Example-based Grasp Adaptation

by

Jiwon Kim

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

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author .....
Department of Electrical Engineering and Computer Science
September 9, 2007

Certified by.....
Tomás Lozano-Pérez
TIBCO Founders Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by.....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

Example-based Grasp Adaptation

by

Jiwon Kim

Submitted to the Department of Electrical Engineering and Computer Science
on September 9, 2007, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Finding a way to provide intelligent humanoid robots with the ability to grasp objects has been a question of great interest. Most approaches, however, assume that objects are composed of primitive shapes such as box, sphere, and cylinder. In the thesis, we explore an efficient and robust method to decide grasps given new objects that are irregularly-shaped (3D polygon meshes). To solve the problem, we use an example-based approach. We first find grasps for objects geometrically similar to those the system has seen before. For example, if the system has been shown a cup being grasped by the handle, it should now be able to grasp any new cup. There are two problems to be solved in order to adapt example grasps to the new object. First, the system should be able to retrieve objects that are geometrically similar to the given object from the database storing previously seen objects. After collecting objects the system knows how to grasp, it needs to adapt example grasps to new object.

Already, there are some working algorithms for the first problem (shape retrieval). Therefore, our main contribution is to present an algorithm that performs grasp adaptation. Before we adapt a grasp, we first find the geometric correspondence between a demo object and new object using probabilistic graphical model. Based on correlation information together with the demo grasp, we generate a grasp for the new object. To ensure that a robot can effectively grasp the object, we adjust the position of grasp contacts until the quality of the grasp is reasonably high. In test cases, the system successfully uses this method to find the correspondence between objects and adapt demo grasps.

Thesis Supervisor: Tomás Lozano-Pérez
Title: TIBCO Founders Professor of Computer Science and Engineering

Acknowledgments

Since I first came to MIT as an undergraduate, I have always been curious about how it is like to think like a researcher. My curiosity would not have been satisfied without the guidance of Prof. Tomás Lozano-Pérez. As my academic, undergraduate research, and master's research advisor, he has been the most influential person in my academic career. He always patiently helped me with solving various technical problems, and I learned from him that researchers should be able to see the big picture of what they are doing. In short, he has been an excellent advisor, and I am very grateful that I have had such a great advisor.

I would like to acknowledge the help of Kaijen Hsiao. She first formulated the main problem tackled in the thesis. She helped me clarify ideas, solve problems and write the thesis. It has been a joy to be around her. Her encouragement and never-ending energy got me to the point of writing the thesis.

Finally, I thank to Ross Glashan for his great 3D object modelling and to Hyunsoo Kim for always listening to my ideas and presentation dry-runs. Also, my friends made my time here very memorable. Lastly, but most importantly, my parents in Korea made everything possible for me to get a good education while I was growing up. I really appreciate their support while I was struggling to come to America for college education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We use our hands to manipulate objects of a wide variety of shapes, sizes and materials. We can perform tricky tasks such as holding a cup or lifting a pen from a table. One of the most basic manipulations is grasping objects. In order to use objects, one must be able to pick them up first. Even though humans can easily pick up objects by wrapping a hand around an appropriate part of the object and lifting the object, it has been very difficult to successfully automate the process of grasping objects using robotic hands.

One simple approach for a robot is to make a robotic hand simply wrap around the object so that the resulting grasp has opposing contacts. As shown in Nguyen's work on force-closure grasps [24], almost any grasp with somewhat opposing contacts can support an object if friction is high enough. Thus, a robot can pick up objects in this manner as long as it can wrap its fingers around an object.

Simple routines solve part of the problem, but more complex manipulations that humans find intuitive remain difficult to reproduce with brute-force solutions. Consider a situation where a robot tries to pick up a cup full of water. If the robot just wraps its fingers around the cup and lifts, it is highly probable that the robot pours water over the floor. We can see that the desired grasp should be adequate for the object type (e.g. cup). We have different grasp approaches for different objects. Even though it is desirable to find a generic solution that reproduces appropriate grasps for all possible objects, finding such a solution seems very difficult.

For such tasks mentioned above, it is often easier to teach the robot how to grasp some demo grasps, and imitate the demo grasps when similar objects are given as the input. In other words, the grasp system generates a grasp for a new object based on the demonstration grasp of a similar object. There are several stages to achieve such a system, and the main goal of this thesis is to provide an ability to adapt a grasp of an object to similar-looking objects.

1.1 Assumptions

Several assumptions are important to the approach taken in this thesis. These include:

1. We have perfect knowledge of object shape.
2. A grasp can be modeled as a set of contact points.
3. The friction coefficient for contacts is set to 0.8, which is comparable to the static coefficient of rubber on dry pavement.

1.2 Approach

A summary of the approach taken for each step in the grasp system is as follows:

1. **Retrieving a similar-looking object:** An algorithm that ranks objects by similarity using geometric properties is used to pick a similar object from a database of demonstration objects. Since there can be several candidates, if the system fails with the first demonstration grasp, the system can try the next candidate if needed.
2. **Matching sampled points on the surfaces of objects:** The system tries to find the point-to-point correspondence between the demo object and the input object.

3. **Adapting the chosen demo grasp to a new object:** The chosen demo grasp is adapted to the input object based on correspondence information. If desired, the system adjusts the contact points of the adapted grasp until the quality of the resulting grasp becomes high.

The overview of the steps is provided in Figure 1-1.

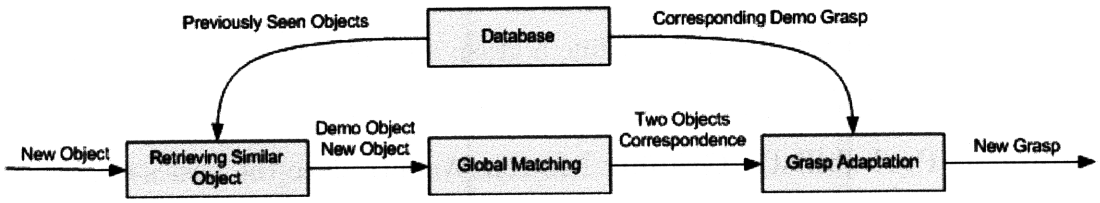


Figure 1-1: System Overview

1.3 Contributions

The focus of the thesis is to build a grasp system that adapts a demo grasp to new object with little or no human supervision. We present novel algorithms for some stages of the adaptation process.

Global Correspondence between Objects: We take the idea of the Correlated Correspondence algorithm (see Chapter 6), an algorithm for 3D surface registration, and apply it to the problem of finding the point-to-point correspondence between similar objects. The algorithm does not need markers, nor does it assume prior knowledge about object shape. We propose a set of potentials used in constructing the probabilistic model over all possible point-to-point correspondences between two meshes.

Grasp Adaptation: We address the problem of adapting a grasp. We propose a simple algorithm that transforms a demo grasp and adjusts the transformed grasp to find a force-closure grasp. Our algorithm has certain limitations addressed in Chapter 7, and the algorithm can be improved by adding more constraints without changing the basic framework.

1.4 Thesis Outline

Below is a summary of the rest of chapters in this thesis.

Chapter 2. Related Work: We introduce previous literature.

Chapter 3. Object Representation: We introduce and discuss the basic object representations, including point clouds and meshes.

Chapter 4. Grasp Quality: We define a grasp and introduce a common way to evaluate the quality of a grasp.

Chapter 5. Probabilistic Graphic Model: We introduce the basics of Markov network. We describe in detail the Loopy Belief Propagation algorithm to perform inference in Markov networks.

Chapter 6. Global Correspondence between Objects: We define the problem of finding the correspondence between two similar objects. We use the Correlated Correspondence algorithm to solve the problem and evaluate the algorithm experimentally on several real-world datasets.

Chapter 7. Grasp Adaptation: We define the problem of grasp adaptation. We present an algorithm based on local search to adapt a grasp and evaluate it experimentally.

Chapter 2

Related Work

Grasp planning has been studied for a long time [19, 27, 8, 9, 18, 31]. Most approaches deal with finding or analyzing sets of contacts on the surface of an object. A common goal of them is to find a high quality grasp according to some quality measure.

Even though there are many methods to achieve the goal mentioned above, the basic approach used in the thesis is very similar to that of the work by Hsiao [9]. In her work, she assumes that the grasp system knows how to grasp some template objects. When new object is given to the system, it retrieves the most similar object (template) from the database, and tries to transform the template grasp to generate a grasp for new object. The idea that a grasp for new object is synthesized by imitating a template grasp is also used in this thesis.

Related research also includes various methods of object manipulations and movements. Approaches of previous works vary in how objects of interest are represented and how grasps and actions are represented.

Algorithms for grasp synthesis that consider complex hand kinematics typically involve treating all or part of an object as a primitive shape (e.g. box, sphere, cylinder, cone) for which previous grasp systems provide good grasp strategies. [29, 30, 10, 3, 9]. However, it may not be always clear how to model irregularly-shaped objects with a given set of primitive shapes. In addition, this approach of modeling objects requires a significant amount of works on the part of the designer. One of the goals in this

thesis is to provide grasp synthesis for irregularly-shaped objects without requiring division of the objects into primitive shapes.

To deal with irregularly-shaped objects, the grasp system requires several algorithms operating on 3-D objects. How these algorithms are used as parts of the system is described in Chapter 1. The first of these algorithms is a shape retrieval algorithm developed by the Princeton Shape Retrieval and Analysis Group [13]. Using Kazhdan's work, the system is capable of retrieving objects that are geometrically similar to the model object. To boost matching and indexing shapes, Kazhdan decomposes a 3-D model into a collection of functions defined on concentric spheres and uses spherical harmonics to discard orientation information for each one. This decomposition yields a shape descriptor that is both orientation-invariant and descriptive.

After retrieving a similar object from the database, we adapt a demonstration grasp for the retrieved object to the input object. To adapt a grasp, we need to know the correspondence between parts of the retrieved object and input object. This correspondence problem is reinterpreted as maximizing probabilities in the corresponding Markov Random Field (MRF). To construct the corresponding Markov network given two objects, I use ideas in Anguelov's work [1]. The network is constructed so that the evaluation of an assignment computes the feature similarity of samples matched between two objects.

In addition to the frameworks used to represent irregularly-shaped objects, there are many different frameworks to represent grasps and actions. Each representation has advantages and disadvantages in terms of encoding and playback [9]. Some of popular representations include finite state automata [25, 26] and relational/contact expression grammars [16, 33, 34]. In relational/contact expression grammars, properties such as "Moving-Up", "Moving-Down" and "Staying" are used as the values of motion features. In a similar way, entities such as "Near" and "Far" are used to describe the state of assembly, grasping or manipulation tasks. Alternatively, in Pollard's work [27], a task is represented by a set of torques at contact points.

In addition to research concerning grasp synthesis, related work includes grasp-quality measures. There are several approaches to quantify the grasp quality; these

can be grouped into two categories. One includes measures associated with the position of the contact points [8, 23]. The other includes measures associated with the hand configuration [15, 36]. In this thesis, we do not use measures associated with the hand configuration. For this reason, they are not discussed in this section.

Measures associated with the position of the contact points can be further divided into three subgroups. The first subgroup of measures includes those that only consider the algebraic properties of the grasp matrix, the mapping between the finger forces and external forces [14]. The next subgroup includes those that are based on the evaluation of certain geometric relations of the contact points [20, 5]. The first and second subgroups of quality measures do not consider any limit on the magnitude of the forces applied by the fingers. This means that in some cases the fingers may have to apply extremely large forces to resist small perturbations. In contrast, measures of the last subgroup consider limitations on the finger forces. One such measure was proposed by C. Ferrari and J. Canny [8]. This measure considers *grasp wrench space* produced by the fingers on the object. The quality is defined as the largest perturbation wrench that the grasp can resist indepedently of its direction. Geometrically, the quality is equivalent to the radius of the largest ball centered at the origin of the wrench space and fully contained in the *grasp wrench space.* This measure is used in this thesis.

Chapter 3

Object Representation

Objects we use everyday are often irregularly-shaped. An example of 3D objects is provided in Figure 3-2. One of the goals in the thesis is to provide a grasping system that works on irregularly-shaped objects without requiring division of objects into primitive shapes. In this chapter, we describe the basics of how 3D objects are modeled, and introduce the type of model chosen for our system. This discussion provides the foundation for algorithms described in later chapters.

Most practical applications approximate 3D surfaces with only a certain degree of accuracy. Moreover, current 3D sensors only provide point samples of surfaces. We will build discretized approximations to surfaces from the point samples. We present two classes of tractable discretizations below: *explicit* and *implicit* representations. *Explicit* representations model the surface directly. *Implicit* representations are in the form of scalar fields and they have gained popularity recently [7, 17].

3.1 Explicit Representation

3D scanning is a very popular way to create 3D object data. A 3D scanner is a device that analyzes a real-world object or environment to collect data on its shape and possibly its appearance such as color and texture. Most current 3D scanners provide object shape in terms of unstructured point clouds by reading samples on surfaces. Sample readings are typically obtained by emitting light rays and measuring

(a) Real-World Objects

(b) Modeled Objects

Figure 3-1: Real-world Objects Modeled as Meshes

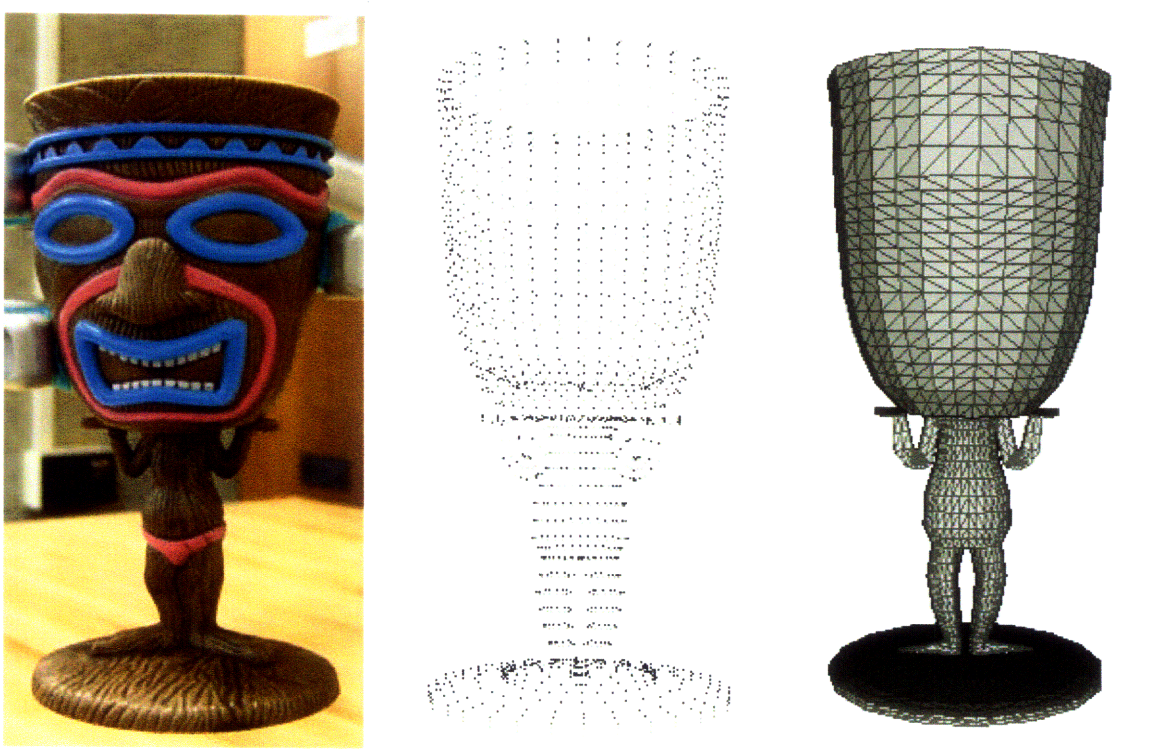
the time of travel from the sensor to the object and back. Each reading measures the coordinates of points on the scanned surface. To obtain the estimates of the normal vectors to the surface at those points, we can perform a post-processing step such as interpolating adjacent sensor readings [21]. The coordinates of sampled points are often called a *point cloud*, which is the simplest representation of the surface.

Definition 3.1.1. *A **point cloud** is a description of the surface X as a collection of the coordinates of surface points.*

Figure 3-2 shows an example of a point cloud.

While point clouds are useful to represent raw data from 3D sensors, they provide only partial information about the surface. Most importantly, it is not easy to retrieve connectivity of points on the surface. To resolve this problem, we use polygon meshes.

A polygon mesh is a very popular 3D object representation in 3D computer graphics. It is a collection of vertices and polygons that define the shape of a polyhedral object. Meshes usually consist of triangles, quadrilaterals or other simple convex polygons. They can approximate any continuous surface arbitrarily well given a fine



(a) A Real-World Object (A Tiki Glass) (b) Corresponding Point Cloud (2489 Points) (c) Corresponding Mesh (2489 Vertices and 4976 Faces)

Figure 3-2: Surface discretization using point clouds and meshes

enough polygon tessellation.

There are many internal representations for meshes. A common representation is to use a list of vertices and faces. A vertex corresponds to a sample point on the surface. Vertices are connected by edges, and faces consist of these edges.

Definition 3.1.2. *A **mesh** M_X is a tessellation of a continuous 3D surface X into a set of faces, or polygons. It can be represented as a collection of vertices and faces:* $M_X = (V_X, F_X)$.

In Definition 3.1.2, $V_X = (v_1, v_2, \cdots, v_m)$ contains the coordinates of the face vertices. The set of faces covering the surface is denoted by $F_X = (f_1, f_2, \cdots, f_m)$. In general, a face can contain an arbitrary number of vertices. An example of mesh is displayed in Figure 3-2.

The normal vectors at sample points can be estimated from vertices and faces. For example, the normal n_{f_k} at face f_k with three vertices $f_{k,1}, f_{k,2}, f_{k,3}$ can be estimated

in the following way.

$$u = (f_{k,2} - f_{k,1}) \times (f_{k,3} - f_{k,1}). \tag{3.1}$$

$$n_{f_k} = \frac{u}{||u||}. \tag{3.2}$$

To estimate the normals to a polygon, we triangularize the polygon and estimate the normals at triangles.

The convention regarding the order of the face vertices is that the vertices specified in a counter-clockwise order will produce an outward pointing normal. The opposite order flips the direction of the normal. The normal at a vertex is estimated by simply averaging the normals of the adjacent faces.

3.2 Implicit Representation

While explicit surface representations model the surface directly, implicit representations are in the form of scalar fields. The most popular implicit representation is the *signed distance map* (SDM) [32]. In SDM, each sample encodes the distance to the nearest point on the curve, with negative values inside the curve. SDM is typically represented over a discrete grid and can be computed from a mesh [17]. In SDM, the isosurface containing all points for which the signed distance is zero corresponds to the original surface.

A SDM is defined everywhere in 3D space relative to an object. This makes it straightforward to relate SDM representations of different objects to each other. In addition, a SDM is smooth and its smoothness facilitates smooth interpolation between objects. For these reasons, SDMs have been used for many algorithms including a famous scanning algorithm by Curless and Levoy [7].

While using SDMs have many advantages, using them also has drawbacks. Most importantly, the map has to be defined for the entire 3D space, which requires more computation and memory than meshes do. In addition, since SDMs do not explicitly model the surface, it is not straightforward to apply combinatorial algorithms to

SDMs.

For the purposes in the thesis, using meshes is a better choice. The first reason is that the Correlated Correspondence algorithm that is introduced in Chapter 6 only works with meshes. Second, since meshes are more popular than SDMs, it is easier to find graphics tools based on meshes. Last, meshes require fewer computations and less space than SDMs. For these reasons, we will use meshes throughout the rest of the thesis.

Chapter 4

Grasp Quality

Before we discuss grasp adaptation, we first need good descriptions of grasps and tasks, and a mechanism to evaluate grasps. In this chapter, we address these issues. There have been many discussions on this topic and this chapter draws from previous literature, especially, from Ferrari and Canny [8], and Pollard [28].

Grasp

This thesis uses a very simple description of a grasp. A geometric model of a target object is given as a mesh (for the definition of mesh, see Chapter 3), and a grasp of that object is formed by placing some number of contacts on the object. It is assumed that all contacts are simple point contacts with friction (hard contact).

Force and Torque

Forces can be exerted on the target object through a grasp. The exerted force on the object is normal to the contact surface. Even though we cannot apply torques at contacts, exerted forces can induce torques around the target object center of mass.

Task

To determine the suitability of the grasp for a particular task, a description of that task is needed. In this thesis, a task is a space of forces and torques that the

hand may be required to apply to the target object in order to perform some function. This space of forces and torques is referred to as the *task wrench space*. The term *wrench* refers to a combined vector of a force and a torque.

Grasp Quality Measure

A grasp quality measure is an estimate of the suitability of a grasp for the task to be performed. That is how well the grasp can withstand the forces and torques of the task. All grasps that are synthesized in this thesis will be evaluated according to this measure. Before defining quality measure in detail, we need some notations.

Notation:

$$\underline{f_i} = \text{applied force at contact } i, \tag{4.1}$$

$$\underline{\tau_i} = \text{torque at contact } i, \ = \lambda(\underline{f_i} \times \underline{d_i}), \tag{4.2}$$

$$\underline{d_i} = \text{vector to object center of mass}, \tag{4.3}$$

$$\lambda = \text{torque to force conversion factor}, \tag{4.4}$$

$$\underline{w_i} = \text{wrench at contact } i = [\underline{f_i}\ \underline{\tau_i}]^T, \tag{4.5}$$

$$c = \text{number of contacts}. \tag{4.6}$$

The grasp wrench space region (GS): A grasp can be characterized based on the set of wrenches that can be applied to the target object through the contacts of a grasp. The unit grasp wrench space region is defined as follows:

$$GS = \{\underline{w} \mid \underline{w} = \sum_{i=1}^{c} \alpha_i \underline{w_i},\ \alpha_i \geq 0,\ \sum_{i=1}^{c} \alpha_i \leq 1,\ |\underline{f_i}| = 1\}. \tag{4.7}$$

In other words, GS is bounded by the convex hull of the contact wrenches formed from unit applied forces at the contacts.

The task wrench space region (TS): A task can be characterized as the space of wrenches that must be applied to the object in order to complete the objective. A common task people use to measure grasp quality is the simple task of countering arbitrary disturbance forces applied to the object. If arbitrary disturbance force of any

direction can be resisted by forces and torques that can be applied through a grasp, this grasp is called *force-closure*. One useful wrench space region that describes this task is the ball-shaped region centered at the origin. A ball of radius κ contains all wrenches whose magnitudes are less than κ:

$$\kappa\, TS_{ball} = \{\underline{w} \mid |\underline{w}| \leq \kappa\}. \tag{4.8}$$

The grasp quality measure (Q): Intuitively, the grasp quality measure is the amount of forces and torques that the robot is capable of resisting while maintaining the grasp. The measure used in the thesis is defined as follows:

$$Q \;=\; max(\kappa) \mid \kappa TS \in GS. \tag{4.9}$$

Geometrically, this is equivalent to the size of the largest wrench space ball that fits completely within the unit grasp wrench space. In other words, grasp quality is the reciprocal of the amount by which GS must be scaled so that it just contains TS.

The grasp quality measure for the grasp shown in Figure 4-1 is 0.35 for task TS_{ball}. Intuitively, this means that if a task wrench must be countered in a direction in which the grasp is weak, the wrenches that must be applied to the object through the contacts of the grasp are approximately three times the magnitude of that task wrench.

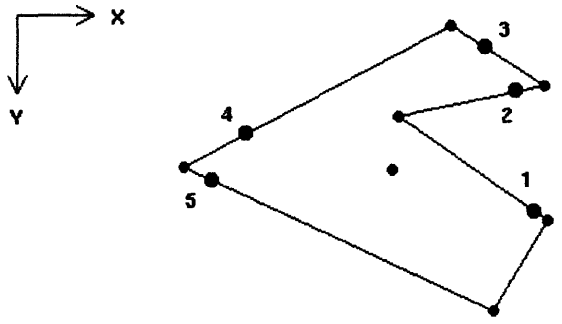


Figure 4-1: An Example Grasp Represented by Numbered Contact Points [28]

In practice, we first need to construct GS before we compute the grasp quality.

Consider that the friction cone at contact i is approximated by a pyramid with s edges, and the force $\underline{f_i}$ can be expressed as a positive linear combination of unitary forces $\underline{f_{i,j}}$, $j = 1, 2, \cdots, s$, along the pyramid edges. The resultant wrench $\underline{w_i}$ produced by $\underline{f_i}$ can be expressed a positive linear combination of the wrenches $\underline{w_{i,j}}$ produced by $\underline{f_{i,j}}$. Now, c fingers produces a resultant wrench on the object given by

$$\underline{w} = \sum_{i=1}^{c} \alpha_i \underline{w_i} \tag{4.10}$$

$$= \sum_{i=1}^{c} \sum_{j=1}^{s} \alpha_{i,j} \underline{w_{i,j}}. \tag{4.11}$$

The set of possible resultant wrenches is the convex hull of the Minkowski sum of the primitive wrenches $\underline{w_{i,j}}$:

$$GS = ConvexHull\left(\bigoplus_{i=1}^{c} \{w_{i,1}, w_{i,2}, \cdots, w_{i,s}\}\right). \tag{4.12}$$

To compute the grasp quality, we need to find the size of the largest wrench space ball that fits within the convex hull. This can be easily done by finding the largest distance to a convex hull face from the origin.

Chapter 5

Probabilistic Graphical Model

Probabilistic graphical models provide a means of encoding the structure of complex environments efficiently. The models are representations of the joint probability distribution over a set of variables, and their inherent structure can be used to perform tasks such as reasoning and learning. In particular, we can reason about the assignments to a large number of variables simultaneously by utilizing the theory of graphical models.

In this chapter, we present the basics of probabilistic graphical models. We first introduce Markov networks, which are used in our algorithms. Then we show how we can perform probabilistic inference in a network. Last, we describe *loopy belief propagation*, an algorithm that calculates conditional probabilities efficiently.

5.1 Markov Random Field

In this section, we briefly introduce undirected graphical models known as *Markov random fields* or *Markov networks* [2]. The semantics of MRFs are similar but simpler than Bayesian networks [2]. The graph represents dependence properties between the variables.

For example, in Figure 5-1, since X_1 and X_2 are connected, they are dependent on each other. In this way, connected variables are correlated. However, if X_2 and X_4 are observed, X_1 and X_3 get disconnected and that means that they are independent

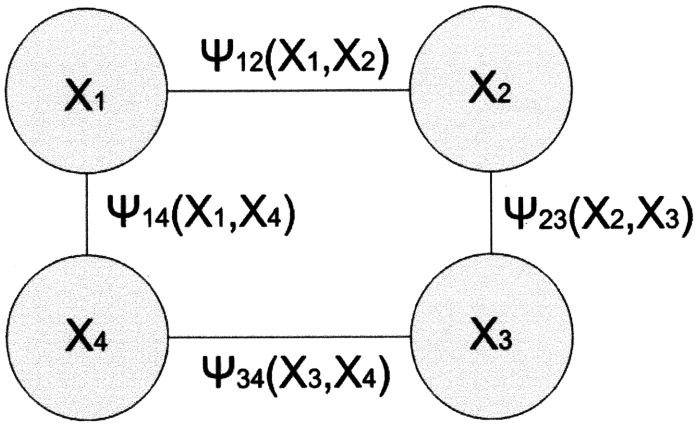


Figure 5-1: Markov Random Field Example

given X_2 and X_4. Similarly, if we remove X_1 and X_3 from the graph, then X_2 and X_4 are no longer dependent.

MRFs can encode dependence structure in joint probability distributions. For the above example, we can define a distribution as the product of non-negative *potential functions* over pairs of variables that specify how the variables depend each other:

$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{14}(x_1, x_4)\psi_{23}(x_2, x_3)\psi_{34}(x_3, x_4), \quad (5.1)$$

where Z is a normalization constant. A normalization constant is required to make distributions of all assignments sum to one. A potential function can take any non-negative real value. In addition, it can be associated with a fully-connected group of variables, called a *clique*. Definition 5.1.1 provides the formal definition of MRF using cliques.

Definition 5.1.1. *A **Markov random field**, or **Markov network** is a representation of a joint probability distribution over a set of variables $X = \{x_1, x_2, \cdots, x_n\}$. The network has vertices that correspond to the variables, and encodes a set of independence assumptions. A potential ψ_c is associated with a clique c in the graph, and is used to define the distribution*

$$P(X) = \frac{1}{Z}\prod_c \psi_c(X_c), \quad (5.2)$$

where X_c denotes the set of variables in the clique c, and the value Z is the normalization factor, defined as

$$Z = \sum_X \prod_c \psi_c(X_c). \tag{5.3}$$

5.2 Probabilistic Inference

Probabilistic inference, the process of computing the posterior distribution of variables given some evidence, is a fundamental task in probabilistic graphical models. It is shown that performing inference in Markov networks is NP-hard [6]. The difficulty of the inference task tends to increase as the graph structure gets more complex. For example, exact inference takes linear time for Markov networks whose underlying graphs are trees, whereas in Markov networks with many cycles, exact inference is intractable.

To evaluate the desired probabilities in Markov networks within a limited time, we often have to approximate inference answers. There are many different approximation methods. We will not review all methods in this thesis. For those who are interested in knowing more about these methods, please read [6],[12], and [22] for *Gibbs sampling*, *mean field* approximation, and *loopy belief propagation*, respectively.

Loopy Belief Propagation is an inference algorithm that is most commonly used in AI. Loopy Belief propagation is used later in this thesis when we discuss correspondence between two objects, and a brief explanation of the algorithm is provided in the next section.

5.3 Loopy Belief Propagation

Loopy Belief Propagation(LBP) is a simple message passing algorithm. The algorithm can approximate conditional probabilities in Markov networks. In case of graphs with cycles, we should repeat iterations of passing messages until probabilities converge.

In this section, we describe a special case of the algorithm, which can be executed on Markov networks with single and pairwise potentials. Theoretically, this case can cover all graphical models since all models can be converted to such Markov networks using the method introduced in the paper by Yedidia *et al.* [35].

Consider a Markov network with discrete variables $X_1, X_2, \cdots, X_n$ and single and pairwise potentials $\psi_i(X_i)$ and $\psi_{i,j}(X_i, X_j)$, respectively. Our task is to evaluate the beliefs $P(X_i)$ for all variables.

The main idea of LBP is that variables send *messages* to their neighbors in the underlying graph in order to update their beliefs. We will use $m_{i\to j}(X_j)$ to denote the message from X_i to X_j. The LBP algorithm maintains an estimate of the marginal probabilities of all variables. We call such estimate *beliefs* and use $b_i(x_i)$ to denote the estimate of the probability that $X_i = x_i$. The beliefs should be normalized (i.e. $\sum_{x_i} b_i(x_i) = 1$) to be used as probabilities.

The beliefs are computed from the incoming messages sent by the variable's neighbors using the following update rule:

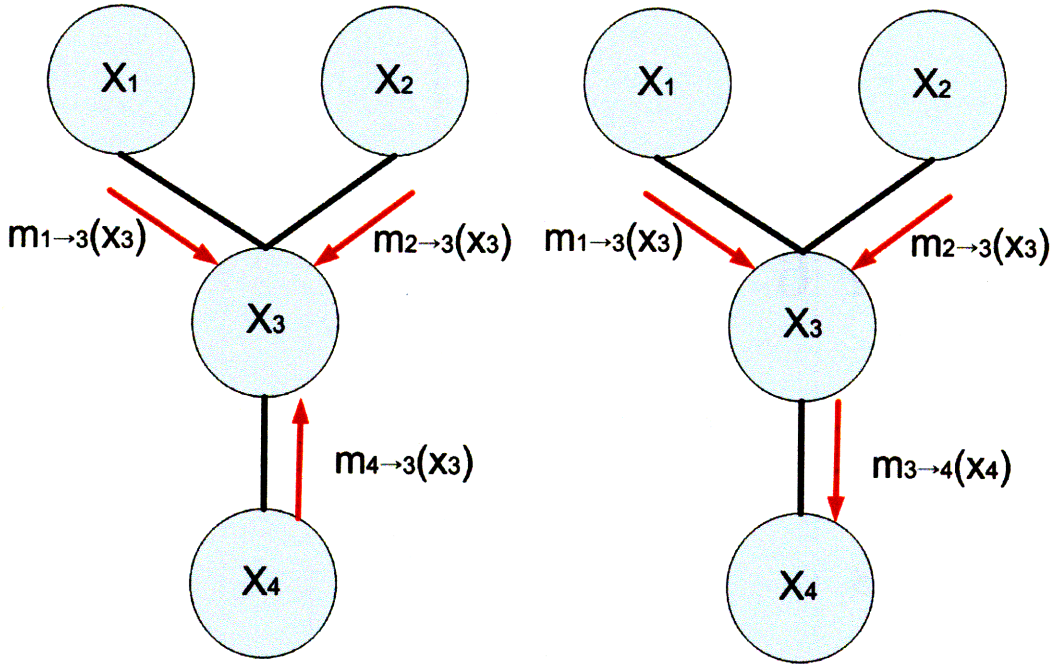
$$b_i(x_i) = \frac{1}{Z}\psi_i(x_i) \prod_{j \in Neighbors(X_i)} m_{j\to i}(x_i). \tag{5.4}$$

In the update rule, Z denotes the normalization constant and $\psi_i(x_i)$ denotes the single potential associated with X_i. An example of the update rule is provided in Figure 5-2. In this case, $b_3(x_3)$ is updated using the messages from the neighbors X_1,X_2, and X_4. Mathematically, the rule is the following:

$$b_3(x_3) = \frac{1}{Z}\psi_3(x_3)m_{1\to 3}(x_3)m_{2\to 3}(x_3)m_{4\to 3}(x_3). \tag{5.5}$$

The messages are also updated based on other messages. Therefore, message update rule is recursively defined as follows:

$$m_{j\to i}(x_i) = \sum_j \psi_j(x_j)\psi_{i,j}(x_i, x_j) \prod_{k \in Neighbors(X_j)\setminus\{X_i\}} m_{k\to j}(x_j). \tag{5.6}$$



(a) Belief Update Rule (b) Message Update Rule

Figure 5-2: Loopy Belief Propagation Examples

Note that the right-hand formula excludes the message from X_i since we are updating the message associated with X_i. In Figure 5-2, the message update rule for $m_{3\to4}(x_4)$ is the following:

$$m_{3\to4}(x_4) \quad = \quad \psi_3(x_3)\psi_{4,3}(x_4, x_3)m_{1\to3}(x_3)m_{2\to3}(x_3). \tag{5.7}$$

The LBP algorithm uses the above rules to estimate beliefs. Initially, all messages and beliefs are uniform (i.e. $m_{j\to i}(x_i) = 1$ and $b_i(x_i) = 1$). Then, we perform updates on messages and compute new beliefs based on these message updates. We repeat this update process until the beliefs converge. The algorithm may not converge in some cases. Therefore, we stop updating after a predefined maximum number of iterations. Convergence conditions are studied in the paper by Yedidia *et al.* [35].

Chapter 6

Global Correspondence between Objects

In this chapter, we address the problem of finding the point-to-point correspondence between the surfaces of two objects. Solving this problem is essential in developing the desired grasp system since we need to know how two objects are related in order to adapt a grasp for one object to the other.

Finding the correspondence between vertices of two meshes is a very difficult problem. First, there are thousands of vertices in each mesh. Since each vertex of one mesh can map to one of thousands of vertices of other mesh, the number of possible correspondences is very large. Second, there is no obvious way to match two objects. Different people can have different ways to match two objects. They can argue over what objectives should be satisfied to give a good matching.

Currently, we have not seen any particular algorithm designed to solve the problem mentioned above. The most relevant research area is *non-rigid 3D registration.* Non-rigid 3D registration is the problem of finding the point-to-point correspondences between the surface of an object and the deformed surface of the original object. In this case, the objective is clearer than the previous case because one surface is the deformed version of the other, whereas the grasping system should be able to deal with two different objects although they are mostly similar. In this chapter, we study ideas from non-rigid 3D registration and apply them to our problem in later sections.

6.1 Correlated Correspondence Algorithm

In this section, we address the problem of *non-rigid 3D registration* which has been studied for many years. Non-rigid 3D registration is the problem of finding the point-to-point correspondence between two deforming surfaces. An example of 3D registration is provided in Figure 6-1. In the figure, the point-to-point correspondence is given by numbered (colored) points.

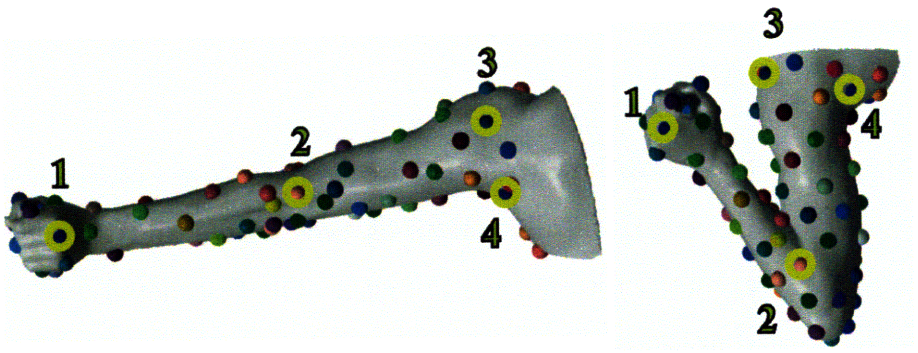


Figure 6-1: An Example of Non-rigid 3D Surface Registration [1]

The registration problem requires search in the space of all possible alignments between two surfaces. Therefore, determining the best correspondence results in a combinatorially large search problem. To make the registration problem tractable, people have developed many algorithms. While many of them assume significant prior knowledge about the objects, the *Correlated Correspondence* algorithm, developed by Anguelov [1], registers deforming surfaces in an unsupervised manner. The algorithm produces reasonable results without prior knowledge about objects. Its only assumption is that the surfaces do not undergo significant topology changes [1].

The algorithm is based on a probabilistic model over the set of possible point-to-point correspondences. For each correspondence, a score is assigned to evaluate the similarity between surface areas. Another score is given to the degree of the deformation. To (approximately) maximize the scores of the probabilistic model, the algorithm uses probabilistic inference introduced in Chapter 5.

In [1], it is reported that the Correlated Correspondence algorithm successfully aligns almost all mesh pairs in the experiment. All running times of the algorithm are

less than 2 minutes. In two cases, the algorithm failed because the torso was flipped, so the front was mapped to the back. This problem arises from ambiguities induced by the fact that the front and back are almost identical. To resolve this problem, we use the shape context algorithm developed by S. Belongie [4] instead of spin images used in Anguelov's original paper. The details are discussed in the next section.

6.2 Correspondence Based On Correlated Correspondence

In this section, we discuss how to find the *correspondence* between two objects. The correspondence is based on all vertices of two objects and aims to correctly align two objects and match the chunks of objects. The Correlated Correspondence algorithm introduced in the previous section is used to find the correspondence. In the following, we discuss the problem definition, the details of the algorithm, and experimental results.

6.2.1 Problem Definition

The correspondence problem is to determine point-to-point correspondences between two surfaces. That is, for each vertex of one object, we find the corresponding vertex of the other object. We assume we are given the complete surface models of two objects. Our ultimate goal in the grasping system is to adapt a grasp for a demonstration object to the new object, utilizing correspondence information. We will call the mesh for the demo object *demo mesh*, and the mesh for new object *new mesh*.

The goal is to match the corresponding parts of the two meshes. We assume that the two meshes are reasonably similar. A matching example is provided in Figure 6-2. Even though there are thousands of vertices for each object, only some of vertices are drawn here for the purpose of visualization. The vertices connected by lines are matched. Note that we have more vertices in new mesh than in demo mesh. The

reason is that by having a fine sampling of new mesh, we have a better chance to correctly assign demo vertices.

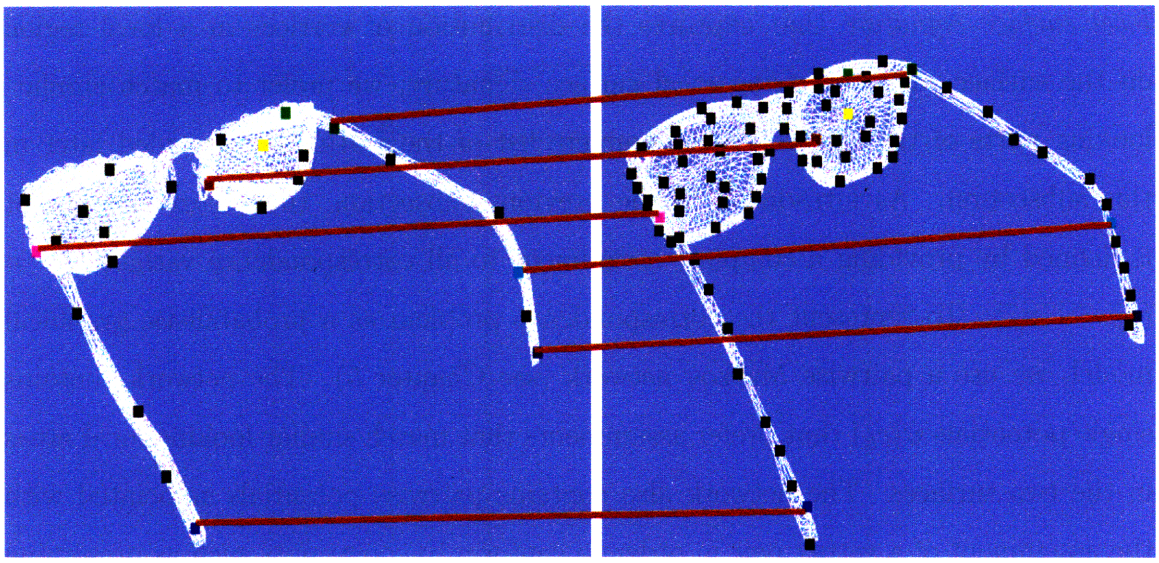


Figure 6-2: Correspondence between the demo mesh and new mesh

The formal problem definition is the following:

Definition 6.2.1. *Given the demo mesh M_X and new mesh M_Y, finding the vertex correspondence $C_{XY} = \{c_1, c_2, \cdots, c_{m_X}\}$ between two meshes is to associate each vertex v_k^X of the demo mesh with a vertex $v_{c_k}^Y$ of new mesh so that the corresponding parts of the objects are matched.*

The variable c_k has a discrete domain containing all vertex indices. Setting $c_k = i$ associates the kth vertex of the demo mesh with the ith vertex of new mesh.

6.2.2 Algorithm

To find the correspondence between two objects, we utilize a probabilistic model. The basic idea is that we give a high likelihood score to vertex assignments that preserve the features of the demo mesh. In other words, if the feature of a set of vertices of the demo mesh is preserved in the set of the corresponding vertices, this assignment is preferred. This idea originates from the paper by Anguelov [1].

There are certain features to be preserved. For example, if the geodesic distance between two vertices of the demo mesh is short, that of the corresponding vertices of

new mesh should be short, too. Similarly, if the distance is long, the corresponding distance should be kept long. Another example of a feature to be preserved is the *local surface signature* that captures the distribution of vertices in a local region on the surface. By preserving local surface signature, we prefer to match similar-looking parts of the surface. The complete list of the features is provided later in this subsection. As introduced in the previous subsection, the vertex assignment is defined by providing a complete assignment to all correspondence variables $C = (c_1, \cdots, c_m)$. To represent the correspondence problem as a probabilistic graphical model, we use a pairwise Markov network (see Chapter 5). The network contains single potentials $\psi(c_k)$ that prefer assignments that match similar-looking local areas in the two surfaces. The network also contains pairwise potentials associated with the assignment of a vertex pair in order to conform to constraints such as preserving geodesic distances. The resulting Markov network is a graphical model of a joint probability distribution of the form $P(C) = \frac{1}{Z} \prod_k \psi_{single}(c_k) \prod_{k,l} \psi_{pairwise}(c_k, c_l)$.

The objective now becomes finding C that maximizes $P(C) = \frac{1}{Z} \prod_k \psi_{single}(c_k) \prod_{k,l} \psi_{pairwise}(c_k, c_l)$ In this way, we preserve the features of the demo vertices that are captured in single and pairwise potentials. To find such C, we use loopy belief propagation to perform probabilistic inference.

Choosing appropriate potentials is a crucial step in order to preserve the features of the demo mesh effectively. In the following, we describe the potentials in detail.

Local Surface Signatures

We prefer to match similar-looking parts of both object surfaces. Therefore, we encode a set of potentials that preserve local surface properties between the demo mesh and new mesh. To capture the signature of local surfaces, we experimented with spin-image features [11] and 3D shape contexts [4]. After the experiment, we decided to use 3D shape contexts. The main reason is that spin images have trouble with bilateral symmetry in objects such as human bodies since spin images are rotationally invariant. In the original paper of the Correlated Correspondence algorithm by Anguelov [1], he also mentions this problem. In contrast, 3D shape contexts can

capture unique surface signatures even when symmetry is present (e.g. the front and back of a torso). For this reason, we use 3D shape context instead of spin images. In this section, we discuss how to use 3D shape context to encode single potentials. However, 3D shape contexts can be easily replaced by other surface signatures including spin images.

Shape context is a local surface descriptor suggested by S. Belongie and J. Malik [4]. The descriptor considers the set of vectors originating from a point to all other sample points on the surface. These vectors express the configuration of the entire object relative to the reference point. Clearly, this set of $m-1$ vectors is a rich description, since as m gets large, the representation of the object becomes exact.

Storing the full set of vectors as a descriptor is inefficient since the description is much too detailed. Instead, we compute the histogram h_i over the set of $m-1$ positions relative to the vertex v_i,

$$h_i(k) = \#\{j \neq i : (v_j - v_i) \in bin(k)\}. \tag{6.1}$$

In 2D, we use bins that are uniform in log-polar space. In this way, the descriptor is more sensitive to nearby sample points than to those farther away. An example of 2D bin is provided in Figure 6-4. In 3D, we define bins similarly with two angles instead of one.

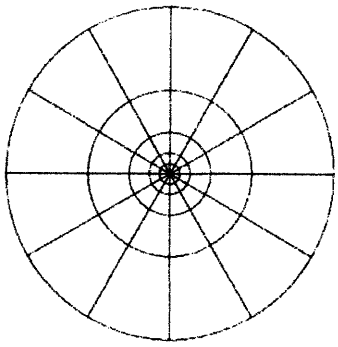


Figure 6-3: An example of a shape context bin over log-polar space

Consider a vertex v_i^X of the first mesh and a vertex v_j^Y of the second mesh. The cost of matching these two vertices is defined as the following:

$$C(v_i^X, v_j^Y) = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}, \tag{6.2}$$

where $h_i(k)$ and $h_j(k)$ denote the K-bin histogram at v_i^X and v_j^Y, respectively.

Using $C(v_i^X, v_j^Y)$, we define single potentials:

$$\psi_s(c_k = i) = N(C(v_k^X, v_i^Y); 0, \sigma_s), \tag{6.3}$$

where $N(x; \mu, \sigma)$ is the probability density function (PDF) of the normal distribution with the mean μ and the variance σ.

3D shape context is very efficient for matching corners. For vertices around corners, only bins that are in some direction are filled in. Therefore, matching corners costs less than matching a corner vertex and a non-corner. In our experiment, the corner vertices of two different glasses were well matched as shown in Figure 6-4.

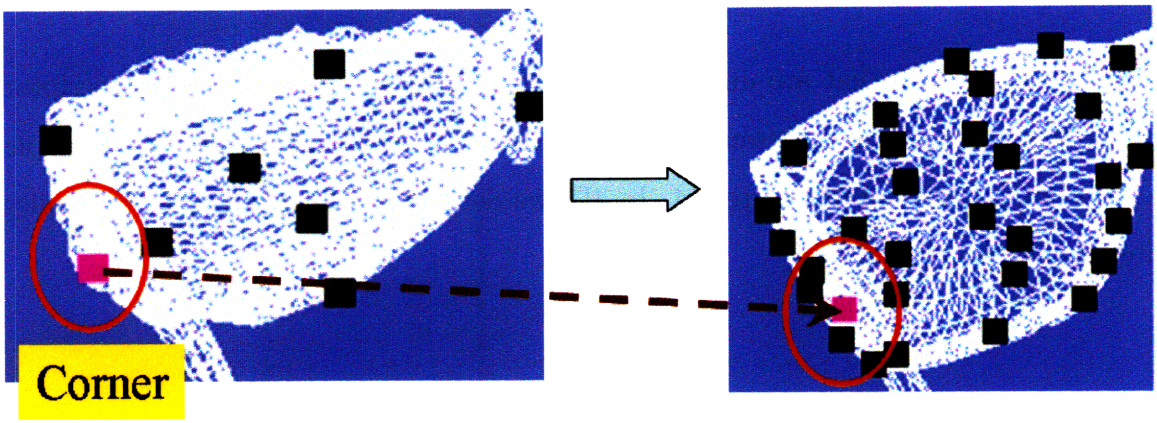


Figure 6-4: 3D shape context is very efficient for matching corners.

Geodesic Distances

In addition to the preservation of local surface signatures, the preservation of distances between vertices is also considered. For example, if the distance between two vertices of the demo mesh is short, the distance between the corresponding vertices of new mesh should be short, too. Similarly, if the distance is long, the corresponding

distance should be kept long.

There are several ways to define the distance between two points. Even though Euclidean distance can be meaningful, it is very sensitive to deformations. If the joint angle between two parts changes, Euclidean distances between vertices in the two parts can change dramatically. In contrast, geodesic distance (i.e. the length of the shortest path on the surface) does not depend on the joint angle. For this reason, we use geodesic distance.

While local surface signature is defined for each vertex, defining geodesic distance requires two vertices. An example of distance preservation is shown in Figure 6-5. Even though only two pairs are highlighted in the figure, we try to preserve geodesic distances for all pairs of vertices. We can achieve the preservation through the use of pairwise potentials in the Markov network:

$$\psi_g(c_k = i, c_l = j) = N(dist(v_i^Y, v_j^Y); dist(v_k^X, v_l^X), \sigma_g), \tag{6.4}$$

where $dist(v_k^X, v_l^X)$ denotes the geodesic distance between the vertices v_k^X, v_l^X. The way this potential works is as follows: when the transformed distance is close to the original distance, the potential value is high, and as the transformed distance deviates from the original, the potential gets lower value.

Cross Product Relations

Preserving local surface signatures and geodesic distances is enough to find a reasonable correspondence between two objects in some cases. In our experiment, we successfully matched two objects when the number of vertices was small enough. However, performing probabilistic inference in graphical models requires a huge number of computations when there are thousands of vertices. This is because there are too many ways to assign vertices. Especially when there are many symmetries present in meshes (e.g. a chair), many vertices share similar local surface signatures and distance relations. To reduce the number of computations necessary to make the probabilities converge, we need another potential.

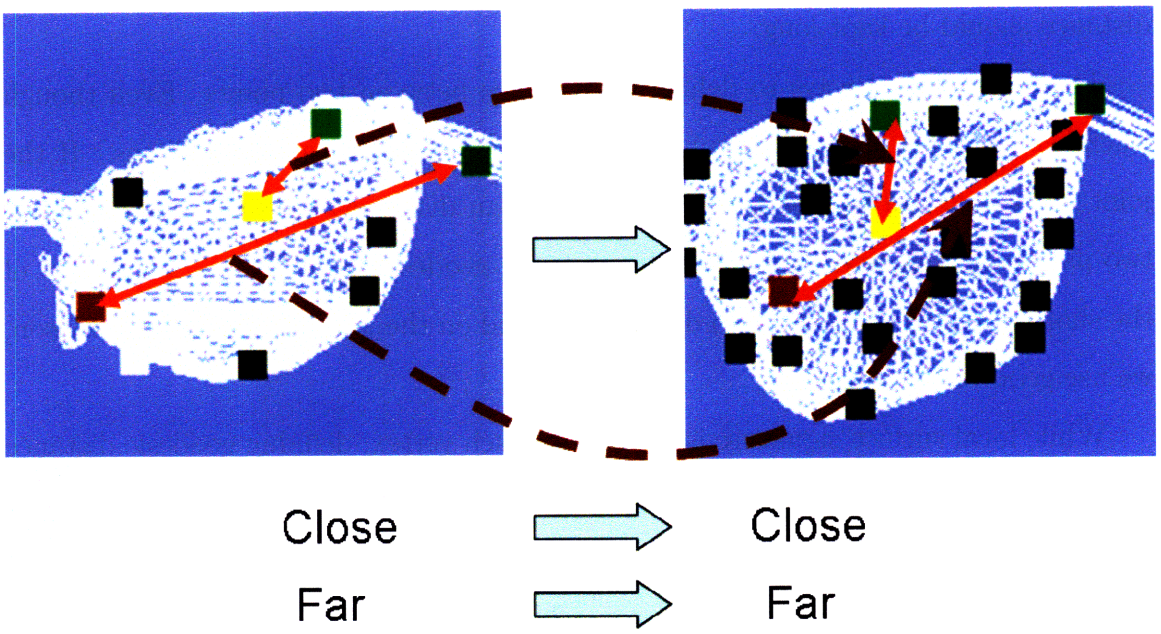


Figure 6-5: Preservation of Geodesic Distance

The potential introduced below makes one strong assumption. We assume that all objects have their z-axis (i.e. the axis oriented vertically) aligned. What this means is that objects have their top and bottom part defined. Imagine a cup placed on a table. There is a common orientation of the cup with respect to the z-axis in that the bottom part of the cup touches the table. Even though it is often the case that we deal with two objects aligned vertically, we can remove this assumption. To do so, we should analyze the distribution of vertex positions and then align objects accordingly before setting up the Markov network.

We first introduce the definition of the potential and then give a simple illustration to show how it works. Before defining the potential, we need the following definitions.

$$cross(v_k^X, v_l^X, v_m^X) = (\vec{v}_l^X - \vec{v}_k^X) \times (\vec{v}_m^X - \vec{v}_k^X), \tag{6.5}$$

$$cross(v_k^X, v_l^X) = \frac{1}{n-2} \cdot \sum_{m \neq k,l} sgn(cross(v_k^X, v_l^X, v_m^X) \cdot \hat{k}), \tag{6.6}$$

where $sgn(x)$ is the sign function and n denotes the number of vertices. The vector function $cross(v_k^X, v_l^X, v_m^X)$ gives the cross product between the vectors starting from v_k^X to v_l^X and v_m^X, respectively. The function $cross(v_k^X, v_l^X)$ gives how vertices are

located in respect to the edge connecting the vertices v_k^X and v_l^X. The geometric interpretation is the following: imagine a plane that contains the edge connecting v_k^X and v_l^X and is perpendicular to the xy-plane. Unless the edge is perpendicular to the xy-plane, this plane is unique. If not unique, $cross(v_k^X, v_l^X, v_m^X)$ will be zero for all v_m^X. Depending on which half-space vertex v_m^X belongs to, we either add to or subtract from $cross(v_k^X, v_l^X)$.

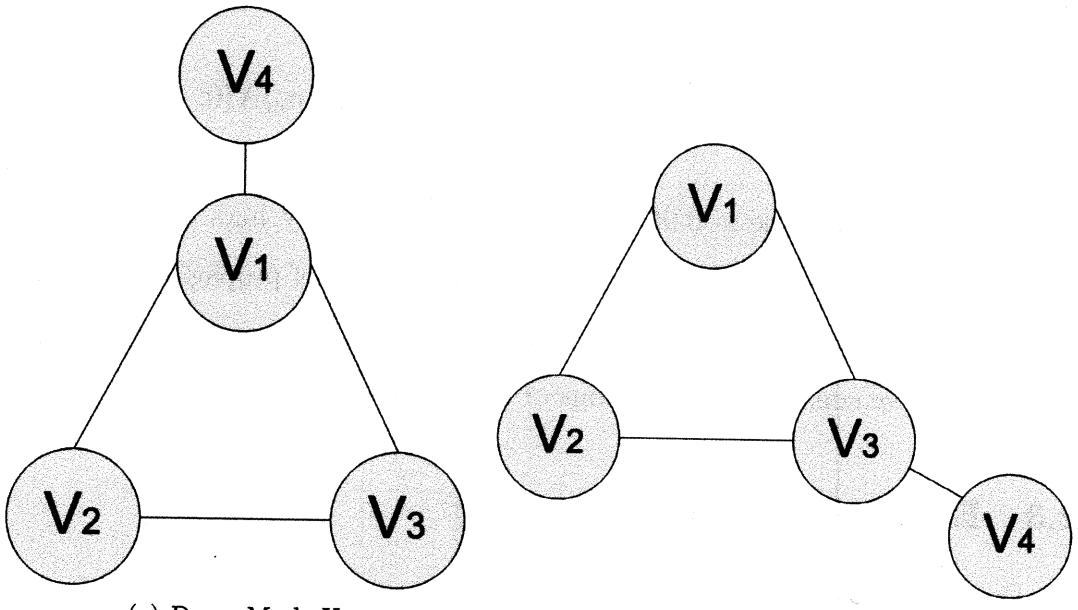
A pairwise potential is defined based on $cross(v_k^X, v_l^X)$ as follows:

$$\psi_c(c_k = i, c_l = j) = N(cross(v_i^Y, v_j^Y); cross(v_k^X, v_l^X), \sigma_c). \tag{6.7}$$

For better understanding of how this potential works, we provide an illustration here. Consider two objects in Figure 6-6. The objects are on the xy-plane. For simplicity, only four vertices are drawn. The best matching in this case is the following:

$$c_1 = 3,\ c_2 = 1,\ c_3 = 2,\ c_4 = 4.$$

Even though we can find a reasonable correspondence only using local surface signa-



(a) Demo Mesh X (b) New Mesh Y

Figure 6-6: Parts of Two Objects

tures and geodesic distances, it does not give much information regarding how v_1^X, v_2^X,

and v_3^X should be placed. The reason is that most of geodesic distances and local surfaces signatures for those vertices are similar. Here, the cross product relations play an important role to differentiate v_1^X from others. The following is the tables for cross product relations.

$cross(v_k^X, v_l^X)$		l			
		1	2	3	4
k	1		0	0	0
	2	0		1	-1
	3	0	-1		1
	4	0	1	-1	

$cross(v_k^Y, v_l^Y)$		l			
		1	2	3	4
k	1		1	0	-1
	2	-1		0	1
	3	0	0		0
	4	1	-1	0	

Table 6.1: Cross Product Relations

If $c_i = i$ for all $i = 1, 2, 3, 4$, the potential value is calculated as follows:

$$\prod_{k,l} \psi_c(c_k, c_l) = N(1; 0, \sigma_c)N(0; 0, \sigma_c) \cdots N(-1; 1, \sigma_c)N(0; -1, \sigma_c). \quad (6.8)$$

When σ_c is 0.4, this value is 1.8684×10^{-22}. Note that the value is very low. Let's consider another case where $c_1 = 3,\ c_2 = 1,\ c_3 = 2,\ c_4 = 4$:

$$\prod_{k,l} \psi_c(c_k, c_l) = N(1; 0, \sigma_c)N(0; 0, \sigma_c) \cdots N(-1; 1, \sigma_c)N(0; -1, \sigma_c). \quad (6.9)$$

When σ_c is 0.4, this value is 0.9692. Note that this is very high compared to the previous case. Therefore, it is highly probable that the belief propagation prefers the latter assignment. In practice, we should take all vertices into account instead of only four, but here, we consider only portions of meshes for illustration.

6.2.3 Experimental Results

In this sub-section, we discuss some experimental results of running the correspondence algorithm introduced in the previous sub-section. For all experiments, the following parameters were used.

$$\sigma_s = 0.03, \sigma_g = 0.3, \sigma_c = 0.4$$

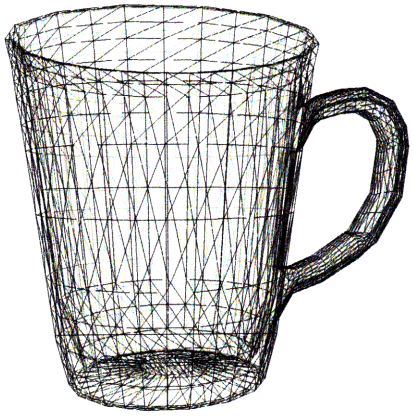
Correspondence between Identical Objects

Here, we test the algorithm on two identical objects, which means the demo mesh and new mesh are the same. Note that we perform subsampling before running the algorithm. Due to the randomness of subsampling, we have two different sets of vertices even when the original meshes are the same. To make them have the same samples, we only perform subsampling once and then use it for both the demo and new mesh in the first experiment. In the second experiment, we will use different subsamples.

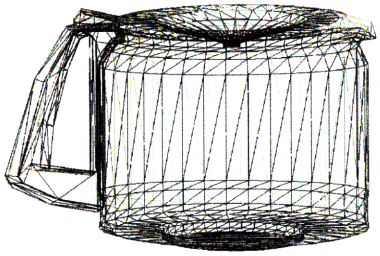
If two meshes and subsampled vertices are the same, the desired correspondence is obviously the identity function (i.e. $c_i = i$ for all i). This is a very easy case since the correspondence of form $c_i = i$ gives the highest potential for all features and therefore, the identity correspondence is the assignment that evaluates to the maximum probability. To make sure of this behavior, we tested the algorithm on 15 objects, where most of them are listed in Figure 3-2. We have achieved the identity correspondence for all test cases and the belief propagation finished in 5 iterations in all cases.

Next, we did the same experiment with different subsamples. The algorithm ran successfully on all 15 cases even though a very small number of vertices were misplaced because we did not have sampling fine enough to make sure that a perfect answer always existed. Here, four selected objects are shown in Figure 6-7, and the results of running the algorithm on them is visualized in Figure 6-8, 6-9, 6-10, and 6-11. Note that only some selected vertices were colored for simplicity. For the details of the number of total vertices and BP iterations, see Table 6.2.

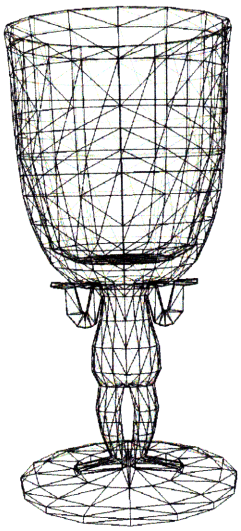
(a) Eye Glasses



(b) Mug

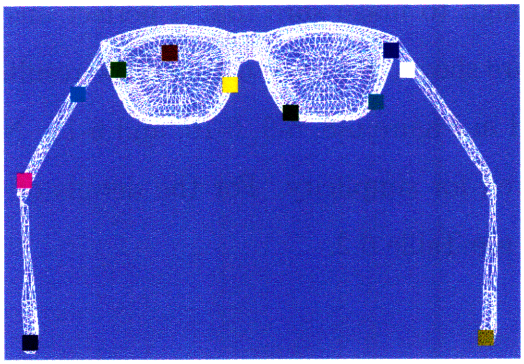


(c) Coffee Pot

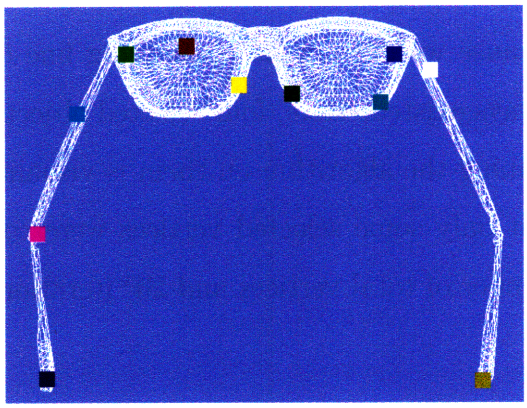


(d) Tiki Glass

Figure 6-7: Test Objects

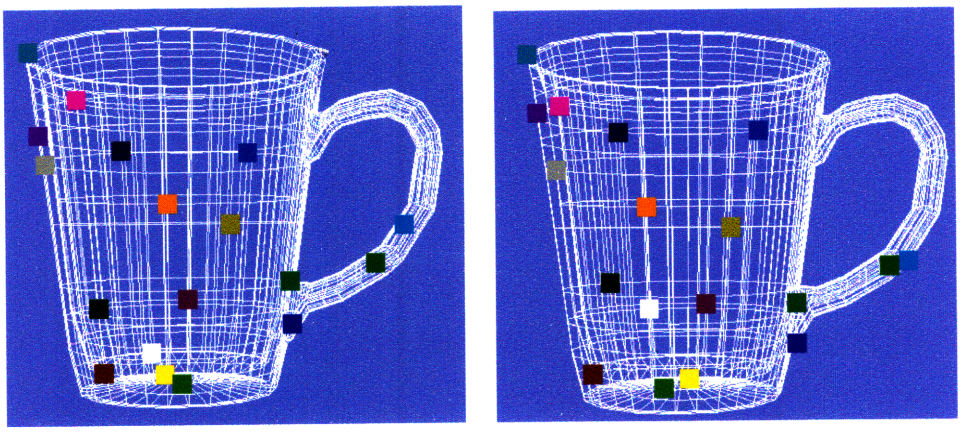


(a) Demo Mesh



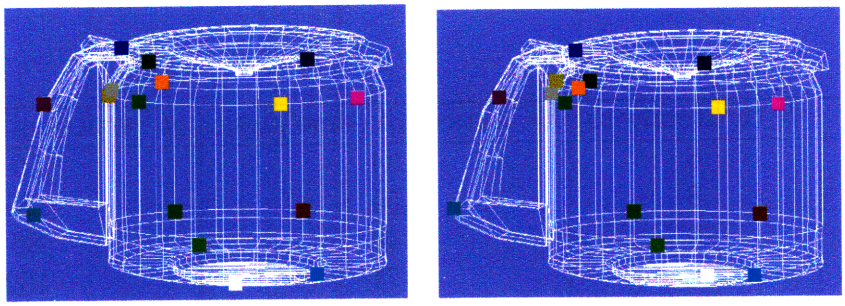
(b) New Mesh

Figure 6-8: Correspondence Result for Eye Glasses



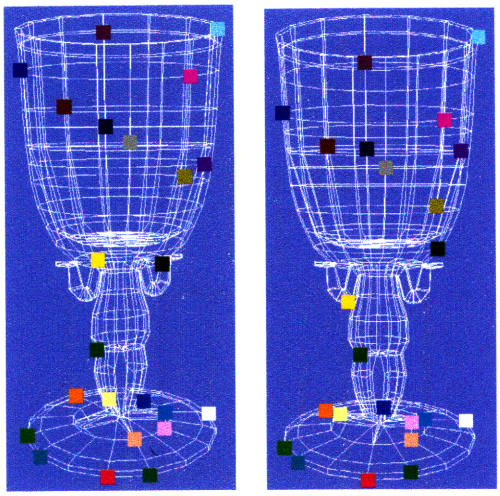
(a) Demo Mesh (b) New Mesh

Figure 6-9: Correspondence Result for Mug



(a) Demo Mesh (b) New Mesh

Figure 6-10: Correspondence Result for Coffee Pot



(a) Demo Mesh (b) New Mesh

Figure 6-11: Correspondence Result for Tiki Glass

Object	# BP iterations	# vertices of demo mesh	# vertices of new mesh
Glasses	6	34	59
Mug	8	55	105
Coffee Pot	6	48	86
Tiki Glass	9	71	102

Table 6.2: Correspondence Algorithm Result

Correspondence under Rotation

We have performed the correspondence algorithm on two objects in which one object is the rotated version of the other. Note that rotation here is restricted to be centered around z-axis. Theoretically, rotation does not change any potential since geodesic distance, local surface signatures, and cross product relations are rotation-invariant. We have performed a similar experiment as before and confirmed that the algorithm gave reasonable results within 10 iterations of BP. We omit the details here because this experiment is very similar to the previous.

Correspondence between Similar Objects

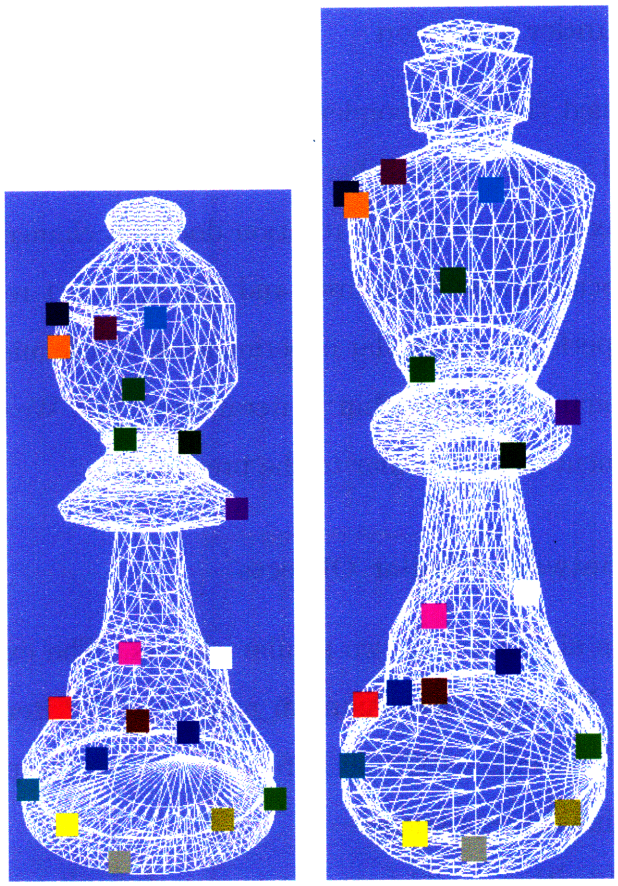
Here, we test the algorithm on two similar objects. The experiment method is the same as before. We limit the maximum number of BP iterations to 30. If the probabilities do not converge within 30 iterations, we use the probabilities after 30 iterations to determine correspondence.

We tested the algorithm on 20 cases. It is very difficult to evaluate the results since there is no way to quantify the quality of a correspondence. Visualization was the only way to see if things went well.

Out of 20 test cases, only 7 cases are introduced in the thesis to save space. The results are drawn in Figures 6-12 through 6-18, and the number of BP iterations taken for each test case is provided in Table 6.3.

Object 1	Object 2	# BP iterations	# vertices (demo)	# vertices (new)
Chess Piece 1	Chess Piece 2	8	57	187
Mug 1	Mug 2	> 30	66	166
Hat 1	Hat 2	> 30	50	158
Mug 3	Mug 4	7	55	202
Glass 1	Glass 2	11	66	171
Glass 3	Glass 4	8	46	241
Flask 1	Flask 2	> 30	58	163

Table 6.3: Correspondence Algorithm Result



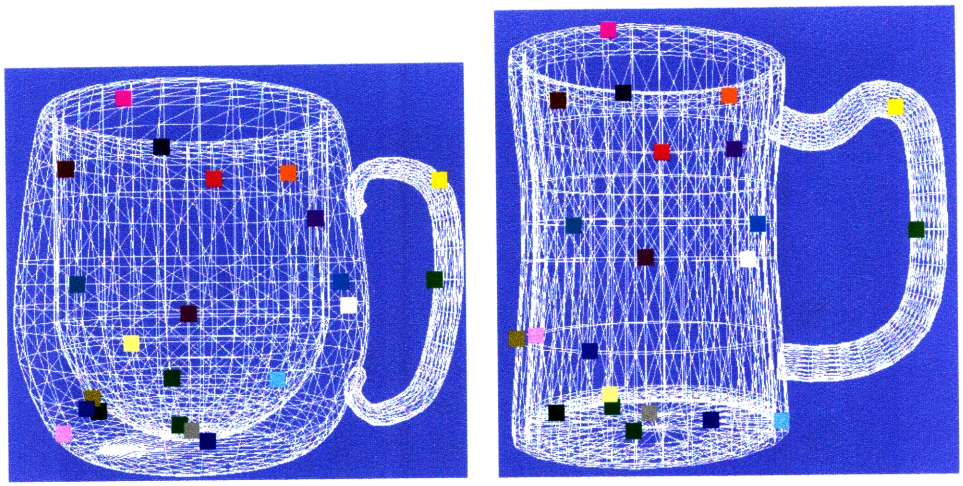
(a) Demo Mesh (b) New Mesh

Figure 6-12: Correspondence Result for Chess Pieces

Correspondence between Different Objects

In the previous experiment, we were not able to find a case in which the correspondence algorithm resulted in a catastrophic failure. However, we still need to know when and how it fails. The easiest way to figure out what happens when the algorithm fails is to try to match two totally different objects.

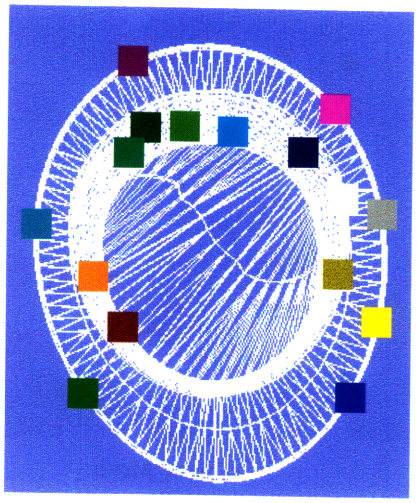
We tried to match a chess piece and a pair of glasses. The belief propagation did not converge in 30 iterations. The result looked very random. The correspondence is shown in Figure 6-19. Even though matching a chess piece and glasses gives nothing interesting, running the algorithm on two very different objects that share some geometric properties can be useful. Figure 6-20 is the result of running the algorithm on a hat and a glass. Interestingly, the algorithm converged and took only 18 iterations. In Figure 6-20, the points are reasonably placed. The points that should be on the



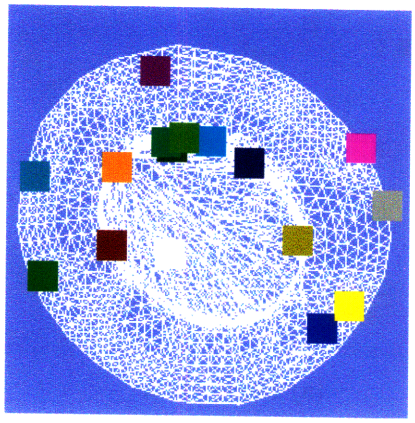
(a) Demo Mesh (b) New Mesh

Figure 6-13: Correspondence Result for Mugs

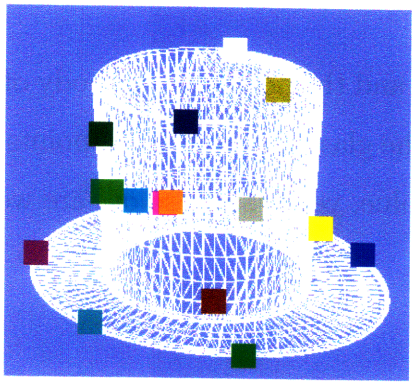
bottom part are on the bottom of the glass, and the points on the body of the hat are also matched to the points on the body of the glass. Note that the points on the bottom part where they are colored as green, blue, yellow, and so on, are well-matched.



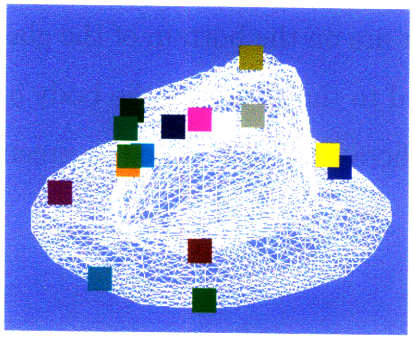
(a) Demo Mesh (Top View)



(b) New Mesh (Top View)

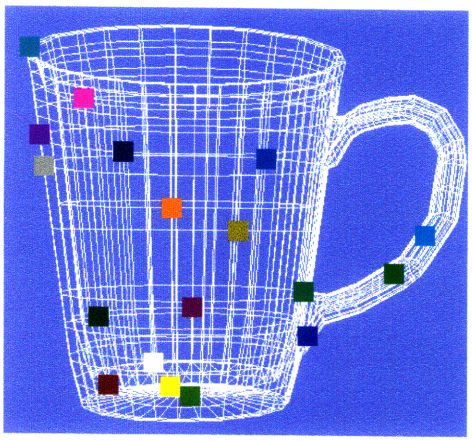


(c) Demo Mesh (Side View)

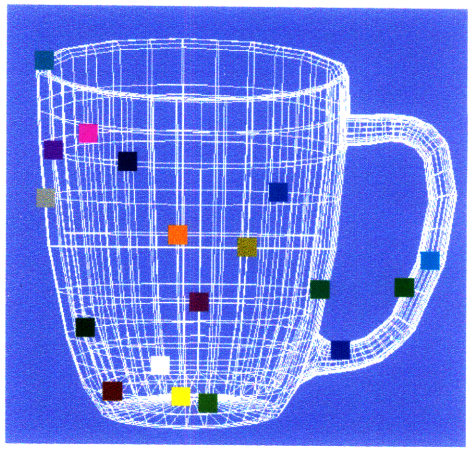


(d) New Mesh (Side View)

Figure 6-14: Correspondence Result for Hats

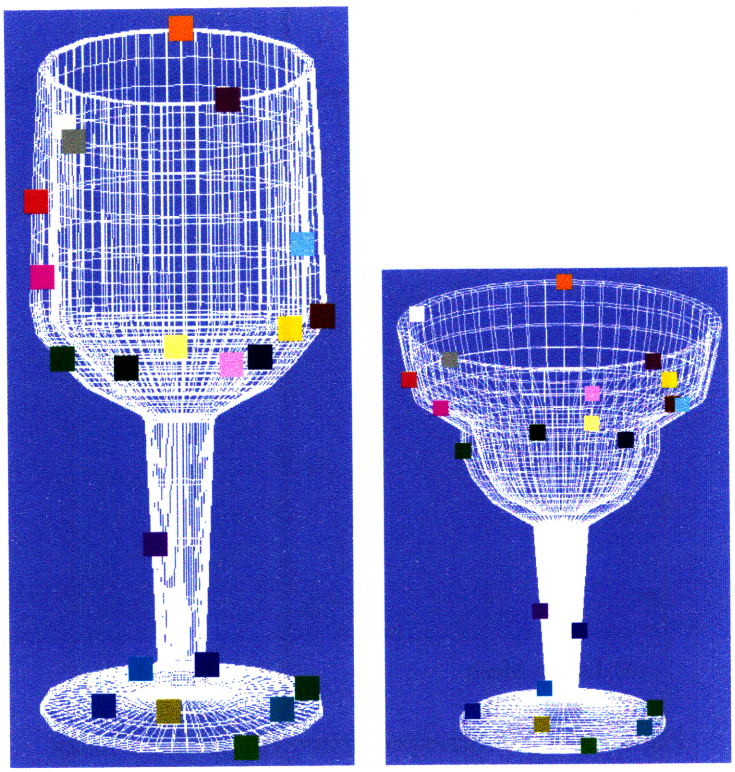


(a) Demo Mesh



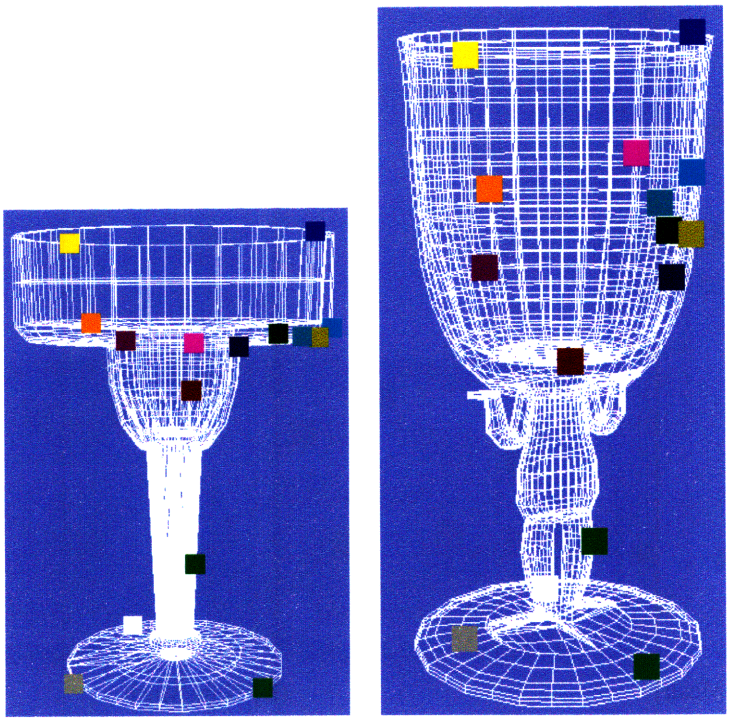
(b) New Mesh

Figure 6-15: Correspondence Result for Mugs



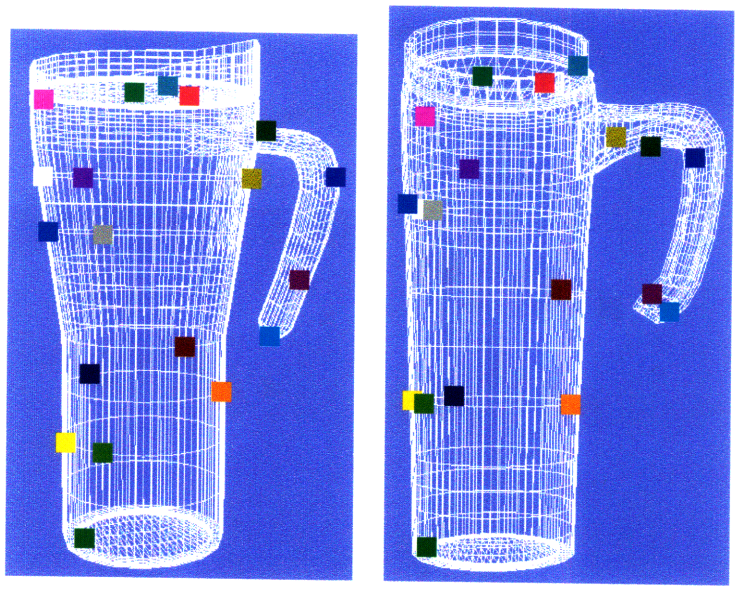
(a) Demo Mesh (b) New Mesh

Figure 6-16: Correspondence Result for Glasses 1



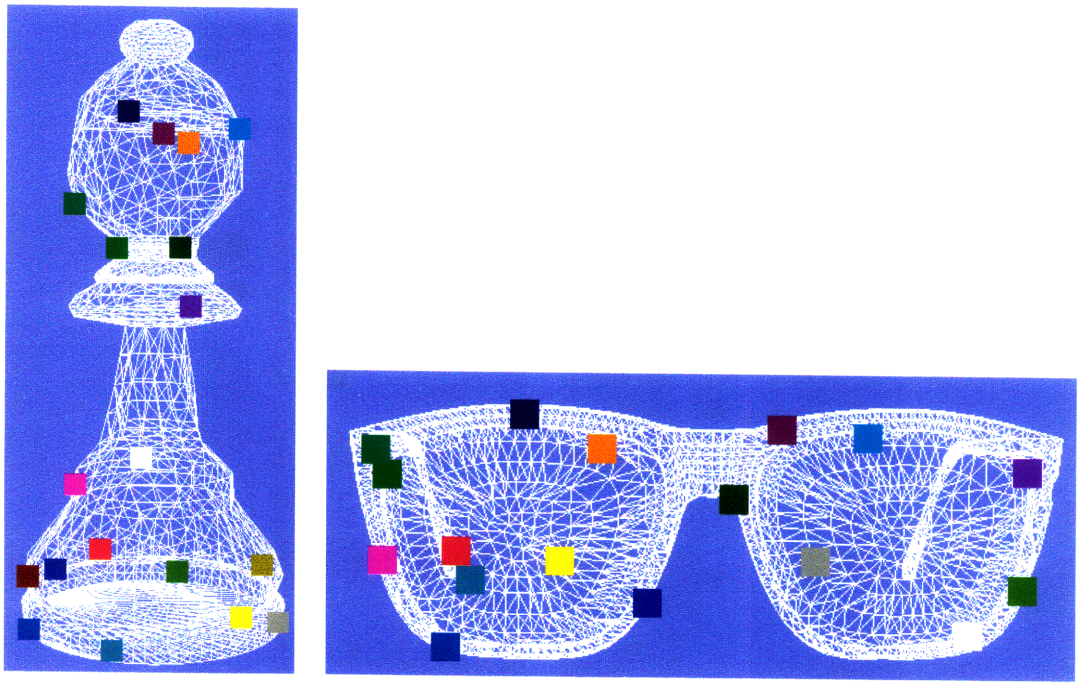
(a) Demo Mesh (b) New Mesh

Figure 6-17: Correspondence Result for Glasses 2



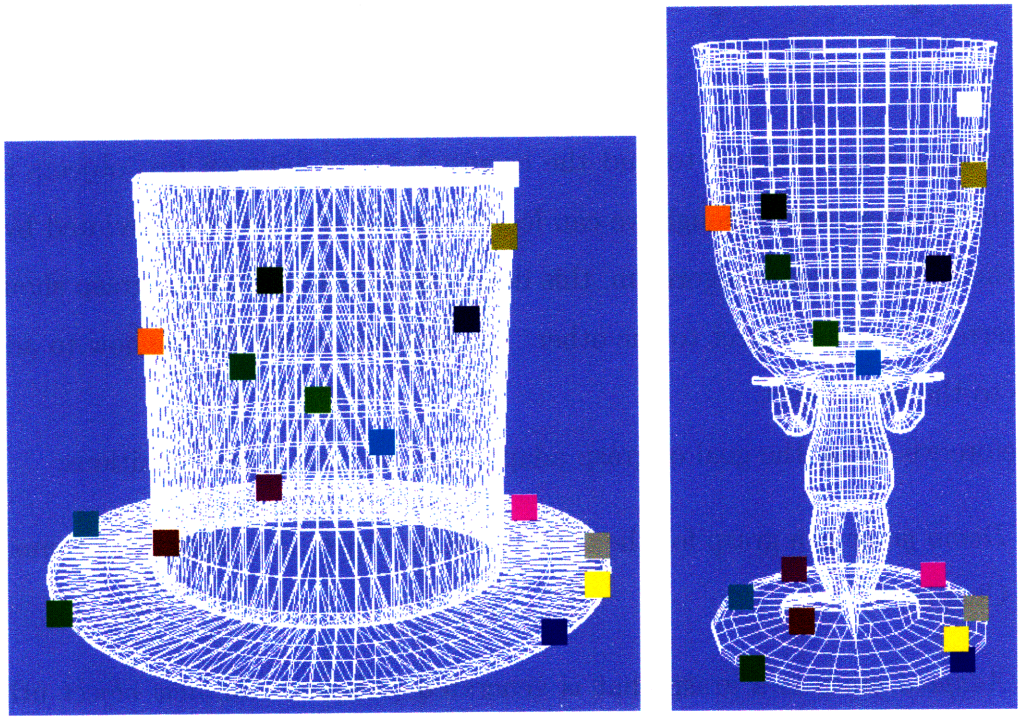
(a) Demo Mesh (b) New Mesh

Figure 6-18: Correspondence Result for Flasks



(a) Demo Mesh (b) New Mesh

Figure 6-19: Correspondence Result for Chess Piece and Eyeglasses



(a) Demo Mesh (b) New Mesh

Figure 6-20: Correspondence Result for Hat and Glass

Chapter 7

Grasp Adaptation

We have discussed how to find the correspondence between two objects in the previous chapter. Even though we can learn how two objects are correlated by running the correspondence algorithm, this does not directly tell how a grasp should be transferred from one object to the other. In this chapter, we discuss how to adapt a grasp to the new object.

Before we define the goal of grasp adaptation, we need some definitions:

1. *Demo grasp* is a grasp for the demonstration object (for definition of *grasp*, see Chapter 4).

2. *Adapted grasp* is a grasp that is generated (adapted) for *new object* utilizing information of demo object, new object and demo grasp.

The goal of grasp adaptation is defined as follows:

1. The adapted grasp should be located around the region of new object that corresponds to the region of demo object in which the demo grasp is placed.

2. The quality of the adapted grasp should be reasonably high.

The first goal ensures that the adapted grasp makes sense to people. Consider a situation in which you transfer a grasp for a cup to the other cup with the similar geometric appearance. If the original grasp is placed around the handle of the cup,

it makes more sense to locate the adapted grasp around the handle of the other cup than around its body.

Even though we can place the adapted grasp in a reasonable region by satisfying the first goal, we should make sure that the quality of the grasp is reasonably high. If the second goal is not satisfied, we can end up with a grasp that is roughly correct in global position, but is not effective in terms of grasp quality discussed in Chapter 4. By satisfying both of the goals, we can make sure that the grasp is well adapted.

7.1 Algorithm

In this section, we propose an algorithm that finds a grasp that satisfies the goals mentioned above. The basic idea is that we generate an initial grasp for new object by transforming each contact point of the demo grasp using global correspondence information. Then, we adjust the points of the initial grasp until we find a *good* grasp. Here, a grasp is *good* if its quality is larger than $\epsilon = 0.01$. Note that a *good* grasp is force-closure.

A contact point of the adapted grasp is one of mesh vertices of new object. This discretization is required because our search space should be tractable. Even though there is no such constraint on the demo grasp, it is possible that the demo grasp is also synthesized and stored by our grasp system. Then, all contact points would be located at the positions of mesh vertices.

In order to find a good grasp, we move the points around. For this exploration, we use *local search*. Local search is an optimization algorithm that moves from solution to solution in the search space until a local maximum is found or a time bound is elapsed. Typically, every candidate solution has several neighbor solutions. In this thesis, the choice of which one to move to is done by taking the one locally maximizing grasp quality. Such heuristic is often called *hill climbing*. A neighbor grasp is generated in the following way: a contact point (mesh vertex) can either stay or move to a neighbor vertex. Obviously, at least one point has to move.

In the above, we have described the basics of the algorithm. In the following, the

steps of the algorithm are described in detail:

1. For each point contact of the demo grasp, find the closest vertex of the demo mesh.

2. Create the initial grasp by taking the corresponding vertices of *new object*, and set this grasp to the current grasp.

3. Repeat the following step until the quality of the current grasp is higher than ϵ.

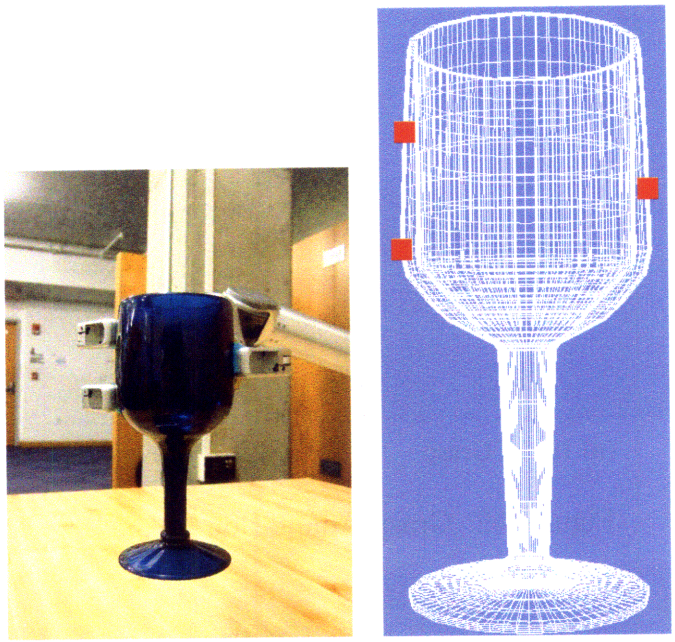
4. Evaluate the quality of all neighbor grasps and take the one maximizing the quality. If its quality is worse than the current grasp, terminate the search. Otherwise, set this grasp to the current grasp.

Under the assumption that the correspondence algorithm successfully matches two objects, the initial grasp contacts are placed in the correct parts of the object. If there is a good grasp that can be achieved by moving some points around, the algorithm is likely to find it since the algorithm tries neighbors first. Because the algorithm can continue moving to new solutions for many rounds, the algorithm might end up with a grasp that looks very different from the demo. This is because finding a force-closure grasp is considered more important than keeping the appearance of the demo grasp. If keeping the demo grasp is more important, we can limit the search space to the grasps that have similar signatures (e.g. appearance) as the demo grasp.

7.2 Experimental Result

We have run the algorithm on ten test cases, and successfully found good grasps in eight cases. In the following, we first introduce some cases where the algorithm succeeds, and then show a failure.

The first test case is glass. The demo grasp is shown in Figure 7-1. We ran the algorithm to adapt this grasp to other three different glasses. The resulting grasps are provided in Figure 7-2, 7-3, and 7-4. The algorithm found force-closure grasps



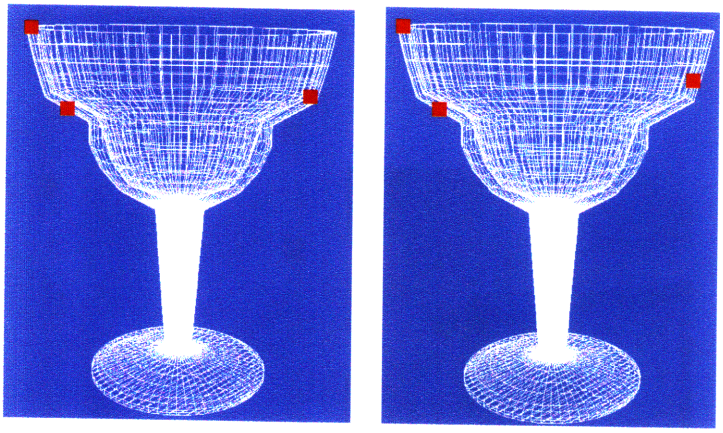
(a) Robot Hand Grasping a Glass (b) Demo Grasp for Glass

Figure 7-1: Picture and Computer Model of Demo Grasp for Glass

after one iteration of local search in the first two cases, and in the last case, the initial grasp was already force-closure so the final grasp is the same as the initial.

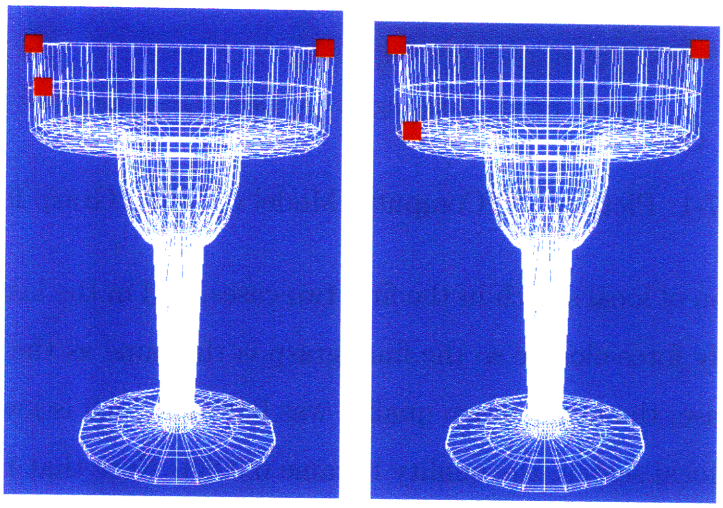
In the first case, the initial grasp quality was 0.0055, which is lower than ϵ. After one iteration of local search, the quality became 0.0368. Note that the contact point on the right side moved slightly upward. In the second case, the quality value changed from 1.52×10^{-5} to 0.0404. Two contact points of the initial grasp were too close, but the algorithm enlarged the space between them in order to increase the grasp quality. In the last case, the initial quality was 0.0328, so there was no local adjustment. In all three cases, the algorithm found grasps that were force-closure and similar to the demo grasp in terms of shape and location.

We have seen some successful cases above. In the following, we describe a limitation of the algorithm. The demo object and grasp are shown in Figure 7-5. This grasp is adapted to the object shown in Figure 7-6, and the initial and final grasps are provided in Figure 7-6 and Figure 7-7, respectively. The initial and final grasp qualities are 0.0041 and 0.0837, respectively. Even though the final grasp is force-closure, this grasp is considered to be a failure for two reasons. First, three points are



(a) Initial Grasp (b) Final Grasp

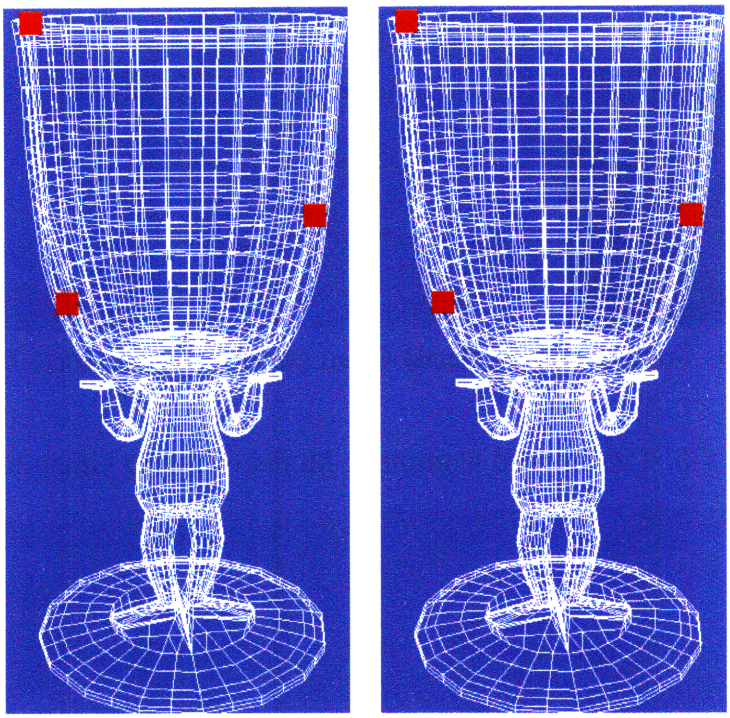
Figure 7-2: Grasp Adaptation to Glass 1



(a) Initial Grasp (b) Final Grasp

Figure 7-3: Grasp Adaptation to Glass 2

not horizontally located whereas those of the demo are horizontally placed. Second, because vertical positions of contact points are too low, it may be impossible for a robot hand to reach the grasp contacts from the top of the object, whereas the demo grasp is top grasp (see Figure 7-5). Some suggestions to make these cases work are provided in the next section.



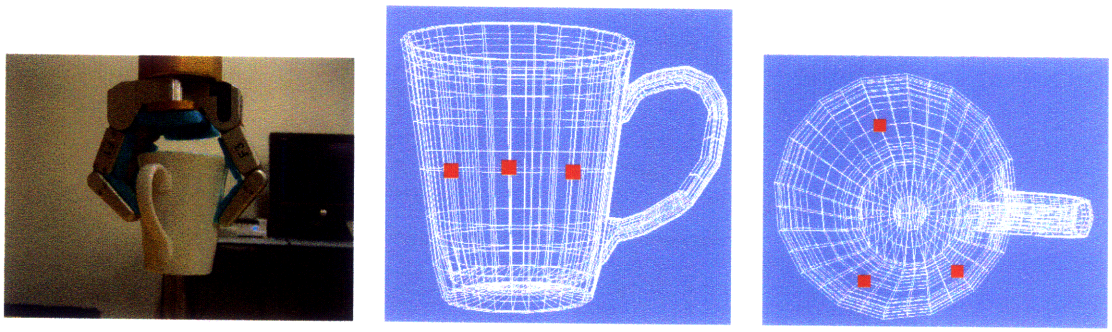
(a) Initial Grasp (b) Final Grasp

Figure 7-4: Grasp Adaptation to Glass 3

7.3 Discussion

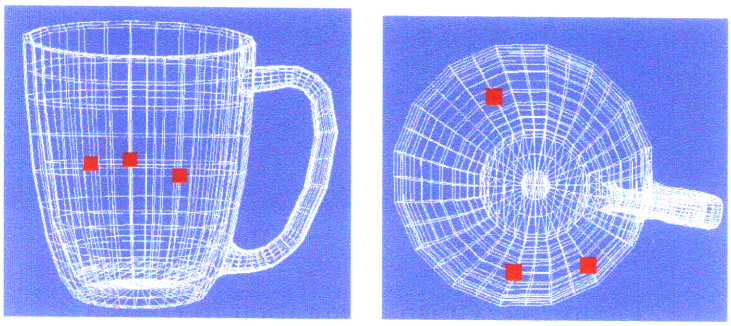
In summary, the grasp adaptation algorithm was successful in 8 out of 10 cases. In most cases, one or two iterations of local search were enough as illustrated in the previous section. Even though the adaptation algorithm works well in general, it has the following pitfalls. First, the final grasp can be very different from the demo grasp if local search takes many rounds of movements. Second, since we do not consider the robot hand kinematics, the generated grasp might not be reached in reality. This is illustrated in the mug case of the previous section. However, it is often the case that the final grasp is feasible since the local search starts from the transformation of the demo grasp and the final grasp does not deviate much from the initial.

To change the behavior of the algorithm, more constraints can be added. One simple constraint is the appearance preservation of the demo grasp. In this way, we can ensure that the robot hand configuration used for the final grasp is the same as that for the demo grasp. In addition to this constraint, one can also limit the search



(a) Robot Hand Grasping a Mug (b) Original Grasp (Mug, Side View) (c) Original Grasp (Mug, Top View)

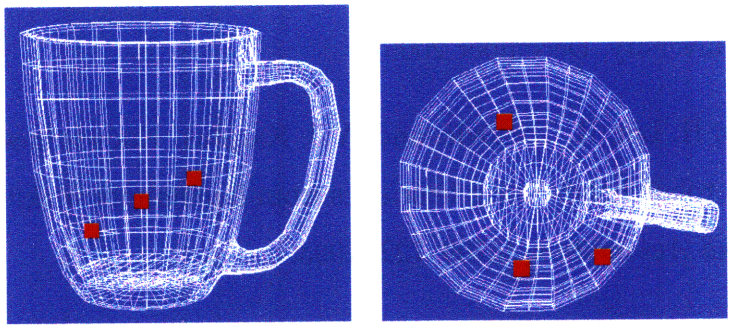
Figure 7-5: Picture and Computer Model of Demo Grasp for Mug



(a) Initial Grasp (Side View) (b) Initial Grasp (Top View)

Figure 7-6: Initial Grasp of Grasp Adaptation (Mug)

space to grasps that are kinematically feasible. This limitation may be essential to use the grasp system in practice, but evaluating the feasibility of grasps in terms of robot hand kinematics is beyond the scope of the thesis.



(a) Final Grasp (Side View) (b) Final Grasp (Top View)

Figure 7-7: Final Grasp of Grasp Adaptation (Mug)

Bibliography

[1] D. Anguelov. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Proceedings of the Neural Information Processing Systems (NIPS) Conference*, 2004.

[2] Drago Anguelov. *Learning Models of Shape from 3D Range Data.* PhD thesis, Stanford University, December 2005.

[3] G.A. Bekey, H. Liu, R. Tomovic, and W.J. Karplus. Knowledge-based control of grasping in robot hands using heuristics from human motor skills. *IEEE Trans. Robotics and Automation*, 9:709–722, 1993.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:509–522, 2002.

[5] E. Chinellato, R.B. Fisher, A. Morales, and A.P. del Pobil. Ranking planar grasp configurations for a three-finger hand. In *Proc. IEEE ICRA 2003*, pages 1133–1138, 2003.

[6] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[7] B. Curless and M. Levoy. A volumetric method of building complex models from range images. In *Proceedings of SIGGRAPH 1996*, pages 303–312, 1996.

[8] C. Ferrari and J. Canny. Planning optimal grasps. In *Proc. IEEE ICRA 1992*, pages 2290–2295, 1992.

[9] K. Hsiao and T. Lozano-Pérez. Imitation learning of whole-body grasps. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[10] Z. Huang, R.Boulic, N. Magnenat Thalmann, and D. Thalmann. A multi-sensor approach for grasping and 3d interaction. In *Proceedings of Computer Graphics International*, 1995.

[11] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching.* PhD thesis, Robotics Institute, Carnegie Mellon University, August 1997.

[12] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In *the NATO Advanced Study Institute on Learning in graphical models*, pages 105–161, 1998.

[13] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Proceedings of Symposium on Geometry Processing*, pages 156–165, 2003.

[14] B. Kim, S. Oh, B. Yi, and I.H. Suh. Optimal grasping based on non-dimensionalized performance indices. In *Proc. IEEE IROS 2001*, pages 949–956, 2001.

[15] C.A. Klein and B.E. Blaho. Dexterity measures for the design and control of kinemat- ically redundant manipulator. *Int. J. Robotics Research*, 6(2):72–83, 1987.

[16] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6):799–822, December 1994.

[17] Michael Leventon. *Statistic models in medical image analysis.* PhD thesis, Massachusetts Institute of Technology, 2000.

[18] Y. Li and N. Pollard. A shape matching algorithm for synthesizing humanlike enveloping grasps. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2005.

[19] Tomás Lozano-Pérez. The design of a mechanical assembly system. Master's thesis, Massachusetts Institute of Technology, 1976.

[20] B. Mirtich and J. Canny. Easily computable optimum grasps in 2d and 3d. In *Proc. IEEE ICRA 1994*, pages 739–747, 1994.

[21] Niloy J. Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of Symposium on Computational Geometry*, pages 322–328, 2003.

[22] K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. In *Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 99)*, pages 467–475, 1999.

[23] Van-Duc Nguyen. Quantitative steinitzs theorem with ap- plications to multifingered grasping. *Discrete and Computational Geometry*, 7(3):295–318, 1992.

[24] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3), June 1988.

[25] H. Ogata and T. Takahashi. Robotic assembly operation teaching in a virtual 151 environment. *IEEE Transactions on Robotics and Automation*, 10:391–399, June 1994.

[26] G. Paul, M. D. Wheeler, and K. Ikeuchi. Modelling human assembly actions from observation. In *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, December, 1996.

[27] N. Pollard and J.K. Hodgins. Generalizing demonstrated manipulation tasks. In *Workshop on the Algorithmic Foundations of Robotics (WAFR '02)*, December 2002.

[28] N. S. Pollard. Synthesizing grasps from generalized prototypes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.

[29] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 339–348, 1991.

[30] R. M. Sanso and D. Thalmann. A hand control and automatic grasping system for synthetic actors. *Comput. Graph. Forum*, 13:167–177, 1994.

[31] A. Saxena and et al. Robotic grasping of novel objects. In *Proceedings of the Neural Information Processing Systems (NIPS) Conference*, 2006.

[32] J. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge Univ. Press, 1999.

[33] J. M. Siskind. Visual event classification via force dynamics. In *Proceedings of AAAI-2000*, 2000.

[34] C. Tung and A. Kak. Automatic learning of assembly tasks using a dataglove system. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 1995.

[35] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. Technical Report TR2000-026, Mitsubishi Electric Research Laboratories, 2000.

[36] T. Yoshikawa. Manipulability of robotic mechanisms. *Int. J. Robotics Research*, 4(2):3–9, 1985.