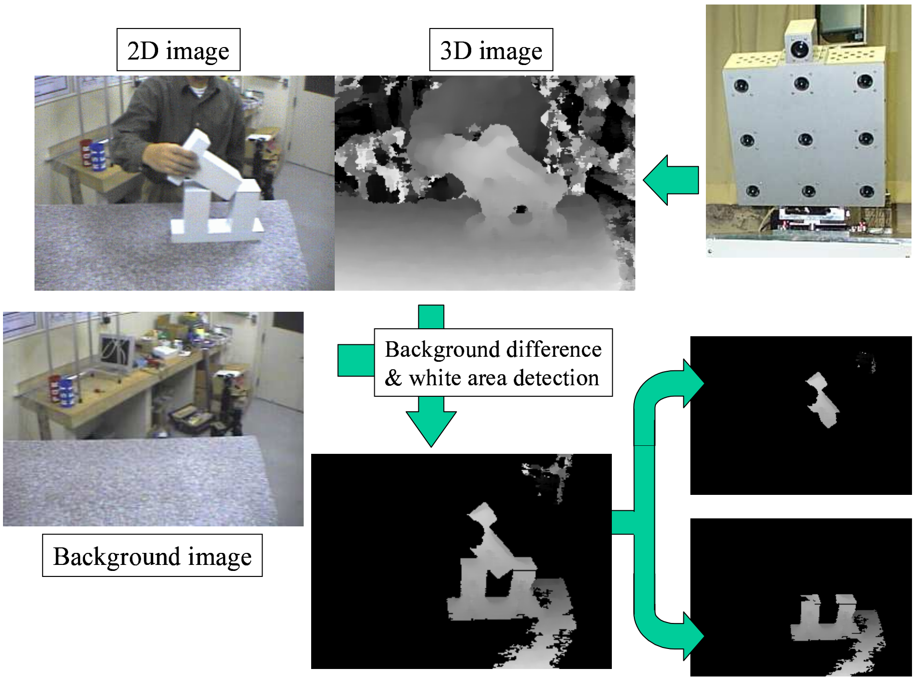


Figure 2.18: Robot vision system

1. Extract only target objects from obtained images and estimate 6-DOF object trajectories.

2. Calculate contact relations and their transitions using the obtained trajectories and 3D CAD models of the target objects.

As mentioned above, estimated 6-DOF trajectories usually include some errors. For correcting the errors, our system first roughly estimates a contact relation using the noise-contaminated trajectories. Next, the system corrects the errors with respect to object configurations using the estimated contact relation. Finally, it removes the contact relations erroneously estimated using the idea of the validity of the transition.

2.7.2 Extraction of Target Objects

First, our system extracts only target objects from images obtained by the vision system. We assume that illumination does not dramatically change and object color is white. Therefore our system extracts the objects using the background subtraction and the white area detection on the intensity image. The disparity image has already been aligned to the intensity image; therefore, our system can easily extract only the objects on the disparity image (See Fig. 2.18).

Next, our system classifies these objects into grasping and environmental objects by using histograms which represent duration of existence of objects on a pixel. Note that an environmental object is always fixed.

Then our system obtains 6-DOF trajectories of a grasping object and a 6-DOF configuration of an environmental object using the three dimensional template matching (3DTM) method proposed by Wheeler et al.[57]. The 3DTM method is a kind of the *iterative closest point* (ICP) method[58]. The advantage of the 3DTM method is that it is robust to noises because it employs the weighted least square method, while the conventional ICP method employs the normal least square method on the other.

2.7.3 Correct Errors With Respect to Object Configurations

Next, we describe two methods for correcting errors with respect to object configurations using a contact relation. One is an improvement of the method proposed by Suehiro et al.[59]. Although their method can deal only with a face-contact relation, our method can deal with any contact relation. We refer to our method as a *non-linear vision error correction.*

The other is an application for calculating the optimal trajectory (We illustrate the method in Appendix A.). We refer to the method as a *linear vision error correction.* The method corrects errors by calculating the optimal trajectory for realizing the transition from a contact relation with no contact primitive to a desired contact relation.

The advantage of a non-linear vision error correction is that it requires only an initial guess with respect to the configurations, which are usually obtained from the observation. The disadvantage is to not always be able to correct the errors. Use of the non-linear optimization method causes the disadvantage. Fortunately, to recognize assembly tasks, we have to extract at most only one configuration of each object against each contact relation.

In contrast, the advantage of a linear vision error correction is to always be able to correct the errors. However the disadvantage is to require one configuration of each object to realize the contact relation, which we would like to extract from the observation. That means that the method is not available to extract assembly task information. However, by using this method, one can extract information about the unconstrained motion. Because our current system employs only the constrained motion, it extracts assembly task information using only a non-linear vision error correction.

Estimation of Contact Relations From Observation

The methods which we will propose in this section require us to roughly estimate contact relations from 3D CAD models and object configurations which include some errors. To estimate the relation, our system calculates the distance between every pair of two object-elements (a vertex, an edge, or a face). If the distance is less than the proper threshold, our system determines that these two object-elements contact each other[2].

Non-linear Vision Error Correction

Next, our system corrects the errors using the roughly estimated contact relation. Generally speaking, calculating configuration of each object which realizes some contact relation is equivalent to solving one system of simultaneous non-linear equations. It is very difficult to solve[3]. Our system calculates such a configuration by using the non-linear optimization method. The method requires the initial guess. Fortunately, a configuration obtained from the vision can be employed as the initial guess[59]. In actuality, the method to correct the errors using a contact relation is as follows:

1. Formulate the relationship between the distance Δ_i between two object-primitives in the i-th contact primitive and the object configurations $\mathbf{q}$ as shown in Equation (2.9).
$$\Delta_i = f_i(\mathbf{q}) \tag{2.9}$$
2. Solve $\bigcap_i \Delta_i = 0$ using the iterative non-linear optimization method.

Our system employs the following optimization, where $\mathbf{q}_0$ is the initial guess, which is obtained from the vision:

1. $\mathbf{q}_c = \mathbf{q}_0$.
2. Answer $\mathbf{q}_c$, if $\max_i f_i(\mathbf{q}_c)$ is sufficiently small.

[2] Xiao and Ji proposed the method to more exactly calculate a contact relation following the metrics of vision errors which they defined[60]. However the method requires much computational time. That is why we employ such a simple calculation.

[3] However, such a problem can be analytically solved in the planer case[61].

3. Linearize $f_i(\mathbf{q})$ using the zeroth and the first order terms of the Taylor series of the equation.

4. Solve one system of simultaneous linearized equations using the singular value decomposition and substitute such a solution to $\mathbf{q}_c$.

5. Go to Step 2.

Linear Vision Error Correction

To correct the errors with respect to object configurations is equivalent to calculating trajectories from them to one of the configurations which realize a desired contact relation. That means that the calculation of the optimal trajectory as mentioned in Appendix A can be employed for such a purpose. When correcting the errors, we can assume that the contact state before the transition includes no contact primitive, because only the end of the trajectory is important.
The advantage of the method is to be able to always correct the errors. Therefore, the method can save the observed configurations of which non-linear vision error correction fails to correct the errors. The data are not directly employed to recognize the assembly task. However, one can extract information of the unconstrained motion using the method. In actuality, the method tends to reduce the amount of the displacement for correcting the error as compared with a non-linear vision error correction.

2.7.4 Correcting Errors With Respect to the Transition

In the previous section, we described two methods for correcting errors with respect to object configurations using roughly estimated contact relations. However, there is a possibility that the contact relation includes some errors. To be more robust to vision errors, our system corrects such estimation errors using the validity of obtained transitions of contact relations, that is, every transition should be valid. Donald proved that, if the translation between two given contact relations A and B is valid, B must include every contact primitive in A, or A must include every contact primitive in B[62]. Note that the inverse condition is not always satisfied. Each contact relation can be represented by a set of the following three contact primitives: vertex-face, face-vertex, and edge-edge contacts. In polyhedral objects, every contact primitive can be represented as a combination of the three contact primitives.

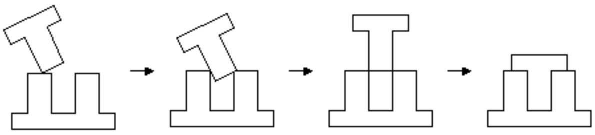


Figure 2.19: Peg-insertion task

To determine the validity of the transitions requires the calculation of connections of c-surfaces which are sets of all object configurations to maintain some contact relation[63]. It is well-known that calculation of c-surfaces is NP-complete[5].
To overcome such a difficulty, our system determines the validity using practical DOF-transitions as shown in Fig. 2.5. If DOF-transitions except for the 20 kinds as shown in Fig. 2.5 appear, the system decides that the transition is invalid. The figure represents only DOF-transitions which practically appear when the performer executes assembly tasks, that is, other DOF-transitions do not practically appear. Because motion DOFs are calculated on the recognition, our system can practically decide the validity of the transitions with less calculation time.

2.8 Experiment

In this section, we illustrate the results of recognizing an assembly task using subskills and critical transitions based on DOF-transitions from the observation.
We employ a peg-insertion task as shown in Fig. 2.19 which looks like a two dimensional case, for verifying our system. We select this example, because 1) various contact relations and their transitions appear and 2) the result is easy to watch and understand. Of course, our system can also recognize three dimensional peg-insertion, etc.

2.8.1 Extracting Assembly Task Information

In this section, we show the result of extracting assembly task information from observation. First, we show the results of correcting the errors using a non-linear vision error correction in Fig. 2.20. Because rough estimation tends to adopt a contact relation with more contact primitives, the system could find a singular contact relation as shown in Fig. 2.20 (b), which plays an important role in recognizing the task.

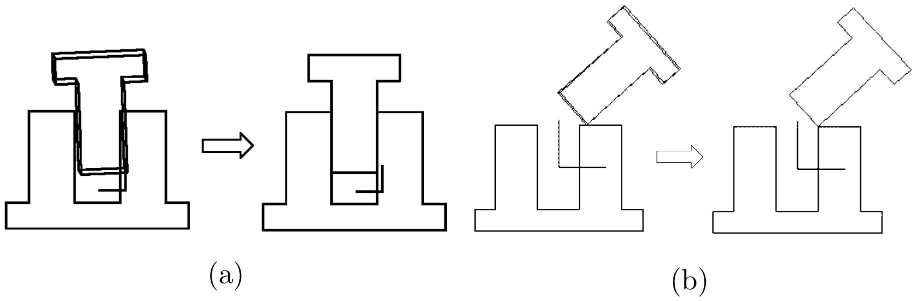


Figure 2.20: Correct errors with respect to object configurations using a contact relation: success cases

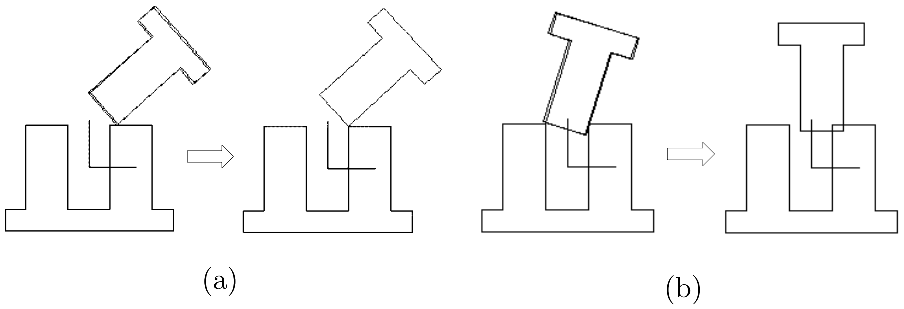


Figure 2.21: Correct errors with respect to object configurations using a contact relation: failure cases

However, the system sometimes failed to correct the errors as shown in Fig. 2.21 (a) and (b), because of failure of rough estimation of a contact relation or inappropriateness of the initial guess employed by the non-linear optimization. In particular, more failures appeared when the performer was just inserting a peg into a hole as shown in Fig. 2.21 (b).

Then, the system determined the validity of the transitions. The system could have obtained the correct transitions shown by outline arrows in Fig. 2.22 by correcting invalid transitions. Note that in Fig. 2.22, two-direction arrows show other valid transitions to be found by the method proposed by Donald[62] and black marks show the transitions rejected by our proposed method[4].

[4] We investigated only transitions between two contact relations obtained from the vision system.

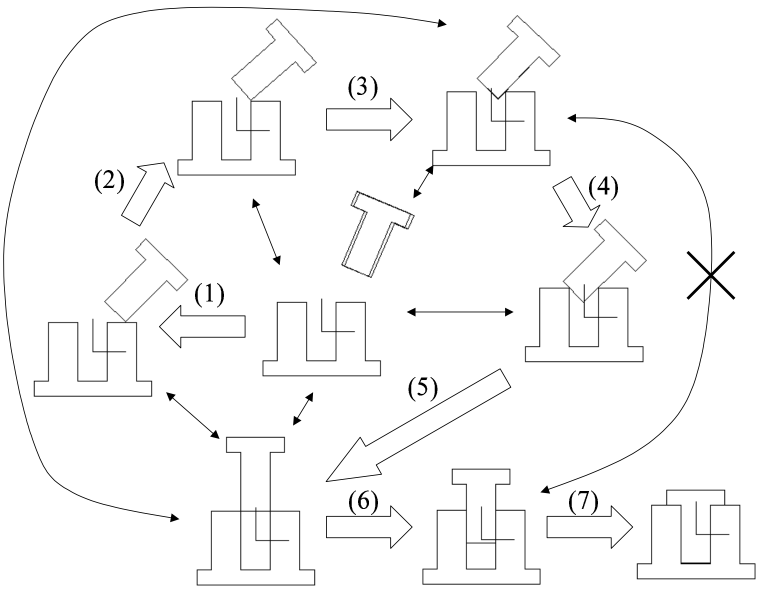


Figure 2.22: Valid transitions of contact relations

2.8.2 Recognizing Assembly Tasks From Assembly Task Information

Figure 2.23 shows the result of recognizing the peg-insertion. Note that numbers under each contact relation represent maintaining, detaching, and constraining DOFs in translation, maintaining, detaching, and constraining DOFs in rotation, and restricted DOFs in translation and rotation from the left. However, in singular contact relations C and F, they represent singular maintaining, singular detaching, and singular constraining DOFs in translation, singular maintaining, singular detaching, and singular constraining DOFs in rotation, and restricted DOFs in translation and rotation. An outline arrow shows a critical transition.

Transition (1) decreases maintaining DOFs in both translation and rotation by one and increases detaching DOFs in both translation and rotation by one, respectively. Therefore the system selects make-contact sub-skills in translation and rotation. These sub-skills align the edge of the peg and the face of the hole. Because an increased detaching DOF in rotation is Type I, the make-contact sub-skill in rotation is Type I. The same DOF-transitions appear in Transition (4) and (7), and the system selects the same sub-skills in the two transitions.

Transition (2) decreases maintaining DOFs in both translation and rotation by two and increases singular maintaining DOFs in both translation and rotation

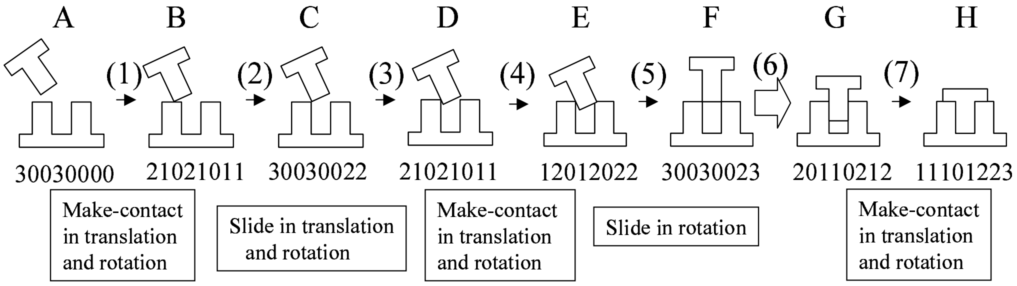

A
B
C
D
E
F
G
H
(1)
(2)
(3)
(4)
(5)
(6)
(7)
30030000
21021011
30030022
21021011
12012022
30030023
20110212
11101223
Make-contact in translation and rotation
Slide in translation and rotation
Make-contact in translation and rotation
Slide in rotation
Make-contact in translation and rotation

Figure 2.23: Result of assembly-task recognition

by two and restricted DOFs in both translation and rotation by one, respectively. Therefore, the system selects slide sub-skills in translation and rotation. These sub-skills align the edge of the peg and the edge of the hole. Furthermore, the transition decreases detaching DOFs in both translation and rotation by one and increases singular maintaining DOFs in both translation and rotation by one, respectively. These DOF-transitions are accompanied by the slide sub-skills.

Transition (5) decreases maintaining DOFs in both translation and rotation by one and increases singular maintaining DOFs in both translation and rotation by one, respectively. However, this transition increases a restricted DOF only in rotation. Therefore, the system selects only a slide sub-skill in rotation.

Transition (3) decreases singular maintaining DOFs in both translation and rotation by two and increases maintaining DOFs in both translation and rotation by two, respectively. Therefore, the system regards these DOF-transitions as a part of the next sub-skill. And this transition decreases singular maintaining DOFs in both translation and rotation by one and increases detaching DOFs in both translation and rotation by one, respectively. Therefore, this transition is not a critical transition.

Transition (6) decreases a singular maintaining DOF in translation by one and a singular maintaining DOF in rotation by two, and increases a constraining DOF in translation by one and a constraining DOF in rotation by two. Therefore this transition is a critical transition.

2.8.3 Efficiency of Linear Vision Error Correction

Finally, we show the efficiency of the linear vision error correction. As mentioned above, the method is basically useless for recognizing assembly tasks, because the

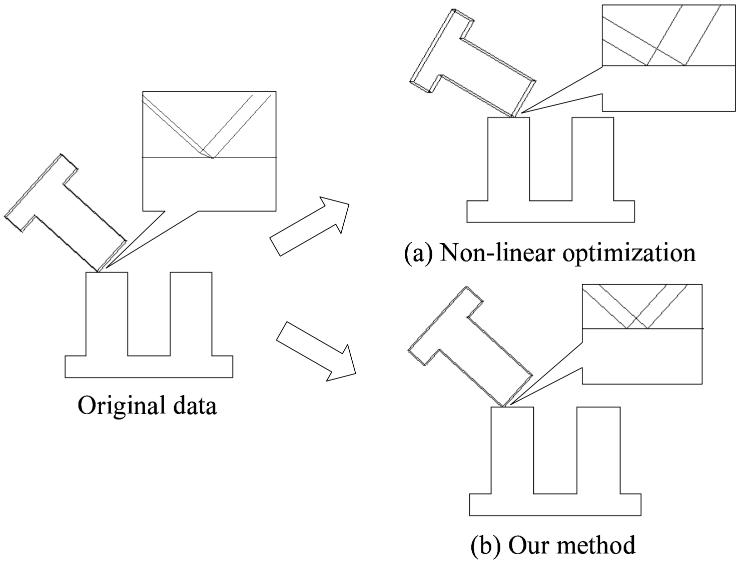


Figure 2.24: Result of Vision error correction 1

method cannot explore an unexplored contact relation from the observation at all. However, the configurations corrected by the method possess preferable characteristics and we can extract information about the unconstrained motion.
The efficiency of the linear vision error correction is as follows:

- Always correct vision errors
- Reduce an amount of the displacement to correct vision errors

Figure 2.24, 2.25 and 2.26 show the efficiency of the linear vision error correction. In each figure, the left represents object configurations obtained from the observation, the top right represents the result of the non-linear vision error correction, and the bottom right represents the result of the linear vision error correction.
In the case shown in Fig. 2.24, the non-linear method can correct the errors. Of course, the linear method can also correct them. However, in the non-linear method, the orientation widely changes to correct the errors. The linear method minimizes the change of the orientation.
In the cases shown in Fig. 2.25 and 2.26, the non-linear method cannot correct the errors. However, the linear method can correct the errors. The method always works well in spite of the difference of restricted DOFs, which is relative to the difficulty of the vision error correction.

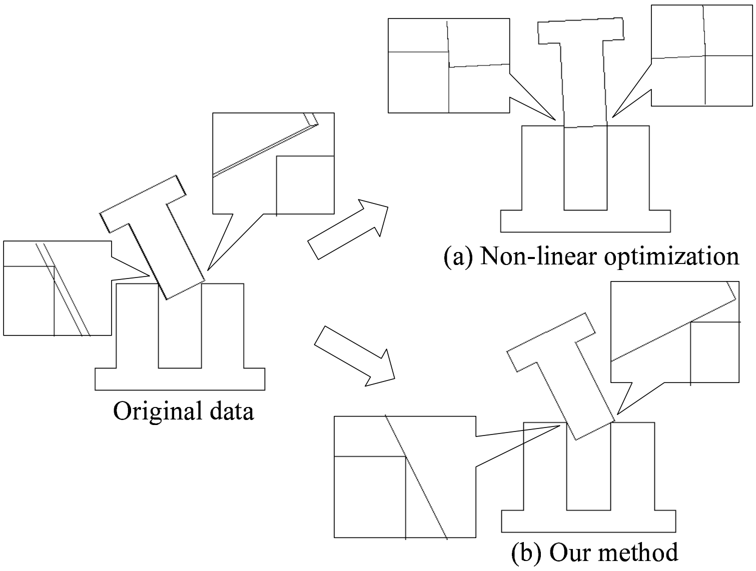


Figure 2.25: Result of Vision error correction 2

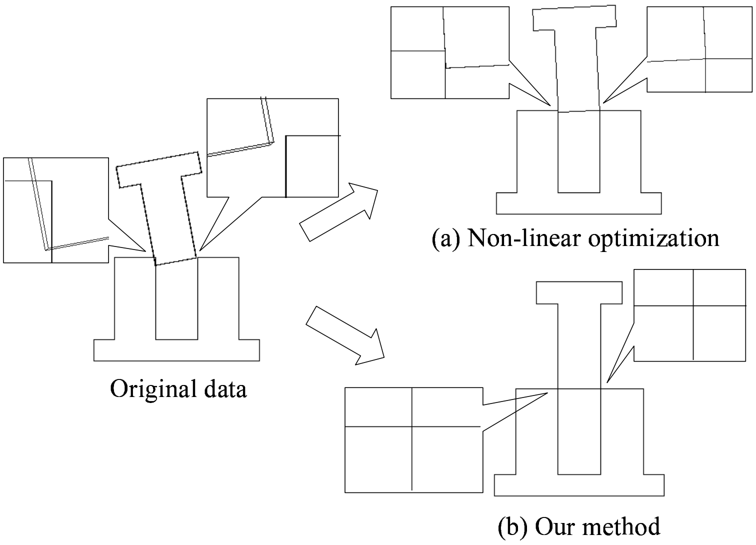


Figure 2.26: Result of Vision error correction 3

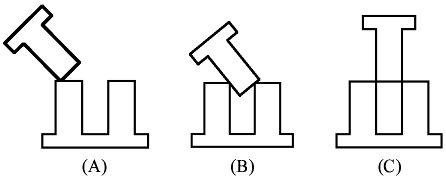


Figure 2.27: Three contact relations which are employed for comparing the non-linear method to the linear method

Next, we show the reduction of an amount of the displacement in the linear vision error correction. In this experiment, we contaminate the correct configurations which satisfy some contact relation, and then correct them using the non-linear and the linear method. Section 4.5.2 describes how to contaminate in detail.
We employed the three contact relations as shown in Fig. 2.27. In every contact relation, the linear method always succeeded in correcting the errors. And the average of an amount of the displacement to correct errors in the linear method was less than one in the non-linear method as shown in Table 2.1. Note that the average was calculated using only the successful case. However the average of an amount of the orientation displacement in the linear method was a little greater than the one in the linear method. The result means that, the non-linear method cannot correct the bigger errors.
In the linear correction, the displacement with respect to the orientation and location was less than the average of amounts of the noises (5[deg] and 10[mm], respectively). Especially, in Contact relation (C), the average of an amount of the displacement with respect to the orientation was equal to the average of an amount of the noise, because the restricted DOF in rotation is three. Of course, the smaller the restricted DOF is, the smaller an amount of the displacement is.
The non-linear method often failed to correct the errors, when a restricted DOF in all motion was three or four. It is easier to solve the non-redundant equation or sparse equation.

Table 2.1: Comparison between the linear method and the non-linear vision error correction. Average is calculated using only the successful case.

Contact Relation (A)

	The non-linear vision error correction	The linear vision error correction
Successful rate [%]	84.6	100.0
Orientation (Average) [deg]	6.80	1.24
Location (Average) [mm]	4.44	2.41

Contact Relation (B)

	The non-linear vision error correction	The linear vision error correction
Successful rate [%]	49.4	100.0
Orientation (Average) [deg]	2.84	1.97
Location (Average) [mm]	4.35	3.79

Contact Relation (C)

	The non-linear vision error correction	The linear vision error correction
Successful rate [%]	82.7	100.0
Orientation (Average)[deg]	2.43	2.51
Location (Average) [mm]	4.82	3.77

Chapter 3

Mathematical Tools For Calculating Constrained Motion Using its Second Order Approximation

In this chapter, we describe our attempt to improve calculation of the constrained motion using the second order approximation of the motion. In the previous chapter, motion DOFs were calculated using the first order approximation (i.e., linearization) of the motion. The linearization characteristics are good because a powerful tool to solve such inequalities, the *theory of the polyhedral convex cones*[54], has already been established. The facility of the solution generates various applications, for example, determining the grasping stability[64] and the difficulty of assembly tasks[31, 37, 38], calculating feasible paths[52, 65], generating a contact relation graph[33, 35], recognizing assembly tasks[6], and so on.

However, the linearization sometimes introduces erroneous solutions[66] and cannot deal with the curvature information, i.e., the applications mentioned above are useless for curved objects. For example, the linearization introduces that the legal displacement of a cubic object in each contact relation as shown in Fig. 3.1 is the same. That is much different from the truth. The purpose of this chapter is to resolve these problems using the second order approximation of the motion.

For the purpose, we first need to formulate the second order approximation of a function which represents the relationship between the motion, i.e., displacement and the distance between two of the following three object primitives: a vertex, an edge, and a face. Next, we introduce a method for calculating maintaining and legal displacement using the second order approximation. In actuality, we examine the effectiveness of the second order approximation as compared with the linearization

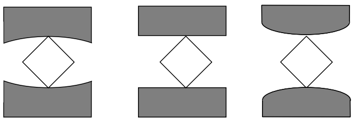


Figure 3.1: Same legal displacement?

through two examples.

Our work described in this thesis is similar to the work of Rimon and Burdick[67]. The novelty of our work is as follows:

- Our work deals with any basic contact relations, the definition of which is presented below, while their work deals only with contact relations which consist of only several face-face contacts, which are included by basic contact relations.

- The purpose of our work is to calculate maintaining or legal displacement itself rather than to index maintaining or legal displacement. Therefore, our work can be employed not only for indexing grasping stability – Rimon and Burdick's purpose – but also for the other purposes mentioned above.

Note that, the contact relation which usually appears is a basic contact relation. One example of a contact relation that is not basic is a singular contact relation as described in Chapter 2.

We concentrate on a two-object relation. One is freely movable (referred to as a *moving object*) and the other is fixed (referred to as a *fixed object*).

The method proposed in this thesis sometimes cannot calculate the displacement. Experimental results show that our method is sufficient to calculate maintaining displacement. The calculation plays a particularly important role for the various applications mentioned above. If a method to deal with a quadratic form with multi-variables is proposed, the complete calculation can be established.

3.1 Preliminaries

3.1.1 Notation

In this chapter, we represent attributes of the moving object or the fixed object by using small letters or capital letters, respectively. For example, a vertex of the moving object is represented by v and an edge of the fixed object is represented by E.

We represent names of vertices, edges, and faces (these three are referred to as *object primitives*) using v, e, and f. Note that the function name which represents a shape of an edge or a face is the same as the name itself. These symbols sometimes have an index. In particular, we employ the symbol with an index for representing an object primitive which is adjacent to another object primitive.

Every edge e has two adjacent faces f_1 and f_2. Every vertex v has n adjacent edges e_i and n adjacent faces f_i. An edge e_i has two adjacent faces f_i and f_{i+1}, where a face f_{n+1} is the same as a face f_1. Two edges e_{i-1} and e_i are adjacent to a face f_i, where an edge e_0 is the same as an edge e_n.

We employ symbols $\mathbf{s}_i$ for representing local displacement of an object and $*_d$ represents the value of $*$ after the displacement. Because we often employ the derivative with respect to $\Delta\theta$, such a derivative is represented by $'$ $\left(\text{for example } \mathbf{s}'_1 = \dfrac{\partial \mathbf{s}_1}{\partial \Delta\theta}\right)$.

3.1.2 Curved Lines and Surfaces

Every edge e is an arbitrary simple curved line, but, we locally substitute its second order approximation. Therefore, any point $\mathbf{x}$ on the edge about a point $\mathbf{x}_c$ is locally represented by Equation (3.1), where $\mathbf{t}$ and $\mathbf{p}$ are tangent and principle normals of the edge at a point $\mathbf{x}_c$, respectively, k is a curvature at a point $\mathbf{x}_c$, and l is the length of the curved line between two points $\mathbf{x}$ and $\mathbf{x}_c$. The range of l is from $-\epsilon$ to ϵ, where ϵ is sufficiently small positive real number.

$$\mathbf{x} = \mathbf{e}(l) = \mathbf{x}_c + \mathbf{t}l + \frac{1}{2}k\mathbf{p}l^2 \tag{3.1}$$

Every face f is an arbitrary simple curved surface, but we locally substitute its second order approximation. Therefore, any point $\mathbf{x}$ on the face is locally represented by an implicit function as Equation (3.2), where $\mathbf{n}$ and m are an outer surface normal and a curvature matrix at a point $\mathbf{x}_c$, respectively. The range of $\mathbf{x}$ is limited by $\rho(\mathbf{x}_c, \mathbf{x}) < \epsilon$, where $\rho(\mathbf{x}_c, \mathbf{x})$ returns the distance between two points $\mathbf{x}_c$ and $\mathbf{x}$.

$$f(\mathbf{x}) = \mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_c) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^T m(\mathbf{x} - \mathbf{x}_c) = 0 \tag{3.2}$$

Note that the distance between the face and some point $\mathbf{x}$ is locally equal to $f(\mathbf{x})$ with respect to the second order approximation.

3.1.3 Convexity

Let A be a set of all inside points of an object. Let f_1 and f_2 be two adjacent faces to an edge e. Then we naturally define the second order convexity of the edge at the neighborhood of a point $\mathbf{x}_c$ on the edge as Equation (3.3).

$$f_1(\mathbf{x}) < 0 \cap f_2(\mathbf{x}) < 0 \ (\forall \mathbf{x} | \mathbf{x} \in A \cap \rho(\mathbf{x}_c, \mathbf{x}) < \epsilon) \tag{3.3}$$

Let f_i be the i-th adjacent face to a vertex v, of which position is $\mathbf{x}_c$. As the same manner, we define the second order convexity of the vertex at its neighborhood as Equation (3.4).

$$\bigcap_i f_i(\mathbf{x}) < 0 \ (\forall \mathbf{x} | \mathbf{x} \in A \cap \rho(\mathbf{x}_c, \mathbf{x}) < \epsilon) \tag{3.4}$$

Every object is represented by a union of several convex objects which include only convex vertices and convex edges. From now on in this thesis, we assume that all vertices and edges are convex.

3.1.4 Screw Representation

To represent local displacement, we employ the screw representation[53]. As mentioned above, in the representation, the displacement is represented as a pair of translation along a *screw axis* and rotation about the same axis. Every displacement can be represented by the screw representation. In this chapter, direction, position, etc. are usually represented with respect to the base coordinate system. Let some displacement be represented in the screw representation, that is, the direction of the screw axis is $\mathbf{a} = (a_x, a_y, a_z)^T$, its position is $\mathbf{c}$, an amount of rotation is $\Delta\theta$ (≥ 0), and an amount of translation is $p\Delta\theta$, where $|\mathbf{a}| = 1$. Note that all variables of the screw representation except for $\Delta\theta$ are functions with respect to $\Delta\theta$. After the displacement, a position of a point $\mathbf{x}$ is calculated by Equation (3.5), where I is a unit matrix.

$$\mathbf{x}_d = \mathbf{d}(\mathbf{x}) = R(\mathbf{x} - \mathbf{c}) + \mathbf{c} + p\mathbf{a} \tag{3.5}$$

$$R = \exp([\mathbf{a}]_\times \Delta\theta) = I + [\mathbf{a}]_\times \Delta\theta + \frac{[\mathbf{a}]_\times^2}{2!}\Delta\theta^2 + \cdots$$

$$[\mathbf{a}]_\times = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}$$

Note that when $p \to \infty$, the displacement is equivalent to pure translation.

After the displacement, any curved line $\mathbf{x} = \mathbf{e}(l)$ and any curved surface $f(\mathbf{x}) = 0$ are represented by Equation (3.6) and Equation (3.7), respectively.

$$\begin{aligned} \mathbf{e}_d(l) &= \mathbf{d}(\mathbf{x}_c) + R\mathbf{t}l + \frac{1}{2}kR\mathbf{p}l^2 && (3.6) \\ f_d(\mathbf{x}) &= (R\mathbf{n}) \cdot (\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) + \frac{1}{2}(\mathbf{x} - \mathbf{d}(\mathbf{x}_c))^T RmR^T(\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) && (3.7) \end{aligned}$$

3.1.5 Derivatives

To formulate the second order approximation, we often employ the derivatives of the functions mentioned above. This section shows derivatives of important functions. The first and second derivatives of R with respect to $\Delta\theta$ are represented by Equation (3.8) and (3.9), respectively.

$$\begin{aligned} R' &= ([\mathbf{a}']_\times \Delta\theta + [\mathbf{a}]_\times) \exp([\mathbf{a}]_\times \Delta\theta) && (3.8) \\ R'' &= ([\mathbf{a}'']_\times \Delta\theta + 2[\mathbf{a}']_\times) \exp([\mathbf{a}]_\times \Delta\theta) \\ &+ ([\mathbf{a}']_\times \Delta\theta + [\mathbf{a}]_\times)^2 \exp([\mathbf{a}]_\times \Delta\theta) && (3.9) \end{aligned}$$

By substituting $\Delta\theta = 0$ to Equation (3.8) and (3.9), Equation (3.10) and (3.11) are obtained, where $\mathbf{s}_1 = \mathbf{a}(0)$ and $\mathbf{s}_3 = 2\mathbf{s}_1' = 2\mathbf{a}'(0)$.

$$\begin{aligned} R'(0) &= [\mathbf{s}_1]_\times && (3.10) \\ R''(0) &= [\mathbf{s}_3]_\times + [\mathbf{s}_1]_\times^2 && (3.11) \end{aligned}$$

The first and second derivatives of $\mathbf{d}(\mathbf{x})$ with respect to $\Delta\theta$ are represented by Equation (3.12) and (3.13), respectively, where $\mathbf{x}$ is constant with respect to $\Delta\theta$.

$$\begin{aligned} \mathbf{d}'(\mathbf{x}) &= R'(\mathbf{x} - \mathbf{c}) - R\mathbf{c}' + \mathbf{c}' + p'\mathbf{a} + p\mathbf{a}' && (3.12) \\ \mathbf{d}''(\mathbf{x}) &= R''(\mathbf{x} - \mathbf{c}) - 2R'\mathbf{c}' - R\mathbf{c}'' + \mathbf{c}'' + p''\mathbf{a} + 2p'\mathbf{a}' + p\mathbf{a}'' && (3.13) \end{aligned}$$

By substituting $\Delta\theta = 0$ to Equation (3.12) and (3.13), Equation (3.14) and (3.15) are obtained, where $\mathbf{s}_2 = \mathbf{c}(0) \times \mathbf{s}_1 + p'(0)\mathbf{s}_1$ and $\mathbf{s}_4 = 2\mathbf{s}_2' - p''(0)\mathbf{s}_1 = 2\mathbf{c}'(0) \times \mathbf{s}_1 + \mathbf{c}(0) \times \mathbf{s}_3 + p''(0)\mathbf{s}_1 + p'(0)\mathbf{s}_3$. Note that $p(0) = 0$.

$$\begin{aligned} \mathbf{d}'(\mathbf{x})|_{\Delta\theta=0} &= \mathbf{s}_1 \times \mathbf{x} + \mathbf{s}_2 && (3.14) \\ \mathbf{d}''(\mathbf{x})|_{\Delta\theta=0} &= \mathbf{s}_3 \times \mathbf{x} + \mathbf{s}_4 + \mathbf{s}_1 \times (\mathbf{s}_1 \times \mathbf{x} + \mathbf{s}_2) && (3.15) \end{aligned}$$

We define the vector $[\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4]$ as the screw vector. We also define two vectors $[\mathbf{s}_1, \mathbf{s}_2]$ and $[\mathbf{s}_3, \mathbf{s}_4]$ as the first and second order screw vectors, respectively. When

$p'(0) \to \infty$, i.e., $\mathbf{s}_1 = \mathbf{s}_3 = \mathbf{0}$, the screw vector represents pure translation. Any screw vectors are valid if $\mathbf{s}_1 = 0 \cap \mathbf{s}_3 \neq 0$ is not satisfied.

The derivative of a curved line $\mathbf{e}(l)$ with respect to $\Delta\theta$ is represented by Equation (3.16), where l is a function with respect to $\Delta\theta$.

$$(\mathbf{e}_d(l))' = \mathbf{e}'_d(l) + \frac{\partial \mathbf{e}_d(l)}{\partial l} l' \tag{3.16}$$

Note that

$$\begin{aligned} \mathbf{e}'_d(l) &= \mathbf{d}'(\mathbf{x}_c) + R'\mathbf{t}l + \frac{1}{2}kR'\mathbf{p}l^2 && (3.17) \\ \frac{\partial \mathbf{e}_d(l)}{\partial l} &= R\mathbf{t} + kR\mathbf{p}l. && (3.18) \end{aligned}$$

By substituting $\Delta\theta = 0$ and $l = 0$ to Equation (3.17) and (3.18), Equation (3.19) and (3.20) are obtained, respectively.

$$\begin{aligned} \mathbf{e}'_d(l)\big|_{\Delta\theta=0,l=0} &= \mathbf{d}'(\mathbf{x}_c)|_{\Delta\theta=0} && (3.19) \\ \left.\frac{\partial \mathbf{e}_d(l)}{\partial l}\right|_{\Delta\theta=0,l=0} &= \mathbf{t} && (3.20) \end{aligned}$$

The derivative of $f_d(\mathbf{x})$ with respect to $\Delta\theta$ is represented by Equation (3.21), where $\mathbf{x}$ is a function with respect to $\Delta\theta$.

$$f_d(\mathbf{x}) = f'_d(\mathbf{x}) + \frac{\partial f_d(\mathbf{x})}{\partial \mathbf{x}} \mathbf{x}' \tag{3.21}$$

Note that

$$\begin{aligned} f'_d(\mathbf{x}) &= (R'\mathbf{n}) \cdot (\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) - (R\mathbf{n}) \cdot \mathbf{d}'(\mathbf{x}_c) + \mathbf{d}'(\mathbf{x}_c)^T RmR^T(\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) \\ &\quad + (\mathbf{x} - \mathbf{d}(\mathbf{x}_c))^T R'mR^T(\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) && (3.22) \\ \frac{\partial f'_d(\mathbf{x})}{\partial \mathbf{x}} &= R\mathbf{n} + RmR^T(\mathbf{x} - \mathbf{d}(\mathbf{x}_c)). && (3.23) \end{aligned}$$

By substituting $\Delta\theta = 0$ and $\mathbf{x} = \mathbf{x}_c$ to Equation (3.22) and (3.23), Equation (3.24) and (3.25) are obtained, respectively.

$$\begin{aligned} f'_d(\mathbf{x})\big|_{\Delta\theta=0,\mathbf{x}=\mathbf{x}_c} &= -\mathbf{n} \cdot \mathbf{d}'(\mathbf{x}_c)|_{\Delta\theta=0} && (3.24) \\ \left.\frac{\partial f'_d(\mathbf{x})}{\partial \mathbf{x}}\right|_{\Delta\theta=0,\mathbf{x}=\mathbf{x}_c} &= \mathbf{n} && (3.25) \end{aligned}$$

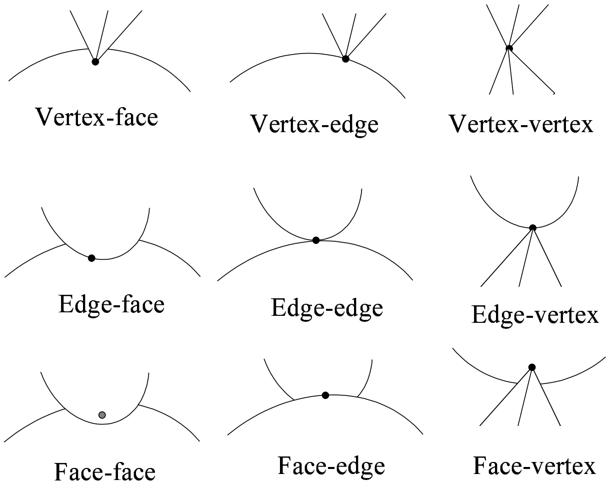


Figure 3.2: Nine types of point-contact primitives

3.2 Formulating the Second Order approximation

In this thesis, we deal with only a basic contact relation between two objects, which is defined below. An arbitrary object is composed of object primitives as follows: vertices, edges (curved lines), and faces (curved surfaces). Basically, every contact relation between two objects can be represented as a set of contact primitives, which are pairs of contacting object primitives. Because three types of object primitives exist, nine types of contact primitives exist as shown in Fig. 3.2. A vertex-face contact means that a vertex of the moving object contacts a face of the fixed object.

To achieve the purpose of this chapter, we must formulate the second order approximation of a function which represents the relationship between the displacement and the distance between two contacting object primitives. First, we consider the formulation with respect to point-contact primitives which realize a point-contact. These nine contact primitives realize point-contacts. In this chapter, we formulate the approximation for the following six point-contact primitives: vertex-face, edge-edge, edge-face, face-vertex, face-edge, and face-face point-contacts. We refer to these six point-contact primitives as *basic point-contact primitives.* We define a basic contact relation as a contact relation which is represented as a set of only basic point-contact primitives. Next we formulate the approximation for a line-contact

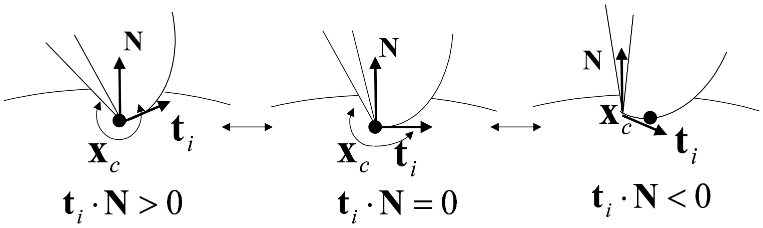

N
$\mathbf{x}_c$
$\mathbf{t}_i$
$\mathbf{t}_i \cdot \mathbf{N} > 0$
N
$\mathbf{x}_c$
$\mathbf{t}_i$
$\mathbf{t}_i \cdot \mathbf{N} = 0$
N
$\mathbf{x}_c$
$\mathbf{t}_i$
$\mathbf{t}_i \cdot \mathbf{N} < 0$

Figure 3.3: A vertex-face point-contact

primitive and a face-contact primitive.

Through the formulation, we found that we had to formulate the approximation not only for point-contact primitives except for basic point-contact primitives, but also for several singular point-contact primitives which include at least one singular object primitive as follows: a half_edge, a half_face, and a quarter_face. The singular object primitives are defined in Appendix B. Fortunately, the second order approximation in any contact primitive is basically formulated by a union of some products of the approximations for basic contact primitives. We describe the formulation in Appendix B.

3.2.1 Vertex-Face Point-Contact

Consider the case that a vertex v contacts a face F at a point $\mathbf{x}_c$ as shown in Fig. 3.3. Because these two do not penetrate each other before the displacement in normal use of the approximation, Equation (3.26) must be satisfied.

$$\bigcap_i \mathbf{t}_i \cdot \mathbf{N} \geq 0 \tag{3.26}$$

Now we assume that $\bigcap_i \mathbf{t}_i \cdot \mathbf{N} > 0$. After the displacement, a point $\mathbf{x}_c$ is represented by $\mathbf{d}(\mathbf{x}_c)$. The distance Δ_{vF} between the vertex and the face after the displacement is calculated by Equation (3.27).

$$\begin{aligned} \Delta_{vF} &= F(\mathbf{d}(\mathbf{x}_c)) \\ &= \mathbf{N} \cdot (\mathbf{d}(\mathbf{x}_c) - \mathbf{x}_c) + \frac{1}{2}(\mathbf{d}(\mathbf{x}_c) - \mathbf{x}_c)^T M (\mathbf{d}(\mathbf{x}_c) - \mathbf{x}_c) \end{aligned} \tag{3.27}$$

The first and second derivatives of Equation (3.27) are represented by Equation (3.28) and (3.29), respectively.

$$\Delta'_{vF} = \mathbf{N} \cdot \mathbf{d}'(\mathbf{x}_c) + \mathbf{d}'(\mathbf{x}_c)^T M (\mathbf{d}(\mathbf{x}_c) - \mathbf{x}_c) \tag{3.28}$$

$$\Delta''_{vF} = \mathbf{N} \cdot \mathbf{d}''(\mathbf{x}_c) + \mathbf{d}'(\mathbf{x}_c)^T M \mathbf{d}'(\mathbf{x}_c) + \mathbf{d}''(\mathbf{x}_c)^T M (\mathbf{d}(\mathbf{x}_c) - \mathbf{x}_c) \tag{3.29}$$

Applying the Taylor expansion to Equation (3.27) about $\Delta\theta = 0$, Equation (3.30) is obtained, where $R_3(\Delta\theta)$ is the third order remainder term.

$$\begin{aligned}\Delta_{vF} &= \mathbf{N}\cdot(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2)\Delta\theta \\ &+ \Big(\mathbf{N}\cdot(\mathbf{s}_3\times\mathbf{x}_c+\mathbf{s}_4)+(\mathbf{N}\times\mathbf{s}_1)\cdot(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2) \\ &\quad+(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2)^T M(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2)\Big)\frac{\Delta\theta^2}{2} \\ &+ R_3(\Delta\theta) \end{aligned} \tag{3.30}$$

Now we consider the legal displacement, i.e., the two objects do not locally penetrate each other about a point $\mathbf{x}_c$. The condition that the vertex is not inside the face is formulated by $\Delta_{vf} \geq 0$. If two objects penetrate each other under the condition, only a part of some adjacent half_edge e_i or quarter_face f_i of a vertex v must be inside the fixed object. As the result, Equation (3.31) must be satisfied (See Fig. 3.3).

$$(R\mathbf{t}_i)\cdot\mathbf{N} < 0 \tag{3.31}$$

Because $(R\mathbf{t}_i)\cdot\mathbf{N}$ is continuous with respect to $\Delta\theta$ and the equation is greater than zero from the assumption when $\Delta\theta = 0$. Therefore $(R\mathbf{t}_i)\cdot\mathbf{N} > 0$ for all i, when $\Delta\theta$ is sufficiently small. Therefore, the displacement which satisfy $\Delta_{vF} \geq 0$ is obviously legal.

If $\mathbf{t}_i\cdot\mathbf{N} = 0$ for some i, we should consider the singular point-contact primitive which consists of a half_edge e_i and a face F. Furthermore, if $\mathbf{t}_{i-1}\cdot\mathbf{N} = 0 \cap \mathbf{t}_i\cdot\mathbf{N} = 0$ for some i, we should consider the singular point-contact primitive which consists of a quarter_face f_i and a face F, where two half_edges e_{i-1} and e_i is adjacent to a quarter_face f_i. As mentioned above, we describe the formulations with respect to the singular contact primitives in Appendix B.

3.2.2 Edge-Face Point-Contact

Consider the case that an edge e contacts a face F at a point $\mathbf{x}_c$ as shown in Fig. 3.4. Because these two do not penetrate each other before the displacement in normal use, Equation (3.32) must be satisfied.

$$\mathbf{t}\cdot\mathbf{N} = 0 \tag{3.32}$$

In this case, we concentrate on the case $\bigcap_i \mathbf{n}_i\times\mathbf{N} \neq \mathbf{0}$. If $\mathbf{n}_i\times\mathbf{N} = \mathbf{0}$ for some i, we should consider the singular point-contact primitive which consists of a half_face f_i and a face F (See Fig. 3.4).

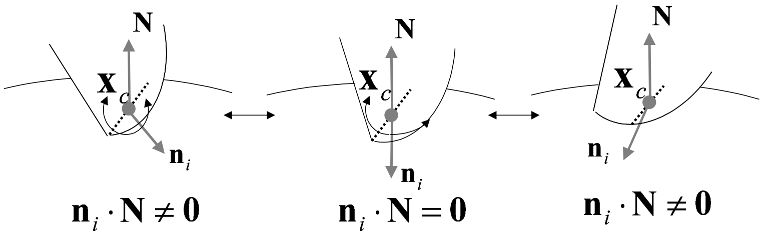

N
$\mathbf{x}_c$
$\mathbf{n}_i$
$\mathbf{n}_i \cdot \mathbf{N} \neq \mathbf{0}$
N
$\mathbf{x}_c$
$\mathbf{n}_i$
$\mathbf{n}_i \cdot \mathbf{N} = \mathbf{0}$
N
$\mathbf{x}_c$
$\mathbf{n}_i$
$\mathbf{n}_i \cdot \mathbf{N} \neq \mathbf{0}$

Figure 3.4: An edge-face point-contact

After the displacement, the edge is represented by $\mathbf{x} = \mathbf{e}_d(l)$. The distance Δ_{ef} between a point on the edge $\mathbf{x} = \mathbf{e}_d(l)$ and the face is represented by Equation (3.33).

$$\begin{aligned}\Delta_{eF} &= F(\mathbf{e}_d(l)) \\ &= \mathbf{N} \cdot (\mathbf{e}_d(l) - \mathbf{x}_c) + \frac{1}{2}(\mathbf{e}_d(l) - \mathbf{x}_c)^T M(\mathbf{e}_d(l) - \mathbf{x}_c) \end{aligned} \tag{3.33}$$

The distance between the edge and the face is naturally defined as the minimum value of Δ_{eF}. When Δ_{eF} is minimum, Equation (3.34) = 0 must be satisfied.

$$\frac{\partial \Delta_{eF}}{\partial l} = \mathbf{N} \cdot \frac{\partial \mathbf{e}_d(l)}{\partial l} + \frac{\partial \mathbf{e}_d(l)}{\partial l}^T M(\mathbf{e}_d(l) - \mathbf{x}_c) \tag{3.34}$$

The derivative of Equation (3.34) with respect to $\Delta\theta$ is represented by Equation (3.35).

$$\begin{aligned}\left(\frac{\partial \Delta_{eF}}{\partial l}\right)' &= \mathbf{N} \cdot \frac{\partial \mathbf{e}'_d(l)}{\partial l} + \mathbf{N} \cdot \frac{\partial^2 \mathbf{e}_d(l)}{\partial l^2} l' + \frac{\partial \mathbf{e}_d(l)}{\partial l}^T M \left(\mathbf{e}'_d(l) + \frac{\partial \mathbf{e}_d(l)}{\partial l} l'\right) \\ &+ (\cdots) M(\mathbf{e}_d(l) - \mathbf{x}_c) \end{aligned} \tag{3.35}$$

By substituting $\Delta\theta = 0$ to Equation (3.35), Equation (3.36) is obtained. Note that $\mathbf{e}_d(l)|_{\Delta\theta=0} = \mathbf{x}_c$, because $l(0) = 0$.

$$\begin{aligned}\left.\left(\frac{\partial \Delta_{eF}}{\partial l}\right)'\right|_{\Delta\theta=0} &= \mathbf{N} \cdot (\mathbf{s}_1 \times \mathbf{t}) + k\mathbf{N} \cdot \mathbf{p} l'(0) \\ &+ \mathbf{t}^T M(\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) + \mathbf{t}^T M \mathbf{t} l'(0) \end{aligned} \tag{3.36}$$

First, we consider the case that $k\mathbf{N} \cdot \mathbf{p} + \mathbf{t}^T M \mathbf{t} \neq 0$. Next we consider the case that $k\mathbf{N} \cdot \mathbf{p} + \mathbf{t}^T M \mathbf{t} = 0$.

Because $k\mathbf{N} \cdot \mathbf{p} + \mathbf{t}^T M \mathbf{t} \neq 0$, Equation (3.33) is minimum, when $l'(0)$ is equal to l_{eF} in Equation (3.37).

$$l_{eF} = -\frac{\mathbf{N} \cdot (\mathbf{s}_1 \times \mathbf{t}) + \mathbf{t}^T M(\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2)}{k\mathbf{N} \cdot \mathbf{p} + \mathbf{t}^T M \mathbf{t}} \tag{3.37}$$

The derivative of Equation (3.33) with respect to $\Delta\theta$ is represented by Equation (3.38).

$$\begin{aligned}
\Delta'_{eF} &= \mathbf{N} \cdot \mathbf{e}'_d(l) + \mathbf{e}'_d(l)^T M(\mathbf{e}_d(l) - \mathbf{x}_c) + \frac{\partial \Delta_{eF}}{\partial l} l' \\
&= \mathbf{N} \cdot \mathbf{e}'_d(l) + \mathbf{e}'_d(l)^T M(\mathbf{e}_d(l) - \mathbf{x}_c) \qquad (3.38)
\end{aligned}$$

The second derivative of Equation (3.33) with respect to $\Delta\theta$ is represented by Equation (3.39).

$$\begin{aligned}
\Delta''_{eF} &= \mathbf{N} \cdot \mathbf{e}''_d(l) + \mathbf{N} \cdot \frac{\partial \mathbf{e}'_d(l)}{\partial l} l' + \mathbf{e}'_d(l)^T M \mathbf{e}'_d(l) + \mathbf{e}'_d(l)^T M {\frac{\partial \mathbf{e}'_d(l)}{\partial l}}^T l' \\
&+ \; (\cdots) M(\mathbf{e}_d(l) - \mathbf{x}_c) \qquad (3.39)
\end{aligned}$$

Applying the Taylor expansion to Equation (3.33) about $\Delta\theta = 0$, we obtain Equation (3.40).

$$\begin{aligned}
\Delta_{eF} &= \mathbf{N} \cdot (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{t}_2)\Delta\theta \\
&+ \; \Big(\mathbf{N} \cdot (\mathbf{s}_3 \times \mathbf{x}_c + \mathbf{s}_2) + (\mathbf{N} \times \mathbf{s}_1) \cdot (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) \\
&\quad + (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2)^T M (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) - l'(0)^2 (k\mathbf{N} \cdot \mathbf{p} + \mathbf{t}^T M \mathbf{t})\Big) \frac{\Delta\theta^2}{2} \\
&+ \; R_3(\Delta\theta) \qquad (3.40)
\end{aligned}$$

Consider the case that $k\mathbf{N} \cdot \mathbf{p} + \mathbf{t}^T M \mathbf{t} = 0$. Now, we formulate only the legal displacement. Of course, the displacement must satisfy $\Delta_{eF} \geq 0$. Note that the term which includes $l'(0)$ is eliminated in the equation. Furthermore, because Equation (3.36) $\neq 0$ means that the edge penetrates the face, the legal displacement must additionally satisfy Equation (3.41).

$$\mathbf{N} \cdot (\mathbf{s}_1 \times \mathbf{t}) + \mathbf{t}^T M (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) = 0 \qquad (3.41)$$

However, this case seldom appears. We do not regard the edge-face point-contact as a basic point-contact primitive.

3.2.3 Face-Face Point-Contact

Consider the case that a face f contacts a face F at a point $\mathbf{x}_c$. Now let the surface normal of a face f be $-\mathbf{N}$, because it is easy to understand the difference of the formulations between in a face-face point-contact and in another point-contact primitives. Note that $-\mathbf{n} = \mathbf{N}$ is always satisfied because two objects do not penetrate each other before the displacement in normal use.

We consider the sum of two distances Δ_{fF} between some point x and both of two faces as Equation (3.42) after the displacement.

$$\begin{aligned}\Delta_{fF} &= f_d(\mathbf{x}) + F(\mathbf{x})\\ &= -(R\mathbf{N})\cdot(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) + \frac{1}{2}(\mathbf{x}-\mathbf{d}(\mathbf{x}_c))^T RmR^T(\mathbf{x}-\mathbf{d}(\mathbf{x}_c))\\ &+ \mathbf{N}\cdot(\mathbf{x}-\mathbf{x}_c) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_c)^T M(\mathbf{x}-\mathbf{x}_c) \end{aligned} \tag{3.42}$$

Because any line of which the direction is $\mathbf{N}$ contacts each face at only one point when $\Delta\theta$ is sufficiently small, the distance between the two faces is defined as the minimum value of Δ_{fF}. When Δ_{fF} is minimum, Equation (3.43) = 0 must be satisfied.

$$\frac{\partial\Delta_{fF}}{\partial\mathbf{x}} = -(R\mathbf{N}) + \mathbf{N} + RmR^T(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) + M(\mathbf{x}-\mathbf{x}_c) \tag{3.43}$$

The derivative of Equation (3.43) with respect to $\Delta\theta$ is represented by Equation (3.44).

$$\begin{aligned}\left(\frac{\partial\Delta_{fF}}{\partial\mathbf{x}}\right)' &= -R'\mathbf{N} + RmR^T(\mathbf{x}'-\mathbf{d}'(\mathbf{x}_c)) + M\mathbf{x}'\\ &+ (\cdots)(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) + (\cdots)(\mathbf{x}-\mathbf{x}_c)\end{aligned} \tag{3.44}$$

By substituting $\Delta\theta = 0$ to Equation (3.44), Equation (3.45) is obtained. Note that $\mathbf{x}(0) = \mathbf{x}_c$.

$$\left.\left(\frac{\partial\Delta_{fF}}{\partial\mathbf{x}}\right)'\right|_{\Delta\theta=0} = \mathbf{N}\times\mathbf{s}_1 + m(\mathbf{x}'(0) - (\mathbf{s}_1\times\mathbf{x}_c + \mathbf{s}_2)) + M\mathbf{x}'(0) \tag{3.45}$$

Equation (3.42) is minimum, when $\mathbf{x}'(0)$ is equal to $\mathbf{x}_{min}$ as Equation (3.46).

$$\mathbf{x}_{min} = (m+M)^{-1}(m(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2) - \mathbf{N}\times\mathbf{s}_1) \tag{3.46}$$

Note that $m + M$ is not a full-rank matrix, because $(m + M)\mathbf{N} = \mathbf{0}$ is always satisfied. Therefore, we calculate a quasi-inverse matrix $(m+M)^{-1}$ using the idea of the singular value decomposition.

The derivative of Equation (3.42) with respect to $\Delta\theta$ is represented by Equation (3.47).

$$\begin{aligned}\Delta'_{fF} &= -(R'\mathbf{N})\cdot(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) - (R\mathbf{N})\cdot(\mathbf{x}'-\mathbf{d}'(\mathbf{x}_c))\\ &+ (\mathbf{x}'-\mathbf{d}'(\mathbf{x}_c))^T RmR^T(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) + (\mathbf{x}-\mathbf{d}(\mathbf{x}_c))^T R'mR^T(\mathbf{x}-\mathbf{d}(\mathbf{x}_c))\\ &+ \mathbf{N}\cdot\mathbf{x}' + \mathbf{x}'^T M(\mathbf{x}-\mathbf{x}_c)\\ &= -(R'\mathbf{N})\cdot(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) + (R\mathbf{N})\cdot\mathbf{d}'(\mathbf{x}_c)\\ &- \mathbf{d}'(\mathbf{x}_c)^T RmR^T(\mathbf{x}-\mathbf{d}(\mathbf{x}_c)) + (\mathbf{x}-\mathbf{d}(\mathbf{x}_c))^T R'mR^T(\mathbf{x}-\mathbf{d}(\mathbf{x}_c))\end{aligned} \tag{3.47}$$

The second derivative of Equation (3.42) with respect to $\Delta\theta$ is represented by Equation (3.48).

$$\begin{aligned}\Delta''_{fF} &= -(R'\mathbf{N})\cdot\mathbf{x}' + 2(R'\mathbf{N})\cdot\mathbf{d}'(\mathbf{x}_c) + (R\mathbf{N})\cdot\mathbf{d}''(\mathbf{x}_c)\\ &- \mathbf{d}'(\mathbf{x}_c)^T RmR^T(\mathbf{x}' - \mathbf{d}'(\mathbf{x}_c)) + (\cdots)(\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) \qquad (3.48)\end{aligned}$$

Therefore, by applying the Taylor expansion to Equation (3.42) about $\Delta\theta = 0$, Equation (3.49) is obtained.

$$\begin{aligned}\Delta_{fF} &= \mathbf{N}\cdot(\mathbf{s}_1\times\mathbf{x}_c + \mathbf{s}_2)\Delta\theta\\ &+ \big(\mathbf{N}\cdot(\mathbf{s}_3\times\mathbf{x}_c + \mathbf{s}_4) - \mathbf{x}'(0)^T(m+M)\mathbf{x}'(0)\\ &\quad + \mathbf{x}'(0)^T(m+M)(\mathbf{s}_1\times\mathbf{x}_c + \mathbf{s}_2)\big)\frac{\Delta\theta^2}{2}\\ &+ R_3(\Delta\theta) \qquad (3.49)\end{aligned}$$

Note that the legal displacement is formulated by $\Delta_{fF}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} \geq 0$ and the formulation is essentially equal to the formulation in [67], although the representation of the displacement is different.

3.2.4 Edge-Edge Point-Contact

Consider the case that an edge e contacts an edge E at a point $\mathbf{x}_c$. In this thesis, we define the distance between two edges as Equation (3.50), where $\mathbf{x}$ and $\mathbf{X}$ are points on edges of the moving and the fixed objects, respectively, $\mathbf{t}$ and $\mathbf{T}$ are tangent vectors on points $\mathbf{x}$ and $\mathbf{X}$, respectively, and the direction of the vector $\mathbf{t}\times\mathbf{T}$ ($\neq \mathbf{0}$) is set up in order to be outward to the fixed object. Note that $-\mathbf{t}\times\mathbf{T}$ must be inward to the fixed object in this case.

$$\min_{\mathbf{x},\mathbf{X}}(\mathbf{t}\times\mathbf{T})\cdot(\mathbf{x}-\mathbf{X}) \qquad (3.50)$$

When the equation is equal to zero, two edges contact each other. And when the equation is less than zero, each edge is inside the opposite object. Otherwise, they detach from each other.

First, we consider the case that $\mathbf{t}\times\mathbf{T}\neq 0$. Note that when the direction of a vector $\mathbf{t}\times\mathbf{T}$ is not outward to the fixed object, $-\mathbf{t}$ is substituted for $\mathbf{t}$, because the direction of a vector $-\mathbf{t}\times\mathbf{T}$ is always outward to the fixed object in this case. Because each edge does not penetrate the opposite object before the displacement in normal use, Equation (3.51) must be satisfied.

$$\left(\bigcap_i \mathbf{t}\cdot\mathbf{N}_i \geq 0\right) \cap \left(\bigcap_i \mathbf{n}_i\cdot\mathbf{T} \geq 0\right) \qquad (3.51)$$

Now we concentrate on the case where Equation (3.52) is satisfied.

$$\left(\bigcap_i \mathbf{t}\cdot\mathbf{N}_i > 0\right) \cap \left(\bigcap_i \mathbf{n}_i\cdot\mathbf{T} > 0\right) \tag{3.52}$$

If Equation (3.53) is satisfied for some i, we should consider the singular point-contact primitive which consists of an edge e and a half_face F_i.

$$\mathbf{t}\cdot\mathbf{N}_i = 0 \tag{3.53}$$

As the same way, if Equation (3.54) is satisfied for some i, we should consider the singular point-contact primitive which consists of a half_face f_i and an edge E.

$$\mathbf{n}_i\cdot\mathbf{T} = 0 \tag{3.54}$$

Furthermore, if Equation (3.55) is satisfied for some i and j, we should consider the singular point-contact primitive which consists of a half_face f_i and a half_face F_j.

$$\mathbf{n}_i\times\mathbf{N}_j = \mathbf{0} \tag{3.55}$$

After the displacement, the distance between the two edges is equal to the minimum value of Δ_{eE} as Equation (3.56).

$$\Delta_{eE} = \left(\frac{\partial\mathbf{e}_d(l)}{\partial l}\times\frac{\partial\mathbf{E}(L)}{\partial L}\right)\cdot(\mathbf{e}_d(l)-\mathbf{E}(L)) \tag{3.56}$$

The derivatives of Equation (3.56) with respect to l and L are represented by Equation (3.57) and (3.58), respectively. Δ_{eE} is minimum with respect to l and L, when these two equations are equal to zero.

$$\begin{aligned}\frac{\partial\Delta_{eE}}{\partial l} &= \left(\frac{\partial^2\mathbf{e}_d(l)}{\partial l^2}\times\frac{\partial\mathbf{E}(L)}{\partial L}\right)\cdot(\mathbf{e}_d(l)-\mathbf{E}(L)) &(3.57)\\ \frac{\partial\Delta_{eE}}{\partial L} &= \left(\frac{\partial\mathbf{e}_d(l)}{\partial l}\times\frac{\partial^2\mathbf{E}(L)}{\partial L^2}\right)\cdot(\mathbf{e}_d(l)-\mathbf{E}(L)) &(3.58)\end{aligned}$$

The derivatives of Equation (3.57) and (3.58) with respect to $\Delta\theta$ are represented by Equation (3.59) and (3.60), respectively.

$$\begin{aligned}\left(\frac{\partial\Delta_{eE}}{\partial l}\right)'\Bigg|_{\Delta\theta=0} &= \left(\frac{\partial^2\mathbf{e}_d(l)}{\partial l^2}\times\frac{\partial\mathbf{E}(L)}{\partial L}\right)\cdot\left(\mathbf{e}'_d(l)+\frac{\partial\mathbf{e}_d(l)}{\partial l}l'\right)\\ &+ (\cdots)(\mathbf{e}_d(l)-\mathbf{E}(L)) &(3.59)\\ \left(\frac{\partial\Delta_{eE}}{\partial L}\right)'\Bigg|_{\Delta\theta=0} &= \left(\frac{\partial\mathbf{e}_d(l)}{\partial l}\times\frac{\partial^2\mathbf{E}(L)}{\partial L^2}\right)\cdot\left(\mathbf{e}'_d(l)+\frac{\partial\mathbf{E}(L)}{\partial L}L'\right)\\ &+ (\cdots)(\mathbf{e}_d(l)-\mathbf{E}(L)) &(3.60)\end{aligned}$$

By substituting $\Delta\theta = 0$ to Equation (3.59) and (3.60), Equation (3.61) and (3.62) are obtained, respectively.

$$\left.\left(\frac{\partial \Delta_{eE}}{\partial l}\right)'\right|_{\Delta\theta=0} = (\mathbf{p} \times \mathbf{T}) \cdot (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2 + \mathbf{t}l'(0)) \tag{3.61}$$

$$\left.\left(\frac{\partial \Delta_{eE}}{\partial L}\right)'\right|_{\Delta\theta=0} = (\mathbf{t} \times \mathbf{P}) \cdot (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2 + \mathbf{T}L'(0)) \tag{3.62}$$

First, we consider the case that $\mathbf{N} \cdot \mathbf{p} \neq 0$ and $\mathbf{N} \cdot \mathbf{P} \neq 0$, where $\mathbf{N} = \mathbf{t} \times \mathbf{T}$. In this case, Equation (3.56) is minimum, when $l'(0)$ is equal to l_{eE} in Equation (3.63) and $L'(0)$ is equal to L_{eE} in Equation (3.64).

$$l_{eE} = \frac{(\mathbf{p} \times \mathbf{T}) \cdot (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2)}{\mathbf{N} \cdot \mathbf{p}} \tag{3.63}$$

$$L_{eE} = \frac{(\mathbf{t} \times \mathbf{P}) \cdot (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2)}{\mathbf{N} \cdot \mathbf{P}} \tag{3.64}$$

The derivative of Equation (3.56) with respect to $\Delta\theta$ is represented by Equation (3.65)

$$\begin{aligned}\Delta'_{eE} &= \left(\frac{\partial \mathbf{e}'_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot (\mathbf{e}_d(l) - \mathbf{E}(L)) \\ &\quad + \left(\frac{\partial \mathbf{e}_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \mathbf{e}'_d(l) + \frac{\partial \Delta_{eE}}{\partial l} l' + \frac{\partial \Delta_{eE}}{\partial L} L' \\ &= \left(\frac{\partial \mathbf{e}'_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot (\mathbf{e}_d(l) - \mathbf{E}(L)) + \left(\frac{\partial \mathbf{e}_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \mathbf{e}'_d(l)\end{aligned} \tag{3.65}$$

The second derivative of Equation (3.56) with respect to $\Delta\theta$ is represented by Equation (3.66).

$$\begin{aligned}\Delta''_{eE} &= \left(\frac{\partial \mathbf{e}'_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \left(\mathbf{e}'_d(l) + \frac{\partial \mathbf{e}_d(l)}{\partial l} l'\right) \\ &+ \left(\left(\frac{\partial \mathbf{e}'_d(l)}{\partial l} + \frac{\partial^2 \mathbf{e}_d(l)}{\partial l^2} l'\right) \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \mathbf{e}'_d(l) \\ &+ \left(\frac{\partial \mathbf{e}_d(l)}{\partial l} \times \frac{\partial^2 \mathbf{E}(L)}{\partial L^2} L'\right) \cdot \mathbf{e}'_d(l) \\ &+ \left(\frac{\partial \mathbf{e}_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \left(\mathbf{e}''_d(l) + \frac{\partial \mathbf{e}'_d(l)}{\partial l} l'\right) \\ &+ (\cdots)(\mathbf{e}_d(l) - \mathbf{E}(L)) \\ &= 2\left(\frac{\partial \mathbf{e}'_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \mathbf{e}'_d(l) + \left(\frac{\partial \mathbf{e}_d(l)}{\partial l} \times \frac{\partial^2 \mathbf{E}(L)}{\partial L^2}\right) \cdot \mathbf{e}'_d(l) L' \\ &+ \left(\frac{\partial^2 \mathbf{e}_d(l)}{\partial l^2} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \mathbf{e}'_d(l) l' + \left(\frac{\partial \mathbf{e}_d(l)}{\partial l} \times \frac{\partial \mathbf{E}(L)}{\partial L}\right) \cdot \mathbf{e}''_d(l) \\ &+ (\cdots)(\mathbf{e}_d(l) - \mathbf{E}(L))\end{aligned} \tag{3.66}$$

By applying the Taylor expansion to Equation (3.56), Equation (3.67) is obtained.

$$\begin{aligned}\Delta_{eE} &= \mathbf{N}\cdot(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{t}_2)\Delta\theta \\ &+ (\mathbf{N}\cdot(\mathbf{s}_3\times\mathbf{x}_c+\mathbf{s}_4) \\ &\quad +((\mathbf{s}_1\times\mathbf{t})\times\mathbf{T})\cdot(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2)+((\mathbf{s}_1\times\mathbf{T})\times\mathbf{t})\cdot(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2) \\ &\quad -\mathbf{N}\cdot(kl'(0)^2\mathbf{p}-KL'(0)^2\mathbf{P}))\,\frac{\Delta\theta^2}{2} \\ &+ R_3(\Delta\theta) \qquad (3.67)\end{aligned}$$

Next, we consider the cases that $\mathbf{N}\cdot\mathbf{p}=0$ and/or $\mathbf{N}\cdot\mathbf{P}=0$. Now, we formulate only the legal displacement. Let $\mathbf{N}\cdot\mathbf{p}$ be equal to zero. In this case, of course, $\Delta_{eE}|_{L'(0)=L_{eE}}\geq 0$ must be satisfied. Note that the term which includes $l'(0)$ is eliminated in the equation. Additionally, Equation (3.68) must be satisfied, because Equation (3.61) must be equal to 0.

$$(\mathbf{p}\times\mathbf{T})\cdot(\mathbf{s}_1\times\mathbf{x}_c+\mathbf{s}_2)=0 \qquad (3.68)$$

In the case that $\mathbf{N}\cdot\mathbf{P}=0$, the legal displacement can be formulated as the same way. However, the cases seldom appear. We do not regard the edge-edge point-contacts as basic point-contact primitives.

Then, we consider the case that $\mathbf{t}\times\mathbf{T}=\mathbf{0}$. In this case, we cannot define the distance between the two edges. Fortunately, because $\mathbf{n}_i\cdot\mathbf{t}=0$ and $\mathbf{N}_i\cdot\mathbf{T}=0$ are satisfied for all i, $\mathbf{n}_i\cdot\mathbf{T}=0$ and $\mathbf{N}_i\cdot\mathbf{t}=0$ are satisfied for all i. That is, we can substitute the approximations in point-contacts between one edge and an adjacent face of the other edge for the edge-edge point-contact.

3.2.5 Face-Vertex and Face-Edge Point-Contacts

Consider the case that a face f contacts a vertex V. Viewing the displacement from the vertex, the face inversely moves. From the definition of the screw vector, the inverse displacement can be represented as $[-\mathbf{s}_1,-\mathbf{s}_2,-\mathbf{s}_3,-\mathbf{s}_4]$. Therefore, the approximation is obtained by substituting $[-\mathbf{s}_1,-\mathbf{s}_2,-\mathbf{s}_3,-\mathbf{s}_4]$ to $[\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3,\mathbf{s}_4]$ of Equation (3.30). As the same way, we obtain the approximation for a face-edge point-contact.

3.2.6 Line-Contact Primitives and Face-Contact Primitives

Now we consider an edge-face line-contact. This contact can be regarded as an infinite number of vertex-face contacts. Therefore, for example, the legal displacement is formulated by Equation (3.69), where the edge is represented as an infinite number of vertices v_i.

$$\bigcap_{i}^{\infty} \Delta_{v_i F} \geq 0 \tag{3.69}$$

It is difficult to solve the infinite number of equations simultaneously. Therefore, it is preferable to obtain the equivalent a finite number of equations by removing redundant equations. Unfortunately, such a finite number of equations are not always obtained. One solution is to approximately substitute a finite number of equations which are selected among the infinite number of equations. In another line-contacts or face-contacts, that is the same.

3.3 Calculating Maintaining and Legal Displacement using the Second Order Approximation

In this section, we consider how to calculate maintaining and legal displacement using the second order approximation. In the result of the formulation, all the second order approximations for basic point-contact primitives can be represented as Equation (3.70). To calculate such displacement, we must examine the characteristic of the equation, where $h_i(\mathbf{s}_1, \mathbf{s}_2)$ is a quadratic form of terms $\mathbf{s}_1$ and $\mathbf{s}_2$.

$$g_i = \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \Delta\theta + \left(\begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) \right) \frac{\Delta\theta^2}{2} \tag{3.70}$$

A coefficient of $\Delta\theta$ of the equation is positive for some screw vector; the equation is positive when $\Delta\theta$ is sufficiently small. That means that the corresponding displacement detaches the corresponding contact-primitive. As the same way, the coefficient of $\Delta\theta$ is negative, the equation is negative, i.e., the corresponding displacement is illegal.

If the coefficient of $\Delta\theta$ is zero, the coefficient of $\Delta\theta^2$ is important to decide the characteristic. If the coefficient is positive, the corresponding displacement detaches the corresponding contact-primitive. In the same way, if the coefficient is negative, the corresponding displacement is illegal. If the coefficient is zero, the equation is zero with respect to the second order approximation. That means that the corresponding displacement maintains the corresponding contact-primitives. Table 3.1 summarizes the contents mentioned above. Note that, in the first order approximation, the characteristics are determined by a coefficient of $\Delta\theta$ only.

Now we describe two types of important calculations as follows:

Table 3.1: Second order displacement

coefficient of $\Delta\theta$	coefficient of $\Delta\theta^2$	contact-primitive
> 0	—	detach
$= 0$	> 0	
	$= 0$	maintain
	< 0	penetrate (illegal)
< 0	—	

- Calculate maintaining displacement
- Calculate legal displacement

The main purpose of the calculation is to examine the range of the first order screw vector in maintaining or legal displacement. The first order screw vector is dominant to the second order screw vector, when $\Delta\theta$ is sufficiently small.

3.3.1 Calculating Maintaining Displacement

In this section, we describe how to calculate maintaining displacement using the second order approximation. From the result of characterization of the approximation, maintaining displacement must satisfy Equation (3.71).

$$\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} = 0 \cap \bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) = 0 \quad (3.71)$$

We calculate the displacement by performing the following steps:

1. Solve Equation (3.72).

$$\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} = 0 \quad (3.72)$$

2. Solve Equation (3.73) under satisfying Equation (3.72).

$$\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) = 0 \quad (3.73)$$

Note that maintaining displacement is calculated by solving only Equation (3.72) in the first order approximation.

The solution of Equation (3.72) is represented as a linear combination of some bases as shown in Equation (3.74), where a_i is any real number.

$$[\mathbf{s}_1, \mathbf{s}_2] = \sum_{i=1}^{r} a_i \mathbf{b_i} \tag{3.74}$$

By substituting (3.74) to (3.73), Equation (3.75) is obtained, where $h_i(a_1, \ldots, a_r)$ is a quadratic form.

$$\bigcap_{i}^{n} \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(a_1, \ldots, a_r) = 0 \tag{3.75}$$

By searching all linearly dependent minimum combinations of $\{[(\mathbf{x}_{ci} \times \mathbf{N}_i)^T, \mathbf{N}_i^T]\}$, equations including terms h_i only are obtained. Concretely, let C be one of the linearly dependent minimum combinations. From the dependency, Equation (3.76) is always satisfied, where $w_i \neq 0$ for all $i \in C$.

$$\sum_{i \in C} w_i \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} = \mathbf{0} \tag{3.76}$$

By applying a dot product of $[\mathbf{s}_3^T, \mathbf{s}_4^T]^T$ to Equation (3.76), Equation (3.77) is obtained.

$$\sum_{i \in C} w_i \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} = 0 \tag{3.77}$$

By substituting Equation (3.75) to Equation (3.77), Equation (3.78) which includes terms h_i only is obtained.

$$\sum_{i \in C} w_i h_i(a_1, \ldots a_r) = 0 \tag{3.78}$$

The equation is also a quadratic form.

Next we decompose these quadratic forms into some linear equations using the following two rules:

- If one of the forms is non-negative or non-positive definite, i.e., its canonical form is $\pm(A_1^2 + \cdots + A_m^2)$, it can be decomposed as Equation (3.79).

$$\bigcap_{i}^{m} A_i = 0 \tag{3.79}$$

- Otherwise, if the rank of one of the forms is two, i.e., its canonical form is $A_1^2 - A_2^2$, it can be decomposed as Equation (3.80).

$$A_1 + A_2 = 0 \cup A_1 - A_2 = 0 \tag{3.80}$$

If some linear equations are obtained, they are substituted into the remaining quadratic forms. The ranks of the quadratic forms decrease after substituting them. By reiterating such linearization, every quadratic form is usually decomposed. As a result, we can decide the range of the first order screw vector. For every first order screw vector in the range, the second order screw vector which satisfies Equation (3.73) always exists.

3.3.2 Calculating Legal Displacement

In this section, we describe how to calculate legal displacement using the second order approximation. From the result of characterization of the approximation, legal displacement must satisfy Equation (3.81).

$$\begin{gathered}\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \geq 0 \cap \\ \left(\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} = 0 \Rightarrow \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) \geq 0 \right)\end{gathered} \tag{3.81}$$

Note that legal displacement is calculated by solving only the upper part of Equation (3.81) in the first order approximation.

To enable easy understanding of the contents of this section, we first consider an essential contact relation. Next we consider any contact relations.

We define an essential contact relation as follows: The set $\{[(\mathbf{x}_{ci} \times \mathbf{N}_i)^T, \mathbf{N}_i^T]\}$ is a minimum set of linearly dependent vectors and always satisfies Equation (3.82), where $w_i > 0$ for all i.

$$\sum_{i=1}^n w_i \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} = \mathbf{0} \tag{3.82}$$

First, we solve the upper part of Equation (3.81), i.e., Equation (3.83).

$$\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \geq 0 \tag{3.83}$$

In this case, Equation (3.83) is equivalent to Equation (3.84), because

$$w_j \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} = -\sum_{i \neq j} w_i \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \leq 0$$

is always satisfied.

$$\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} = 0 \tag{3.84}$$

The solution can be represented by Equation (3.74).
Next we solve the lower part of Equation (3.81). Because Equation (3.84) is always satisfied when Equation (3.83) is satisfied, Equation (3.85) must be satisfied.

$$\bigcap_i^n \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) \geq 0 \tag{3.85}$$

From Equation (3.82), Equation (3.86) is obtained.

$$\sum_{i=1}^n w_i \left(\begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) \right) - \sum_{i=1}^n w_i h_i(\mathbf{s}_1, \mathbf{s}_2) = 0 \tag{3.86}$$

From Equation (3.85), (3.86) and (3.74), we obtain Equation (3.87).

$$\sum_i^n w_i h_i(a_1, \ldots, a_r) \geq 0 \tag{3.87}$$

In the same manner as the calculation of maintaining displacement, Equation (3.87) may be decomposed into some linear equations:

- If it is non-positive definite, i.e., its canonical form is $-A_1^2 - \cdots - A_m^2$, it can be decomposed into Equation (3.88).

$$\bigcap_i^m A_i = 0 \tag{3.88}$$

- Otherwise, if its rank is two, i.e., its canonical form is $A_1^2 - A_2^2$, it can be decomposed into Equation (3.89).

$$(A_1 + A_2 \geq 0 \cap A_1 - A_2 \geq 0) \cup (A_1 + A_2 \leq 0 \cap A_1 - A_2 \leq 0) \tag{3.89}$$

As a result, we may be able to determine the range of the first order screw vector. For every first order screw vector in the range, the second order screw vector which satisfies Equation (3.86) always exists. Unfortunately, there is no guarantee that we will always be able to decompose into some linear equations.

Next, we consider any contact relations. First we solve Equation (3.83). As a result, the solution can be represented by a linear sum of some bases as Equation (3.90), where $a_i \geq 0$ for all i.

$$[\mathbf{s}_1, \mathbf{s}_2] = \sum a_i \mathbf{b_i} \tag{3.90}$$

If the set $\{[(\mathbf{x}_{ci} \times \mathbf{N}_i)^T, \mathbf{N}_i^T]\}$ is linearly independent, the second order screw vector which satisfies Equation (3.81) always exists for the any $[\mathbf{s}_1, \mathbf{s}_2]$ which satisfies Equation (3.90), because $\begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix}$ is set up to any preferable value for all i.

Then we consider the case that more than one linearly dependent minimum combinations of $\{[(\mathbf{x}_{ci} \times \mathbf{N}_i)^T \mathbf{N}_i^T]\}$ exist. In this case, we solve Equation (3.81) by a method similar to the essential contact relation. Let C be one of the combinations. We solve the lower part of Equation (3.81). First we solve Equation (3.91).

$$\bigcap_{i \in C} \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} = 0 \tag{3.91}$$

The solution is represented by Equation (3.92).

$$[\mathbf{s}_1, \mathbf{s}_2] = \sum a_i^C \mathbf{b_i^C} \tag{3.92}$$

We should solve Equation (3.93) for the range of the first order screw vector $[\mathbf{s}_1, \mathbf{s}_2]$ which satisfies Equation (3.90) and Equation (3.92). Note that the range is formulated like Equation (3.90).

$$\bigcap_{i \in C} \begin{pmatrix} \mathbf{x}_{ci} \times \mathbf{N}_i \\ \mathbf{N}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) \geq 0 \tag{3.93}$$

However, it is difficult to solve Equation (3.93) under the range. We approximately solve Equation (3.93) for the range of the first order screw vector $[\mathbf{s}_1, \mathbf{s}_2]$ which satisfies Equation (3.90) and Equation (3.92), where a_i in Equation (3.90) is assumed to be any real number.

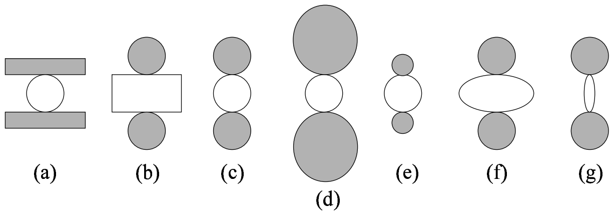


Figure 3.5: Example

From the dependency, we obtain Equation (3.76). If $w_i > 0$ for all i, we obtain Equation (3.94).

$$\sum_{i \in C} w_i h_i(\cdots) \geq 0 \tag{3.94}$$

The equation may be decomposed into several linear equations in the same way as the essential contact relation. If some linear equations are obtained, they are substituted to remaining quadratic forms. Their ranks may decrease after substituting them. Unfortunately, every quadratic form sometimes cannot be decomposed. However, if all quadratic forms can be decomposed, we can determine the range of the first order screw vector. For every first order screw vector in the range, the second order screw vector which satisfies Equation (3.81) always exists.

3.4 Examples

In this section, we describe two applications of the second order approximation. The first one is to determine grasping stability. The second one is to generate a contact relation graph.

3.4.1 Determining Grasping Stability Using the Second Order Approximation

In this example, we consider stability of seven types of grasps as shown in Fig. 3.5. The purpose of this thesis is not to index grasping stability but rather, to calculate the displacement. Therefore, we illustrate only an example to determine the stability using the maintaining and legal displacement. First, we calculate maintaining and legal displacement using the second order approximation in every grasp. Next, using the result, we determine the stability.

In all grasps, we set up the origin of the base coordinate system on the center of the white grasping object. Directions of the x-axis, the y-axis, and the z-axis are rightward, the depth direction, and upward, respectively. Every contact primitive is a face-face point-contact, where two faces have various curvature matrices case by case. The positions of the two point-contacts are $(0, 0, 1)$ and $(0, 0, -1)$, respectively. The direction of every surface normal is $(0, 0, \pm 1)$. Every contact relation is an essential contact relation.

In all grasps, the maintaining and legal displacement calculated using only the first order approximation can be represented by a linear sum of the following first order screw vectors:

$$\{(1, 0, 0, 0, 0, 0), (0, 1, 0, 0, 0, 0), (0, 0, 1, 0, 0, 0), \\ (0, 0, 0, 1, 0, 0), (0, 0, 0, 0, 1, 0)\} \tag{3.95}$$

The grasping stability which the first order approximation determines is the same for each grasp. The result is completely different from our intuition. Next we show the difference between the displacement calculated by the first and the second order approximation.

In Grasp (a), a sphere is grasped by two flat fingertips. The range of the first order screw vector does not change. Any legal displacement maintains the contact relation.

In Grasp (b), a box is grasped by two sphere fingertips, of which the radius is one. The range of the first order screw vector degenerates. Any legal displacement maintains the contact relation.

In Grasp (c), a sphere is grasped by two sphere fingertips. The radius of every sphere is one. In Grasp (d), a sphere is grasped by two sphere fingertips, which are twice as large as the grasped sphere. In Grasp (e), a sphere is grasped by two sphere fingertips, which are twice as small. The range of the first order screw vector does not change, but the displacement corresponding to some range of the first order screw vector translates maintaining displacement to detaching displacement.

In Grasp (f), a wide ellipse is grasped by two sphere fingertips. Unfortunately, maintaining and legal displacement cannot be linearized. The range of the first order screw vector degenerates and the displacement corresponding to some range of the first order screw vector translates maintaining displacement to detaching displacement.

In Grasp (g), a narrow ellipse is grasped by two sphere fingertips. The range of the first order screw vector does not change. Furthermore, the displacement corresponding to some range of the first order screw vector translates maintaining displacement to detaching displacement.

We determine the grasping stability by the following rules:

- The grasping becomes stabler if the range of legal displacement becomes narrower.
- The grasping becomes stabler if the range of detaching displacement becomes narrower.

Following the rules, Grasp (b) is the stablest of all and Grasp (g) is the most unstable of all. Grasps (c), (d), and (e) are equally as stable as each other. However, these three are the second most unstable of all. As a result, we find that the size of the fingertip has no effect on the grasping stability in these grasps. It is somewhat difficult to determine the stability of Grasp (a) and (f), because Grasp (a) is stabler than Grasp (f) when applying the former rule and Grasp (f) is stabler when applying the latter rule. The results are equivalent to our intuition.

3.4.2 Generating the Contact Relation Graph Using the Second Order Approximation

In this section, we describe a method for generating a contact relation graph using the second order approximation. Although the algorithm shown in this section has not been completely established, the basic idea is applicable not only for graph generation but also for path planning.

As mentioned above, methods[34, 35] for automatically generating a contact relation graph have been proposed. Almost all of them employ the non-linear optimization method, which requires the initial guess, i.e., approximate object configurations to realize a desired contact relation. However the method to obtain the appropriate initial guess has not yet been proposed, as far as we know. In this section, we aim to speed up the generation by setting up the appropriate initial guess. In this example, we assume that an object configuration which satisfies a contact relation with more contact primitives is given.

In actuality, we describe a method to obtain the appropriate initial guess with respect to the three contact relations with a fewer contact-primitives as shown at the top right (face-face contact), bottom left (edge-face contact), and bottom right (vertex-face contact) in Fig. 3.6, given the object configuration which satisfies an edge-edge contact as shown to the top left in the figure. We first calculate the

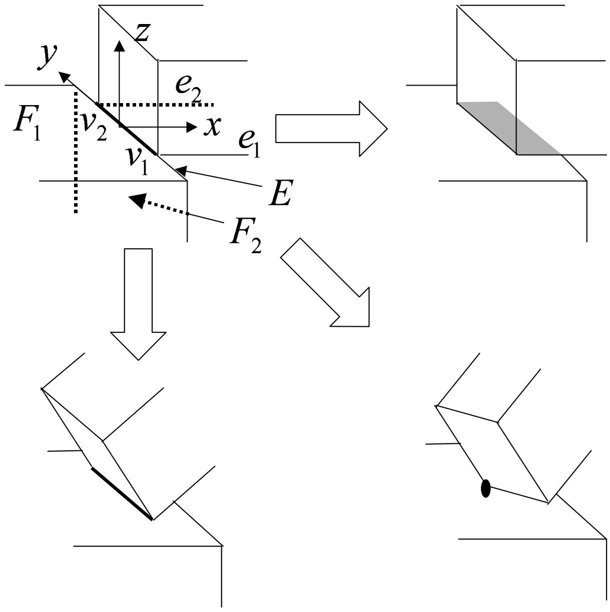


Figure 3.6: Set up the appropriate initial guess using an object configuration which satisfy a contact relation with more contact primitives

displacement from the configuration in the edge-edge contact to the configuration in the desired contact relation. Then we calculate the desired configuration using the displacement.

In this example, we assume that a part of the screw vector $\mathbf{s}_3$ is set to zero, i.e., the axis direction of rotation is constant while rotating. Considering to move a robot arm using the displacement obtained in this example, the assumption is preferable. Furthermore, the corresponding displacement can be analytically calculated. We describe the calculation in Appendix C.

We set up the base coordinate system as follows: The origin of the system is the center of the contacting line. The directions of the x-axis and z-axis are directions of surface normals of a face F_2 and a face F_1, respectively. The direction of the y-axis is set up in order that the system becomes the right-hand coordinate system (See Fig. 3.6).

First we consider the face-face contact. In this case, the displacement satisfies Equation (3.96), because the distance between two faces is equal to zero.

$$\Delta_{v_1 F_1} = 0 \cap \Delta_{v_2 F_1} = 0 \cap \Delta_{e_1 E} = 0 \cap \Delta_{e_2 E} = 0 \tag{3.96}$$

And it satisfies Equation (3.97), because the edge-edge contact needs to be detached.

$$\Delta_{v_1F_2} < 0 \cap \Delta_{v_2F_2} < 0 \tag{3.97}$$

As one of the solution for these equations using only the coefficients of $\Delta\theta$, i.e., the first order approximation, we obtain the first order screw vector $(0, 0, 0, -1, 0, 0)$. The vector represents translation toward the $-\mathbf{x}$ direction and, in actuality, the displacement correctly realizes the configuration in the face-face contact. Note that, by substituting the first order screw vector to the coefficients of $\Delta\theta^2$, Equation (3.98) is obtained, where $\mathbf{N} = (0, 0, 1)^T$.

$$\mathbf{N} \cdot \mathbf{s}_4 = 0 \tag{3.98}$$

That means that the translation where its initial velocity is $-\mathbf{x}$, and its acceleration vector is on the xy-plane is the desired displacement. The displacement also correctly realizes the configuration.

Next we consider the edge-face contact. In this case, the displacement satisfies Equation (3.99).

$$\Delta_{v_1F_1} = 0 \cap \Delta_{v_2F_1} = 0 \tag{3.99}$$

And it satisfies Equation (3.100).

$$\Delta_{e_1E} > 0 \cap \Delta_{e_2E} > 0 \cap \Delta_{v_1F_2} < 0 \cap \Delta_{v_2F_2} < 0 \tag{3.100}$$

However it is impossible to solve the two equations using only the coefficients of $\Delta\theta$. Although the screw vector $(0, 0, 0, -1, 0, 0)$ detaches a contact e_3-E, that vector does not detach a contact e_1-E and a contact e_2-E.

Then we solve Equation (3.99) using the second order approximation. That calculation is equivalent to calculating maintaining displacement. As a result, the solution $[\mathbf{s}_1, \mathbf{s}_2]$ is represented by Equation (3.101) or (3.102), where a_i is any real number.

$$[\mathbf{s}_1, \mathbf{s}_2] = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0.7071 & -0.7071 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.7071 & 0.7071 & 0 & 0 \end{pmatrix} \tag{3.101}$$

$$[\mathbf{s}_1, \mathbf{s}_2] = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.102}$$

Then we solve Equation (3.100). The coefficients of $\Delta\theta$ in Δ_{e_1E} and Δ_{e_2E} are always zero. Therefore the coefficients of $\Delta\theta^2$ must be greater than zero.

First, we consider the case that the first order screw vector is represented by Equation (3.101). By substituting the equation to the coefficients of $\Delta\theta^2$, Equation (3.103) is obtained. Note that the coefficients of $\Delta_{v_1F_2}$ and $\Delta_{v_2F_1}$ are ignored, because their coefficients of $\Delta\theta$ can be set not to zero by selecting the first order screw vector.

$$\begin{aligned} \Delta_{v_1F_1} &: \mathbf{N}\cdot\mathbf{s}_4 &=& \ 0 \\ \Delta_{v_2F_1} &: \mathbf{N}\cdot\mathbf{s}_4 &=& \ 0 \\ \Delta_{e_1E} &: \mathbf{N}\cdot\mathbf{s}_4 &>& \ 0 \\ \Delta_{e_1E} &: \mathbf{N}\cdot\mathbf{s}_4 &>& \ 0 \end{aligned} \tag{3.103}$$

Obviously, no solution exists.

Next, we consider the case that the first order screw vector is represented by Equation (3.102). By substituting the equation to the coefficients of $\Delta\theta^2$, Equation (3.104) is obtained, where $\mathbf{a} = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}$ and

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

$$\begin{aligned} \Delta_{v_1F_1} &: \mathbf{N}\cdot\mathbf{s}_4 + \mathbf{a}^T A\mathbf{a} &=& \ 0 \\ \Delta_{v_2F_1} &: \mathbf{N}\cdot\mathbf{s}_4 + \mathbf{a}^T A\mathbf{a} &=& \ 0 \\ \Delta_{e_1E} &: \mathbf{N}\cdot\mathbf{s}_4 - \mathbf{a}^T A\mathbf{a} &>& \ 0 \\ \Delta_{e_2E} &: \mathbf{N}\cdot\mathbf{s}_4 - \mathbf{a}^T A\mathbf{a} &>& \ 0 \end{aligned} \tag{3.104}$$

From the equation, Equation (3.105) should be satisfied.

$$\mathbf{a}^T A\mathbf{a} < 0 \tag{3.105}$$

We set up $\mathbf{a} = (1, 0, -1)$, which satisfies Equation (3.105). By substituting it to Equation (3.105), Equation (3.106) is obtained.

$$\begin{aligned} \Delta_{v_1F_1} &: \mathbf{N}\cdot\mathbf{s}_4 - 1 &=& \ 0 \\ \Delta_{v_2F_1} &: \mathbf{N}\cdot\mathbf{s}_4 - 1 &=& \ 0 \\ \Delta_{e_1E} &: \mathbf{N}\cdot\mathbf{s}_4 + 1 &>& \ 0 \\ \Delta_{e_2E} &: \mathbf{N}\cdot\mathbf{s}_4 + 1 &>& \ 0 \end{aligned} \tag{3.106}$$

We set up $\mathbf{s}_4 = (0, 0, 1)$. As a result, we obtain the screw vector $(0, -1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 1)$. The vector obviously detaches a contact e_3-E. The accuracy of the solution is estimated using the distance between the vertex and the face. Figure 3.7 shows the accuracy with respect to the screw vector and a screw vector $(0,-1,0,-1,0,0,0,0,0,0,0,0)$, which have only the effect of the first order screw vector.

In the vertex-face contact, we conclude that the screw vector $(0,1,-1,0,0,0,0,0,0,0,0,1)$ is one of approximate displacement. Figure 3.8 shows the accuracy with respect to the screw vector and a screw vector $(0,1,-1,0,0,0,0,0,0,0,0,0)$, which have only the effect of the first order screw vector.

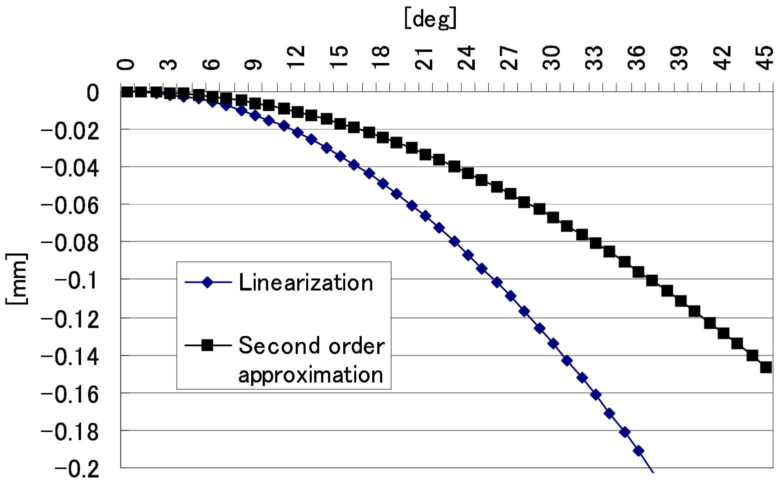


Figure 3.7: Accuracy of solution of the first and the second order approximation

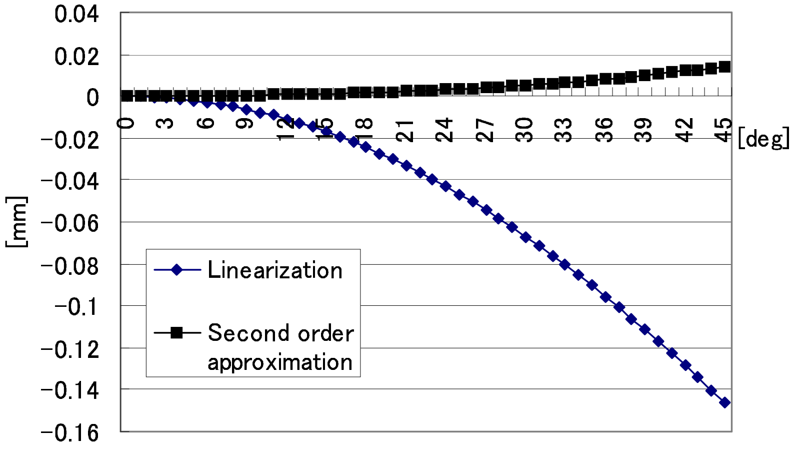


Figure 3.8: Accuracy of solution of the first and the second order approximation

Chapter 4

Estimating Parameters of Joints from Observation

In this chapter, we deal with tasks for manipulating linkages connected by some kind of a joint, for example, for rotating a doorknob, pulling open a drawer, inserting a screw with a screw driver, etc. In this case, we determine the constrained motion from a kind of a joint. Thus, it corresponds to an abstract task representation. Manipulation of linkages also requires us to know the parameters of the joint (in the case of rotating a doorknob, the axis direction and the center of rotation). They are essential for executing movement primitives.

Here, we propose a method for obtaining the parameters of some types of joints from the observation data. Note that the type of the joint is known in advance. The observation data consist of relative configurations between the two linkages connected by the joint. Of course the data include some errors. We assume that these errors are the Gaussian noise.

We concentrate on dealing with the following three types of joints (See Fig. 4.1):

- Prismatic joint (Fig. 4.1 (a))
- Revolute joint (Fig. 4.1 (b))
- Screw joint (Fig. 4.1 (c))

These joints often appear in objects which are manipulated daily.

In addition, we illustrate a method to estimate parameters of another kind of a joint. The difficulty in estimating them is caused by estimating parameters with respect to an object orientation (for example, the axis direction in the revolute

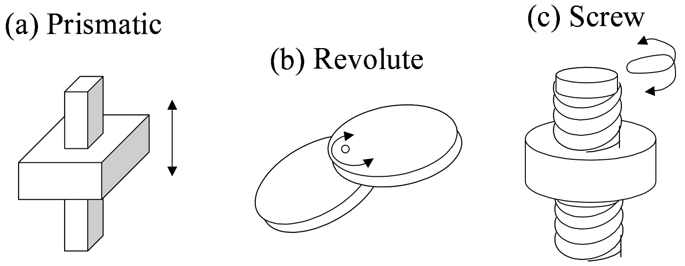


Figure 4.1: Our target linkages

joint). Therefore, we illustrate a method for estimating the parameters with respect to the orientation for any joints.

4.1 Preliminary

Let Θ be a 3×3 orthogonal matrix which represents the transformation between two object orientations. The matrix can be represented by rotation about an axis $\mathbf{n}$ by θ radian. In this thesis, we define $|\theta|$ as the metric of the matrix. $||\Theta||$ denote the metric.

From now on, we consider estimating the parameters given the type of joints and several relative configurations $\mathbf{q}_i = ({}^B\mathbf{t}^i_A, {}^B\Theta^i_A)$ of an object A with respect to the coordinate system of an object B obtained from the observation, where ${}^B\mathbf{t}^i_A$ is a three dimensional vector which represents the location and ${}^B\Theta^i_A$ is a 3×3 orthogonal matrix which represents the orientation. Note that we present an index i (≥ 1), which represents a time step, as a superscript in these terms.

The relative configurations usually include some errors. Let $\hat{\mathbf{q}}_i = ({}^B\hat{\mathbf{t}}^i_A, {}^B\hat{\Theta}^i_A)$ be the true (estimated) relative configurations. Let $\Delta\mathbf{q}_i = (\Delta\mathbf{t}^i, \Delta\Theta^i)$ be the difference between the obtained and the true relative configuration.

Now, we consider the following three types of joints (See Fig. 4.1): a prismatic joint, a revolute joint, and a screw joint.

4.2 Estimating Parameters of a Prismatic Joint

4.2.1 Formulation

Parameters of a prismatic joint consist of relative orientation ${}^B\Theta_A$ of an object A with respect to the coordinate system of an object B and the movable directions ${}^A\mathbf{l}$

and ${}^{B}\mathbf{l}$ with respect to the coordinate system of two objects A and B, respectively, where $|{}^{A}\mathbf{l}| = |{}^{B}\mathbf{l}| = 1$.

The prismatic joint requires us to satisfy Equation (4.1), (4.2), and (4.3) for all i, where we assume that ${}^{B}\hat{\mathbf{t}}_{A}^{i+1} \neq {}^{B}\hat{\mathbf{t}}_{A}^{i}$.

$$\begin{aligned} {}^{B}\Theta_{A} &= {}^{B}\hat{\Theta}_{A}^{i} && (4.1) \\ {}^{B}\mathbf{l} &= \frac{{}^{B}\hat{\mathbf{t}}_{A}^{i+1} - {}^{B}\hat{\mathbf{t}}_{A}^{i}}{|{}^{B}\hat{\mathbf{t}}_{A}^{i+1} - {}^{B}\hat{\mathbf{t}}_{A}^{i}|} && (4.2) \\ {}^{A}\mathbf{l} &= {}^{B}\Theta_{A}^{T}\,{}^{B}\mathbf{l} && (4.3) \end{aligned}$$

4.2.2 Parameter Estimation

In practice, the parameter estimation has to be performed from erroneous relative configurations. A reasonable estimation is to solve Equation (4.1), (4.2) and (4.3) by minimizing the error correction terms $\Delta\mathbf{t}_i$ and $\Delta\Theta_i$. Such minimization tends to reserve the information about the unconstrained motion, because the method does not contaminate the original data as far as possible.

First we estimate the true relative orientation ${}^{B}\hat{\Theta}_{A}$. $\Delta\Theta^{i}$ is represented by Equation (4.4).

$$\Delta\Theta^{i} = {}^{B}\Theta_{A}^{i}\,{}^{B}\hat{\Theta}_{A}^{T} \quad (4.4)$$

We assume that a transformation matrix $\Delta\Theta^{i}$ is represented by rotation about an axis $\mathbf{l}_i$ by θ_i radian. Then, Equation (4.5) is obtained by applying a trace to both sides of Equation (4.4).

$$\mathrm{Tr}({}^{B}\Theta_{A}^{i}\hat{\Theta}^{T}) = \mathrm{Tr}(\Delta\Theta^{i}) = 1 + 2\cos\theta_i \quad (4.5)$$

Because $1 - \cos\theta_i$ is minimum when $||\Delta\Theta^{i}|| = |\theta_i|$ is minimum, we estimate the parameter ${}^{B}\hat{\Theta}_{A}$ by minimizing Equation (4.6) using the non-linear optimization method.

$$\sum_{i}(1 - \cos\theta_i) = \sum_{i}\frac{3 - \mathrm{Tr}({}^{B}\Theta_{A}^{i}\,{}^{B}\hat{\Theta}_{A}^{T})}{2} \quad (4.6)$$

Next we estimate the parameter ${}^{B}\mathbf{l}$; this can be accomplished by applying the principle component analysis (PCA) to ${}^{B}\mathbf{t}_{A}^{i}$. Concretely, we set up the covariant matrix as Equation (4.7), where ${}^{B}\bar{\mathbf{t}}_{A}$ is the average of ${}^{B}\mathbf{t}_{A}^{i}$.

$$\left(\sum_{i}{}^{B}\mathbf{t}_{A}^{i} - {}^{B}\bar{\mathbf{t}}_{A}\right)^{T}\left(\sum_{i}{}^{B}\mathbf{t}_{A}^{i} - {}^{B}\bar{\mathbf{t}}_{A}\right) \quad (4.7)$$

${}^{B}\mathbf{l}$ is equal to the eigen vector with the maximum eigen value of the covariant matrix. ${}^{A}\mathbf{l}$ is calculated by Equation (4.3).

4.3 Estimating Parameters of a Revolute Joint

4.3.1 Formulation

Parameters of the revolute joint consist of the axis directions ${}^{A}\mathbf{l}$ and ${}^{B}\mathbf{l}$ and the centers of the axis ${}^{A}\mathbf{c}$ and ${}^{B}\mathbf{c}$ with respect to the coordinate system of two objects A and B, respectively, where $|{}^{A}\mathbf{l}| = |{}^{B}\mathbf{l}| = 1$.

The revolute joint requires us to satisfy Equation (4.8) and (4.9) for all i.

$$\begin{aligned} {}^{B}\mathbf{l} &= {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\mathbf{l} &(4.8)\\ {}^{B}\mathbf{c} &= {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\mathbf{c} + {}^{B}\hat{\mathbf{t}}^{i}_{A} &(4.9)\end{aligned}$$

4.3.2 Parameter Estimation

First we estimate the axis directions ${}^{A}\mathbf{l}$ and ${}^{B}\mathbf{l}$ using Equation (4.8). Equation (4.10) is obtained by reforming Equation (4.8), using the term $\Delta\Theta^{i}$.

$${}^{B}\mathbf{l} = \Delta\Theta^{i}\ {}^{B}\Theta^{i}_{A}\ {}^{A}\mathbf{l} \qquad (4.10)$$

Now we present $\Delta\Theta^{i}$ as a product of two transformation matrices as shown Equation (4.11).

$$\Delta\Theta^{i} = R({}^{B}\mathbf{l}, \theta_{1i})R(\mathbf{l}_{i}, \theta_{2i}) \qquad (4.11)$$

A transformation matrix $R(\mathbf{l}, \theta)$ presents the transformation to rotate about an axis $\mathbf{l}$ by θ radian and ${}^{B}\mathbf{l} \cdot \mathbf{l}_{i} = 0$ is always satisfied for all i.

By substituting Equation (4.11) to Equation (4.10), Equation (4.12) is obtained.

$$R({}^{B}\mathbf{l}, \theta_{1i})^{T}\ {}^{B}\mathbf{l} = R(\mathbf{l}_{i}, \theta_{2i}){}^{B}\Theta^{i}_{A}\ {}^{A}\mathbf{l} \qquad (4.12)$$

The vector value of $R({}^{B}\mathbf{l}, \theta_{1i})^{T}\ {}^{B}\mathbf{l}$ is constant for any θ_{1i}. That means that we cannot decide whether rotation about the axis ${}^{B}\mathbf{l}$ is essential or erroneous. Therefore $\theta_{1i} = 0$ is assumed to minimize $\Delta\Theta^{i}$.

By multiplying ${}^{B}\mathbf{l}^{T}$ to both sides of Equation (4.12) from the right, Equation (4.13) is obtained, where θ_{i} is used as a substitute for θ_{2i} to simply present it.

$$R(-\mathbf{l}_{i}, \theta_{i}){}^{B}\mathbf{l}{}^{B}\mathbf{l}^{T} = {}^{B}\Theta^{i}_{A}\ {}^{A}\mathbf{l}{}^{B}\mathbf{l}^{T} \qquad (4.13)$$

The left side of Equation (4.13) can be expressed as Equation (4.14), where $[*]_{\times}$ is a skew symmetry matrix (See Section 3.1.4).

$$(I - \sin\theta_{i}[\mathbf{l}_{i}]_{\times} + (1 - \cos\theta_{i})[\mathbf{l}_{i}]^{2}_{\times}){}^{B}\mathbf{l}{}^{B}\mathbf{l}^{T} \qquad (4.14)$$

Because

$$\begin{aligned}
\mathrm{Tr}({}^B\mathbf{l}{}^B\mathbf{l}^T) &= 1 \\
\mathrm{Tr}([\mathbf{l}_i]_\times {}^B\mathbf{l}{}^B\mathbf{l}^T) &= 0 \\
\mathrm{Tr}([\mathbf{l}_i]_\times^2 {}^B\mathbf{l}{}^B\mathbf{l}^T) &= ({}^B\mathbf{l}\cdot\mathbf{l}_i)^2 - 1 = -1
\end{aligned}$$

are always satisfied, Equation (4.15) is obtained.

$$\mathrm{Tr}(R(-\mathbf{l}_i,\theta_i){}^B\mathbf{l}{}^B\mathbf{l}^T) = \cos\theta_i \tag{4.15}$$

As the same way in the prismatic joint case, we estimate the parameter ${}^A\mathbf{l}$ and ${}^B\mathbf{l}$ by minimizing Equation (4.16) using the non-linear optimization method.

$$\sum_i (1-\cos\theta_i) = \sum_i (1 - \mathrm{Tr}({}^B\Theta_A^i\ {}^A\mathbf{l}{}^B\mathbf{l}^T)) \tag{4.16}$$

Next we calculate ${}^B\hat{\Theta}_A^i$. We can calculate $\Delta\Theta^i$ using the cross product for ${}^B\Theta_A^i\ {}^A\mathbf{l}$ and ${}^B\mathbf{l}$. Therefore, we can calculate ${}^B\hat{\Theta}_A^i$.
Then we estimate parameters ${}^A\mathbf{c}$ and ${}^B\mathbf{c}$ using Equation (4.9). Equation (4.17) is obtained by reforming Equation (4.9) using the term $\Delta\mathbf{t}^i$.

$$\Delta\mathbf{t}^i = {}^B\mathbf{c} - {}^B\hat{\Theta}_A^i\ {}^A\mathbf{c} - {}^B\mathbf{t}_A^i \tag{4.17}$$

We can estimate the parameters using the least square method, i.e., we have only to solve Equation (4.18), where $A_i = \begin{pmatrix} -{}^B\hat{\Theta}_A^i & I \end{pmatrix}$.

$$\left(\sum_i A_i^T A_i\right)\begin{pmatrix} {}^A\mathbf{c} \\ {}^B\mathbf{c} \end{pmatrix} = \sum_i A_i^T\ {}^B\mathbf{t}_A^i \tag{4.18}$$

Note that a matrix $\sum_i A_i^T A_i$ may not be full-rank because of an ambiguity of the center of rotation. We solve the equation using the singular value decomposition (SVD).

4.4 Estimating Parameters of a Screw Joint

4.4.1 Formulation

Parameters of the screw joint consist of a ratio r between two amounts of translation and rotation, ${}^A\mathbf{q}_{LA} = ({}^A\mathbf{t}_{LA}, {}^A\Theta_{LA})$ which is the configuration of the screw coordinate system Σ_{LA} with respect to the coordinate system of an object A, and ${}^{B}\mathbf{q}_{LB} = ({}^{B}\mathbf{t}_{LB}, {}^{B}\Theta_{LB})$, which is the configuration of the screw coordinate system Σ_{LB} with respect to the coordinate system of an object B. We assume that the axis direction and the center of rotation are the direction of the z-axis and the origin of the screw coordinate system, respectively.

The screw joint requires us to satisfy Equation (4.19) and (4.20) for all i, where $\mathbf{z} = (0, 0, 1)^T$.

$$\begin{aligned} {}^{B}\Theta_{LB}\mathbf{z} &= {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\Theta_{LA}\mathbf{z} && (4.19) \\ {}^{B}\mathbf{c} &= {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\mathbf{c} + {}^{B}\hat{\mathbf{t}}^{i}_{A} - r\theta_i\, {}^{B}\Theta_{LB}\mathbf{z} && (4.20) \end{aligned}$$

θ_i represents the rotation angle and can be calculated by Equation (4.21), where $k_i \in Z$.

$$\theta_i = ||{}^{B}\Theta^{iT}_{LB}\ {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\Theta^{i}_{LA}|| + 2k_i\pi \qquad (4.21)$$

Although the value of k_i cannot be uniquely decided by the equation only, we decide k_i using the continuity of θ_i, i.e., using that $|\theta_i - \theta_{i-1}| < \pi$ is satisfied for all i.

4.4.2 Parameter Estimation

First we estimate the parameters ${}^{A}\Theta_{LA}$ and ${}^{B}\Theta_{LB}$. In actuality, we can estimate only ${}^{A}\Theta_{LA}\mathbf{z}$ and ${}^{B}\Theta_{LB}\mathbf{z}$. We estimate these parameters using Equation (4.19) at first. By substituting ${}^{A}\mathbf{l}$ and ${}^{B}\mathbf{l}$ for ${}^{A}\Theta_{LA}\mathbf{z}$ and ${}^{B}\Theta_{LB}\mathbf{z}$, respectively, Equation (4.19) is equivalent to Equation (4.8). That means that these two vectors can be estimated by the same method that was used for the revolute joint. As a result, the z-axes of the screw coordinate systems Σ_{LA} and Σ_{LB} are obtained, respectively. We are permitted to set up the x-axis and the y-axis as we like; therefore, ${}^{A}\Theta_{LA}$ and ${}^{B}\Theta_{LB}$ are set up to one of any solution.

Next we estimate the parameters ${}^{A}\mathbf{c}$, ${}^{B}\mathbf{c}$, and r. Equation (4.22) is obtained by reforming Equation (4.20) using the term $\Delta\mathbf{t}_i$, where $\Delta\theta_i$ is the error with respect to rotation about the axis ${}^{B}\mathbf{l}$.

$${}^{B}\mathbf{c} = {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\mathbf{c} + {}^{B}\mathbf{t}^{i}_{A} + \Delta\mathbf{t}^{i} - r(\theta_i + \Delta\theta_i){}^{B}\mathbf{l} \qquad (4.22)$$

Because $\Delta\theta_i$ and r are usually very small, we eliminate the term $r\Delta\theta_i\ {}^{B}\mathbf{l}$ from the equation. As a result, Equation (4.23) is obtained.

$$\Delta\mathbf{t}^{i} = {}^{B}\mathbf{c} - {}^{B}\hat{\Theta}^{i}_{A}\ {}^{A}\mathbf{c} + r\theta_i\ {}^{B}\mathbf{l} - {}^{B}\mathbf{t}^{i}_{A} \qquad (4.23)$$

We can estimate the parameters using the least square method, i.e., we solve Equation (4.24), where $A_i = \left(\begin{array}{ccc} -{}^B\hat{\Theta}^i_A & I & \theta_i \mathbf{l}_b \end{array} \right)$.

$$\left(\sum_i A_i^T A_i \right) \left(\begin{array}{c} {}^A\mathbf{c} \\ {}^B\mathbf{c} \\ r \end{array} \right) = \sum_i A_i^T \; {}^B\mathbf{t}^i_A \tag{4.24}$$

4.5 Implementation and Experiment

4.5.1 Implementation

In this section, we describe our implementation. For the implementation, we solve the following two issues:

- How are the axis direction and object orientation represented?
- What kind of a non-linear optimization method is employed?

With respect to the first issue, we represent the axis direction using the spherical coordinate of which the radius is one. For example, ${}^A\mathbf{l}$ and ${}^B\mathbf{l}$ are represented by Equation (4.25).

$$\begin{aligned} {}^A\mathbf{l} &= (\sin\alpha\cos\beta, \sin\alpha\sin\beta, \cos\alpha)^T \\ {}^B\mathbf{l} &= (\sin\phi\cos\gamma, \sin\phi\sin\gamma, \cos\phi)^T \end{aligned} \tag{4.25}$$

We represent the orientation using roll-pitch-yaw angles. For example, ${}^B\Theta_A$ is represented by Equation (4.26).

$${}^B\Theta_A = \left(\begin{array}{ccc} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\alpha & \cos\beta\sin\gamma & \cos\alpha\cos\gamma \end{array} \right) \tag{4.26}$$

Although these representations have several singularities, which sometimes prevent convergence, a serious problem does not appear in the actual use.

We employ the conjugate gradient method as the non-linear optimization method. With respect to the prismatic joint, the initial guess, which is required by the method, is set up to ${}^B\Theta^1_A$. With respect to the revolute and screw joints, the initial guess of the axis direction ${}^A\mathbf{l}_{in}$ and ${}^B\mathbf{l}_{in}$ is set up as follows:

1. Select i and j, where $||{}^B\Theta^{iT}_A \; {}^B\Theta^j_A||$ is sufficiently near to $\frac{\pi}{2}$.

2. Decide ${}^{A}\mathbf{l}_{in}$ by solving $R({}^{A}\mathbf{l}_{in}, \theta) =^{B} \Theta_{A}^{iT}\ {}^{B}\Theta_{A}^{j}$.

3. Decide ${}^{B}\mathbf{l}_{in}$ using ${}^{B}\mathbf{l}_{in} =^{B} \Theta_{A}^{i}\ {}^{A}\mathbf{l}_{in}$.

Note that Equation (4.27) is always satisfied under the noiseless situation.

$$ {}^{B}\hat{\Theta}_{A}^{iT}\ {}^{B}\hat{\Theta}_{A}^{j}\ {}^{A}\mathbf{l} =^{A} \mathbf{l} \tag{4.27}$$

That means that a transformation matrix ${}^{B}\hat{\Theta}_{A}^{iT}\ {}^{B}\hat{\Theta}_{A}^{j}$ can be represented as rotation about ${}^{A}\mathbf{l}$.

4.5.2 Simulation Experiment

At first, we examine the performance of our proposed method using artificial data. Because the true parameters are known, it is possible to estimate the performance. The purpose of this experiment is as follows:

- Compare our proposed method with another method
- Examine the relationship between the accuracy and the experimental set up

In actuality, we executed the experiment by performing the following steps:

1. Randomly set up the parameters
2. Generate the relative configurations without noise according to the parameters
3. Contaminate the configurations with noise
4. Estimate the parameters using the noise-contaminated configurations
5. Compare the estimated parameters with the true parameters

We examined the performance by reiterating these steps 1000 times.
With respect to Step 3., the noise-contaminated configurations are generated using Equation (4.28), where $\Delta\mathbf{t}$ and $\Delta\Theta$ are noises with respect to the location and the orientation.

$$\begin{aligned} {}^{B}\mathbf{t}_{A}^{i} &= \Delta\mathbf{t} + {}^{B}\hat{\mathbf{t}}_{A}^{i} \\ {}^{B}\Theta_{A}^{i} &= \Delta\Theta\, {}^{B}\hat{\Theta}_{A}^{i} \end{aligned} \tag{4.28}$$

$|\Delta\mathbf{t}|$ is randomly set up according to the rectangular distribution of which interval is 0 to 10[mm]. Of course, the direction of $\Delta\mathbf{t}$ is randomly set up. $\Delta\Theta$ is calculated by Equation (4.29), where $\Delta\theta$ is randomly set up according to the rectangular distribution of which interval is 0 to 5[deg] and $\mathbf{l}$ is randomly set up.

$$\Delta\Theta = R(\mathbf{l}, \Delta\theta) \tag{4.29}$$

The intervals are set up based on the performance of our vision system.

We first compared our proposed method with another method. We selected the following method (referred to as *Method A*) as the comparison method: That method estimates the axis direction using only two orientations ${}^{B}\Theta_{A}^{i}$ and ${}^{B}\Theta_{A}^{j}$ where $||{}^{B}\Theta_{A}^{iT}\ {}^{B}\Theta_{A}^{j}||$ is the nearest to $\dfrac{\pi}{2}$.

Although it is possible to estimate the axis direction using the average of the axis directions estimated by the above method, we need to calculate summation in the S^2 space and such calculation is not general. Therefore, such an estimation method is unfamiliar, we believe.

The focus is on the estimation of the axis directions. We employed the following nine types of the experimental set up:

- The number of configurations : 50, 100, or 200
- An amount of rotation : 45[deg], 90[deg], or 180[deg]

Table 4.1 shows the result of the comparison.

From the table, we discovered the following things: The performance of our proposed method is superior to Method A. The accuracy of Method A suddenly declines if an amount of rotation is less than $\dfrac{\pi}{2}$ and is about constant with respect to the number of configurations. The accuracy of our proposed method is improved by increasing the number of the configurations and an amount of rotation.

Next, we examined the relationship between the accuracy and the experimental set up. We employed the same experimental set up. Table 4.2 shows the result. As a result, we discovered the following things: For all parameters, the inaccuracy is about inversely proportional to an amount of rotation, i.e., if the amount is doubled, the accuracy is doubled. The square of the accuracy is proportional to the number of configurations, i.e., the accuracy doubles if the number increases 4 $(= 2^2)$ times. As a result, it is more difficult to estimate the parameters of the revolute joint with a smaller amount of rotation.

Next, we examined the relationship between the accuracy and the experimental set up in the prismatic joint. We employed the following nine types of the experimental set up:

- The number of configurations : 50, 100, or 200

Table 4.1: Comparison between our proposed method and another method

Difference between the estimated and the true axis direction with respect to ${}^{A}\mathbf{l}$

Amount of rotation - # of configurations	Method A Average (SD)[deg]	Our method Average (SD)[deg]
45[deg] - 50	3.857 (2.104)	1.330 (0.6914)
45[deg] - 100	3.800 (2.157)	0.9352 (0.5167)
45[deg] - 200	3.884 (2.143)	0.6989 (0.3877)
90[deg] - 50	2.128 (1.120)	0.6916 (0.3455)
90[deg] - 100	2.053 (1.147)	0.4881 (0.2552)
90[deg] - 200	2.077 (1.132)	0.3485 (0.1891)
180[deg] - 50	1.999 (1.117)	0.3951 (0.2000)
180[deg] - 100	2.051 (1.177)	0.2690 (0.1424)
180[deg] - 200	2.044 (1.130)	0.1904 (.09464)

Difference between the estimated and the true axis direction with respect to ${}^{B}\mathbf{l}$

Amount of rotation - # of configurations	Method A Average (SD)[deg]	Our method Average (SD)[deg]
45[deg] - 50	3.909 (2.122)	1.329 (0.6838)
45[deg] - 100	3.741 (2.157)	0.9411 (0.5109)
45[deg] - 200	3.862 (2.126)	0.6904 (0.3826)
90[deg] - 50	2.096 (1.199)	0.6890 (0.3474)
90[deg] - 100	2.051 (1.158)	0.4929 (0.2558)
90[deg] - 200	2.060 (1.134)	0.3397 (0.1907)
180[deg] - 50	2.058 (1.151)	0.3818 (0.1970)
180[deg] - 100	2.048 (1.170)	0.2701 (0.1468)
180[deg] - 200	2.042 (1.151)	0.1934 (0.1010)

Table 4.2: Relationship between the accuracy and the experimental set up

Difference between the estimated and the true axis direction with respect to $^{A}\mathbf{l}$

	50	100	200
	Average (SD)[deg]	Average (SD)[deg]	Average (SD)[deg]
45[deg]	1.330 (0.6914)	0.9352 (0.5167)	0.6989 (0.3877)
90[deg]	0.6916 (0.3455)	0.4881 (0.2551)	0.3485 (0.1891)
180[deg]	0.3951 (0.2000)	0.2690 (0.1424)	0.1904 (.09464)

Difference between the estimated and the true axis direction with respect to $^{B}\mathbf{l}$

	50	100	200
	Average (SD)[deg]	Average (SD)[deg]	Average (SD)[deg]
45[deg]	1.329 (0.6838)	0.9411 (0.5109)	0.6904 (0.3826)
90[deg]	0.6890 (0.3474)	0.4929 (0.2558)	0.3397 (0.1907)
180[deg]	0.3818 (0.1970)	0.2701 (0.1468)	0.1934 (0.1010)

Difference between the estimated and the true axis direction with respect to $^{A}\mathbf{c}$

	50	100	200
	Average (SD)[mm]	Average (SD)[mm]	Average (SD)[mm]
45[deg]	2.632 (1.378)	1.870 (0.9719)	1.317 (0.6787)
90[deg]	1.394 (0.7260)	0.9468 (0.4800)	0.6748 (0.3424)
180[deg]	0.8007 (0.3834)	0.5706 (0.2735)	0.4047 (0.1928)

Difference between the estimated and the true axis direction with respect to $^{B}\mathbf{c}$

	50	100	200
	Average (SD)[mm]	Average (SD)[mm]	Average (SD)[mm]
45[deg]	2.640 (1.384)	1.858 (0.9538)	1.327 (0.6823)
90[deg]	1.398 (0.7326)	0.9509 (0.4869)	0.6697 (0.3396)
180[deg]	0.8043 (0.3817)	0.5836 (0.2772)	0.3967 (0.1923)

Table 4.3: Relationship between the accuracy and the experimental set up

Difference between the estimated and the true axis direction with respect to ${}^{A}\mathbf{l}$

	50 Average (SD)[mm]	100 Average (SD)[mm]	200 Average (SD)[mm]
50[mm]	3.003 (1.649)	2.384 (1.302)	2.061 (1.173)
100[mm]	2.103 (1.173)	1.787 (1.095)	1.709 (1.106)
200[mm]	1.743 (1.154)	1.690 (1.154)	1.604 (1.153)

Difference between the estimated and the true axis direction with respect to ${}^{B}\mathbf{l}$

	50 Average (SD)[mm]	100 Average (SD)[mm]	200 Average (SD)[mm]
50[mm]	2.412 (1.261)	1.689 (0.8845)	1.212 (0.6207)
100[mm]	1.222 (0.6029)	0.8123 (0.4324)	0.5790 (0.3070)
200[mm]	0.5718 (0.3063)	0.4148 (0.2256)	0.2954 (0.1532)

- An amount of translation : 50[mm], 100[mm], or 200[mm]

Table 4.3 shows the result. The table shows the following things: It is more inaccurate to estimate ${}^{A}\mathbf{l}$ than ${}^{B}\mathbf{l}$, because ${}^{A}\mathbf{l}$ is estimated using Equation (4.3). The inaccuracy with respect to ${}^{B}\mathbf{l}$ is about inversely proportional to an amount of translation and the square of the accuracy is proportional to the number of the configurations. As the result, it is more difficult to estimate the parameters of the prismatic joint with a smaller amount of translation.

We then examined the relationship between the accuracy and the experimental set up in the screw joint. Because we have already examined the accuracy with respect to the axis direction using examples of the revolute joint, we concentrate on the parameter r.

We employed the following nine types of the experimental set up:

- The number of configurations : 50, 100, or 200
- An amount of rotation : 360[deg]
- Ratio between two amounts of translation and rotation: 0.2, 1.0, or 5.0

We estimated the accuracy using the absolute value of subtraction from the estimated parameter to the true parameter. Table 4.3 shows the result: The square

Table 4.4: Relationship between the accuracy and the experimental set up

	50 Average (SD)	100 Average (SD)	200 Average (SD)
0.2	0.2073 (0.1607)	0.1492 (0.1127)	0.1104 (.08078)
1.0	0.2105 (0.1613)	0.1454 (0.1098)	0.1037 (.07513)
5.0	0.1958 (0.1472)	0.1446 (0.1074)	0.1042 (.07886)

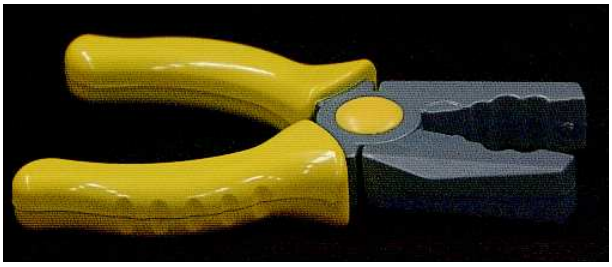


Figure 4.2: Toy plier

of the accuracy is proportional to the number of configurations. The accuracy is insensitive to the ratio. However, the proportional error is bigger according to the decrease of the ratio.

4.5.3 Experiment With Actual Objects

In actuality, we estimated the parameters of the revolute joint which a toy plier equips as shown in Fig. 4.2 using our proposed method. We estimated that by performing the following steps:

1. Measure 3D shape of the plier using a Laser Range Finder (Vivid 910 produced by Konika Minolta Holdings, INC.[68]) at several times.

2. Align one of two pairs connected by the revolute joint between two range images

3. Model the each pair by subtracting one range image from the aligned range image (See Fig. 4.3)

4. Obtain several relative configurations using their models from observation.

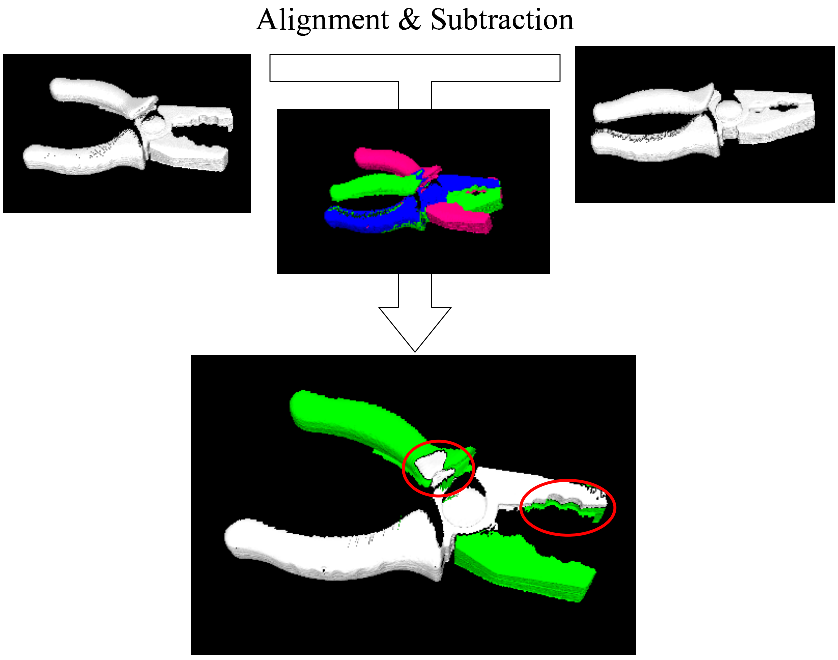


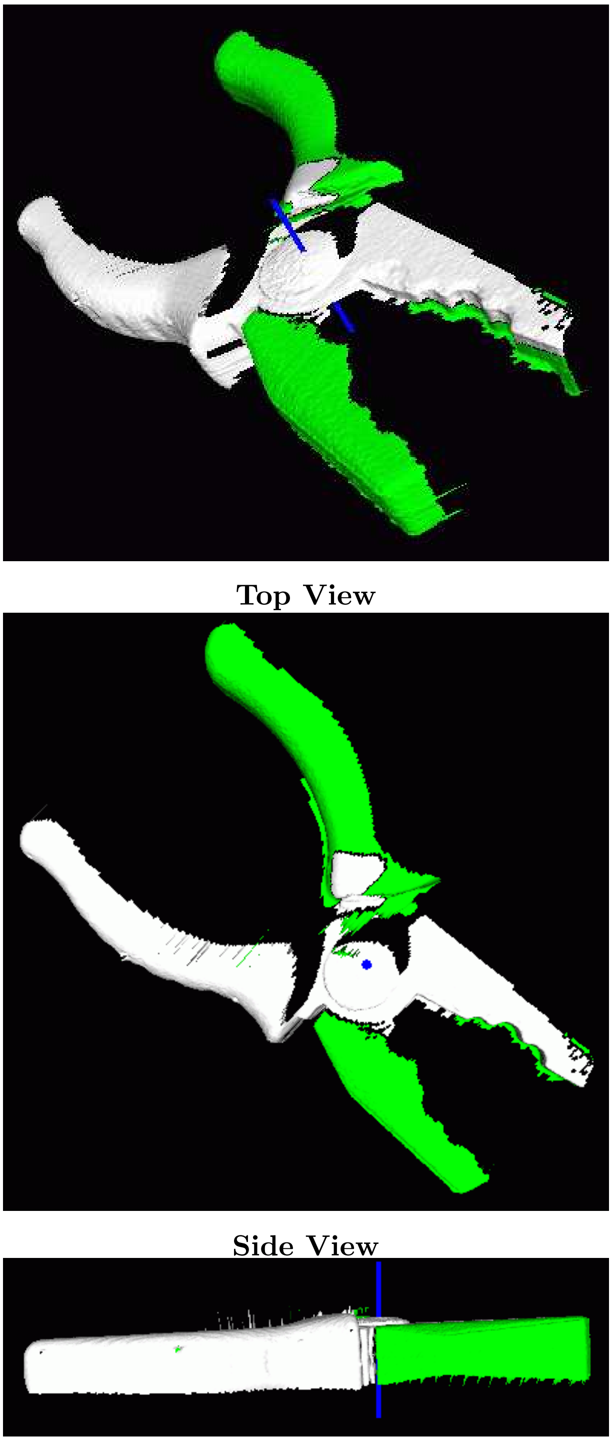
Figure 4.3: 2-part division

5. Estimate the parameters using the configurations

Figure 4.3 shows the result of modeling the two pairs. Although it locally failed to separate one range image into two pairs (see the areas surrounded by the red circles in the figure), it is enough to estimate configurations of each pair from the observation.
In the toy plier, an amount of rotation is limited up to about 30[deg]. Because it is difficult to estimate the parameters using the stereo vision which we employ throughout this thesis, we obtain five relative configurations from the range images. Figure 4.4 shows the estimation result. The blue line in the figure represents the estimated rotational axis.

4.6 Estimating Parameters of Other Types of Joints

As mentioned above, the difficulty in estimating the parameters is caused by estimating parameters with respect to an object orientation. For example, the center of rotation can be estimated by solving only simultaneous linear equations in revolute and screw joints. However, estimating the axis direction of rotation requires



Top View

Side View

Figure 4.4: Estimated parameters of a rotational mechanical joint

us to calculate the non-linear equation, of which the estimation is more difficult.

In this section, we illustrate a method for estimating the parameters with respect to an object orientation of other types of joints. Any joints are classified into four types based on a DOF with respect to orientation, i.e., a restricted DOF in rotation. The range of the restricted DOF in rotation is from zero to three.

Consider the joints of which the restricted DOF in rotation are three, for example, the prismatic joint. The parameters consist of the true relative orientation between the two pairs, and the estimation has already been described.

Consider the joints of which the restricted DOF in rotation are two, for example, the revolute joint and the screw joint. The parameters consist of the axis directions, and the estimation has already been described.

Consider the joints of which the restricted DOF in rotation is zero, for example, the spherical joint. There are no parameters with respect to the orientation. Errors with respect to the orientation cannot be corrected, because the restricted DOF in rotation is zero, i.e., any orientation can be realized.

In the following section, we describe how to estimate parameters of the joints of which the restricted DOF is one.

4.6.1 Formulation

The joint of which the restricted DOF in rotation is one is generally composed of two revolute joints as shown in Fig. 4.5, where the two axis directions are perpendicular to each other.

The parameters of the joint consist of ${}^{A}\Theta_{LA}$, which is the orientation of the coordinate system of the joint Σ_{LA} with respect to the coordinate system of an object A, and ${}^{B}\Theta_{LB}$, which is the orientation of the coordinate system of the joint Σ_{LB} with respect to the coordinate system of an object B, respectively. Let ${}^{LB}\Theta^{i}_{LA}$ be a transformation matrix from the coordinate system Σ_{LA} to the coordinate system Σ_{LB} at time i. From the structure of the joint, ${}^{LB}\Theta^{i}_{LA}$ is formulated by Equation (4.30).

$$ {}^{LB}\Theta^{i}_{LA} = R_z(\theta_{1i})R_y(\theta_{2i}) \tag{4.30} $$

The joint requires us to satisfy Equation (4.31).

$$ {}^{LB}\Theta^{i}_{LA} =^{B} \Theta^{T}_{LB} \; {}^{B}\hat{\Theta}^{i}_{A} \; {}^{A}\Theta_{LA} \tag{4.31} $$

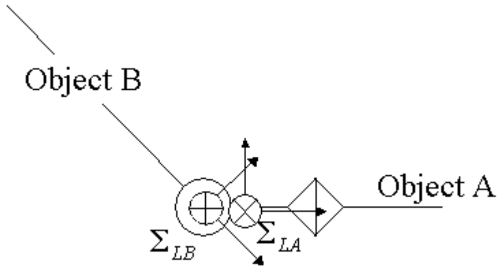


Figure 4.5: Joint with 2-axis-rotation

4.6.2 Parameter Estimation

From Equation (4.30) and (4.31), Equation (4.32) is obtained.

$$R_z(\theta_{1i}) =^B \Theta_{LB}^T \, {}^B\hat{\Theta}_A^i \, {}^A\Theta_{LA} R_y(-\theta_{2i}) \tag{4.32}$$

By multiplying $\mathbf{y} = (0,1,0)^T$ to both sides of Equation (4.32) from the right, Equation (4.33) is obtained, where $\mathbf{x} = (1,0,0)^T$ and $s_{1i}, s_{2i} \in R$.

$$s_{1i}\mathbf{x} + s_{2i}\mathbf{y} =^B \Theta_{LB}^T \, {}^B\hat{\Theta}_A^i \, {}^A\Theta_{LA}\mathbf{y} \tag{4.33}$$

By multiplying $\mathbf{z}^T$ to both sides of Equation (4.33) from the left, Equation (4.34) is obtained.

$$0 = \mathbf{z}^T \, {}^B\Theta_{LB}^T \, {}^B\hat{\Theta}_A^i \, {}^A\Theta_{LA}\mathbf{y} \tag{4.34}$$

By reforming Equation (4.34) using the term $\Delta\Theta_i$, Equation (4.35) is obtained.

$$0 = \mathbf{z}^T \, {}^B\Theta_{LB}^T \Delta\Theta_i \, {}^B\Theta_A^i \, {}^A\Theta_{LA}\mathbf{y} \tag{4.35}$$

Let ${}^A\mathbf{l}_i$ and ${}^B\mathbf{l}$ be ${}^B\Theta_A^i \, {}^A\Theta_{LA}\mathbf{y}$ and ${}^B\Theta_{LB}\mathbf{z}$, respectively. By reforming Equation (4.35) using these terms, Equation (4.36) is obtained.

$$0 = \, {}^B\mathbf{l}^T \Delta\Theta_i \, {}^A\mathbf{l}_i \tag{4.36}$$

The equation means that ${}^B\mathbf{l}$ is perpendicular to $\Delta\Theta_i \, {}^A\mathbf{l}_i$. Equation (4.37) is satisfied, when $||\Delta\Theta_i||$ is minimum (See Fig. 4.6).

$$\Delta\theta_i = ||\Delta\Theta_i|| = \left|\cos^{-1}({}^B\mathbf{l}^T \, {}^A\mathbf{l}_i) - \frac{\pi}{2}\right| = |\sin^{-1}({}^B\mathbf{l}^T \, {}^A\mathbf{l}_i)| \tag{4.37}$$

Because $|\Delta\theta_i|$ is minimum when $\sin^2\Delta\theta_i$ is minimum, we can estimate ${}^A\Theta_{LA}$ and ${}^B\Theta_{LB}$ by minimizing Equation (4.38) using the non-linear optimization method.

$$\sum_i \sin^2\Delta\theta_i = \sum_i ({}^B\mathbf{l}^T \, {}^A\mathbf{l}_i)^2 = \sum_i (\mathbf{z}^T \, {}^B\Theta_{LB}^T \, {}^B\Theta_A^i \, {}^A\Theta_{LA}\mathbf{y})^2 \tag{4.38}$$

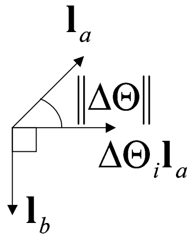


Figure 4.6: Minimum $||\Delta\theta||$

4.6.3 Example

We examined the success rate of estimating the parameters of the joints of which the restricted DOF in rotation is one. In the implementation, we represented the orientation of the coordinate systems of the joint with respect to two objects, A and B, using roll-pitch-yaw angles and we employed the conjugate gradient method as the non-linear optimization method. However, we set up the initial guess of the two orientations to the identity matrix, because we could not design a method to set up the appropriate initial guess.
The experimental set up was as follows:

- Amounts of rotation about two axes are 90[deg].
- The number of the relative orientation is 100.

Among 1000 trials, we were able to estimate the parameters for 677 trials. If the angle between the estimated and the true axis direction was less than 2.5[deg], which is the average of amounts of noise, we regarded the estimation as a success. To set up the appropriate initial guess can increase the success rate, we believe.

Chapter 5

Recognizing Knot-Tying Tasks Using One Rope from Observation

In this section, we describe a method for recognizing a knot-tying task; our method is based on the LFO paradigm. Although there are two implementation styles in the LFO paradigm, i.e., motion-based and state-based methods, it is impossible to apply the motion-based method to such a task, because the same knot-tying cannot be realized, even if a rope is moved in the same manner as that of the performer. Therefore, we apply the state-based method.
To apply it, we first should solve the following problems:

- How should the state of the task be represented?

- What are sufficient movement primitives for the task?

As far as we know, the solutions have not yet been discovered. In this section, we describe our solutions, which are introduced from the knowledge of the knot theory[11].
First we describe P-data representation, which we employ as the knot-state representation. P-data can sufficiently represent topological information of a knot and is insensitive to meaningless changes of a knot shape. Furthermore, the reversibility between a knot configuration and the P-data is guaranteed by the knot theory and the graph theory[30].
Next we define four movement primitives for the task and propose a method for selecting the corresponding movement primitives from transitions of P-data. Theoretically, any robot can execute knot-tying tasks by implementing four movement primitives.

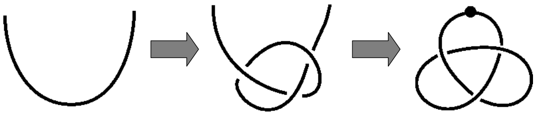


Figure 5.1: Knot

5.1 Knot-State Representation

5.1.1 Knot Projection

A tangled loop as shown to the right in Fig. 5.1 is defined as a knot in the knot theory. Theoretically, the knot is a simple closed curve with no thickness in 3D space. Fig. 5.1 reasonably illustrates a knot in 3D space on a 2D plane by applying an orthographic or perspective projection from an appropriate viewpoint. In such a projected image, several intersections appear. And in each intersection, the farthest strand, relative to the viewpoint, is drawn with gaps around the intersection.

For recoverability of essential information of a knot from a projected image, this drawing must satisfy the following conditions:

- Each intersection is a point-intersection.
- One strand goes across the other one in any intersections
- Any three strands do not intersect at the same point

By applying such a drawing style, we can correctly recover essential information of a knot from this figure. This drawing is defined as a knot projection. Fortunately, such a projected image can always be obtained from any knot by selecting an appropriate viewpoint[69].

However, our target knot is not a closed curve and it has two ends. We append a drawing rule that **any end is not on a strand**. Similarly, in the case of an open curve, a knot projection can always be obtained from any knot. From now on, we represent a knot configuration using a knot projection.

5.1.2 P-Data

In constructing a knot-state representation, the following two conditions are required:

1. A knot state is represented as an abstract data structure that does not depend on parametric information.

2. The transformation from a knot projection to a knot state is reversible. At least, the original projection can be restored from a knot state.

The first condition should be satisfied because it is unfavorable to change the state by locally moving an intersection point, transforming the shape of a strand, and so on. The second condition should be satisfied for solving a so-called path-planning problem.

We chose *P-data* representation[70][1], because it is simple and yet it satisfies both conditions. Original P-data presentation is employed for a tangled loop. Thus, some changes are necessary to apply the representation to our target knot which has two terminal ends.

In the original P-data representation, a tangled loop is converted into P-data through the following process:

1. At first we choose one point (referred to as a *selective point*) on the knot except for intersections.

2. From the selective point, we follow the knot toward one of two directions until we reach the point again. If we encounter an intersection, we number the intersection according to the order of encounter. Therefore, each intersection has two numbers.

3. We re-follow the knot. When we encounter an intersection, we record both intersection numbers. We also determine a relative vertical position (over/under) and a sign plus/minus from the relation of the direction of the following and the crossed strand.

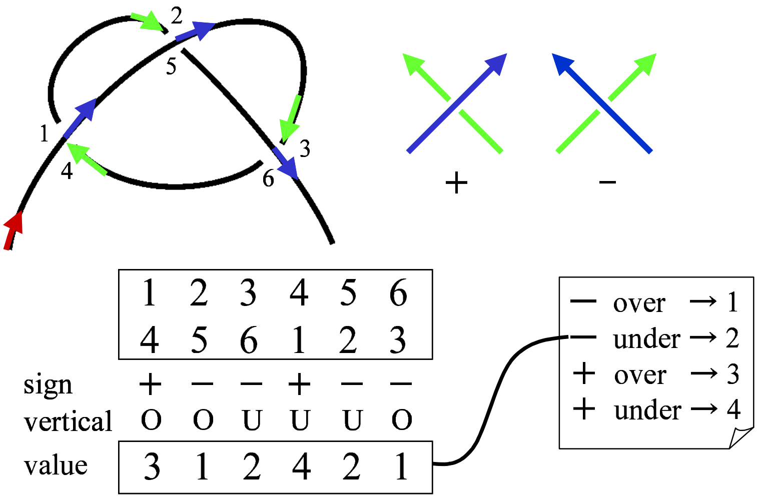
4. Finally, we code relative vertical positions and signs (referred to as *attributes*) into a set of numbers as follows: 1: over/$-$, 2: under/$-$, 3: over/$+$, 4: under/$+$.

The sign is determined by the sign of the following equation:

$$(\vec{l}_{over} \times \vec{l}_{under}) \cdot \vec{e_z}$$

Where $\vec{l}_{over}$ and $\vec{l}_{under}$ are the direction vectors of the upper and lower strands, respectively, at their intersection point. e_z is a unit vector parallel to the z axis

[1] [70] calls this presentation *perfect P-data*, but in this thesis, we simply call this *P-data*.



	1	2	3	4	5	6
	4	5	6	1	2	3
sign	+	−	−	+	−	−
vertical	O	O	U	U	U	O
value	3	1	2	4	2	1

−	over	→ 1
−	under	→ 2
+	over	→ 3
+	under	→ 4

Figure 5.2: P-data

(upward from this thesis). If the upper strand passes from the left to the right of the lower strand, the sign is plus. And if the upper strand passes from the right to the left, the sign is minus.

Our target knot is not a tangled loop and has two ends. So one of the two ends (referred to as a *selective end*) is employed as the selective point. Therefore, the following direction is uniquely determined by selection of the selective end.

By performing the above process, P-data for the knot in Fig. 5.2 is obtained as follows:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 3 & 1 & 2 & 4 & 2 & 1 \end{array}$$

5.1.3 Reversibility

A theorem in the graph theory guarantees that a knot projection can be restored from P-data[70]. Note that, in the knot theory, a knot is a closed curve. Fortunately, the reversibility is also guaranteed for our target knots. For proving the reversibility, we begin with the definition of terminologies and notations.

Definition 1 *The i-th intersection is defined as the i-th intersection encountered in the process of converting a knot projection into P-data.*

Definition 2 *A segment is defined as a local strand between two of the intersections and terminals which are continuously encountered. The i-th segment is defined as the i-th segment encountered in the process of converting a knot projection into P-data.*

Definition 3 *Given P-data P, $n(P)$ is defined as the number of columns of the P-data, i.e., twice the number of intersections in the knot projection.*

Definition 4 $\sigma(i|P)$ *is defined as the number assigned to the i-th intersection other than i. When it is clear what the state P is, we briefly describe it as* $\sigma(i)$.

Definition 5 *attr*$(i|P)$ *is defined as the attribute (1: over/−, ...) of the i-th intersection. When it is clear what the state P is, we briefly describe it as attr*(i).

For example, in the above P-data, $n(P) = 6$, $\sigma(2|P) = 5$, and $\mathrm{attr}(3|P) = 2$. Obviously, if $\sigma(i|P) = j$ is satisfied, $\sigma(j|P) = i$ must be satisfied, i.e., the commutative law is always satisfied.

Definition 6 *A* reducible *P-data is defined as the one which satisfies the following condition, where* $1 \leq i < j \leq n(p) \cap (i \neq 1 \cup j \neq n(p))$ *is satisfied:*

$$\exists i, j, \forall k, i \leq \sigma(k) \leq j (i \leq k \leq j) \tag{5.1}$$

An irreducible *P-data is one which is not reducible.*

Figure 5.3 shows knots which generate irreducible and reducible P-data, respectively. It is easy to find out that a knot which generates reducible P-data consists of some knots which generate irreducible P-data. Therefore, if we can reverse any irreducible P-data to a knot projection, we can reverse all P-data by appropriately connecting the knot projections. From now on, we concentrate on the reversibility of irreducible P-data.

Definition 7 *A* knot graph *is defined as a graph whose vertices and edges correspond to intersections and segments of a knot projection, respectively. However the first and the last segments are not included in the edges of the knot graph and the two ends of a knot projection are not included in the vertices of the knot graph. Note that, in a knot graph, an attribute of each intersection is disregarded.*

Figure 5.4 shows an example of a knot graph. Note that it is easy to convert a knot graph into a knot projection, if an attribute of each intersection is given.
Now, we introduce some propositions which are necessary to prove the reversibility.

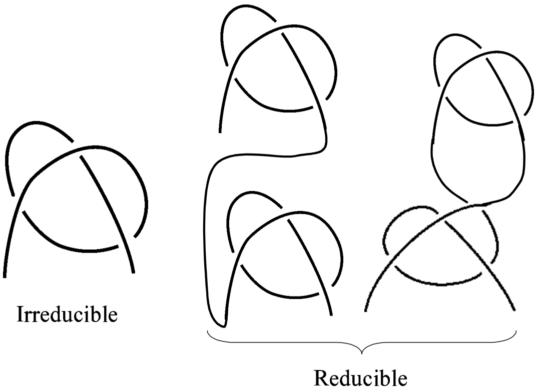


Figure 5.3: Knot projections which generate irreducible and reducible P-data, respectively

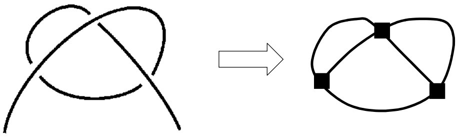


Figure 5.4: Knot projection to knot graph

Proposition 7 *The graph which is dual to any knot graph is a connected graph, where the external face and its connections to inner faces are disregarded in a dual graph.*

Figure 5.5 shows an example of a dual graph.

Proof

If the external face and its connections are not disregarded, the dual graph is obviously a connected graph. Therefore, we need to prove that the disregard does not cause a break in the connectivity. Now we assume that the dual graph is not a connected graph. Consider only all edges and vertices adjacent to the external face in the original graph. This graph is referred to as an external graph. The external graph is obviously a connected graph because our target knot is created by only one rope. If the external graph is equivalent to one loop, the dual graph is obviously a connected graph.

Because the external graph is not equivalent to one loop, we assume that two loops exist (See Fig. 5.6). In this case, two loops must be connected to each other by a

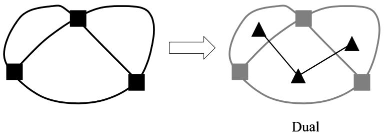


Figure 5.5: Dual graph of a knot graph

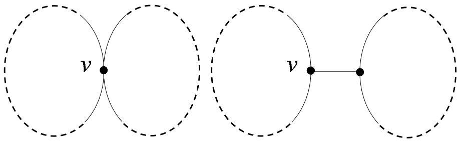


Figure 5.6: Necessary condition for disconnect of a dual graph

vertex v or one edge e adjacent to two vertices v and v' as shown in Fig. 5.6. Consider the process of converting a knot projection into P-data. When following a knot from the selective end, the following three cases are possible:

1. The case to enter and exit from a vertex v
2. The case to enter from a vertex v and stop at the end inside the loop
3. The case to begin from the end inside the loop and exit from a vertex v

Let a vertex v have two numbers i and j, where $i < j$. In the first case, each vertex inside the loop is k-th encountered, where $i < k < j$. This contradicts that P-data which is obtained from a knot projection corresponding to the original graph is irreducible. In the second case, each vertex inside the loop is k-th encountered, where $1 \leq k < j$, and $j \neq n(P)$. This also contradicts. In the third case, each vertex inside the loop is k-th encountered, where $i < k \leq n(P)$, and $i \neq 1$. This also contradicts. In the same way, in the case that more than two loops exist, we can introduce the contradiction that the original graph is not irreducible.

□

Proposition 8 *A graph given by applying the barycentric subdivision to a knot graph is a three-connected graph.*

The graph as shown at the bottom in Fig. 5.7 is the result of applying the barycentric subdivision to the graph shown at the top left in the figure. The subdivided graph is obtained by performing the following steps (See Fig. 5.7):

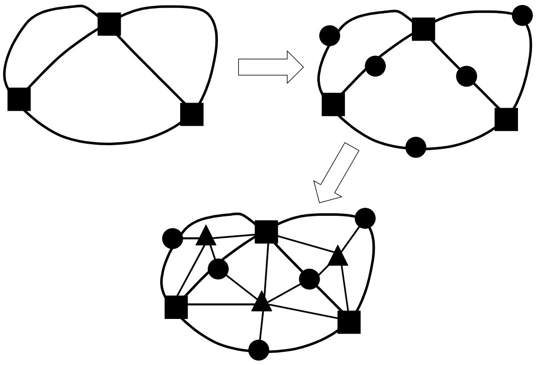


Figure 5.7: Barycentric subdivision

1. Append a vertex to the middle of each edge and connect it to the two vertices adjacent to the edge. In this time, the original edge is removed.

2. Append a vertex to the barycenter of each face except for the external face and connect it to all vertices on the boundary of the face.

Proof

Now we try to judge the connectivity of any two vertices in a graph G' which is made from the subdivided graph by removing two vertices. When two vertices are on the same face of the original graph, a path between the two vertices always exists for any G', because the graph which corresponds to one of the faces of the original knot graph is obviously a wheel graph, which is a three-connected graph. Next consider the case where they are not on the same face. Let the two vertices be on faces F_1 and F_n. From Proposition 7, a sequence $F_1 \to \cdots \to F_n$ is obtained, where F_i is adjacent to F_{i+1} with a shared edge for all i. Because each face is a wheel graph, that is, a three-connected graph, and three entrances exist between the two adjacent faces, a path between these two vertices always exist for any G'. Therefore, the subdivided graph is a three-connected graph.

□

Proposition 9 *A knot graph obtained from irreducible P-data is uniquely embedable into a sphere S^2. The way to embed is topologically unique except for its reverse.*

Proof

From Proposition 8, a graph obtained by applying the barycentric subdivision to the knot graph is a three-connected graph and obviously a planar graph. Unique embedability of a planar three-connected graph has already been proved by Whitney[71].

□

Fortunately, we resolve the ambiguity of the reverse using an attribute of an intersection. We obtain a knot projection from irreducible P-data by performing the following steps:

1. Extract all faces by searching shorter loops using the breadth first search from a knot graph. Each edge is twice employed as a component of a face.

2. Select one of the faces as the external face.

3. Apply the barycentric subdivision.

4. Embed it using the algorithm proposed in [70].

There is one caution on the process of extracting all faces as follows: Consider the vertex which has four adjacent edges. Let the vertex be corresponding to i-th and j-th intersections. At this time, these edges are $i-1$-th, i-th, $j-1$-th, and j-th segments. Loops including $i-1$-th and i-th segments do not compose a face, because one strand always goes across the other strand in any intersection as mentioned above. In the same manner, loops including $j-1$-th and j-th segments do not compose a face.

5.2 Definition of Movement Primitives

We define movement primitives as follows:

- They move only one segment at a time.

- They directly change one P-data to another P-data without any intermediate P-data.

We employ the Reidemeister moves[69] as such movement primitives. Reidemeister moves are employed to investigate characteristics of a tangled loop in the knot theory.

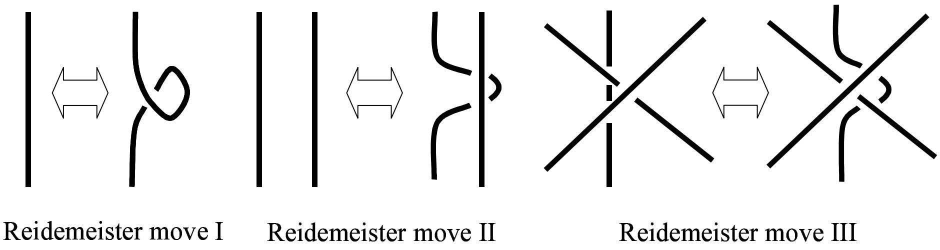


Figure 5.8: Reidemeister moves

5.2.1 Reidemeister Moves

Two knots are equivalent if one can be deformed to the other without cutting the string. Note that a knot is allowed to stretch and to shorten.

Reidemeister proved that any equivalent knot can be obtained from the current knot by stretching, shortening and finite repetitions of three types of moves referred to as Reidemeister moves[69]. Note that stretching and shortening do not change P-data, but these three types of Reidemeister moves do. Figure 5.8 shows the Reidemeister moves.

The Reidemeister move I adds or removes one intersection by creating or destroying a simple loop. The Reidemeister move II adds or removes two intersections, while one strand crosses to the other. The Reidemeister move III allows a strand to be moved to the other side of a crossing. Therefore, the number of intersections is not changed.

Note that the algorithm to effectively search a sequence of Reidemeister moves to translate one knot projection to another equivalent one has not yet been designed. However, it is possible to determine that two given knot projections are not equivalent using a so-called *polynomial invariant*[11].

5.2.2 Cross Move

A knot in the knot theory is a closed curve and does not have open ends. However, we have to deal with an open curve. An open curve is not considered to be a knot in the knot theory. Therefore, we additionally define the Cross move as a movement primitive. The Cross move is an operation in which one end of the rope crosses some segment as shown in Fig. 5.9 and adds or removes one intersection.

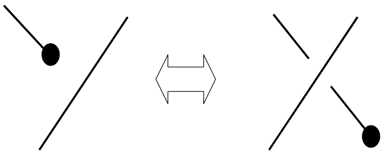


Figure 5.9: Cross Move

5.2.3 Sufficiency of the Movement Primitives

Every P-data transition can be divided into the following two types:

1. An end of the knot crosses another segment.

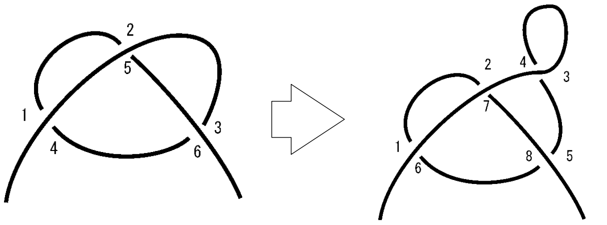
2. No end crosses any segment.

In the first case, every transition can be realized by the Cross move. In the second case, every transition can be realized by several repetitions of Reidemeister moves and this is easily introduced from Reidemeister's proof about knot-equivalence. However, there is no guarantee that such a transition is realized by only one of the three Reidemeister moves (See the definition of movement primitives). That guarantee is very important. If it is not satisfied, our proposed method cannot recognize such a move, even if the move is an essential movement primitive for a knot-tying task. Although the movement primitive may be discovered, it is easy to improve the system to recognize and execute the primitive, because it is equivalent to several repetitions of three types of Reidemeister moves.

Fortunately, various kinds of knot-tying can be represented by a sequence of these four movement primitives (Cross move and Reidemeister moves I, II, and III). For our current analysis (overhand knot, eight knot, bowline knot, harness hitch, bow tie, single loop bow, two-half knot, and taut-line hitch), the Reidemeister move III has not been employed. Because it is a redundant move when tying a knot, it seldom appears.

5.3 Selection of Corresponding Movement Primitives from P-data Transitions

In this section, we describe a method for selecting the corresponding one of three types of Reidemeister moves and a Cross move from P-data transitions. In the following, P_t denotes the P-data obtained from a knot projection at time t. Without



1	2	3	4	5	6
4	5	6	1	2	3
3	1	2	4	2	1

1	2	3	4	5	6	7	8
6	7	4	3	8	1	2	5
3	1	1	2	2	4	2	1

Figure 5.10: Change of P-data under the Reidemeister move I

any loss of generality, P-data before the transition has a fewer intersections than the one after the transition, that is,

$$n(P_{t-1}) \leq n(P_t).$$

Now we assume that a selective end in the process of converting a knot projection into P-data is not changed before and after the transition.

5.3.1 Reidemeister move I

Figure 5.10 shows an example of the Reidemeister move I. After applying the Reidemeister move I to the knot projection at time $t-1$, one intersection is added in the knot projection at time t. Therefore, Equation (5.2) must be satisfied.

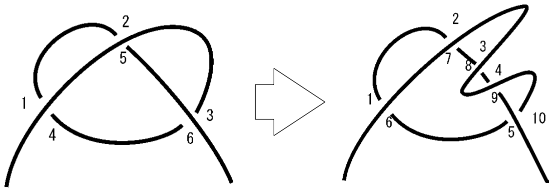
$$n(P_t) = n(P_{t-1}) + 2 \tag{5.2}$$

Now we assume that the Reidemeister move I is applied to the i-th segment (the third segment in the case as shown in Fig. 5.10). The additional intersection has two continuous numbers. Therefore, Equation (5.3) must be satisfied.

$$\sigma(i|P_t) = i + 1 \tag{5.3}$$

Because states of other intersections are not changed, when we remove the i-th and $(i+1)$-th columns from P_t and reorder the intersection numbers, i.e., subtract the intersection numbers from $i+2$ to $n(P_t)$ by two, P_t must become equal to P_{t-1}. Such removing and reordering is defined as $R_I(P_t, i)$.

Conversely, if we find i ($1 \leq i \leq n(P_{t-1}) + 1$) which satisfies $\sigma(i|P_t) = i + 1$ and $R_I(P_t, i) = P_{t-1}$, we conclude that the Reidemeister move I occurs at the i-th



1	2	3	4	5	6
4	5	6	1	2	3
3	1	2	4	2	1

1	2	3	4	5	6	7	8	9	10
6	7	8	9	10	1	2	3	4	5
3	1	3	1	2	4	2	4	2	1

Figure 5.11: Change of P-data under the Reidemeister move II

segment at time $t-1$. Note that several ways may exist to realize a desired P-data transition. In the case as shown in Fig. 5.10, we find that the Reidemeister move I occurs at the third segment at time $t-1$.

5.3.2 Reidemeister Move II

Figure 5.11 shows an example of the Reidemeister move II. After applying the Reidemeister move II to the knot projection at time $t-1$, two intersections are added in the knot projection at time t. Therefore, Equation (5.4) must be satisfied.

$$n(P_t) = n(P_{t-1}) + 4 \tag{5.4}$$

Now we assume that the Reidemeister move II is applied to the i-th and j-th segments (the third and sixth segments in the case as shown in Fig. 5.11). In the process of converting a knot projection into P-data, the two additional intersections are continually encountered. Therefore, Equation (5.5) or Equation (5.6) must be satisfied, where $1 \leq i < j \leq n(P_{t-1}) + 1$.

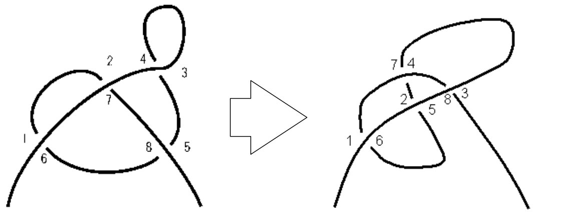
$$\sigma(i|P_t) = j + 2 \quad \cap \quad \sigma(i+1|P_t) = j + 3 \tag{5.5}$$

$$\sigma(i|P_t) = j + 3 \quad \cap \quad \sigma(i+1|P_t) = j + 2 \tag{5.6}$$

Furthermore these relative vertical positions (over/under) must be the same and these signs must be different. Therefore, Equation (5.7) must be satisfied.

$$|\mathrm{attr}(i|P_t) - \mathrm{attr}(i+1|P_t)| = 2 \tag{5.7}$$

Because states of other intersections are not changed, when we remove the i-th, i+1-th, j+2-th, and j+3-th columns from P_t and reorder the intersection numbers,



1	2	3	4	5	6	7	8
6	7	4	3	8	1	2	5
3	1	1	2	2	4	2	1

Type A

1	2	3	4	5	6	7	8
6	5	8	7	2	1	4	3
3	1	1	2	2	4	1	2

Type H

Figure 5.12: Change of P-data under the Reidemeister move III

i.e., subtract the intersection numbers from $i+2$ to $j+1$ by two and subtract the intersection numbers from $j+4$ to $n(P_t)$ by four, P_t must become equal to P_{t-1}. Such removing and reordering is defined as $R_{II}(P_t, i, j)$.

Conversely, if we find i, j $(1 \leq i < j \leq n(P_{t-1})+1)$ which satisfy Equation (5.5) or (5.6), Equation (5.7), and $R_{II}(P_t, i, j) = P_{t-1}$, we conclude that the Reidemeister move II occurs between the i-th and j-th segments at time $t-1$. We calculate $R_{II}(P_t, i, j)$ toward i and j $(1 \leq i < j \leq n(P_{t-1})+1)$ in order. Fortunately j satisfies

$$j = \frac{\sigma(i|P_t) + \sigma(i+1|P_t) - 5}{2}.$$

Therefore, we can select the Reidemeister move II by $O(n(P_{t-1}))$-time[2]. In the case as shown in Fig. 5.11, we find that the Reidemeister move II occurs between the third and sixth segments, between the first and fourth segments, or between the second and fifth segments at time $t-1$. Figure 5.11 looks occurring the first case. Of course, the last two cases can realize such a transition.

5.3.3 Reidemeister Move III

Figure 5.12 shows an example of the Reidemeister move III. After applying the Reidemeister move III to the knot projection at time $t-1$, no intersection is added in the knot projection at time t. Therefore, Equation (5.8) must be satisfied.

$$n(P_t) = n(P_{t-1}) \tag{5.8}$$

[2] In addition, comparison of two sets of P-data requires $O(n(P_{t-1}))$-time.

The focus is on three important segments for the Reidemeister move III. Now, we name them Segment A, B, and C by the following rules:

- Segment A goes over the other two.
- Segment C goes under the other two.
- Segment B is the remaining segment.

Let Segment A, B, and C correspond to the i-th, j-th, and k-th segments, respectively, where $2 \leq i, j, k \leq n(P_t) - 1$. These three segments compose three intersections. Each intersection has two numbers of the set $\{i-1, i, j-1, j, k-1, k\}$, and does not have any numbers that the others have.

Proposition 10 $\sigma(i) \neq i - 1 \cap \sigma(j) \neq j - 1 \cap \sigma(k) \neq k - 1$

Proof
Each intersection is composed of two of the three segments.

□

Corollary 1 $\sigma(i-1) = k-1$ *or* k*, when* $\sigma(i) = j-1$ *or* j*. Similarly,* $\sigma(i-1) = j-1$ *or* j*, when* $\sigma(i) = k-1$ *or* k*.*

Proof
We assume that the first condition is not satisfied. Then $\sigma(i-1) = k-1 \cap \sigma(i) = k$ or $\sigma(i-1) = k \cap \sigma(i) = k-1$. From the fact that each intersection does not have any numbers that the others have, $\sigma(j-1) = j$ must be satisfied. However that contradicts Proposition 10. In the same way, the second condition can be easily proved.

□

From Proposition 10 and Corollary 1, relations between these segments can be classified into eight types as follows:

A. $\sigma(i-1) = j-1$, $\sigma(i) = k-1$, $\sigma(j) = k$
B. $\sigma(i-1) = j-1$, $\sigma(i) = k$, $\sigma(j) = k-1$
C. $\sigma(i-1) = j$, $\sigma(i) = k-1$, $\sigma(j-1) = k$
D. $\sigma(i-1) = j$, $\sigma(i) = k$, $\sigma(j-1) = k-1$
E. $\sigma(i-1) = k-1$, $\sigma(i) = j-1$, $\sigma(j) = k$
F. $\sigma(i-1) = k$, $\sigma(i) = j-1$, $\sigma(j) = k-1$
G. $\sigma(i-1) = k-1$, $\sigma(i) = j$, $\sigma(j-1) = k$
H. $\sigma(i-1) = k$, $\sigma(i) = j$, $\sigma(j-1) = k-1$

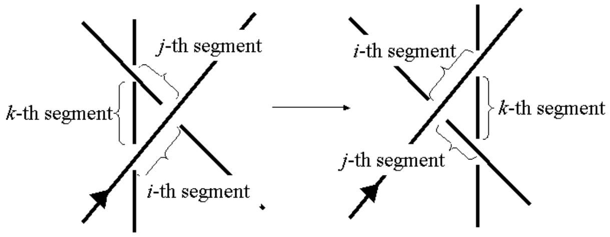


Figure 5.13: Characteristics of Reidemeister Move III

Proposition 11 *We assume that the Reidemeister move III changes P-data P_{t-1} to P-data P_t. The orders of the three important segments in the state P_{t-1} are equal to the orders in the state P_t, respectively.*

Proof
Trivial.

□

Proposition 12 *If the $i-1$-th intersection is composed of the i-th and j-th segments before the translation, the $i-1$-th intersection is composed of the i-th and k-th segments after the transition. Similarly, if the i-th intersection consists of the i-th and j-th segments before the transition, the i-th intersection is composed of the i-th and k-th segments after the transition. Even if the role of i, j, and k shifts, this proposition is satisfied.*

Proof
Trivial from Fig. 5.13. We assume that we follow a knot toward the arrow direction as shown in Fig. 5.13 in the process of converting a knot projection into P-data. From Proposition 11, the three segments have the same encountering order before and after the transition. However, the crossed order is changed before and after the transition.

□

Corollary 2 $\sigma(i-1|P_{t-1}) \neq \sigma(i|P_t)$. *This proposition is satisfied, even if j or k is used as a substitute for i.*

Proof
We assume $\sigma(i-1|P_{t-1}) = \sigma(i|P_t) = l$, where l is equal to one of $\{j-1, j, k-1, k\}$. From the commutative law, $\sigma(l|P_{t-1}) = i-1$ and $\sigma(l|P_t) = i$ must be satisfied. This means the l-th intersection is composed of the i-th and l (or $l+1$)-th segments before and after the transition. That contradicts Proposition 12.

□

Proposition 13 *Among the eight types mentioned above, the following four transitions are permitted.*

- *Type A* ↔ *Type H*
- *Type B* ↔ *Type G*
- *Type C* ↔ *Type F*
- *Type D* ↔ *Type E*

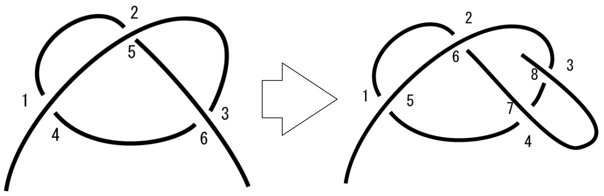
Proof
Trivial from Proposition 12 and Corollary 2.

□

Because states of another intersections are not changed, when we remove the $i-1$-th, i-th, $j-1$-th, j-th, $k-1$-th, and k-th columns from P_t and P_{t-1}, P_t becomes equal to P_{t-1}. Such removing is defined as $R_{III}(P, i, j, k)$. Therefore, the algorithm to decide if the Reidemeister move III is applied or not is as follows:

1. Search i-th, j-th, and k-th segments which compose a triangle before and after the transition
2. Decide on the type for these three segments before and after the transition
3. Check the legality of the transition based on Proposition 13
4. Check $R_{III}(P_t, i, j, k) = R_{III}(P_{t-1}, i, j, k)$

In the case as shown in Fig. 5.12, we find that the Reidemeister move III occurs among the third, fifth and eighth segments at time $t-1$.



1	2	3	4	5	6
4	5	6	1	2	3
3	1	2	4	2	1

1	2	3	4	5	6	7	8
5	6	8	7	1	2	4	3
3	1	4	2	4	2	1	3

Figure 5.14: Change of P-data under the Cross move

5.3.4 Cross Move

Figure 5.14 shows an example of the Cross move. After applying the Cross move to the knot projection at time $t-1$, one intersection is added in the knot projection at time t. Therefore, Equation (5.9) must be satisfied.

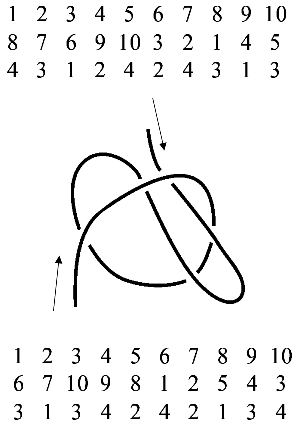
$$n(P_t) = n(P_{t-1}) + 2 \tag{5.9}$$

Because in the Cross move, one of two ends crosses over or under a segment, the additional intersection is a neighbor to the first or last segment after the transition. Therefore, it is easy to find out to which segments the cross move is applied from $\sigma(1|P_t)$ or $\sigma(n(P_t)|P_t)$.

Now we assume that the Cross move is applied to the selective end and the i-th segment. The additional intersection has the number $i+1$. Therefore $i+1 = \sigma(1|P_t)$ must be satisfied. When we remove the first and $i+1$-th columns from P_t and reorder the intersection numbers, i.e., subtract the intersection numbers from two to i by one, from $i+2$ to $n(P_t)$ by two, P_t must become equal to P_{t-1}. Such removing and reordering is defined as $C_s(P_t)$.

Next, we assume that the Cross move is applied to the other end and the j-th segment. The additional intersection has the number j. Therefore $j = \sigma(n(P_t)|P_t)$ must be satisfied. In the same fashion, when we remove the last and $\sigma(n(P_t)|P_t)$-th columns from P_t and reorder the intersection numbers, i.e., subtract the intersection numbers from $j+1$ to $n(P_t)-1$ by one, P_t must become equal to P_{t-1}. Such removing and reordering is defined as $C_e(P_t)$.

Conversely, if $C_s(P_t) = P_{t-1}$ ($C_e(P_t) = P_{t-1}$) is satisfied, we conclude that the Cross move occurs between the selective end and $(\sigma(1|P_t) - 1)$-th segment (the other end and $\sigma(n(P_t)|P_t) - th$ segment).



1	2	3	4	5	6	7	8	9	10
8	7	6	9	10	3	2	1	4	5
4	3	1	2	4	2	4	3	1	3

1	2	3	4	5	6	7	8	9	10
6	7	10	9	8	1	2	5	4	3
3	1	3	4	2	4	2	1	3	4

Figure 5.15: Two equivalent P-data

5.3.5 Ambiguity With Respect to Selective End

Until the previous section, we assumed that a selective end in the process of converting a knot projection into P-data is not changed before and after the transition. Now we consider eliminating this assumption.
As shown in Fig. 5.15, the knot projection is converted into two different P-data based on the difference of a selective end. We define these two sets of P-data as equivalent P-data sets. Between the P-data P and the equivalent P-data P_{eq}, the following relation should be satisfied:

Proposition 14 $\sigma(N(P) - i + 1|P_{eq}) = N(P) - \sigma(i|P) + 1$

Proof
Considering the conversion into P-data, the i-th intersection when converting into P-data P is the $(N(P) - i + 1)$-th intersection when converting into P-data P_{eq}.

□

Proposition 15 $attr(N(P) - i + 1|P_{eq}) = attr(i|P)$

Proof
Trivial. Of course, the vertical position, i.e., over or under, of the i-th intersection when converting into P-data P is equal to that of the $(N(P)-i+1)$-th intersection

when converting into P-data P_{eq}. Furthermore, the sign of the i-th intersection when converting into P-data P is equal to that of the $(N(P)-i+1)$-th intersection when converting into P-data P_{eq}. The direction vector of every strand turns back, but

$$(\vec{l}_{over} \times \vec{l}_{under}) \cdot \vec{e_z} = (-\vec{l}_{over} \times -\vec{l}_{under}) \cdot \vec{e_z}$$

is always satisfied.

□

Using Proposition 14 and 15, we can easily convert P-data into equivalent P-data. Because P-data is reversible to a knot projection as proved in Section 5.1.3, we can resolve the ambiguity by applying the rule for selecting corresponding movement primitives to both P-data and the equivalent P-data.

5.4 Examples

5.4.1 Reversibility of P-data

We show the result of reversibility of P-data by converting a knot projection into a knot projection again through P-data conversion. First, we convert the knot projection as shown at the top left in Fig. 5.16 into P-data. As a result, we obtain the following P-data:

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 3 & 4 & 3 & 4 & 3 & 4 \end{matrix}$$

Next, we convert such irreducible P-data into a knot projection again as the algorithm described in Section 5.1.3. First, we convert P-data into a knot graph, which has three (= 6/2) vertices $\{v_1, v_2, v_3\}$ and five edges $\{e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_3\}, e_3 = \{v_3, v_1\}, e_4 = \{v_1, v_2\}, e_5 = \{v_2, v_3\}\}$.
We extract the following four faces by searching a shorter loop using the breadth first search:

$$f_1 = \{e_1, e_4\}, f_2 = \{e_2, e_5\}, f_3 = \{e_1, e_5, e_3\}, f_4 = \{e_4, e_2, e_3\}$$

From the caution about the search in Section 5.1.3, we should not regard $\{e_1, e_2, e_3\}$ and $\{e_3, e_4, e_5\}$ as a face, because these sets include e_1 and e_2, or e_4 and e_5, simultaneously.

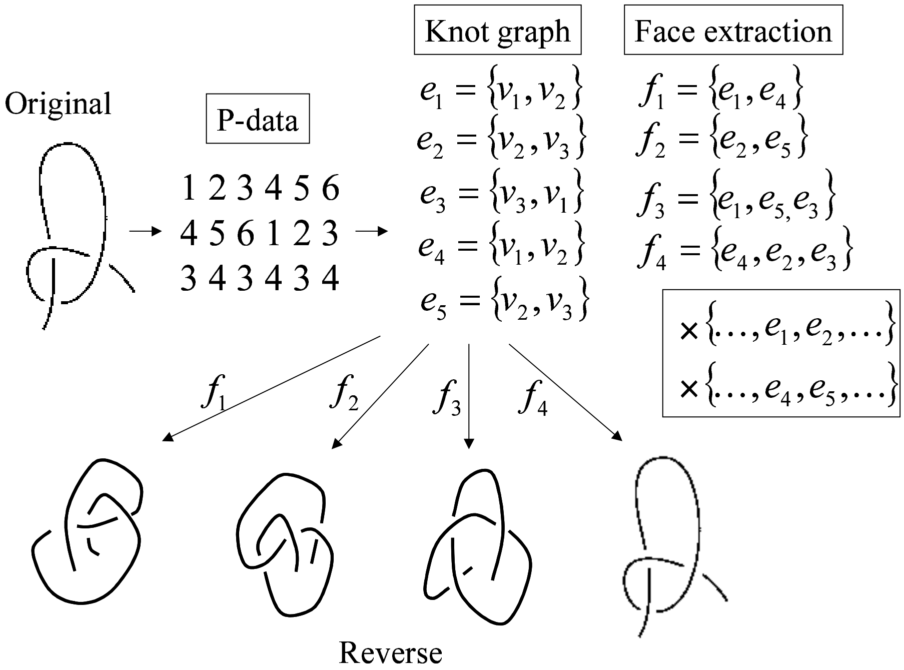


Figure 5.16: Convert a knot projection into a knot projection again through P-data conversion

When we select a face f_4 as an external face, we obtain a knot projection as shown at the bottom right in Fig. 5.16, through applying the barycentric subdivision and embedding it using the algorithm proposed by [70]. That is an equivalent knot projection to the original one. By selecting another three faces as external faces, we obtain anther three knot projections, respectively. Note that these three projections look different from the original knot, because these knots are illustrated in R^2 space. Of course, they are equivalent in S^2 space.

5.4.2 Recognition of Movement Primitives

Figure 5.17 shows the result of the analysis of the bowline knot. Note that we correctly select the selective end that is shown by the black point in Fig. 5.17 in every knot projection. First, the system converts each knot projection into P-data. The number array under each knot projection is the P-data. Next, the system recognizes the knot-tying task as a sequence of movement primitives that have been already defined. In each transition, the number of columns increases by two; therefore, the Cross move ($C_s(*)$ (to the selective end) or $C_e(*)$ (to the other end)

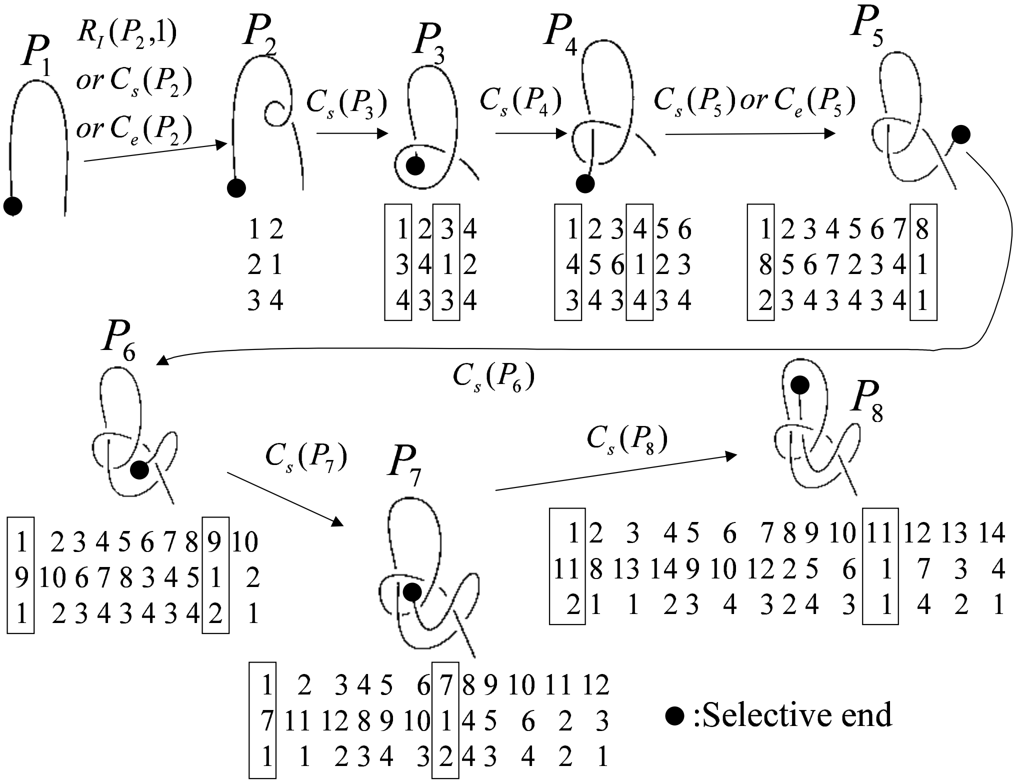


Figure 5.17: Analysis of the bowline knot

in the figure) or the Reidemeister move I ($R_I(*)$ in the figure) may occur.

In all transitions, the system can select the correct movement primitives. Among them, in the first and the fourth transitions, the system selects more than one movement primitives. Actually, in the fourth transition, the Cross move $C_s(P_4)$ is more appropriate by observing the transformation of a knot projection. Because P-data does not include parametric information, the system cannot select one of the two movement primitives. However, both of them can realize the transition; therefore, this ambiguity is not a problem. In the first transition, that is the same.

Figure 5.18 shows the efficiency of the system with respect to the ambiguity of selection of the selective end. In this example, we select the wrong one of two ends as the selective end in the third, sixth, and seventh knot projections. The wrong selection usually leads to the result that the system selects no movement primitive in spite of the fact that a movement primitive to realize the transition exists.

In the fifth transition, the system selects no movement primitive from the transition

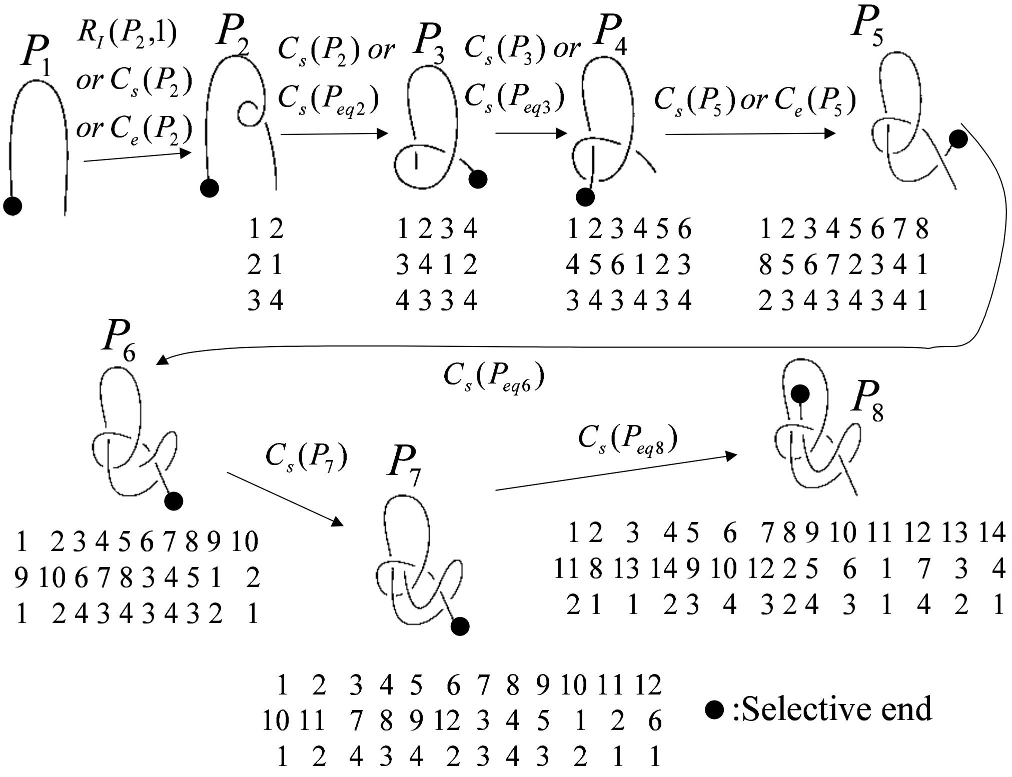


Figure 5.18: Analysis of the bowline knot when some selective ends are not correctly selected

P_5 to P_6, however selects the Cross move $C_s(P_{eq6})$, which is a correct one, from the transition P_5 to P_{eq6}, which is equivalent P-data of P_6. As a result, we find that selection of the selective end is wrong in the sixth P-data.

In the sixth transition, the system selects a correct movement primitive from the transition P_6 to P_7; however, the selection was wrong in the sixth P-data. Therefore, we find that the selection is wrong in the seventh P-data.

In the second transition, the system cannot decide if the selection is correct or wrong, because P_3 is equal to P_{eq3}, which is the equivalent P-data of P_3.

Figure 5.19 shows the result of the analysis of the bow knot. Now, we correctly select the selective end in every knot projection. In this example, the Reidemeister move II ($R_{II}(*)$ in the figure) is employed to realize the fourth, sixth, and seventh transitions. It is certain that these transitions can be realized by several repetitions of the Cross move. However, it is natural that they, especially the seventh

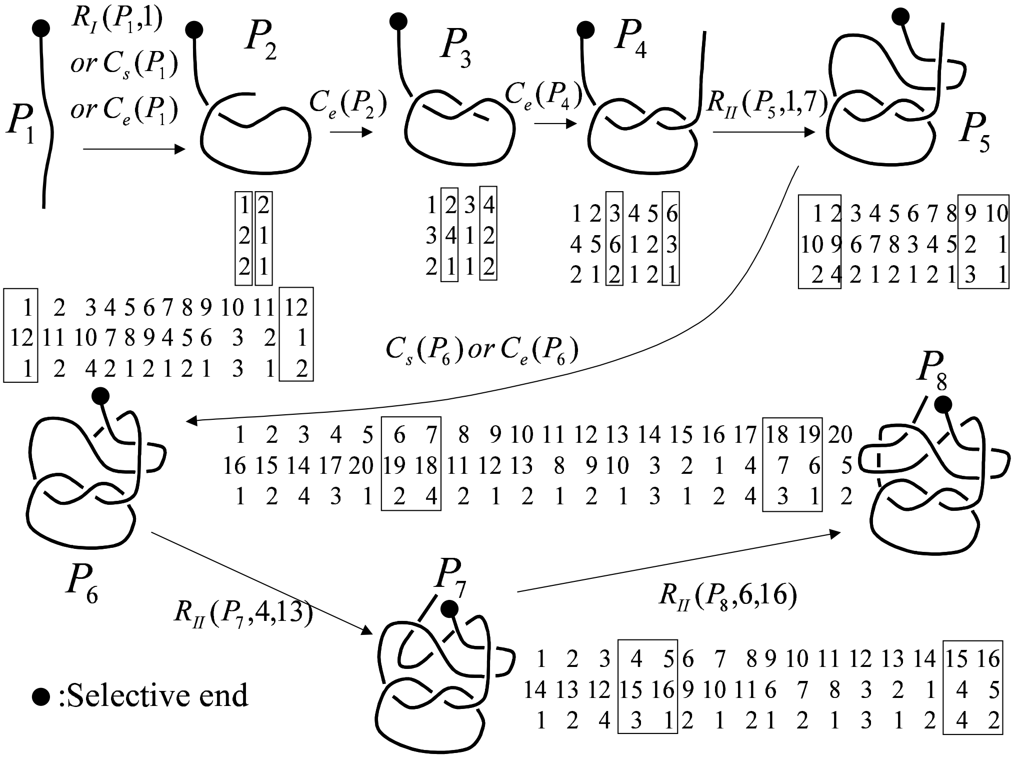


Figure 5.19: Analysis of the bow knot

transition, are realized by the Reidemeister move II. The method in [49] considers only the Cross move, that is, it must tie the bow knot in an unnatural way.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

In this thesis, we have proposed novel methods that enable robots to recognize tasks for manipulating various kinds of objects. First, we propose a method for recognizing assembly tasks using two polyhedral objects. In Chapter 2, we describe how to observe and recognize them. Concretely speaking, we define *Motion Degrees of Freedom* (DOF) to index the quality of the constrained motion. Using them enables us to efficiently define sufficient movement primitives for assembly tasks. Note that we describe the execution in Appendix A.

Next, in Chapter 3, we propose a mathematical tool for improving calculation of the constrained motion using the second order approximation of the motion. As a result, the tool is also able to deal with the curvature information, i.e., curved objects.

In Chapter 4, we propose a method for estimating the parameters of a *joint*. The estimation is essential for manipulating linkages which are connected by a joint, for example, rotating a doorknob, pulling open a drawer, inserting a screw with a screw driver, etc. Each task is a very common everyday task.

In Chapter 5, we propose a method for recognizing knot-tying tasks using one rope. We introduce the P-data representation, which is employed to represent a knot state, and define sufficient movement primitives for the tasks.

Below, we concretely conclude each method.

6.1.1 Assembly Tasks using Two Rigid Polyhedral Objects

In this thesis, we have proposed a method for observing and recognizing assembly tasks using two rigid polyhedral objects. First we propose a method for recognizing assembly tasks from transitions of contact relations. For the purpose of the recognition, we improve motion DOFs in only translation defined in [6] to deal with rotation. We describe a method for calculating such DOFs. And we define DOFs of axis directions for improving the recognition.

Next, using the idea of practical DOF-transitions, we define sufficient sub-skills which are movement primitives for assembly tasks, and critical transitions which are the important transitions for a smooth process of an assembly task. Then we propose a method for selecting the corresponding sub-skills and extracting critical transitions using DOF-transitions.

Then we describe a method for observing the task under the existence of vision errors. To be robust to vision errors, we propose a method for correcting them using roughly estimated contact relations and validity of the transitions. And we propose a method for correcting the errors by applying the calculation of the optimal trajectories. We describe the calculation in Appendix A. Although the method is useless for recognizing the tasks, it is very useful for extracting information about the unconstrained motion. Such motion may include the technique for well executing the task, we believe.

Finally, we examine the effectiveness of our method by recognizing a peg-insertion operation using it. We think that, because our method can deal with a peg-insertion operation where various kinds of contact relations appear, our method is apropos for various kinds of assembly tasks.

6.1.2 Second Order approximation

In this thesis, we have proposed a method for formulating and calculating maintaining and legal displacement using the second order approximation of a function which represents the relationship between the displacement and the distance between two of the following three object primitives: a vertex, an edge, and a face.

First, we formulate the second order approximation for basic point-contact primitives as follows: vertex-face, edge-edge, edge-face, face-vertex, face-edge, and face-face point-contacts. These contact primitives play an important role in formulating the maintaining and legal displacement of any basic contact relations. A contact relation which appears is usually a basic contact relation.

Second, we describe that a line- or face-contact primitive can be regarded as an infinite number of basic point-contact primitives. However, it cannot always be represented as an equivalent finite number of basic point-contact primitives. Through the formulation, we also illustrate the necessity of formulation of some special point-contact primitives, which include at least one singular object primitive.

Then we propose a method for calculating the displacement. All target equations look very different from one another; however, all of them are generally represented as Equation (3.70) in Chapter 3. We calculate the displacement by linearizing a quadratic form through converting its canonical form. Therefore, in the case where every quadratic form cannot be linearized, our method cannot calculate the displacement. To establish a method to completely calculate the displacement is now an outstanding problem. However, we believe that our method proposed in this thesis is sufficient for calculating the maintaining displacement from experimental results.

Finally, we examine the effectiveness of our proposed method by applying it to two examples. First, we describe how it is applied to an example to determine grasping stability. Because the purpose of the method is not to index the displacement but rather, to calculate the displacement itself, we analyze grasping stability using the idea of maintenance of a contact relation and the range (DOF) of the legal displacement. Our intuition agrees with the calculation result.

Second, we describe how we apply the method to an example of generating a contact relation graph. The graph is very useful for planning assembly tasks, as mentioned above. In this example, we propose a method for generating the desired trajectory using the second order approximation. The example shows the applicability of the second order approximation.

6.1.3 Extracting Parameters of Joints

We have proposed a method for estimating the parameters of a joint from observation, given a type of the joint and several relative configurations between two linkages which are connected by the joint. Of course, the observation includes some errors.

First, we concentrate on three types of joints as follows: a prismatic joint, a revolute joint, and a screw joint. These three are so common that they often appear. We describe how we formulate the conditions which the parameters must satisfy and estimate them by minimizing the error function which we define.

Next, we examine the effectiveness of our method through simulation experiments and actual experiments. In the simulation experiment, we illustrate the statistical characteristics of our method. As a result, the inaccuracy is about inversely proportional to the amount of translation or rotation and square of the accuracy is proportional to the number of the configurations. The result is adequate from a statistical viewpoint.

Finally, we propose a method for estimating the parameters of any types of joints with respect to an object orientation. The difficulty of estimating the parameters mainly causes the difficulty of estimating the parameters with respect to an object orientation. Therefore, it becomes possible to estimate the parameters of any types of joints using our proposed method, we believe.

6.1.4 Knot Tying Tasks with One Rope

We have propose a method to recognize knot-tying tasks using one rope. By applying the knot theory, we introduce a method for representing a knot state in a P-data representation, define four sufficient movement primitives, and describe a method for selecting the corresponding one from P-data transitions.

First, we introduce a knot projection and P-data representation to represent an abstract information of a knot projection. A P-data representation has several desirable characteristics as follows:

- P-data representation is an abstract data structure for representing only the topological information. It ignores a slight difference in the position of an intersection, which is unimportant in the process of knot-tying tasks.
- A knot projection can be reversed from P-data representation. That is important in solving a so-called path planning problem.

We prove such a reversibility by showing a method for converting irreducible P-data into a planar three-connected graph.

Next, we define the following movement primitives for knot tying tasks: the Reidemeister move I, II, and III, and the Cross move. From Reidemeister's proof about knot-equivalence, a transition between two equivalent knots which are tangled loops can be realized by shortening, stretching, and repeating the Reidemeister moves. The proof guarantees that a transition where any end of a rope does not cross over or under a strand can be realized by repetitions of them. However, our target rope has two ends and the Cross move is necessary to realize the transition where one of them crosses over or under a strand.

Then, we propose a method for selecting the corresponding movement primitive from a P-data transition. A P-data transition is uniquely determined from the type and parameters (for example, to which segment is the movement primitive applied?) of the movement primitive. Conversely, we can select the corresponding movement primitive by a whole search against all candidates. At the same time, we also solve the problem of ambiguity of selection of the selective ends using equivalent P-data.

Finally, we show examples of analysis of knot-tying tasks. Our method selects the corresponding movement primitives, even if selection of the selective end is wrong. Especially, in a bow knot, which requires Reidemeister move II, our method can recognize such a knot-tying task.

6.2 Future Directions

In this thesis, we have described methods for recognizing the following tasks:

- Assembly tasks using two rigid polyhedral objects
- Tasks for manipulating linkages connected by a join
- Knot-tying tasks using one rope

It is very meaningful to have proposed methods to recognize these tasks, because these tasks are often executed every day. However, there are various other kinds of manipulation tasks. We should propose a method to deal with such tasks. Among them, it would be easy to design a method to deal with assembly tasks using not only polyhedral, but also curved objects using the second order approximation. We believe that motion DOFs can be calculated using the second order approximation.

In this thesis, each system deals only with its target task. It is very important to integrate such systems in order to generate a general system that is able to deal with all kinds of manipulation tasks, we believe.

With regard to the integration, it is easy to integrate systems that deal with the first and the second tasks because it is very easy to formulate the constrained motion, i.e., to calculate motion DOFs (For example, Mason's works[44]), if the parameters of the joint are known.

Although we do not consider states of joints in this thesis, some joint may have several states. For example, consider the case of rotating a doorknob. There are the following two states: One is that a doorknob can rotate both clockwise and counter-clockwise. The other is that a doorknob can rotate either clockwise or counter-clockwise.

By representing the task of rotating a doorknob as transitions between the two states, we can execute the rotation using the sub-skills. In this case, the DOF with respect to the axis direction of rotation is a maintaining DOF in the former state and the DOF is a detaching DOF in the latter state. In short, opening a door can be executed by a make-contact sub-skill in rotation because a DOF-transition from a maintaining DOF to a detaching DOF in rotation occurs. Through the integration, the system can share several movement primitives.

However, with regard to the case of turning a crank, such a DOF-transition does not appear, because the joint has only one state. To recognize such a task requires analyzing the unconstrained motion. The linear vision error correction enables such an analysis, we believe.

Then, we consider integrating systems that deal with the first and the third tasks. The integration is very difficult because movement primitives of each system can be employed for the target tasks only and movement primitives for manipulation between rigid and string-like deformable objects have not yet been determined. Such tasks are so applicable that we are ambitious to try the integration.

Appendix A

Sub-skill Implementation

In this appendix, we describe a method to implement sub-skills, which we define as movement primitives for assembly tasks. Of course, to implement them, one should consider intrinsic characteristics of a robot arm and its end effector.

Therefore, we first describe a method to calculate trajectories of the grasping object to realize the desired transitions. Such calculation is relatively independent of the characteristics. As mentioned above, the trajectory usually cannot be uniquely decided. Therefore, we first define the optimal trajectory and we propose a method to calculate the optimal trajectory using only a linear solution. Note that the trajectory is formulated as non-linear equations.

We set up the following assumptions:

1. The trajectory is locally represented by translation or rotation of which the axis direction is constant.

2. There is no obstacle on the optimal trajectory.

With regard to how to implement sub-skills on a robot in actuality, the first assumption is preferable. The second assumption means that we do not deal with the obstacle avoidance problem[5]. Although the assumption may be strict, it is possible to solve such a problem using the unconstrained motion obtained from the observation, we believe.

Next we illustrate the actual implementation on our test-bed[72]. Various basic ideas which we employ to implement sub-skills can be applied to another robot arm, we believe.

A.1 Preliminaries

As mentioned above, the legal infinitesimal displacement is formulated by Equation (A.1), where $[\mathbf{S}_0, \mathbf{S}_1]$ is a screw vector[1][53] which represents the displacement and $\mathbf{P}_i$ is a contact position of the i-th contact primitive.

$$\bigcap_{i}^{N} \bigcup_{j}^{M(i)} \mathbf{F}_{ij} \cdot \mathbf{S}_1 + (\mathbf{P}_i \times \mathbf{F}_{ij}) \cdot \mathbf{S}_0 \geq 0 \tag{A.1}$$

In assembly tasks, it is preferable to calculate trajectories which maintain some contact relation for robustness against execution errors. The derivative of such trajectories, i.e., the infinitesimal displacement must satisfy Equation (A.2).

$$\bigcap_{i}^{N} \bigcap_{j}^{M(i)} \mathbf{F}_{ij} \cdot \mathbf{S}_1 + (\mathbf{P}_i \times \mathbf{F}_{ij}) \cdot \mathbf{S}_0 = 0 \tag{A.2}$$

Converting $\cup$ into $\cap$ in the equation, Equation (A.2) is a simple system of simultaneous linear equalities. To simply denote the equation, we merge two $\cap$. As a result, Equation (A.2) is converted to Equation (A.3).

$$\bigcap_{i}^{n} \mathbf{F}_{i} \cdot \mathbf{S}_1 + (\mathbf{P}_i \times \mathbf{F}_{ij}) \cdot \mathbf{S}_0 = 0 \tag{A.3}$$

Note that the rank of Equation (A.3) is equal to a restricted DOF in all motions. Because a screw vector $[\mathbf{0}, \mathbf{S}_1]$ represents pure translation, Equation (A.4) which are obtained by substituting $\mathbf{S}_0 = \mathbf{0}$ to Equation (A.3) represents the translational infinitesimal displacement.

$$\bigcap_{i}^{n} \mathbf{F}_{i} \cdot \mathbf{S}_1 = 0 \tag{A.4}$$

Note that the rank of Equation (A.4) is equal to a restricted DOF in translation. Next, we obtain simultaneous linear equalities which include a term $\mathbf{S}_0$, but do not include a term $\mathbf{S_1}$ from Equation (A.3), by performing the following steps:

1. Search all linearly dependent minimum combinations of $\{\mathbf{F}_i\}$
2. Obtain the equalities by erasing a term $\mathbf{S}_1$ from equations which are included in the combinations.

[1] In this appendix, every displacement is calculated using the first order approximation only.

The equalities represent any axis directions in the rotational displacement to maintain a contact relation.

Concretely, let C be one of the linearly dependent minimum combinations. Because of the dependency, Equation (A.5) always satisfies, where $w_i \neq 0$ for all $i \in C$.

$$\sum_{i \in C} w_i \mathbf{F}_i = \mathbf{0} \tag{A.5}$$

Applying a dot product of $\mathbf{S}_1$ to both sides of Equation (A.5), Equation (A.6) is obtained.

$$\sum_{i \in C} w_i \mathbf{F}_i \cdot \mathbf{S}_1 = \mathbf{0} \tag{A.6}$$

By repeating to substitute Equation (A.6) to Equation (A.2) for all the combinations, we obtain Equation (A.7). As mentioned above, the equation represents any axis directions in the rotational displacement to maintain a contact relation.

$$\bigcap_{i}^{m} \mathbf{G}_i \cdot \mathbf{S}_0 = 0 \tag{A.7}$$

Note that the rank of Equation (A.7) is equal to a restricted DOF in rotation.

A.2 Calculate Optimal Trajectories

In this section, we propose a method to calculate the optimal trajectories given a valid transition from one contact relation to the other contact relation[2] and object configurations in each contact relation.

We assume that all restricted DOFs before the transition are not greater than all restricted DOFs after the transition. If the condition is not satisfied, we first calculate the trajectory to realize the inverse transition and then invert it. Note that there is no case that one of restricted DOFs before the transition is greater than the DOF after the transition and the other of restricted DOFs before the transition is less than the DOF after the transition, because of the validity of the transition.

Any position, direction, etc. are represented with respect to the coordinate system of the environmental object unless otherwise noted. We concentrate on calculating the trajectory of the grasping object with respect to the system.

As mentioned above, because of the validity of the transition, restricted DOFs in translation and rotation before the transition are not greater than the restricted DOFs after the transition. If restricted DOFs in rotation before and after the

[2] We describe a method to decide the validity in Chapter 2.

transition are equal to each other (referred to as a *translational case*), we define the trajectory which consists of only translational displacement as the optimal trajectory. If the restricted DOF before the transition is less than the DOF after the transition (referred to as a *rotational case*), the optimal trajectory includes rotational displacement. We first describe a translational case and next describe a rotational case.

A.3 Translational Case

A.3.1 Definition of Optimal Trajectory

Consider a transition from a contact relation C_s to a contact relation C_e in a translational case. Let $\mathbf{q}_s = (\mathbf{t}_s, \Theta)$ and $\mathbf{q}_e = (\mathbf{t}_e, \Theta)$ be configurations of the grasping object before and after the transition, respectively, where $\mathbf{t}_*$ $(\in R^3)$ and Θ $(\in SO(3)$: a 3×3 orthogonal matrix) represent location and orientation of the grasping object, respectively.

In this case, as mentioned above, the maintaining infinitesimal translational displacement $\Delta\mathbf{t}$ $(\in R^3)$ before and after the transition is formulated by one system of simultaneous linear equalities as Equation (A.8) and (A.9), where $F_s \in R^{l\times 3}$, $F_e \in R^{m\times 3}$.

$$\begin{aligned} F_s\Delta\mathbf{t} &= 0 \quad \text{(A.8)} \\ F_e\Delta\mathbf{t} &= 0 \quad \text{(A.9)} \end{aligned}$$

Because all objects are polyhedral, some straight trajectory can realize the transition while maintaining a contact relation C_s. A straight trajectory is formulated by $\mathbf{q}(s) = (\mathbf{t}(s), \Theta(s))$ $(0 \le s \le 1)$, where $\mathbf{t}(s) = s\Delta\mathbf{t}_d + \mathbf{t}_s$ and $\Theta(s) = \Theta$ for all s. The trajectory should satisfy a contact relation C_s when $0 \le s < 1$ and a contact relation C_e when $s = 1$.

We define the trajectory which satisfies Equation (A.10) as the optimal trajectory, where $\mathbf{T}_e$ is a solution space of Equation (A.9).

$$\mathbf{l} \cdot \Delta\mathbf{t}_d = 0 \;\; (\forall \mathbf{l} \in \mathbf{T}_e) \quad \text{(A.10)}$$

Any translational displacement which does not satisfy the equation maintains not only a contact relation C_s, but also a contact relation C_e, i.e., it is redundant for the transition. That is why we define such a trajectory as the optimal trajectory. This optimization tends to minimize the amount of translation.

Table A.1: Classification based on rank

Rank F_s	Rank F_e	
0	3	Case 1.
1		
2		
0	2	Case 2.
1		
0	1	Case 3.

A.3.2 Calculating Optimal Trajectory

For calculating the optimal trajectory, we have only to decide $\Delta\mathbf{t}_d$. Because the trajectory maintains a contact relation C_s, $\Delta\mathbf{t}_d$ must satisfy Equation (A.8). Because Rank(F_s) < Rank(F_e), there are six cases with respect to the difference of Rank(F_s) and Rank(F_e) as shown in Table A.1, where Rank(F) returns the rank of a matrix F. Now we introduce a method to calculate the optimal trajectory in the following three cases:

- Rank(F_s) = 0, 1, or 2, Rank(F_e) = 3
- Rank(F_s) = 0 or 1, Rank(F_e) = 2
- Rank(F_s) = 0, Rank(F_e) = 1

Case 1. Rank(F_s) = 0, 1, or 2, Rank(F_e) = 3
Because Rank(F_e) = 3, the location of the grasping object after the transition is uniquely decided. As a result, $\Delta\mathbf{t}_d$ is calculated by Equation (A.11).

$$\Delta\mathbf{t}_d = \mathbf{t}_e - \mathbf{t}_s \tag{A.11}$$

Case 2. Rank(F_s) = 0 or 1, Rank(F_e) = 2
Figure A.1 shows an example in the case Rank(F_s) = 0, Rank(F_e) = 2. Because Rank(F_e) = 2, a solution space $\mathbf{T}_e$ is one dimensional, i.e., a straight line. Therefore, $\Delta\mathbf{t}_d$ satisfies Equation (A.10), if and only if $\Delta\mathbf{t}_d$ satisfies Equation (A.12), where $\mathbf{t}_r \in \mathbf{T}_e - \{\mathbf{0}\}$.

$$\Delta\mathbf{t}_d \cdot \mathbf{t}_r = 0 \tag{A.12}$$

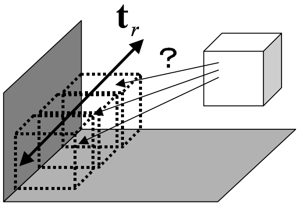


Figure A.1: Redundant translational displacement in Case 2

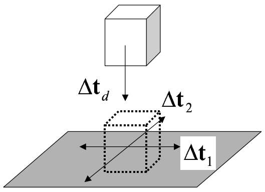


Figure A.2: Redundant translational displacement in Case 3

The object configuration after the transition is formulated by $\mathbf{q} = (\mathbf{t}_e + u\mathbf{t}_r, \Theta)(u \in R)$. As the result, $\Delta\mathbf{t}_d$ is formulated by Equation (A.13).

$$\Delta\mathbf{t}_d = \mathbf{t}_e + u\mathbf{t}_r - \mathbf{t}_s \tag{A.13}$$

By substituting Equation (A.13) to Equation (A.12), u is calculated as Equation (A.14).

$$u = \frac{(\mathbf{t}_s - \mathbf{t}_e) \cdot \mathbf{t}_r}{|\mathbf{t}_r|^2} \tag{A.14}$$

Thus, we obtain $\Delta\mathbf{t}_d$.

Case 3. Rank$(F_s) = 0$, Rank$(F_e) = 1$
Let $\{\Delta\mathbf{t}_1, \Delta\mathbf{t}_2\}$ be a set of bases of a solution space of Equation (A.9). From Equation (A.10), $\Delta\mathbf{t}_d$ can be formulated by Equation (A.15) (See Fig. A.2).

$$\Delta\mathbf{t}_d = u(\Delta\mathbf{t}_1 \times \Delta\mathbf{t}_2) \tag{A.15}$$

u can be calculated using the condition that the object configuration after the transition satisfies a contact relation C_e. Thus, we obtain $\Delta\mathbf{t}_d$.

A.4 Rotational Case

Next we describe a method to calculate the optimal trajectory in a rotational case. Generally speaking, calculating a trajectory which includes rotational displacement requires us to solve simultaneous non-linear equations. The solution is very difficult. We assume that rotation is limited to one where the axis direction of rotation is constant while rotating. The equation with respect to the axis direction has already been formulated as Equation (A.7). Therefore, we resolve the difficulty of the solution by performing the following steps: First, we calculate the axis direction and the amount of rotation. Next we calculate the trajectory by solving the relationship between a rotation angle and an object location. Note that each calculation is linear.

A.4.1 Definition of Optimal Trajectory

Consider a transition from a contact relation C_s to a contact relation C_e in a rotational case. Let $\mathbf{q}_s = (\mathbf{t}_s, \Theta_s)$ and $\mathbf{q}_e = (\mathbf{t}_e, \Theta_e)$ be configurations of the grasping object before and after the transition, respectively.
In this case, as mentioned above, the axis direction $\mathbf{a}$ ($\in R^3$) with respect to maintaining displacement before and after the transition should satisfy Equation (A.16) and (A.17).

$$\begin{aligned} G_s\mathbf{a} &= \mathbf{0} \qquad (A.16) \\ G_e\mathbf{a} &= \mathbf{0} \qquad (A.17) \end{aligned}$$

Because the axis direction is constant, the trajectory is formulated by $\mathbf{q}(s) = (\mathbf{t}(s), \Theta(s))(0 \leq s \leq 1)$. $\mathbf{t}(s) \in R$. $\Theta(s)$ represents uniform rotation, i.e., is formulated by Equation (A.18), where $R(\mathbf{a}, \theta)$ is a 3×3 orthogonal matrix to represent rotation about the axis $\mathbf{a}$ by θ ($\in R$) radian.

$$\Theta(s) = R(\mathbf{a}_d, s\theta_d)\Theta_s \qquad (A.18)$$

The trajectory should satisfy a contact relation C_s when $0 \leq s < 1$ and a contact relation C_s when $s = 1$.
We define the trajectory which satisfies Equation (A.19) as the optimal trajectory, where $\mathbf{A}_e$ is a solution space of Equation (A.17).

$$\mathbf{l} \cdot \mathbf{a}_d = 0 \ \ (\forall \mathbf{l} \in \mathbf{A}_e) \qquad (A.19)$$

The optimization tends to minimize the amount of rotation.

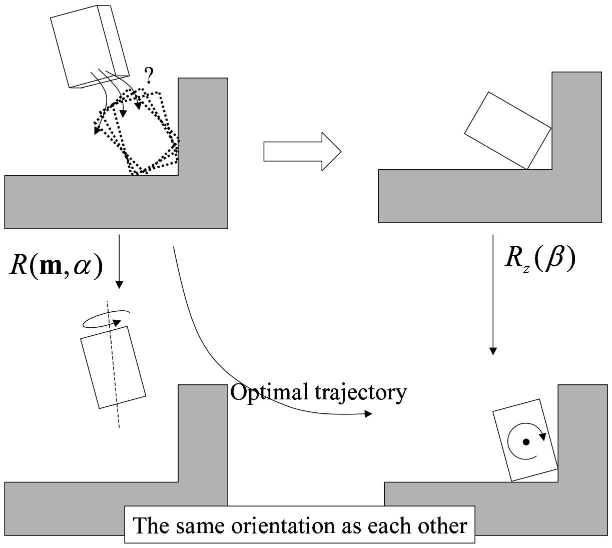


Figure A.3: Redundant displacement in rotation in Case 2

A.4.2 Deciding Axis Direction $\mathbf{a}_d$ and Amount of Rotation θ_d

There are six cases with respect to the difference of Rank(G_s) and Rank(G_e). Now we introduce a method to decide the axis direction $\mathbf{a}_d$ and the amount of rotation θ_d in the following four cases:

- Rank(G_s) = 0, 1, or 2, Rank(G_e) = 3
- Rank(G_s) = 0, Rank(G_e) = 2
- Rank(G_s) = 1, Rank(G_e) = 2
- Rank(G_s) = 0, Rank(G_e) = 1

Case 1. Rank(G_s) = 0, 1, or 2, Rank(G_e) = 3
As the same manner in a translational case, because Rank(G_e) = 3, the orientation of the grasping object after the transition is uniquely decided. Therefore, we can decide $\mathbf{a}_d$ and θ_d using orientations Θ_s, Θ_e of the grasping object before and after the transition.

Case 2. Rank(G_s) = 0, Rank(G_e) = 2
Because Rank(G_e) $\neq$ 3, the orientation of the grasping object after the transition cannot be uniquely decided as shown in Fig. A.3. We decide the direction and the amount by removing redundant orientation displacement.

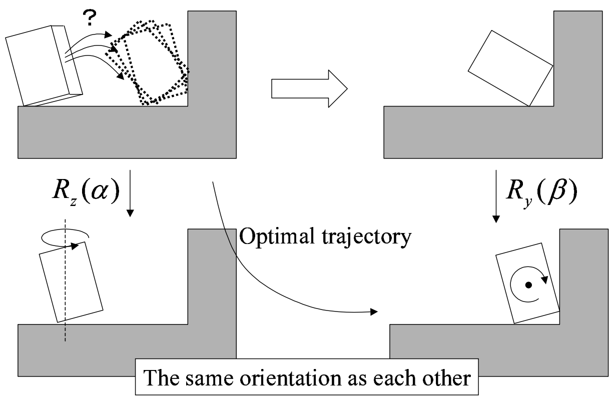


Figure A.4: Redundant orientation displacement in Case 3

Let $\{\mathbf{a}_1\}$ be a set of orthonormal bases of a solution space of Equation (A.17) and $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ be a set of orthonormal bases of a solution space of Equation (A.16)[3]. The two sets share $\mathbf{a}_1$.

Let Σ_a be the coordinate system of which directions of the z-axis, the y-axis, and the x-axis are equal to $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$, respectively. Let ${}^a\Theta_s$ and ${}^a\Theta_e$ be 3×3 orthogonal matrices which represent orientations of the grasping object with respect to the system Σ_a before and after the transition, respectively.

From the setting, rotation about $\mathbf{a}_1$ maintains a contact relation C_e. As shown in Fig. A.3, the orientation ${}^a\Theta_e$ (the top right of the figure) can be transformed to the orientation ${}^a\Theta_s$ (the top left of the figure) by the following steps:

1. Rotation about the z-axis while maintaining a contact relation C_e
2. Rotation about an axis $\mathbf{m}$ on the xy-plane

As the result, Equation (A.20) is always satisfied, where $R_*(\theta)$ is a 3×3 orthogonal matrix to represent rotation about the $*$-axis by θ radian.

$$R(\mathbf{m}, \alpha)^a\Theta_s = R_z(\beta)^a\Theta_e \tag{A.20}$$

$\alpha, \beta, \mathbf{m}$ can be solved using the equation. Because the first rotation is a redundant displacement for the transition, the optimal orientation displacement is equal to $R(\mathbf{m}, \alpha)$. Thus, we obtain $\mathbf{a}_d$ and θ_d.

[3] It is equal to the R^3 space.

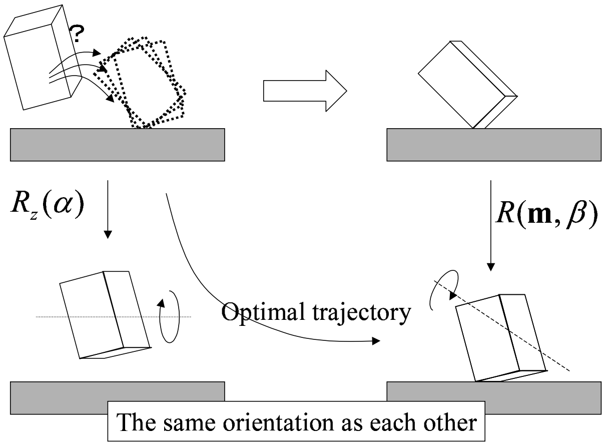


Figure A.5: Redundant orientation displacement in Case 4

Case 3. Rank$(G_s) = 1$, Rank$(G_e) = 2$
Let $\{\mathbf{a}_1\}$ be a set of orthonormal bases of a solution space of Equation (A.17) and $\{\mathbf{a}_1, \mathbf{a}_2\}$ be a set of orthonormal bases of a solution space of Equation (A.16). Unfortunately, we cannot always set up the two sets which share $\mathbf{a}_1$, i.e., a solution space of Equation (A.17) is not always a subspace of a solution space of Equation (A.16). In this section, we deal only with the case where the two sets can be set up. We describe another case in Section A.6.
Let Σ_a be the coordinate system of which directions of the y-axis and the z-axis are equal to $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively. Let ${}^a\Theta_s$ and ${}^a\Theta_e$ be 3×3 orthogonal matrices which represent the orientations with respect to the system Σ_a before and after the transition, respectively.
The orientation ${}^a\Theta_e$ (the top right of Fig. A.4) can be transformed to the orientation ${}^a\Theta_s$ (the top left of Fig. A.4) by the steps as shown in Fig. A.4. As a result, Equation (A.21) is always satisfied.

$$R_z(\alpha)^a\Theta_s = R_y(\beta)^a\Theta_e \tag{A.21}$$

α, β can be solved using the equation. Because the orientation displacement $R_y(\beta)$ is redundant for the transition, the optimal orientation displacement is equal to $R_z(\alpha)$. Thus, we obtain $\mathbf{a}_d$ and θ_d.

Case 4. Rank$(G_s) = 0$, Rank$(G_e) = 1$
Let $\{\mathbf{a}_1, \mathbf{a}_2\}$ be a set of orthonormal bases of a solution space of Equation (A.17) and $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ be a set of orthonormal bases of a solution space of Equation (A.16).

The two sets share $\mathbf{a}_1$ and $\mathbf{a}_2$. Let Σ_a be the coordinate system of which directions of the x-axis, the y-axis, and the z-axis are equal to $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$, respectively. Let ${}^a\Theta_s$ and ${}^a\Theta_e$ be 3×3 orthogonal matrices which represent the orientations with respect to the system Σ_a before and after the transition, respectively.

The orientation ${}^a\Theta_e$ (the top right of Fig. A.5) can be transformed to the orientation ${}^a\Theta_s$ (the top left of Fig. A.5) by the steps as shown in Fig. A.5. As the result, Equation (A.22) is always satisfied, where $\mathbf{m}$ is on the xy-plane.

$$R_z(\alpha)^a\Theta_s = R(\mathbf{m}, \beta)^a\Theta_e \tag{A.22}$$

$\alpha, \beta, \mathbf{m}$ can be solved using the equation. Because the orientation displacement $R(\mathbf{m}, \beta)$ is redundant for the transition, the optimal orientation displacement is equal to $R_z(\alpha)$. Thus, we obtain $\mathbf{a}_d$ and θ_d.

A.4.3 Formulating the Relationship Between the Rotation Angle and the Location

Let $\mathbf{t}(s)$ and $\Theta(s)$ be the location and the orientation on the trajectory. $\Theta(s)$ has already been decided by the method as mentioned above.

Generally speaking, the configuration which satisfies some contact relation is formulated by one system of non-linear equations as Equation (A.23)[41].

$$\bigcap_{i=1}^{l} f_i(\mathbf{t}(s), \Theta(s)) = 0 \tag{A.23}$$

Only the term $\Theta(s)$ yields non-linearity of the equation. Therefore, Equation (A.23) becomes one system of linear equations when s is decided, because $\Theta(s)$ is constant in this time.

Because a robot is usually controlled at discrete time steps, it is enough to calculate the configurations which are sufficiently sampled on the trajectory. It is easy to solve the configuration, even if the solution is redundant.

A.5 Example of Calculation of Optimal Trajectories I

We employed the transitions and the configurations in each contact relation as shown in Fig. 2.22 in Section 2.8.

The initial object configuration was set up to align depth directions of the peg and the hole in order to satisfy the condition to be able to set up the sets in Case 3.

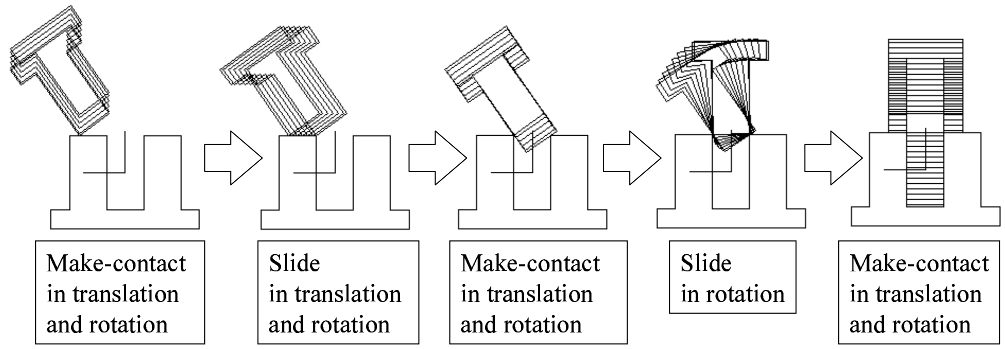


Figure A.6: Optimal trajectory

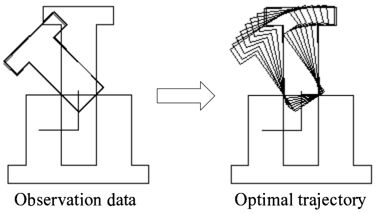


Figure A.7: A trajectory obtained from the observation (the left) and the optimal trajectory (the right)

Figure A.6 shows that the calculated optimal trajectory in VR space. Of course, the calculation was performed in the 3 dimensional space.

Figure A.7 shows the configurations obtained from the observation (the left of the figure) and the configurations calculated (the right of the figure) in the fifth transition (corresponding to a slide sub-skill in rotation in the figure). Sufficient configurations cannot be obtained from the observation, because of the difficulty of correcting vision errors, i.e., the non-linear optimization method cannot obtain sufficient configurations. However, the calculation of the optimal trajectory employes only linear solution. Therefore, the method can obtain sufficient configurations.

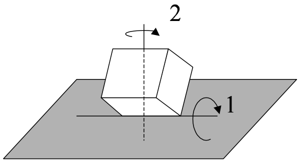


Figure A.8: Edge-face contact

A.6 Problem of the Calculation and its Solution

A.6.1 Over-Approximation With Respect to Formulation of the Legal Infinitesimal Displacement

Through calculation of the optimal trajectory in various cases, we found out that the calculation failed when a restricted DOF in rotation translated from 0 to 1 or from 1 to 3. In this section, we describe the reason and the solution.

Equation (A.1) is the Taylor series of the displacement up to order 1. The displacement is actually formulated as a non-linear equation. Generally speaking, the linearization sometimes introduces an erroneous solution as mentioned above. And our proposed method cannot work well, when a restricted DOF is one before or after the transition.

For example, in an edge-face contact case as shown in Fig. A.8, Equation (A.7) answers that rotation about an axis of which a direction is represented as a linear sum of the edge direction and the surface normal of the contacting face maintains a contact relation. Of course, rotation about an axis of which a direction is equal to the edge direction or the surface normal (Axis 1 and 2 as shown in Fig. A.8, respectively) maintains it. However, rotation about the axis except for the two does not maintain it. That is, Equation (A.7) introduces erroneous solutions. As a result, calculating the optimal trajectory is failed.

Although we can employ the second order approximation to solve the problem, we now introduce the easier solution. First we assume that a restricted DOF in rotation is one, only when the contact relation consists of one edge-face contact or one face-edge contact only. That is an adequate assumption, because contact relations where a restricted DOF in rotation is one are usually either these two.

In an edge-face contact case as shown in Fig. A.8, the orientation displacement to first rotate about Axis 1 by β radian and to next rotate about Axis 2 by α radian maintains the contact relation. Such a displacement is formulated by Equation

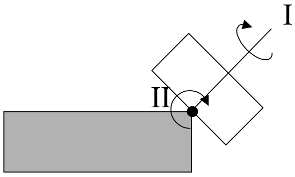


Figure A.9: Face-edge contact

(A.24), where **l** and **n** are directions of Axis 1 and 2, respectively. Note that the direction of Axis 2 is constant with respect to rotation about Axis 1.

$$R(\mathbf{n}, \alpha)R(\mathbf{l}, \beta) \tag{A.24}$$

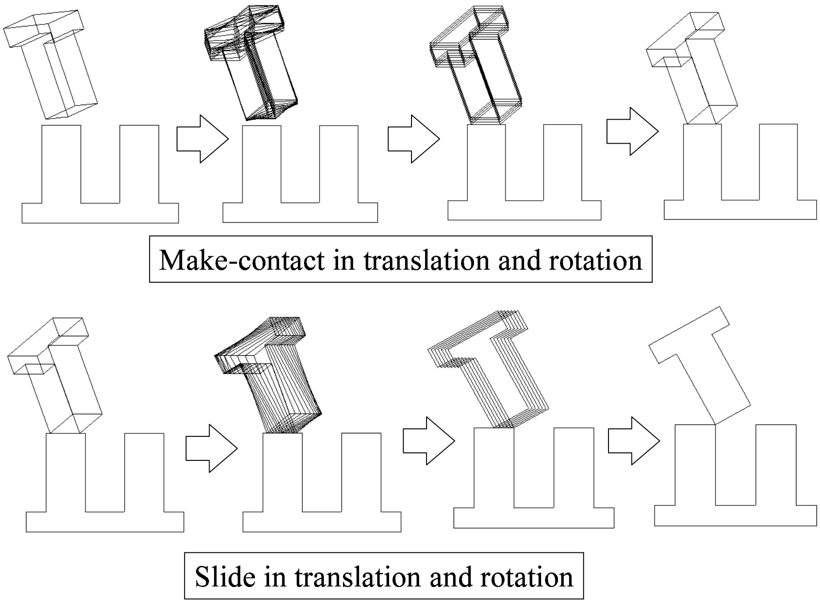
In a face-edge contact case as shown in Fig. A.9, the orientation displacement to first rotate about Axis I by β radian and to next rotate about Axis II by α radian maintains the contact relation. Such a displacement is formulated by Equation (A.25) where **n** and **l** are directions of Axis I and II, respectively. Note that the direction of Axis II is constant with respect to rotation about Axis I.

$$R(\mathbf{l}, \alpha)R(\mathbf{n}, \beta) \tag{A.25}$$

When a restricted DOF in rotation translates 0 to 1, the axis direction and the amount of rotation can be calculated by the equation which is obtained by substituting Equation (A.24) or (A.25) into a term $R(\mathbf{m}, \beta)$ in Equation (A.22). When the DOF translates 1 to 3, they can be calculated by solving ${}^{A}\Theta_e \, {}^{a}\Theta_s^{-1}$ = Equation (A.24) or (A.25). Then, we can calculate the optimal trajectory in these two cases. The optimal trajectory requires us to rotate twice.

A.6.2 Two Sets Cannot Be Set Up in Case 3.

In this section, we consider that the two sets cannot be set up in Case 3 as mentioned above. We also assume the same assumption about the restricted DOF in rotation in the previous section. In an edge-face contact case as shown in Fig. A.8, rotation about Axis 2 enables a solution space of Equation (A.17) to be a subspace of a solution space of Equation (A.16). After that, the optimal trajectory is calculated by applying the method in Case 3. In practical transition, $\mathbf{a}_2$ is always equal to Axis 2. The optimal trajectory can be calculated by only applying the method in Case 3, where $\mathbf{a}_1$ is a basis of Equation (A.17) and $\mathbf{a}_2$ is the surface normal. Note that one does not need rotation about Axis 2 to set up two sets.



Make-contact in translation and rotation

Slide in translation and rotation

Figure A.10: Optimal Trajectory

As the same way, in a face-edge contact as shown in Fig. A.9, rotation about Axis II can realize the condition. In practical transition, $\mathbf{a}_1$ is always equal to Axis II. The optimal trajectory calculated by only applying the method in Case 3, where $\mathbf{a}_1$ is a basis of Equation (A.17) and $\mathbf{a}_2$ is the surface normal, when the object orientation is ${}^{a}\Theta_s$.

A.6.3 Example of Calculation of Optimal Trajectories II

In this example, the initial object configuration is set up not to align depth directions of the peg and the hole. Figure A.10 shows the first and second transitions (Transition (1) and (2) as shown in Fig. 2.22), which clearly illustrate the effect of the improvement as mentioned above. Note that other transitions are essentially the same as the previous example.

In this case, because these two transitions require both translational and rotational displacement, first the optimal rotational displacement is operated and then the optimal translational displacement is operated[4]. The calculation is complete when the restricted DOF in rotation is one before or after the transition.

[4] The optimal trajectory to simultaneously translate and rotate can be calculated by dealing with the redundancy of Equation (A.23). However, that is beyond the scope of this thesis.

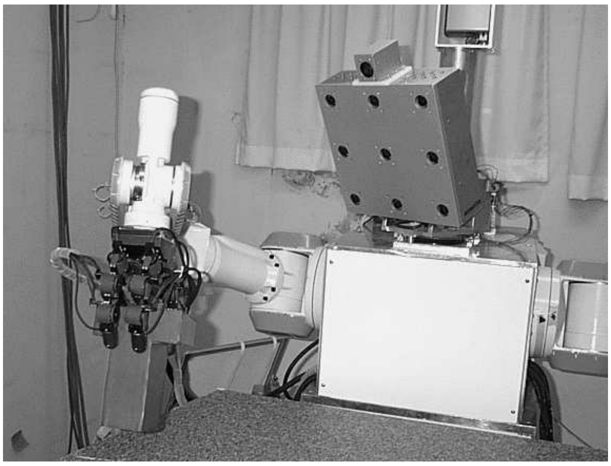


Figure A.11: Our test bed

A.7 Implementation on Our Test-Bed

Until the previous section, we described a method to calculate the optimal trajectory. If every object configuration were precisely known and a robot arm could be perfectly controlled, the task could be achieved by moving the grasping object along the trajectory. However, because that condition is not usually satisfied, sensor feedback is necessary for achievement of the task. In this section, we describe an example of implementing sub-skills on our test-bed[72] as shown in Fig. A.11. It is equipped with dual 7 DOF robot arms (PA-10 produced by Mitsubishi Heavy Industries, Ltd.[73]) , a hand with multi-fingers at the end of each robot arm, and multi-baseline real-time stereo vision system[56]. The hand, as shown in Fig. A.12, consists of four fingers. Each finger has three joints (servo motor produced by Yasukawa Electric Corp.) and is equipped with a force/torque sensor at the tip.

A.7.1 Control

Because execution errors occur for various reasons (misalignment of objects, etc.), sensor feedback is necessary to correct such errors.
For easy implementation, we set up the following assumption:

1. Employ the robot hand only for grasping
2. Move the robot arm so slowly that the effect of inertia is sufficiently small
3. Set up appropriate compliance in each finger joint

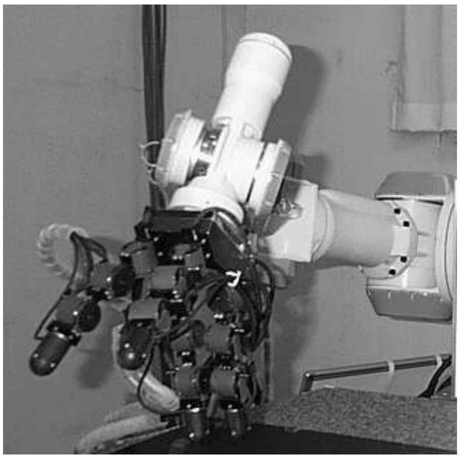


Figure A.12: Robot arm

4. Generate only the contact relations and their transitions which have already been obtained from the observation

5. Know the positional relationship between the grasping object and the hand, which little changes on the execution

An assembly task is achieved using sub-skills by performing the following steps: First, the robot obtains the configuration of the environmental object from the observation. As a result, the robot obtains the optimal trajectories of the grasping object and the robot arm with respect to the coordinate system of the robot. Next, the robot executes a sub-skill until its end condition is satisfied. Once the condition is satisfied, the robot executes the next sub-skills until the task has been accomplished. We describe the end condition in Section A.7.3.

Each sub-skill is implemented by frequently calling a position-control command. Note that force-control is also realized using a position-control command. Generally speaking, a position-control command is implemented on almost all of robot arms. Let $\mathbf{q}_d(t)$ be the configuration of a robot arm at step t which is sufficiently sampled on the optimal trajectory. At step t, the configuration $\mathbf{q}(t)$ which is sent to a robot arm through a position-control command is calculated by Equation (A.26), where $\mathbf{f}$ and $\mathbf{f}_d$ are the actual value and the desired value of the force/torque sensor, respectively, and $\mathbf{K}$ is a gain parameter. Note that Δs is a sufficiently small number to satisfy Condition 2.

$$\mathbf{q}(t) = \mathbf{q}(t-1) + (\mathbf{q}_d(\Delta s \cdot t) - \mathbf{q}_d(\Delta s \cdot (t-1)) + \mathbf{K}(\mathbf{f} - \mathbf{f}_d) \tag{A.26}$$

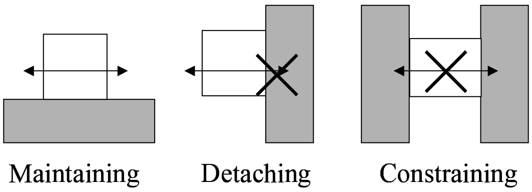


Figure A.13: Maintaining, detaching, and constraining DOFs in translation

In this implementation, magnitude of force/torque applied to the grasping object is estimated using a force/torque sensor at the tip of each finger. Unfortunately, the magnitude of torque cannot be estimated, i.e., we cannot implement the control using magnitude of torque, because of errors with respect to the grasping position, etc.

A.7.2 Control to Correct Errors With Respect to Location

In this section, we describe a method to decide parameters $\mathbf{K}$ and $\mathbf{f_d}$ in Equation (A.26) with respect to the location. Basically, we can decide only signs of the parameters and cannot decide appropriate magnitude of the parameters. Such magnitude should be decided from the knowledge acquired through the execution. While executing a sub-skill, each DOF with respect to translation belongs to one of maintaining, detaching, and constraining DOFs in translation as shown in Fig. A.13. In the direction of the basis corresponding to a maintaining DOF, any small errors are insensitive for the execution. Therefore we do not need to apply force control to the direction. We should apply force control to the direction of the basis corresponding to another two DOFs for correcting execution errors. In the direction of the basis corresponding to a detaching DOF, execution errors are removed by pushing the grasping object toward the direction corresponding to the illegal displacement by sufficient small force. Although execution errors do not always arise in the direction of the basis corresponding to a constraining DOF, we set up the desired force value to zero, because extra binding force does not arise. As a result, the parameters are decided by the following rules:

- Not react in the direction of the basis corresponding to a maintaining DOF
- Push the grasping object toward the direction of the illegal displacement by sufficient small force in the direction of the basis corresponding to a detaching DOF

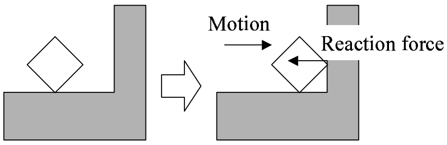


Figure A.14: Make-contact sub-skill

- Not arise force in the direction of the basis corresponding to a constraining DOF

A.7.3 End Condition

In this section we describe the end condition of sub-skills. In this implementation, we decide the condition using the value of a force/torque sensor[5]. All sub-skills are classified into three types (make-contact, detach-contact, and slide sub-skills) and the difference among the three is derived by the difference of the end conditions.

End Conditions of Make-Contact and Detach-Contact Sub-Skills

In a make-contact sub-skill as shown in Fig. A.14, reactive force (torque) arises by contacting a dead-end contact primitive in the end of the sub-skill. Therefore, we can decide the end when magnitude of a force/torque sensor with respect to the direction of the basis corresponding to the DOF-transition suddenly increases. In a detach-contact sub-skill, we can decide the end when the magnitude of a force/torque sensor suddenly decreases.

End Condition of Slide Sub-Skill

In a slide sub-skill as shown in Fig. A.15, some support contact primitives are transformed to singular contact primitives at the end of the sub-skill. Now we focus on the contacting face (the face including the two edges in an edge-edge contact case) in one of the support contact primitives. In the transformation, magnitude of a force sensor with respect to the surface normal suddenly decreases. Therefore, we can decide the end using this decrease. However, we should consider

[5] Kitagaki et al. decide the condition by monitoring motion of pseudo contact points, which can be calculated the difference of the value of a force/torque sensor[74]. Shimokura and Mutoh decide the condition by motion of the grasping object which is compliant to environmental constraint[75].

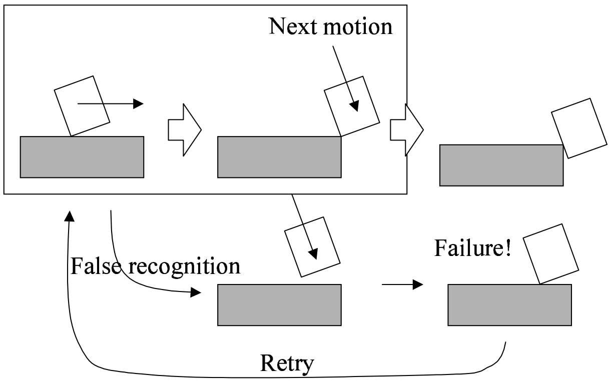


Figure A.15: Slide sub-skill

that such a decrease also occurs in the case to fail to maintain the contact relation, which is not the end.

One method to solve the ambiguity is to employ visual feedback when decreasing the magnitude. However, it is difficult to classify these two types of decreases if the observation is poor. An easier method to solve the ambiguity is to call the next sub-skill in the meantime. If the decrease is not caused by the end, the magnitude suddenly increases after a brief interval, because the next sub-skill can be executed during a brief duration (See Fig. A.15).

A.8 Experiment

For verifying our sub-skill implementation, we made our test-bed execute the peg-insertion as shown in Fig. A.16, which looks like a two dimensional case. Although it is preferable to verify the implementation using three dimensional peg-insertion, we select the peg-insertion to satisfy Condition 4, i.e., to satisfy the following condition:

- Minimize an amount of increase of restricted DOFs every transition

- Translate only the contact relation obtained from the observation

In three dimensional peg-insertion, the possibility not to satisfy the second condition (For example the peg passes through the top of the hole.) is very high. Inversely, in the case to guarantee that these two conditions are always satisfied, the difficulty of two types of peg-insertion are equivalent to each other, i.e., our

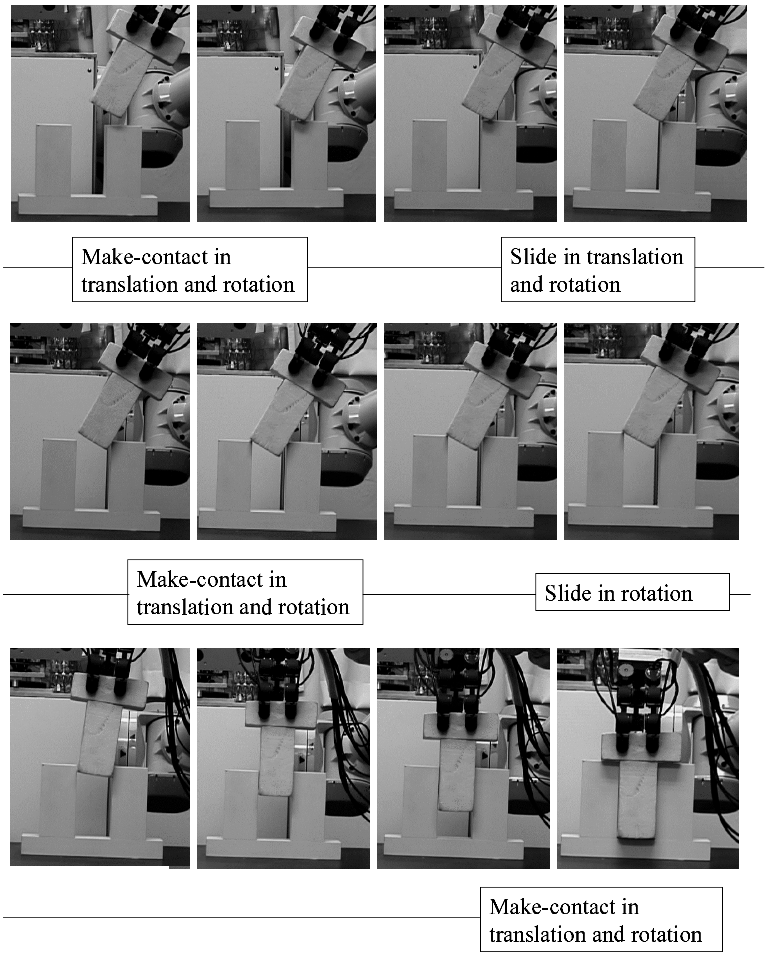

Make-contact in translation and rotation
Slide in translation and rotation
Make-contact in translation and rotation
Slide in rotation
Make-contact in translation and rotation

Figure A.16: Robot execution

test-bed can achieve these tasks. Of course, motion of sub-skills is not limited to planar motion. Figure A.16 shows the actual execution by our test-bed.

In this experiment, magnitude of a desired force $\mathbf{f}_d$ and a gain $\mathbf{K}$ are manually set up by trial and error. Figure A.17 shows the force along the vertical direction at each time, when first executing a make-contact sub-skill and next executing a slide sub-skill (A top part of Fig. A.16). The desired force was set up to 50[gf], which is shown by the dotted line in Fig. A.17.

At about 5.2[s], a force/torque sensor detected the sudden increase and the robot decided the end of a make-contact sub-skill. After that, the value of the force was steadily converged to the desired value 50[gf]. At about 15.4[s], the robot detected

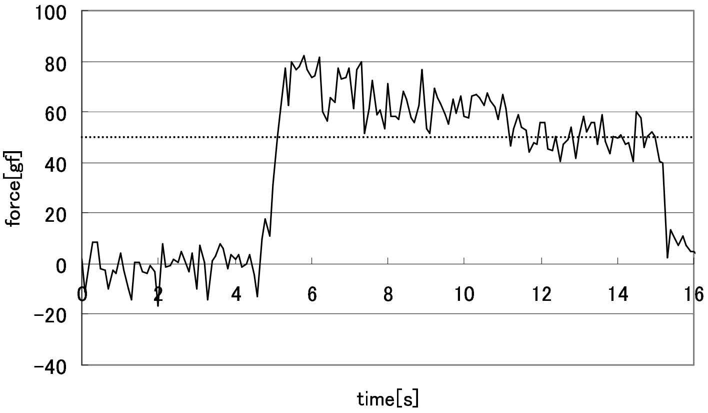


Figure A.17: Force sensor date from the 1st sub-skill to the 3rd sub-skill

the end of a slide sub-skill from the decrease caused by the transformation from the edge-face contact to the parallel edge-edge contact and executed the next sub-skill. Because the transition is not a critical transition, i.e., the process of the task is insensitive to the small execution errors, the next sub-skill was executed without any exception.

Figure A.18 shows the force along the vertical direction at each time, when first executing a slide sub-skill in rotation and next executing a make-contact sub-skill (A middle part of Fig. A.16).

This transition is a critical transition, i.e., the process of the task is sensitive to the execution errors. The robot decided the end of a slide sub-skill from the decrease at (a), (b), (c), and (d) in Fig. A.16 and called the next make-contact sub-skill in the meantime. However, the sudden reactive force occurred[6], because of incomplete orientation alignment for the insertion. Then the robot summed up the failure to detect the end and re-executed the slide sub-skill.

The robot decided the end from the decrease at (e) in Fig. A.16 again. In this time, the reactive force did not suddenly occur. Therefore the robot continued to execute the sub-skill. However, slightly incomplete orientation alignment immediately increased binding force as shown in the figure. In this experiment, the increase was caused by not applying feedback using torque.

[6] In this experiment, that means the reactive force was detected within 1[s].

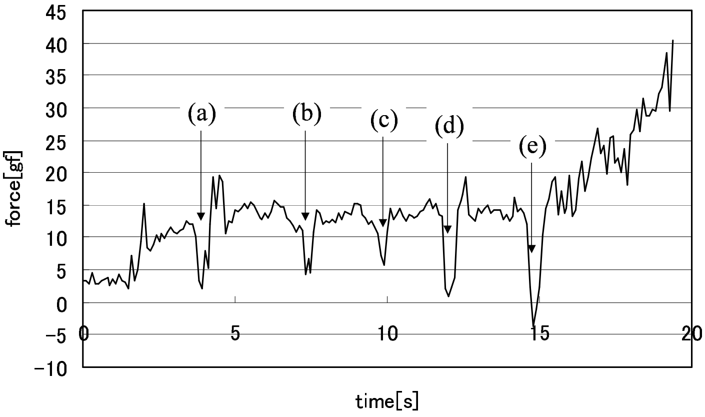


Figure A.18: Force sensor data in "slide in rotation" sub-skill

A.9 Discussion

A.9.1 Skill-Based Manipulation System

In this thesis, we adopt the concept of the *Skill-Based Manipulation System*[76, 77, 75, 78], which deal with assembly tasks. In this section, we describe the similarity and difference between the past research and our method. Suehiro and Takase first classified any contact relations based on the number of the point-contact, and next introduced movement primitives from the transitions of the class of contact relations. Note that they deal only with planar motion and they claim that any motion in the task locally can be represented as planar motion[76]. They proposed the following seven movement primitives:

- Move-to-touch (Make-contact sub-skill from the contact relation with no contact primitive)
- Slide-to-touch (Make-contact sub-skill in translation from any contact relation except for the contact relation with no contact primitive)
- Vertex-to-edge (Type I make-contact sub-skill in rotation)
- Rotate-to-touch (Type II make-contact sub-skill in rotation)
- Change (1. Slide sub-skill in translation and then make-contact or slide sub-skill in translation, or 2. slide sub-skill in rotation and then make-contact or

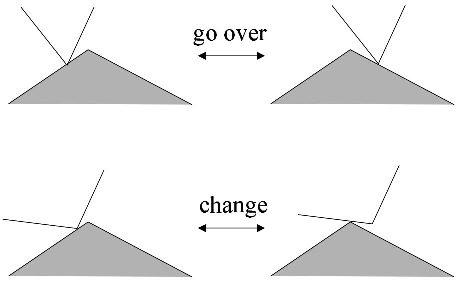


Figure A.19: "Go over" skill and "change" skill

slide sub-skill in rotation)

- Go-over (1. Slide sub-skill in translation and then make-contact or slide sub-skill in translation, or 2. slide sub-skill in rotation and then make-contact or slide sub-skill in rotation)
- Rotate-to-insert (Slide sub-skill in rotation and then make-contact or slide sub-skill in translation)

The corresponding sub-skills are shown between parentheses.
The differences between the movement primitives proposed by them and our sub-skills are as follows:

- Some kinds of the movement primitives consist of a pair of a slide sub-skill and one sub-skill.
- Change (*change* in Fig. A.19) and Go-over (*go over* in Fig. A.19) is mapped into the same sequence of sub-skills.
- There is not the movement primitive which consists of a slide sub-skill in translation and then a make-contact or slide sub-skill in rotation. In actually, this combination seldom appears.

With the exception of a few differences, these two are very similar to each other. Therefore our sub-skills are very flexible to apply the result of past research.

A.9.2 Control to Remove Errors With Respect to Orientation

Each DOF with respect to the axis direction of rotation, i.e., an orientation belongs to one of maintaining, detaching, and constraining DOFs in rotation as shown in

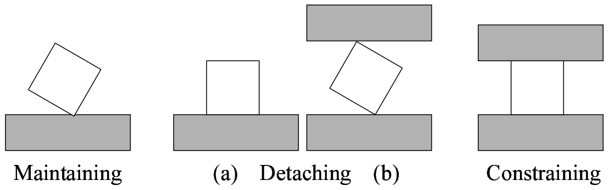


Figure A.20: Maintaining, detaching, and constraining DOFs in rotation

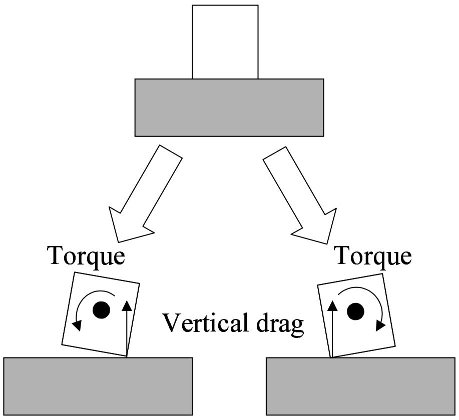


Figure A.21: Control with respect to axis direction of rotation corresponding to Type I detaching DOF

Fig. A.20. As the same manner in the location case, feedback by torque should be applied to the axis direction corresponding to detaching and constraining DOFs. In the axis direction corresponding to a constraining DOF, the desired torque value is set up to zero, because extra binding force does not arise.

As shown in Fig. A.20, there are two types of detaching DOFs. One is the Type I detaching DOF and the object can rotate about the axis both clockwise and counter-clockwise (See Fig. A.20 (a)); the other is the Type II detaching DOF and the object can rotate about the axis either clockwise or counter-clockwise (See Fig. A.20 (b)). In the latter type, execution errors are corrected by twisting the object toward the direction corresponding to the illegal displacement by a sufficiently small torque. In the former type, because unexpected torque arises when detaching one of contact primitives, we set up the desired torque to zero (See Fig. A.21).

Furthermore, we should consider the set up of the center of compliance. Shimizu and Kosuge proposed a method to formulate the valid area of the center as simultaneous linear inequalities[79]. Their method only deals with planar motion, but it is possible to improve the method to deal with 3 dimensional motion, we believe.

A.9.3 Improvement of Sub-Skill Implementation

It is strongly desirable to improve present sub-skill implementation. In the section, we describe our improvement.

First, the assumption to generate only the contact relations and their transitions which has already been obtained from the observation is not usually satisfied. For example, the peg is directly inserted to the hole in the peg-insertion. To eliminate the assumption, we need to solve the following three questions:

1. What kind of an unexpected contact relation occurs because of execution errors?

2. How is the unexpected contact relation detected?

3. What kind of a sub-skill is executed for returning to the planned transition.

With respect to the first question, several methods exist to search the neighbor contact relations to some contact relation[32, 33]. Although almost all of these methods employ the non-linear optimization method, we propose a method to speed up the method by giving a better initial guess in Chapter 3.

With respect to the second question, Yu et al. proposed the method to estimate a present contact relation by slightly moving the grasping object[80]. It is possible to efficiently estimate by predicting unexpected contact relations in advance, we believe.

With respect to the third question, it is easy to select an appropriate sub-skill to return, if the second question is solved. It is essential to implement these three to improved the execution.

In this thesis, we do not consider a method to decide magnitude of the gain parameter or the desired force/torque values. For example, Matsuoka et al. proposed the method to obtain the appropriate parameters through repetition of the task[81]. However, the method requires the preparation of two modules to detect the failure of the task and to adjust the parameters to avoid the failure in advance. These modules are manually implemented for each task and the method to automatically generate the two modules has not been developed yet. To estimate an unexpected contact relation overcomes the problem, we believe.

Appendix B

Formulating Legal Displacement using the Second Order Approximation

In this appendix, we describe how to formulate legal displacement for any contact relations using the second order approximation. For this purpose, we must formulate the approximation for all point-contact primitives and singular point-contact primitives which include at least one singular object primitive. Unfortunately, we have not formulated legal displacement in a few singular point-contact primitives yet. This formulation is an outstanding problem.

We begin by defining singular object primitives. Next, we describe how we formulate legal displacement in all contact primitives which have not been formulated in Chapter 3. Although, we have not been completely able to examine the validity and the efficiency of the formulation, the basic ideas which we illustrate in this appendix are very useful, we believe. In this appendix, we concentrate on formulating legal displacement. Although the approximations are different depending on the types of displacement, it is easy to formulate other types of displacement by applying the results described in this appendix.

B.1 Singular Object Primitives

The representation for a curved line as mentioned in Chapter 3 is assumed to be able to define tangent and principle normals at a point $\mathbf{x}_c$. However, if a point $\mathbf{x}_c$ is on a vertex, these cannot be defined. In this case, we determine these two values using limit values from either below or above. Of course, the equation is meaningful when either $l \geq 0$ or $l \leq 0$. We define this curved line as a *half_edge*. In this case, we set up $\mathbf{t}$ along the edge from the vertex. Therefore the equation is

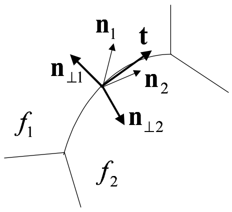


Figure B.1: Definition of $\mathbf{n}_{\perp i}$

meaningful when $l \geq 0$.

In the representation for a curved surface, we should pay attention when a point $\mathbf{x}_c$ is on an edge or a vertex. If the point is on an edge except for a vertex, this face is define as a *half_face.* A half_face can be represented by the same equation, but the range of a point $\mathbf{x}$ on the face is limited by $\mathbf{x} \cdot \mathbf{n}_{\perp} \geq 0$. Let two faces f_1 and f_2 be adjacent to an edge e. $\mathbf{n}_{\perp i}(i = 1 \text{ or } 2)$ is defined as Equation (B.1) (See Fig. B.1). Note that an edge e is convex.

$$\mathbf{n}_{\perp i} = \begin{cases} \mathbf{n}_i \times \mathbf{t} & ((\mathbf{n}_i \times \mathbf{t}) \cdot \mathbf{n}_{2-i} < 0) \\ -\mathbf{n}_i \times \mathbf{t} & ((\mathbf{n}_i \times \mathbf{t}) \cdot \mathbf{n}_{2-i} > 0) \end{cases} \tag{B.1}$$

If the point is on a vertex, this face f_i is defined as a *quarter_face.* A quarter_face can be represented by the same equation, but the range of a point $\mathbf{x}$ on the face is limited by $w_1 \geq 0$ and $w_2 \geq 0$, where $\mathbf{x} - \mathbf{x}_c = w_1\mathbf{t}_{i-1} + w_2\mathbf{t}_i + w_3\mathbf{n}_i$. w_1 and w_2 are calculated by Equation (B.2). Note that $\mathbf{t}_{i-1} \cdot \mathbf{n}_i = 0$, $\mathbf{t}_i \cdot \mathbf{n}_i = 0$, and $\mathbf{t}_{i-1} \neq \pm\mathbf{t}_i$, i.e., the inverse matrix $b_{i-1,i}^{-1}$ always exists.

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = b_{i-1,i}^{-1} \begin{pmatrix} \mathbf{t}_{i-1} \cdot (\mathbf{x} - \mathbf{x}_c) \\ \mathbf{t}_i \cdot (\mathbf{x} - \mathbf{x}_c) \end{pmatrix} \tag{B.2}$$

$$b_{i-1,i} = \begin{pmatrix} 1 & \mathbf{t}_{i-1} \cdot \mathbf{t}_i \\ \mathbf{t}_{i-1} \cdot \mathbf{t}_i & 1 \end{pmatrix}$$

We define these three as *singular object primitives.*

B.2 Formulating Legal Displacement

As mentioned in Chapter 3, if the approximation for A-B contact has already been formulated, the approximation for B-A contact is easily formulated. Therefore, we concentrate on only A-B contact below.

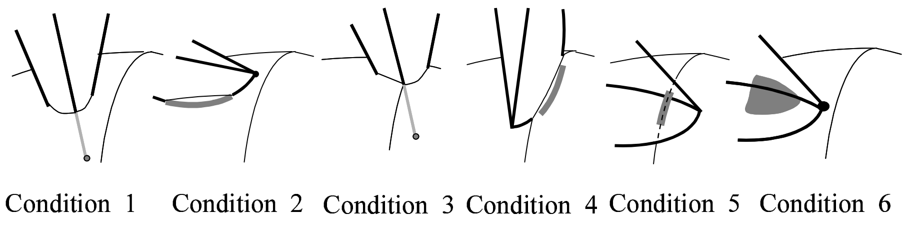


Figure B.2: Six penetration cases in a vertex-edge contact

B.2.1 Point-Contact Primitives

Vertex-Edge Point-Contact

Consider the case that a vertex v contacts an edge E at a point $\mathbf{x}_c$. We formulate the legal displacement by inverting the illegal displacement. If two objects locally penetrate each other after the displacement, a part of object primitives of the moving object is inside the fixed object. Concretely, at least one of the following six conditions must be satisfied (See Fig. B.2):

1. One adjacent half_edge e_i of a vertex v penetrates some adjacent half_face F_j of an edge E, where $F_j(\mathbf{d}(\mathbf{x}_c)) < 0$.
2. One adjacent half_edge e_i of a vertex v penetrates some adjacent half_face F_j of an edge E, where $F_j(\mathbf{d}(\mathbf{x}_c)) \geq 0$.
3. One adjacent half_edge e_i of a vertex v penetrates an edge E, where a vertex v is inside the fixed object.
4. One adjacent half_edge e_i of a vertex v penetrates an edge E, where a vertex v is not inside the fixed object.
5. One adjacent quarter_face f_i of a vertex v penetrates an edge E.
6. One adjacent quarter_face f_i of a vertex v penetrates some adjacent half_face F_j of an edge E.

First we consider Condition 1. Positional relationship between the half_edge and the half_face is classified into the following three types, where $\text{Proj}(\mathbf{a}, \mathbf{b}) = \mathbf{a} - (\mathbf{a} \cdot \mathbf{b})\mathbf{b}$ and $\mathbf{N}_{sep} = \dfrac{\mathbf{N}_{\perp 1} + \mathbf{N}_{\perp 2}}{|\mathbf{N}_{\perp 1} + \mathbf{N}_{\perp 2}|}$. (See Fig. B.3):

- $\text{Proj}(\mathbf{t}_i, \mathbf{N}_{sep}) \cdot \text{Proj}(\mathbf{N}_{\perp j}, \mathbf{N}_{sep}) > 0$ (Fig. B.3 (a))

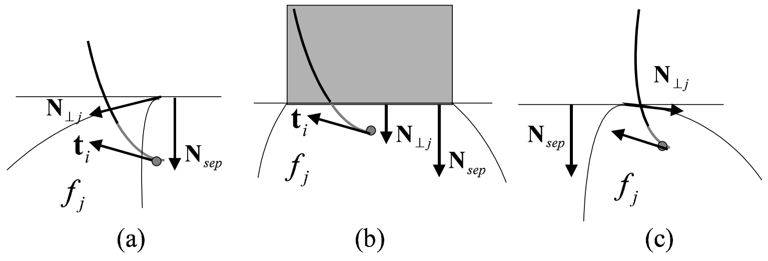


Figure B.3: Positional relationship between a half_edge and a half_face

- Proj$(\mathbf{t}_i, \mathbf{N}_{sep}) \cdot$ Proj$(\mathbf{N}_{\perp j}, \mathbf{N}_{sep}) = 0$ (Fig. B.3 (b))
- Otherwise (Fig. B.3 (c)),

In the first case, the penetration condition is formulated by Equation (B.3).

$$\Delta_{e_iE} < 0 \cap \Delta_{vF_j} < 0 \tag{B.3}$$

We omit a description of the method to set up $l'(0)$ and $L'(0)$ in Δ_{e_iE}, because the method is the same as that described in Chapter 3.

In the second type, Δ_{e_iE} cannot be employed because the outward normal cannot be defined. In this case, both $\pm\mathbf{t}_i \times \mathbf{T}$ are inward or outward to the fixed object. However, we can formulate the displacement as Equation (B.4).

$$\Delta_{vF_1} < 0 \cap \Delta_{vF_2} < 0 \tag{B.4}$$

In reality, this penetration is equivalent to the penetration in Condition 3.

We consider the third type. Because of the convexity of a vertex, the angle between $\mathbf{t}_i$ and $\mathbf{t}_{i+1}$ is less than 180 degrees for all i. As a result, another edge of which the positional relationship to the half_face is the first type always exists and obviously penetrates the half_face. Therefore we do not need to consider the third type.

Next we consider Condition 2. Such a penetration requires us to satisfy Equation (B.5).

$$\mathbf{t}_i \cdot \mathbf{N}_j = 0 \tag{B.5}$$

Conversely, if the equation is satisfied, we should consider the condition that a half_edge e_i penetrates a half_face F_j. The condition is formulated below.

Because we have already considered Condition 3, we next consider Condition 4. Realizing the penetration requires us to satisfy Equation (B.6).

$$\mathbf{t}_i \times \mathbf{T} = 0 \tag{B.6}$$

Conversely, if the equation is satisfied, we should consider the penetration. In this case, the penetration is formulated by Equation (B.7).

$$\bigcap_{j=1}^{2} l_{e_iF_j} > 0 \cap \Delta_{e_iF_j} < 0 \tag{B.7}$$

We do not describe the method to set up $l'(0)$ in $\Delta_{e_iF_j}$. The equation means that a half_edge e_i penetrates two faces that have the same equations as two half_faces F_1 and F_2. Section B.2.2 concretely describes the formulation for a half_edge-face point-contact.

Then we consider Condition 5. Condition 5 is usually accompanied by Conditions 1 or 2. For occurrence of Condition 5 only, Equation (B.8) must be satisfied.

$$\mathbf{n}_i \cdot \mathbf{T} = \mathbf{0} \tag{B.8}$$

Conversely, if the equation is satisfied, we should consider the condition that a quarter_face f_i penetrates an edge E_j. The condition is formulated below.

Finally we consider Condition 6. Condition 6 is also usually accompanied by at least one of Condition 1, 2, and 5. For occurrence of Condition 6 only, Equation (B.9) must be satisfied.

$$\mathbf{n}_i \times \mathbf{N}_j = \mathbf{0} \tag{B.9}$$

Conversely, if the equation is satisfied, we should consider the condition that a quarter_face f_i penetrates a half_face F_j. The condition is formulated below.

As a result, if all penetration conditions are represented by a union of Equation (B.3), the legal displacement is formulated by Equation (B.10). For other cases, we can easily formulate the displacement.

$$\bigcap_{i,j} (\Delta_{e_iE} \geq 0 \cup \Delta_{vF_j} \geq 0) \tag{B.10}$$

Vertex-Vertex Point-Contact

Consider the case that a vertex v contacts a vertex V at the point $\mathbf{x}_c$. Basically speaking, the legal displacement in a vertex-vertex contact can be formulated as the same way as a vertex-edge contact. If two objects locally penetrate each other after the displacement, at least one of the following seven conditions must be satisfied:

1. One adjacent half_edge e_i of a vertex v penetrates some adjacent quarter_face F_j of a vertex V, where $F_j(\mathbf{d}(\mathbf{x}_c)) < 0$.

2. One adjacent half_edge e_i of a vertex v penetrates some adjacent quarter_face F_j of a vertex V, where $F_j(\mathbf{d}(\mathbf{x}_c)) \geq 0$.

3. One adjacent half_edge E_j of a vertex V penetrates some adjacent quarter_face f_i of a vertex v, where $f_i(\mathbf{d}(\mathbf{x}_c)) < 0$.

4. One adjacent half_edge E_j of a vertex V penetrates some adjacent quarter_face f_i of a vertex v, where $f_i(\mathbf{d}(\mathbf{x}_c)) \geq 0$.

5. One adjacent half_edge e_i of a vertex v penetrates some adjacent half_edge E_j of a vertex V, where a vertex v is inside the fixed object.

6. One adjacent half_edge e_i of a vertex v penetrates some adjacent half_edge E_j of a vertex V, where a vertex v is not inside the fixed object.

7. One adjacent quarter_face f_i of a vertex v penetrates some adjacent quarter_face F_j of a vertex V.

Each condition is similar to the corresponding condition in a vertex-edge case. First, we consider Condition 1. The positional relationship between these two is updated as follows:

- $w_1 \cdot w_2 > 0$
- $w_1 = 0$
- $w_2 = 0$
- otherwise,

where

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = B_{j-1,j}^{-1} \begin{pmatrix} \mathbf{t}_i \cdot \mathbf{T}_{j-1} \\ \mathbf{t}_i \cdot \mathbf{T}_j \end{pmatrix}$$

$$B_{j-1,j} = \begin{pmatrix} 1 & \mathbf{T}_{j-1} \cdot \mathbf{T}_i \\ \mathbf{T}_{j-1} \cdot \mathbf{T}_i & 1 \end{pmatrix}.$$

In the first type, the penetration condition is formulated by Equation (B.11).

$$\Delta_{e_i E_{j-1}} < 0 \cap \Delta_{e_i E_j} < 0 \cap \Delta_{vF_j} < 0 \tag{B.11}$$

In the second type, the penetration condition is formulated by Equation (B.12).

$$\Delta_{vF_{j-1}} < 0 \cap \Delta_{vF_j} < 0 \tag{B.12}$$

In the third type, the penetration condition is formulated by Equation (B.13).

$$\Delta_{vF_j} < 0 \cap \Delta_{vF_{j+1}} < 0 \tag{B.13}$$

Actually, these two types of penetration are equivalent to the penetration in Condition 5. As the same manner in Condition 1 of a vertex-edge contact, we do not need to consider the fourth type.
If Equation (B.5) is satisfied, we should consider Condition 2, i.e., a half_edge e_i penetrates a quarter_face F_j.
Next, we consider Condition 3. As the same way in Condition 1, the positional relationship between these two is classified as follows:

- $W_1 \cdot W_2 > 0$
- $W_1 = 0$
- $W_2 = 0$
- otherwise,

where

$$\begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = b_{i-1,i}^{-1} \begin{pmatrix} \mathbf{t}_{i-1} \cdot \mathbf{T}_j \\ \mathbf{t}_i \cdot \mathbf{T}_j \end{pmatrix}.$$

In the first type, the penetration condition is formulated by Equation (B.14).

$$\Delta_{e_{i-1}E_j} < 0 \cap \Delta_{e_iE_j} < 0 \cap \Delta_{f_iV} < 0 \tag{B.14}$$

In the second type, the penetration condition is formulated by Equation (B.15).

$$\Delta_{f_{i-1}V} < 0 \cap \Delta_{f_iV} < 0 \tag{B.15}$$

In the third type, the penetration condition is formulated by Equation (B.16).

$$\Delta_{f_iV} < 0 \cap \Delta_{f_{i+1}V} < 0 \tag{B.16}$$

Actually, these two types of penetration are equivalent to the penetration in Condition 5. As the same manner in Condition 1 of a vertex-edge contact, we do not need to consider the fourth type.
If Equation (B.17) is satisfied, we should consider Condition 4, i.e., a half_edge E_j penetrates a quarter_face f_i.

$$\mathbf{n}_i \cdot \mathbf{T}_j = 0 \tag{B.17}$$

We have already considered Condition 5.
As the same manner in Condition 4 of a vertex-edge contact, we should consider Condition 6, if Equation (B.18) is satisfied.

$$\mathbf{t}_i \times \mathbf{T}_j = 0 \tag{B.18}$$

Such a penetration is formulated as Equation (B.19).

$$\begin{aligned} & l_{e_iF_j} > 0 \cap L_{e_iF_j} > 0 \cap \Delta_{e_iF_I} < 0 \\ & \qquad \cap\, l_{e_iF_{j+1}} > 0 \cap L_{e_iF_{j+1}} > 0 \cap \Delta_{e_iF_{II}} < 0 \end{aligned} \tag{B.19}$$

The equation means that a half_edge e_i penetrates two half_faces F_I and F_{II}. The half_face F_I has the same equation as a quarter_faces F_j and its boundary is an expansion of a half_edge E_{j-1}. As the same manner, the half_face F_{II} has the same equation as a quarter_faces F_{j+1} and its boundary is an expansion of a half_edge E_{j+1}. Section B.2.2 concretely describe the formulation for a half_edge-half_face point-contact.
If Equation (B.9) is satisfied, we should consider Consider 7, i.e., a quarter_face f_i penetrates a quarter_face F_j. These two penetration conditions are formulated below.
As a result, if all penetration conditions are represented by a union of Equation (B.11) and (B.14), the legal displacement is formulated by Equation (B.20). For another cases, we can easily formulate the displacement.

$$\begin{aligned} & \bigcap_{i,j}(\Delta_{e_iE_{j-1}} \geq 0 \cup \Delta_{e_iE_j} \geq 0 \cup \Delta_{vF_j} \geq 0) \\ & \qquad \cap \bigcap_{i,j}(\Delta_{e_{i-1}E_j} \geq 0 \cup \Delta_{e_iE_j} \geq 0 \cup \Delta_{f_iV} \geq 0) \end{aligned} \tag{B.20}$$

B.2.2 Singular Point-Contact

Half_Edge-Face Point-Contact

Consider the case that a half_edge e_i contacts a face F at a point $\mathbf{x}_c$ as shown in Fig. B.4. In this case $\mathbf{t}_i \cdot \mathbf{N} = 0$ must be satisfied. This contact is accompanied by a vertex-face point-contact. Note that the vertex-face point-contact has already been included in the legal displacement condition in normal use. In contrast to an edge-face point-contact case, we should consider the range of l in this case.
We formulate a not-penetration condition, i.e., the legal displacement, by inverting a penetration condition. First, we consider the case that $k_i\mathbf{N} \cdot \mathbf{p}_i + \mathbf{t}_i^T M \mathbf{t}_i \neq 0$.

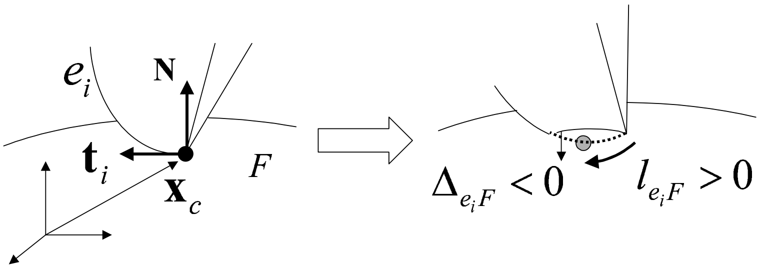


Figure B.4: A vertex-face contact and a half_edge-face contact

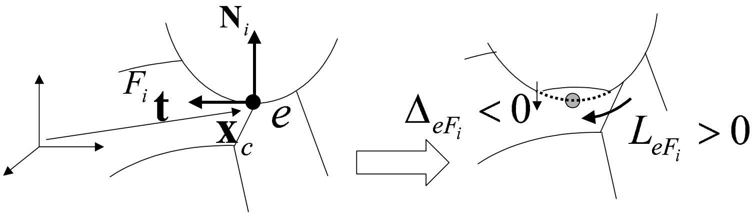


Figure B.5: An edge-edge point-contact and an edge-half_face point-contact

As mentioned above, l_{e_iF} is calculated by Equation (3.37) in an edge-face contact case. Because the vertex is not inside the fixed object, the penetration requires $l_{e_iF} > 0$, which means that the point which minimizes Δ_{e_iF} is on the half_edge (See Fig. B.4). Of course, the penetration requires $\Delta_{e_iF}|_{l'(0)=l_{e_iF}} < 0$. As the result, a not-penetration condition is formulated by Equation (B.21). Note that l_{e_iF} is linear with respect to $\mathbf{s}_1$ and $\mathbf{s}_2$.

$$l_{e_iF} \leq 0 \cup \Delta_{e_iF}|_{l'(0)=l_{e_iF}} \geq 0 \tag{B.21}$$

Next, we consider the case that $k_i\mathbf{N}\cdot\mathbf{p}_i + \mathbf{t}_i^T M\mathbf{t}_i = 0$. In this case, Equation (3.36) is constant with respect to $l'(0)$, and its value is uniquely decided from $\mathbf{s}_1$ and $\mathbf{s}_2$. Because the vertex is not inside the fixed object, the penetration requires Equation (3.36) < 0, which means that the bigger $l'(0)$ becomes, the less Δ_{e_iF} becomes. As a result, a not-penetration condition is formulated by Equation (B.22).

$$\mathbf{N}\cdot(\mathbf{s}_1 \times \mathbf{t}_i) + \mathbf{t}_i^T M(\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) \geq 0 \tag{B.22}$$

Edge-Half_Face and Edge-Quarter_Face Point-Contacts

Consider the case that an edge e contacts a half_face F_j at a point $\mathbf{x}_c$ as shown in Fig. B.5. In this case $\mathbf{t} \cdot \mathbf{N}_j = 0$ must be satisfied. This contact is accompanied to an edge-edge point-contact. Note that the edge-edge point-contact has already been included in the legal displacement condition in normal use. In contrast to an edge-face point-contact case, we should consider that the domain of $F_j(\mathbf{X})$, i.e., $L_\perp = (\mathbf{e}(l) - \mathbf{x}_c) \cdot \mathbf{N}_{\perp j} \geq 0$.
The derivative of $L_\perp$ with respect to $\Delta\theta$ is represented by Equation (B.23).

$$L'_\perp = \left(\mathbf{e}'(l) + \frac{\partial \mathbf{e}(l)}{\partial l} l'\right) \cdot \mathbf{N}_{\perp j} \tag{B.23}$$

By substituting $\Delta\theta = 0$ to Equation (B.23), Equation (B.24) is obtained.

$$L'_\perp(0) = (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2 + \mathbf{t}l'(0)) \cdot \mathbf{N}_{\perp j} \tag{B.24}$$

Now we consider the case that $k\mathbf{N}_j \cdot \mathbf{p} + \mathbf{t}^T M_j \mathbf{t} \neq 0$. In an edge-face contact case, $l'(0) = l_{eF_j}$. By substituting l_{e_jF} to $l'(0)$ in Equation (B.24), Equation (B.25) is obtained.

$$L_{eF_j} = (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2 + \mathbf{t}l_{eF_j}) \cdot \mathbf{N}_{\perp j} \tag{B.25}$$

Note that L_{eF_j} is linear with respect to $\mathbf{s}_1$ and $\mathbf{s}_2$, because l_{eF_j} is linear with respect to these two.
Because the distance between the two edges is not less than zero, the penetration requires $L_{eF_j} > 0$. Of course, the penetration requires $\Delta_{eF_j}|_{l'(0)=l_{eF_j}} < 0$. As the result, a not-penetration condition is formulated by Equation (B.26).

$$L_{eF_j} \leq 0 \cup \Delta_{eF_j}|_{l'(0)=l_{eF_j}} \geq 0 \tag{B.26}$$

Next, consider the case that an edge e contacts a quarter_face F_j at a point $\mathbf{x}_c$. In this case $\mathbf{t} \cdot \mathbf{N}_j = 0$ must also be satisfied and we should consider that the domain of $F_j(\mathbf{X})$, i.e., $W_1 > 0$ and $W_2 > 0$ must be satisfied after the displacement, where

$$\mathbf{e}_d(l) - \mathbf{x}_c = W_1\mathbf{T}_{j-1} + W_2\mathbf{T}_j + W_3\mathbf{N}_j. \tag{B.27}$$

By substituting $\Delta\theta = 0$ to the derivative of Equation (B.27) with respect to $\Delta\theta$, Equation (B.28) is obtained.

$$\left(\mathbf{e}'(l) + \frac{\partial \mathbf{e}(l)}{\partial l} l'\right) = W'_1(0)\mathbf{T}_{j-1} + W'_2(0)\mathbf{T}_j + W'_3(0)\mathbf{N}_j \tag{B.28}$$

In this case, $l'(0) = l_{eF_j}$. Because $W_1(0) = W_2(0) = 0$, $W'_1(0) > 0$ and $W'_2(0) > 0$ must be satisfied. From Equation (B.28), $W'_1(0)$ and $W'_2(0)$ are solved by Equation (B.29).

$$\begin{pmatrix} W'_1(0) \\ W'_2(0) \end{pmatrix} = B^{-1}_{j-1,j} \begin{pmatrix} (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2 + \mathbf{t}l_{eF_j}) \cdot \mathbf{T}_{j-1} \\ (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2 + \mathbf{t}l_{eF_j}) \cdot \mathbf{T}_j \end{pmatrix} \tag{B.29}$$

Note that $W'_1(0)$ and $W'_2(0)$ is linear with respect to $\mathbf{s}_1$ and $\mathbf{s}_2$, because $B_{j-1,j}$ is constant and l_{eF_j} is linear with respect to these two.
Of course, the penetration requires $\Delta_{eF_j}|_{l'(0)=l_{eF_j}} < 0$. As the result, a not-penetration condition is formulated by Equation (B.30).

$$W'(0) \leq 0 \cup W'(1) \leq 0 \cup \Delta_{eF_j}|_{l'(0)=l_{eF_j}} \geq 0 \tag{B.30}$$

We have not formulated the legal displacement yet in the two point-contact primitives, when $k\mathbf{N}_j \cdot \mathbf{p} + \mathbf{t}^T M_j \mathbf{t} = 0$. The formulation is an outstanding problem. Fortunately, the cases seldom appear.

Half_face-Face and Quarter_face-Face Point-Contacts

Consider the case that a half_face f_i contacts a face F at a point $\mathbf{x}_c$. In this case $\mathbf{n}_i \times \mathbf{N} = \mathbf{0}$ must be satisfied. This contact is accompanied by an edge-face point-contact. Note that the contact has already been included in the legal displacement condition in normal use. In contrast to a face-face point-contact case, we should consider that the domain of $f_i(\mathbf{x})$ after the displacement, i.e., $x_\perp = (\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) \cdot (R\mathbf{n}_{\perp i}) \geq 0$.
The derivative of $x_\perp$ with respect to $\Delta\theta$ is represented by Equation (B.31).

$$x'_\perp = (\mathbf{x}' - \mathbf{d}'(\mathbf{x}_c)) \cdot (R\mathbf{n}_{\perp i}) + (\mathbf{x} - \mathbf{d}(\mathbf{x}_c)) \cdot (R'\mathbf{n}_{\perp i}) \tag{B.31}$$

By substituting $\Delta\theta = 0$ to Equation (B.31), Equation (B.32) is obtained.

$$x'_\perp(0) = (\mathbf{x}'(0) - (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2)) \cdot \mathbf{n}_{\perp i} \tag{B.32}$$

In a face-face point-contact case, $\mathbf{x}'(0) = \mathbf{x}_{min}$ in Equation (3.46). By substituting $\mathbf{x}_{min}$ to $\mathbf{x}'(0)$ in Equation (B.32), Equation (B.33) is obtained.

$$x_{f_iF} = -((m_i + M)^{-1}(M(\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) + \mathbf{N} \times \mathbf{s}_1)) \cdot \mathbf{n}_{\perp i} \tag{B.33}$$

Note that x_{f_iF} is linear with respect to $\mathbf{s}_1$ and $\mathbf{s}_2$.

Because the distance between the two edges is not less than zero, the penetration requires $x_{f_iF} > 0$. Of course, it requires $\Delta_{f_iF}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} < 0$. As the result, a not-penetration condition is formulated by Equation (B.34).

$$x_{f_iF} \leq 0 \cup \Delta_{f_iF}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} \geq 0 \tag{B.34}$$

Next, consider the case that a quarter_face f_i contacts a face F at a point $\mathbf{x}_c$. In this case $\mathbf{n}_i \times \mathbf{N} = \mathbf{0}$ must also be satisfied. Let two edges e_{i-1} and e_i be adjacent to a quarter_face f_i. Then, we should consider the domain of $f_i(\mathbf{x})$ after the displacement, i.e., $w_1 > 0$ and $w_2 > 0$, where

$$\mathbf{x} - \mathbf{d}(\mathbf{x}_c) = w_1 R\mathbf{t}_{i-1} + w_2 R\mathbf{t}_i + w_3 R\mathbf{n}_i. \tag{B.35}$$

The derivative of Equation (B.35) with respect to $\Delta\theta$ is represented by Equation (B.36).

$$\begin{aligned}\mathbf{x}' - \mathbf{d}'(\mathbf{x}_c) = {} & w_1' R\mathbf{t}_{i-1} + w_1 R'\mathbf{t}_{i-1} \\ & + w_2' R\mathbf{t}_i + w_2 R'\mathbf{t}_i + w_3' R\mathbf{n}_i + w_3 R'\mathbf{n}_i\end{aligned} \tag{B.36}$$

By substituting $\Delta\theta = 0$ to Equation (B.36), Equation (B.37) is obtained. Note that $w_1(0) = w_2(0) = w_3(0) = 0$.

$$\mathbf{x}'(0) - (\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) = w_1'(0)\mathbf{t}_{i-1} + w_2'(0)\mathbf{t}_i + w_3'(0)\mathbf{n}_i \tag{B.37}$$

In this case, $\mathbf{x}'(0) = \mathbf{x}_{min}$. Because $w_1(0) = w_2(0) = 0$, $w_1'(0) > 0$ and $w_2'(0) > 0$ must be satisfied. From Equation (B.37), $w_1'(0)$ and $w_2'(0)$ are solved by Equation (B.38).

$$\begin{pmatrix} w_1'(0) \\ w_2'(0) \end{pmatrix} = b_{i-1,i}^{-1} \begin{pmatrix} -((m_i + M)^{-1}(M(\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) + \mathbf{N} \times \mathbf{s}_1) \cdot \mathbf{t}_{i-1} \\ -((m_i + M)^{-1}(M(\mathbf{s}_1 \times \mathbf{x}_c + \mathbf{s}_2) + \mathbf{N} \times \mathbf{s}_1) \cdot \mathbf{t}_i \end{pmatrix} \tag{B.38}$$

Note that $w_1'(0)$ and $w_2'(0)$ is linear with respect to $\mathbf{s}_1$ and $\mathbf{s}_2$, because $b_{i-1,i}$ is constant.

Of course, the penetration requires $\Delta_{f_iF}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} < 0$. As the result, a not-penetration condition is formulated by Equation (B.39).

$$w_1'(0) \leq 0 \cup w_2'(0) \leq 0 \cup \Delta_{f_iF}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} \geq 0 \tag{B.39}$$

Half_Face-Half_Face and Quarter_Face-Half_Face Point-Contacts

Consider the case that a half_face f_i contacts a half_face F_j at a point $\mathbf{x}_c$. In this case, $\mathbf{n}_i \times \mathbf{N}_j = \mathbf{0}$ must be satisfied. This contact is accompanied by an edge-edge point-contact in normal use. The contact has already been included in the legal displacement condition, i.e., the distance between the two edges is not less than zero. First the penetration requires Equation $(B.33) > 0$, considering the domain of $f_i(\mathbf{x})$.

Now we consider the domain of $F_j(\mathbf{X})$. Then we should consider the sign of Equation (B.40).

$$X_\perp = \mathbf{X} \cdot \mathbf{N}_{\perp j} \tag{B.40}$$

By substituting $\Delta\theta = 0$ to the derivative of Equation (B.40) with respect to $\Delta\theta$, Equation (B.41) is obtained.

$$X'_\perp(0) = \mathbf{X}'(0) \cdot \mathbf{N}_{\perp j} \tag{B.41}$$

By substituting $\mathbf{x}_{min}$ to $\mathbf{X}'(0)$ in Equation (B.41), Equation (B.42) is obtained.

$$X_{f_iF_j} = \mathbf{x}_{min} \cdot \mathbf{N}_{\perp j} \tag{B.42}$$

The penetration requires $X_{f_iF_j} > 0$ and $\Delta_{f_iF_j}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} < 0$. As the result, the not-penetration condition is formulated by Equation (B.43).

$$x_{f_iF_j} \leq 0 \cup X_{f_iF_j} \leq 0 \cup \Delta_{f_iF_j}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} \geq 0 \tag{B.43}$$

Next, consider the case that a quarter_face f_i contacts a half_face F_j at a point $\mathbf{x}_c$. In this case, $\mathbf{n}_i \times \mathbf{N}_j = \mathbf{0}$ must also be satisfied. First the penetration requires $w'_1(0) > 0$ and $w'_2(0) > 0$, considering the domain of $f_i(\mathbf{x})$. These two terms are calculated by Equation (B.38).

Next the domain of $F_j(\mathbf{X})$, $X_{f_iF_j} > 0$ and $\Delta_{f_iF_j}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} < 0$ are must be satisfied. As the result, the not-penetration condition is formulated by Equation (B.44).

$$w'_1(0) \leq 0 \cup w'_2(0) \leq \cup X_{f_iF_j} \leq 0 \cup \Delta_{f_iF_j}|_{\mathbf{x}'(0)=\mathbf{x}_{min}} \geq 0 \tag{B.44}$$

Half_Edge-Half_Face and Half_Edge-Quarter_Face Point-Contacts

Consider the case that a half_edge e_i contacts a half_face F_j at a point $\mathbf{x}_c$. In this case, $\mathbf{t}_i \cdot \mathbf{N}_j = 0$ must be satisfied. This contact is accompanied by a vertex-edge or vertex-vertex point-contact. We only need to formulate the penetration, where the half_edge penetrates the fixed object and the vertex which is its terminal

is not inside the fixed object. Unfortunately, we have not formulated the legal displacement in the case $k_i\mathbf{N}_j \cdot \mathbf{p}_i + \mathbf{t}_i^T M_j \mathbf{t}_i = 0$. Therefore, we assume that $k_i\mathbf{N}_j \cdot \mathbf{p}_i + \mathbf{t}_i^T M_j \mathbf{t}_i \neq 0$.
From the penetration condition, Equation (B.45) must be satisfied.

$$l_{e_iF_j} > 0 \tag{B.45}$$

Because the half_edge penetrates the half_face, Equation (B.46) must be satisfied. Note that this condition is equal to the condition that an edge must penetrate a half_face.

$$L_{e_iF_j} > 0 \cap \Delta_{e_iF_j}|_{l'(0)=l_{e_iF_j}} < 0 \tag{B.46}$$

As a result, such a penetration condition is formulated by Equation (B.47).

$$l_{e_iF_j} > 0 \cap L_{e_iF_j} > 0 \cap \Delta_{e_iF_j}|_{l'(0)=l_{e_iF_j}} < 0 \tag{B.47}$$

Next, consider the case that a half_edge e_i penetrates a quarter_face F_j at a point $\mathbf{x}_c$. In this case, $\mathbf{t}_i \cdot \mathbf{N}_j = 0$ must be satisfied. This contact is also accompanied by a vertex-edge or vertex-vertex point-contact and we only need to formulate the penetration, where the half_edge penetrates the fixed object and the vertex which is its terminal is not inside the fixed object. Let an edge E_{j-1} and E_j be adjacent to a quarter_face F_j. Because the half_edge penetrates the quarter_face, Equation (B.48) must be satisfied, where $W_1'(0)$ and $W_2'(0)$ is calculated by Equation (B.29). Note that this condition is equal to the condition that an edge penetrates a quarter_face.

$$W_1'(0) > 0 \cap W_2'(0) > 0 \cap \Delta_{e_iF_j} < 0 \tag{B.48}$$

As a result, such a penetration condition is formulated by Equation (B.49).

$$l_{e_iF_j} > 0 \cap W_1'(0) > 0 \cap W_2'(0) > 0 \cap \Delta_{e_iF_j} < 0 \tag{B.49}$$

Quarter_Face-Quarter_Face Point-Contact

Consider the case that a quarter_face f_i contacts a quarter_face F_j at a point $\mathbf{x}_c$. In this case, $\mathbf{n}_i \times \mathbf{N}_j = 0$ must be satisfied. This contact is also accompanied by a vertex-edge or vertex-vertex contact. We only need to formulate the penetration where a quarter_face f_i locally penetrates a quarter_face F_j of the fixed object and the adjacent half_edges and vertex are not inside the object.
Let edge e_{i-1} and e_i be adjacent to a quarter_face f_i. Considering the domain of $f_i(\mathbf{x})$ after the displacement, Equation (B.50) must be satisfied, where $w_1'(0)$ and

$w_2'(0)$ is calculated by Equation (B.38).

$$w_1'(0) > 0 \cap w_2'(0) > 0 \tag{B.50}$$

Let edge E_{j-1} and E_j be adjacent to a quarter_face F_j. Next we consider the domain of $F_j(\mathbf{X})$. Then $W_1 > 0$ and $W_2 > 0$ must be satisfied after the displacement, where

$$\mathbf{X} - \mathbf{x}_c = W_1\mathbf{T}_{j-1} + W_2\mathbf{T}_j + W_3\mathbf{N}_j. \tag{B.51}$$

By substituting $\Delta\theta = 0$ to the derivative of Equation (B.51) with respect to $\Delta\theta$, Equation (B.52) is obtained.

$$\mathbf{X}'(0) = W_1'(0)\mathbf{T}_{j-1} + W_2'(0)\mathbf{T}_j + W_3'(0)\mathbf{N}_j \tag{B.52}$$

In this case, $\mathbf{X}'(0) = \mathbf{x}_{min}$. Because $W_1(0) = W_2(0) = 0$, $W_1'(0) > 0$ and $W_2'(0) > 0$ must be satisfied. From Equation (B.52), $W_1'(0)$ and $W_2'(0)$ are solved by Equation (B.53).

$$\begin{pmatrix} W_1'(0) \\ W_2'(0) \end{pmatrix} = B_{j-1,j}^{-1} \begin{pmatrix} \mathbf{x}_{min} \cdot \mathbf{T}_{j-1} \\ \mathbf{x}_{min} \cdot \mathbf{T}_j \end{pmatrix} \tag{B.53}$$

Note that $W_1'(0)$ and $W_2'(0)$ is linear with respect to $\mathbf{s}_1$ and $\mathbf{s}_2$, because $B_{j-1,j}$ is constant and $\mathbf{x}_{min}$ is linear with respect to these two.
Of course $\Delta_{f_iF_j} < 0$ must be satisfied. As a result, the penetration condition is formulated by Equation (B.54).

$$w_1'(0) > 0 \cap w_2'(0) > 0 \cap W_1'(0) > 0 \cap W_2'(0) > 0 \cap \Delta_{f_iF_j} < 0 \tag{B.54}$$

B.3 Summary of Formulation

The approximations with respect to the legal displacement are classified into the following two types:

- The equation is represented by a product of the some components as Equation (B.55) (referred to as a *basic form*).

$$\bigcap_i g_i(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \Delta\theta) \geq 0 \tag{B.55}$$

- The equation is represented by a union of some basic forms as Equation (B.56).

$$\bigcup_i \bigcap_j g_{ij}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \Delta\theta) \geq 0 \tag{B.56}$$

The legal displacement in a basic contact relation can be represented by a basic form. Contact relations where the legal displacement cannot be represented by a basic form are very *unstable* like singular contact relations in Chapter 2. The contact relations instantaneously appear through an assembly task.

Furthermore, each component $g_i(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \Delta\theta)$ is classified into the following two types:

- Include a term $\Delta\theta$ (for example, Δ_{vF})
- Not include the term (for example, l_{eF})

The former component is represented as Equation (B.57) as mentioned in Chapter 3.

$$g_i = \begin{pmatrix} \mathbf{x}_i \times \mathbf{n}_i \\ \mathbf{n}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \Delta\theta + \left(\begin{pmatrix} \mathbf{x}_i \times \mathbf{n}_i \\ \mathbf{n}_i \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_3 \\ \mathbf{s}_4 \end{pmatrix} + h_i(\mathbf{s}_1, \mathbf{s}_2) \right) \frac{\Delta\theta^2}{2} \tag{B.57}$$

The latter component is represented as Equation (B.58).

$$g_i = \begin{pmatrix} \mathbf{w}_{1i} \\ \mathbf{w}_{2i} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \tag{B.58}$$

The latter component appears if, and only if, the contact relation includes some singular point-contact primitives. Fortunately, the latter component is similar to a coefficient of $\Delta\theta$ of the former component. That similarity is useful to solve Equation (B.55) and (B.56).

Appendix C

Convert Second Order Screw Representation to Object Configuration

Now we assume that the axis direction is constant to an rotation angle θ, that is, one of the screw vector $\mathbf{s}_3$ is equal to zero. Given the screw vector, we introduce the transformation matrix $[\Theta(\theta), \mathbf{t}(\theta)]$, where θ is an amount of rotation, $\Theta(\theta)$ is a 3×3 orthogonal matrix which represents the transformation with respect to orientation, and $\mathbf{t}(\theta)$ is a three-dimensional vector which represents the transformation with respect to location.

From the screw vector, the axis direction of rotation $\mathbf{a}$, the initial position of the instantaneous center of rotation $\mathbf{c}(0)$ and the derivative of the position $\mathbf{c}'(0)$ with respect to θ on $\theta = 0$ can be calculated, where $\mathbf{c}(0) \cdot \mathbf{a} = 0$ and $\mathbf{c}'(0) \cdot \mathbf{a} = 0$. Of course, Equation (C.1) is always satisfied, where $R(\mathbf{a}, \theta)$ represent rotation about the axis $\mathbf{a}$ by θ radian.

$$\Theta(\theta) = R(\mathbf{a}, \theta) \tag{C.1}$$

Therefore, we have only to solve $\mathbf{t}(\theta)$.

Now we assume that the derivatives of all functions are constant with respect to θ. Therefore the center is formulated by Equation (C.2).

$$\mathbf{c}(\theta) = \mathbf{c}(0) + \mathbf{c}'(0)\theta \tag{C.2}$$

Now, we consider the relationship between the instantaneous center of rotation and the transformation matrix. The difference $[\Delta\Theta, \Delta\mathbf{t}]$ between the two transformation matrices $[R(\mathbf{a}, \theta + \Delta\theta), \mathbf{t}(\theta + \Delta\theta)]$ and $[R(\mathbf{a}, \theta), \mathbf{t}(\theta)]$ is represented by

Equation (C.3), where $\Delta\theta \neq 0$.

$$\begin{aligned} \Delta\Theta &= R(\mathbf{a}, \Delta\theta) \\ \Delta\mathbf{t} &= \mathbf{t}(\theta + \Delta\theta) - R(\mathbf{a}, \Delta\theta)\mathbf{t}(\theta) \end{aligned} \tag{C.3}$$

The finite center of rotation $\mathbf{c}$ satisfies Equation (C.4), where I is a unit matrix.

$$(I - \Delta\Theta)\mathbf{c} = \Delta\mathbf{T} \tag{C.4}$$

Note that the instantaneous center of rotation is defined by the limit $\lim_{\Delta\theta\to 0} \mathbf{c}$.
Now we temporarily consider the equation on the coordinate system of which the direction of the z-axis is equal to the axis direction. In this case, the values with respect to the z-axis is meaningless on the equation. From now on, we ignore that part.
Then, because $I - \Delta\Theta$ is a full-rank matrix, $\mathbf{c}$ is calculated by Equation (C.5), where $\mathbf{z} = (0, 0, 1)^T$.

$$\mathbf{c} = \mathbf{t}(\theta) - \Delta\theta(I - R(\mathbf{z}, \Delta\theta))^{-1} \frac{(\mathbf{t}(\theta + \Delta\theta) - \mathbf{t}(\theta))}{\Delta\theta} \tag{C.5}$$

We consider the limit $\lim_{\Delta\theta\to 0} \Delta\theta(I - R(\mathbf{z}, \Delta\theta))^{-1}$. $I - R(\Delta\theta)$ can concretely be written as Equation (C.6).

$$I - R(\mathbf{z}, \Delta\theta) = \begin{pmatrix} 1 - \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & 1 - \cos\Delta\theta \end{pmatrix} \tag{C.6}$$

By applying the inverse operation to Equation (C.6), Equation (C.7) is obtained.

$$(I - R(\mathbf{z}, \Delta\theta))^{-1} = \begin{pmatrix} \frac{1}{2} & \frac{\sin\Delta\theta}{2(1-\cos\Delta\theta)} \\ -\frac{\sin\Delta\theta}{2(1-\cos\Delta\theta)} & \frac{1}{2} \end{pmatrix} \tag{C.7}$$

Note that

$$\lim_{\Delta\theta\to 0} \frac{\Delta\theta \sin\Delta\theta}{2(1 - \cos\Delta\theta)} = 1.$$

Therefore, Equation (C.8) is obtained.

$$\lim_{\Delta\theta\to 0} \Delta\theta(I - R(\mathbf{z}, \Delta\theta))^{-1} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{C.8}$$

Because the instantaneous center of the rotation is formulated by $\lim_{\Delta\theta\to 0} \mathbf{c}$, Equation (C.9) must be satisfied.

$$\mathbf{t}'(\theta) = A\mathbf{t}(\theta) - A\mathbf{c}(0) - A\mathbf{c}'(0)\theta \tag{C.9}$$

$$A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

By solving Equation (C.9), Equation (C.10) is obtained, where $\mathbf{b}$ is an integral constant.

$$\mathbf{t}(\theta) = \mathbf{c}(0) + R\left(\mathbf{z}, \frac{\pi}{2} - \theta\right)\mathbf{c}'(0) - \mathbf{c}'(0)\theta + R(\mathbf{z}, \theta)\mathbf{b} \tag{C.10}$$

Because $\mathbf{t}(0) = 0$, Equation (C.11) is obtained.

$$\mathbf{t}(\theta) = (I - R(\mathbf{z}, \theta))\left(\mathbf{c}_0 + R\left(\mathbf{z}, \frac{\pi}{2}\right)\mathbf{c}'(0)\right) - \mathbf{c}(0)'\theta \tag{C.11}$$

By converting the original coordinate system, we finally obtain Equation (C.12).

$$\mathbf{t}(\theta) = (I - R(\mathbf{a}, \theta))\left(\mathbf{c}(0) + R\left(\mathbf{a}, \frac{\pi}{2}\right)\mathbf{c}'(0)\right) - \mathbf{c}(0)'\theta \tag{C.12}$$

References

[1] *http://www.sony.net/Products/aibo/.*

[2] *http://world.honda.com/ASIMO/.*

[3] *http://www.kawada.co.jp/ams/hrp-2/index_e.html.*

[4] *http://www.humanoid.waseda.ac.jp/index.html.*

[5] J. C. Latombe : "Robot motion planning," *Kluwer Academic Publishers.*

[6] K. Ikeuchi and T. Suehiro : "Toward an assembly plan from observation part i: Task recognition with polyhedral objects," *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 3, Jun. 1994.

[7] Y. Kuniyoshi, M. Inaba, and H. Inoue : "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 6, Dec. 1994.

[8] S. Schaal : "Is imitation learning the route to humanoid robots?," *Trends in Cognitive Sciences*, Vol. 3, pp. 233 – 242, 1999.

[9] R. Dillmann and M. Bordegoni : "Learning robot behavior and skills based on human demonstration and advice: The machine learning paradigm," *Int. Symp. on Robotics Research*, 1999.

[10] V. Gallese and A. Goldman : "Mirror neurons and the simulation theory of mind-reading," *Trends in Cognitive Sciences*, Vol. 2, No. 12, pp. 493 – 501, 1998.

[11] C. C. Adams : "The Knot Book: An Elementary Introduction to the Matematicla Theory of Knots," New York: W. H. Freeman, 1994.

[12] M. Skubic and R. Volz : "Acquiring robust, force-based assembly skills from human demonstration," *IEEE Trans. on Robotics and Automation*, Vol. 16, No. 6, 2000.

[13] T. Fukuda, M. Nakaoka, T. Ueyama, and Y. Hasegawa : "Direct teaching and error recovery method for assembly task based on a transition process of a constraint condition," *IEEE Int. Conf. on Robotics and Automation*, pp. 1518 – 1523, 2001.

[14] T. Takahashi, H. Ogata, and S. Muto : "Robotic assembly operation based on task-level teaching in virtual reality," *IEEE Int. Conf. on Robotics and Automation*, 1992.

[15] H. Onda, T. Ogasawara, H. Hirukawa, K. Kitagaki, A. Nakamura, and H. Tsukune : "A telerobotics system using planning functions based on manipulation skills and teaching-by-demonstration technique in vr," *Journal of the Robotics Society of Japan*, Vol. 18, No. 7, pp. 979 – 994, 2000.

[16] Y. Maeda, N. Ishido, H. Kikuchi, and T. Arai : "Teaching of grasp/graspless manipulation for industrial robots by human demonstration," *IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 1523 – 1528, 2002.

[17] M. Ehrenmann, R. Zollner, S. Knoop, and R. Dillmann : "Sensor fusion approaches for observation of user actions in programing by demonstration," *IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 2001.

[18] K. Ogawara, S. Iba, T. Tanuki, H. Kimura, and K. Ikeuchi : "Acquiring hand-action models by attention point analysis," *IEEE Int. Conf. on Robotics and Automations*, 2001.

[19] B. Keni, K. Ogawara, K. Ikeuchi, and R. Dillmann : "A hidden markov model based sensor fusion approach for recgnizing continuous human grasping sequences," *Third IEEE Int. Conf. on Humanoid Robots*, 2003.

[20] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato : "A kendama learning robot based on bi-directional theory," *Neural Networks*, Vol. 9, No. 8, pp. 1281 – 1302, 1996.

[21] H. Miyamoto and M. Kawato : "A tennis serve and upswing learning robot based on bi-directional theory," *Neural Networks*, pp. 1331 – 1344, 1998.

[22] S. Schaal, C. G. Atkeson, and S. Vijayakumar : "Real-time robot learning with locally weighted statistical learning," *IEEE Int. Conf. on Robotics and Automation*, 2000.

[23] T. Shibata and S. Schaal : "Fast learning of biomimetic oculomotor control with nonparametri8c regression networks," *IEEE Int. Conf. on Robotics and Automation*, pp. 3847 – 3854, 2000.

[24] S. Vijayakumar and S. Schaal : "Fast and efficient incremental learning for high-dimensional movement systems," *IEEE Int. Conf. on Robotics and Automation*, 2000.

[25] A. J. Ijspeert, J. Nakanishi, and S. Schaal : "Movement imitation with nonlinear dynamical systems in humanoid robots," *IEEE Int. Conf. on Robotics and Automation*, pp. 1398 – 1403, 2002.

[26] T. Inamura, Y. Nakamura, H. Ezaki, and I. Toshima : "Imitation and primitive symbol acquisition of humanoids by the integrated mimesis loop," *IEEE Int. Conf. on Robotics and Automation*, pp. 4208 – 4213, 2001.

[27] C. G. Atkeson and S. Schaal : "Robot learning from demonstration," *Int. Conf. on Machine Learning*, Vol. 2, No. 12, pp. 12 – 20, 1997.

[28] S. B. Kang and K. Ikeuchi : "Toward automatic robot instruction from perception - recognizing a grasp from observation," *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 4, pp. 432 – 443, Aug. 1993.

[29] M. Tsuda, T. Takahashi, and H. Ogata : "Generation of an assembly-task model analyzing human demonstration," *Journal of the Robotics Society of Japan*, Vol. 18, No. 4, pp. 535 – 544, 2000.

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

[31] R. E. Jones, R. H. Wilson, and T. L. Calton : "On constraints in assembly planning," *IEEE Trans. on Robotics and Automaiton*, Vol. 14, No. 6, pp. 849 – 963, 1998.

[32] S. Hirai, H. Asada, and H. Tokumaru : "Kinematic analysis of contact state transitions in assembly operations and automatic generation of transition net-

work," *Journal of the Society of Instrument and Control Engineers*, Vol. 24, No. 4, 1998.

[33] J. Xiao and X. Ji : "A divide-and-merge approach to automatic generation of contact states and planning of contact motion," *IEEE Int. Conf. on Robotics and Automation*, pp. 750 – 756, 2000.

[34] B. B. Goeree, E. D. Fasse, and M. M. Marefat : "Determining feasible contact state of pairs of spatial polyhedra," *IEEE Int. Conf. on Robotics and Automation*, pp. 1396 – 1401, 2000.

[35] F. Pan and J. M. Schimmels : "Efficient contact state graph generation for assembly application," *IEEE Int. Conf. on Robotics and Automation*, pp. 2592 – 2598, 2003.

[36] C. Ong and E. G. Gilbert : "Growth distance: New measures for object separation and penetration," *IEEE Int. Trans. on Robotics and Automation*, Vol. 12, No. 6, pp. 888 – 903, 1996.

[37] Y. Yokokohji : "Classification of contact states and planning of assembly sequence," *Journal of the Robotics Society of Japan*, Vol. 11, No. 2, pp. 185 – 191, 1993.

[38] Y. Yu and T. Yoshikawa : "Evaluation of contact stability between objects," *Journal of the Robotics Society of Japan*, Vol. 18, No. 7, pp. 1026 – 1033, 2000.

[39] M. Uchiyama and K. Imahashi : "Coimputer-aided rule generation for a rule-based artificial dexterity system," *Journal of the Robotics Society of Japan*, Vol. 12, No. 3, pp. 459 – 465, 1994.

[40] B. J. McCarragher and H. Asada : "A discrete event approach to the control of robotic assembly tasks," *IEEE Int. Conf. on Robotics and Automation*, pp. 331 – 336, 1997.

[41] H. Hirukawa : "On motion planning of polyhedra in contact," *WAFR*, 1996.

[42] X. Ji and J. Xiao : "Planning motion compliant to complex contact states," *IEEE Int. Conf. on Robotics and Automation*, pp. 1512 – 1517, 2001.

[43] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars : "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 4, pp. 566 – 580, Aug. 1996.

[44] M. T. Mason : "Compliance and force control for computer controlled manipulators," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 11, No. 6, Jun. 1981.

[45] John H. Challis : "Estimation of the finite center of rotation in planar movements," *Medical Engineering and Physics*, Vol. 23, pp. 227 – 233, 2001.

[46] German K. M. Cheung, Simon Baker, and Takeo Kanade : "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture," *Proc. of CVPR*, 2003.

[47] H. Wakamatsu and T. Wada : "Modelling of string objects for their manipulation," *Journal of the Robotics Society of Japan*, Vol. 16, No. 2, pp. 145 – 148, 1998.

[48] M. Inaba and H. Inoue : "Hand eye coordination in rope handling," *Journal of Robotics Society Japan*, Vol. 3, No. 6, pp. 32 – 41, 1985.

[49] J. E. Hopcroft, J. K. Kearne, and D. B. Krafft : "A case study of flexible object manipulation," *Int. Journal of Robotics Research*, Vol. 10, No. 1, pp. 41 – 50, 1991.

[50] J. Wolter and E. Kroll : "Toward assembly sequence planning with flexible parts," *IEEE Int. Conf. on Robotics and Automation*, pp. 1517–1524, 1996.

[51] M. Yamada, R. Budiarto, H. Seki, and H. Itoh : "Topology of cat's cradle diagrams and its characterization using knot polynomials," *Journal of Information Processing Society of Japan*, Vol. 38, No. 8, pp. 1873 – 1582, 1997.

[52] H. Hirukawa, T. Matsui, and K. Takase : "Automatic determination of possible velocity and applicable force of frictionless objects in contact from a geometric model," *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 3, pp. 309 – 322, 1994.

[53] M. S. Ohwovoriole and B. Roth : "An extension of screw theory," *Journal of Mechanical Design*, Vol. 103, pp. 725 – 735, Oct. 1981.

[54] H. W. Kuhn and A. W. Tucker : "Linear inequalities and related systems," *Annals. of Mathematics Studies*, Vol. 38, , 1956.

[55] J. Miura and K. Ikeuchi : "Task-oriented generation of visual sensing strategies in assembly tasks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 2, February 1998.

[56] *http://www.komatsu.co.jp/research/study56.htm.*

[57] M. D. Wheeler and K. Ikeuchi : "Sensor modeling, probabilistic hypothesis generation, and robust localization for object recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, pp. 252 – 265, 1995.

[58] P. Besl and N. McKay : "A method for registration of 3-d shapes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, 1992.

[59] T. Suehiro and K. Ikeuchi : "Towards an assembly plan from observation: Part ii: Correction of motion parameters based on fact contact constraints," *IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 2096 – 2102, Jul. 1992.

[60] J. Xiao and L. Zhang : "Toward obtaining all possible contacts - growing a polyhedron by its location uncertainty," *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 4, pp. 553 – 565, 1996.

[61] J. Xiao and L. Zhang : "Contact constraint analysis and determination of geometrically valid contact formations from possible contact primitives," *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 3, pp. 456 – 466, June 1997.

[62] B. R. Donald : "A search algorithm for motion planning with six degrees of freedom," *Artificial Intelligence*, Vol. 31, No. 3, pp. 295 – 353, 1987.

[63] T. Lozano-Perez, M. T. Mason, and R. H. Taylor : "Automatic synthesis of fine-motion strategies for robotics," *Int. Journal of Robotics Research*, Vol. 3, No. 1, pp. 3 – 24, 1984.

[64] A. Bicchi and V. Kumar : "Robotic grasping and contact: A review,", 2000.

[65] S. Hirai : "Kinematics and statics of manipulation using the theory of polyhedral convex cones and their application to the planning of manipulative operations," *Journal of the Robotics Society of Japan*, Vol. 17, No. 1, pp. 68 – 83, 1999.

[66] H. Hirukawa : "Representation and analysis of constraints for motions of objects in assembly processes," *Journal of the Robotics Society of Japan*, Vol. 11, No. 2, pp. 192 – 200, 1993.

[67] E. Rimon and J. W. Burdick : "Mobility of bodies in contact – part i: A 2nd-order mobility index for multiple-finger grasps," *IEEE Trans. on Robotics and Automation*, Vol. 14, No. 5, pp. 696 – 708, 1998.

[68] *http://www2.konicaminolta.jp/products/industrial/instrument/vivid/vivid910.html.*

[69] K. Reidemeister : "KNOT THEORY," BCS Associates, 1983.

[70] M. Ochiai, S. Yamada, and E. Toyoda : "Computer Aided Knot Theory," Makino Shoten, 1996.

[71] H. Whitney : "Congruent graphs and the connectivity of graphs," *Amer. J. Math.*, Vol. 54, pp. 150 – 168, 1932.

[72] K. Ogawara, J. Takamatsu, S. Iba, T. Tanuki, Y. Sato, A. Saegusa, H. Kimura, and K. Ikeuchi : "Acquiring hand-action models in task and behavior levels by a learning robot through observing human demonstrations," *The First IEEE-RAS Int. Conf. on Humanoid Robots*, 2000.

[73] *http://www.sdia.or.jp/mhikobe-e/products/mechatronic/e_index.html.*

[74] K. Kitagaki, T. Suehiro, and T. Ogasawara : "Monitoring of a pseudo contact point for fine manipulation," *IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 757 – 762, 1996.

[75] K. Shimokura and S. Muto : "A study of a manipulation system with a detection module for contact-state-transition," *Journal of the Robotics Society of Japan*, Vol. 12, No. 6, pp. 837 – 845, 1994.

[76] T. Suehiro and K. Takase : "Skill based manipulation system," *Journal of the Robotics Society of Japan*, Vol. 8, No. 5, pp. 551 – 562, 1990.

[77] T. Hasegawa, T. Suehiro, and K. Takase : "A model-based manipulation system with skill-based execution," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 5, pp. 535 – 545, 1992.

[78] K. Shimokura and S. Muto : "Assembly task-oriented manipulation system(atoms) with direct-drive manipulator and multi-task controller," *Journal of the Robotics Society of Japan*, Vol. 13, No. 8, pp. 1190 – 1198, 1995.

[79] M. Shimizu and K. Kosuge : "Planar parts mating tasks using structured compliance," *Journal of the Robotics Society of Japan*, Vol. 20, No. 8, pp. 852 – 859, 2002.

[80] Y. Yu, T. Yoshikawa, and S. Tsujio : "Automatic generation of probing operation for estimating contact position between object and environment," *Journal of the Robotics Society of Japan*, Vol. 16, No. 3, pp. 417 – 424, 1998.

[81] T. Matsuoka, T. Hasegawa, K. Honda, and T. Kiriki : "Manipulation system by multi-fingered hand based on task observation and task evaluation," *Journal of the Robotics Society of Japan*, Vol. 17, No. 5, pp. 696 – 703, 1999.