UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

A 3D FEATURE-BASED OBJECT RECOGNITION SYSTEM FOR GRASPING

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

DI WANG
Norman, Oklahoma
2007

A 3D FEATURE-BASED
OBJECT RECOGNITION SYSTEM
FOR GRASPING

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

By

Prof. Andrew H. Fagg

Prof. Dean F. Hougen

Prof. Amy McGovern

Dedication

This thesis is dedicated to Aijun Wang and Huiping Zheng. Also, to Dr. Andrew Fagg for all the help in this research.

Acknowledgments

Special thanks to Dr. Andrew Fagg for mentoring, teaching, guiding, and encouraging this endeavor. This work has been supported in part by the University of Oklahoma and by the National Science Foundation under Grant No. 0453545.

Contents

List Of Tables

List Of Figures

Abstract

Robotic grasping and vision are two important domains in robotics research. Recent research has shown that infants pay more visual attention to hand movement than faces during their early development (Smith et al. 2007). This suggests that the growth of visual and grasping skills should be intertwined from the beginning of an agent's "life." In this thesis, I take several steps to bridge the gap between robot grasping and vision. Following the work of Coelho and Grupen, I first approach the problem of grip selection on an object as one of haptic search in which the fingers are incrementally displaced until a suitable grasp is found. I propose a method that can dynamically detect the local curvature of the object and modify the incremental displacements as a function of whether the surface is concave or convex. This approach dramatically extends the set of applicable objects for which stable grasps can be found. However, as a gradient descent method, this approach suffers from the problem of local minima. One way to combat this is to use *a priori* knowledge derived from an image of the object to select an initial placement of the hand near the correct position and orientation. I propose a method of visually identifying the direction from which the object is viewed (which is referred to as *aspect recognition*). This will ultimately be used to estimate the pose of the object, and consequently to place the hand before haptic exploration commences. The aspect recognition model is trained using experience from simultaneously manipulating and watching the target object. Constellations of image features are used to represent the appearance of the object at certain viewing angles. Clusters of viewing angles in which a given constellation is viewable are represented as probability density functions defined over the unit sphere. By combining the evidence from several observed constellations, one can infer an estimate of viewing angle given a novel image. I demonstrate empirically that constellations which are locally robust and yet selective over all possible viewing angles lead to improved aspect recognition rates over the ones that do not satisfy these properties.

Chapter 1

Introduction

Some of the most state-of-the-art robots in the world are humanoid: they can see, hear, speak, move and manipulate the environment. Both robotic vision and manipulation are important: a robot needs to interact with the world and needs to know what objects are around it and where they are located. Both problems are difficult due to the dynamic and stochastic nature of the world and the large amount of information that may be brought to bear. In order to manipulate an object, a robot first needs to grasp it stably. In order to grasp the object, the robot must utilize the information from both vision and haptic systems. In this thesis, I am taking several steps to bridge the gap between vision and grasping following these basic ideas.

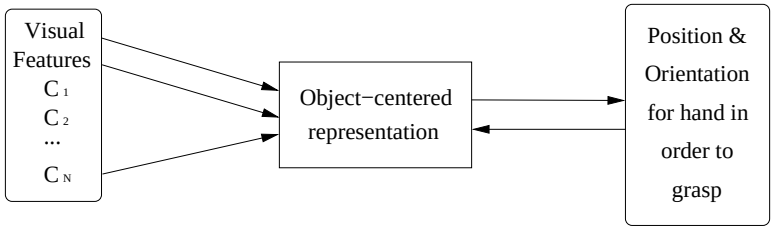


Figure 1.1: Overall structure of a vision-grasping system.

The relationship between vision and grasping can be described by Figure 1.1. First of all, visual features can suggest different grasp types and hand orientations. I propose to mediate this process through an object-centered representation, which could include object identity, position, pose and shape. This object-centered representation is more stable than the raw visual features, since the latter is subject to lighting conditions, occlusions, viewing angle and scale change. In this way, the entire system is separated into two parts. The goal for the first part is to capture a model that includes a set of visual features and a quantitative evaluation of how each feature

matches an object as it is rotated in 3D. Given such a model and a new image of an object, we should then be able to infer the object pose relative to the camera (down to symmetries). The second part corresponds to the set of grasp affordances that describe the different ways that the hand could be placed relative to the object in order to grasp it, and is addressed in part in previous work (de Granville, Southerland and Fagg 2006). Also, the act of executing a grasp can contribute evidence for the object-centered representation. In this thesis, I focus on the first part.

A classic way to approach robotic grasping is to use an *a priori* model of the manipulated object. However, we would like the robot to be able to figure out by itself how to grasp an unknown object. Teichmann and Mishra (1994) and Coelho and Grupen (1997) suggest that one way to accomplish this is for a robot to haptically explore an object until a quality grasp is found. A quality grasp can be defined as one that minimizes the total force and torque applied to the object by the hand at the set of contacts (Coelho and Grupen 1997). Coelho and Grupen suggest that we can formulate this search in terms of a gradient descent problem in which the robot successively probes an object and then moves the contacts. Based on this idea, I have reformulated Coelho and Grupen's algorithm to operate strictly in Cartesian space as opposed to a coordinate system that describes the surface of the object. The *force controller* (the controller that reduces the total force applied to the object), *moment (torque) controller* (the controller that reduces the total moment) and any other high level controllers can be combined together using null space projection (Platt et al. 2002). However, in the previous work, the force control law assumes that the finger moves on a convex surface. This will cause the finger to move in the wrong direction when the local surface is concave. I propose a switching controller that can choose a different control law depending on whether the local surface is convex or concave (Wang et al. 2007). I have shown in simulation experiments that the switching controller consistently outperforms the original approach over a wide range of parameter settings.

However, since this controller is based on a gradient descent algorithm, it inevitably suffers from the problem of local minima. As a compensation, a robot can use vision to pre-shape its hand into a position and orientation in which our grasp controller is more likely to succeed. Such hand poses can be associated with the appearance of an object either by observing the grasps of a human teacher or by using the grasp controller to find quality grasps in a global search. Piater and Grupen (2002) use appearance features to pre-shape a robot hand, which is then guided by

haptics during the execution of a grasp. These features are chosen so that a successful grasp is much more likely than when random initial hand poses are chosen. As a result, the system learns object-specific grasp parameters by itself and increases the quality of the haptic grasp system considerably.

In their paper, Piater and Grupen construct an association from a single constellation of edgels (short edges found in the image) to the set of possible grasps. An alternative approach is to use the responses of all constellations of edgels which vote for every grasp type. My hypothesis is that by combining decisions from a set of constellations, we can achieve a more robust system. In this thesis, I use visual features extracted from a set of images containing objects to be grasped in order to predict successful grasp type. Here, each grasp type is a combination of the approach direction (top or side) and the 3D hand orientation (two or three degrees of freedom are constrained) that one can use to grasp a target object. The construction of this constellation-grasp association is formulated as a supervised learning process. The algorithm extracts features from training images and ultimately learns a function that maps feature responses to grasp types. After learning enough features, the algorithm can achieve high accuracy of predicting the grasp types associated with two simple objects contained in the test images.

Another limitation of Piater and Grupen's work is that the objects are planar and the camera view is always from the top of the objects. I propose a method that explicitly acknowledges the fact that appearance of an object changes as the viewing angle changes, and models an object as a set of constellations that cover different object aspects. One of the key challenges to efficiently applying this approach to the 3D domain is to select constellations that do not incorporate non-object features (including those of the background or shadows), and yet are useful in identifying the viewing angle. I propose to first employ pre-filtering using a small set of images with nearby viewing angles. We select constellations that can be consistently observed in this set of neighborhood images. Specifically, I hypothesize that appropriate feature constellations should be robust to small changes in object viewing angle, but selective to a subset of possible viewing angles.

I formulate the process of acquiring a visual model as a supervised learning problem in which a set of <image, object orientation> tuples are provided. The object orientations correspond to points on a unit sphere with the object in the center (we call this the *aspect sphere*). A constellation is locally robust if it can be observed in a set of images with similar viewing angles (aspects) and globally discriminative if it

cannot generally be observed when the viewing angle changes substantially. The set of points on the aspect sphere where a constellation can be observed is summarized by a probability density function (pdf). In turn, these pdfs give us a principled way to estimate the aspect of a known object in an unknown scene given that a set of constellations are observed.

This thesis is organized as follows: In Chapter 2, I discuss previous work on object recognition and grasping; in Chapter 3, I give some statistics and mathematics background on probability distributions on the unit sphere; in Chapter 4, I propose a haptic-based grasp controller; in Chapter 5, I focus on the 2D object recognition problem and in Chapter 6, I focus on the 3D aspect recognition problem.

Chapter 2

Related Work on Grasping and Vision

2.1 Robotic Grasping

Robotic grasping approaches have traditionally relied on *a priori* knowledge of the geometry of the object being grasped or on estimates of the geometry based on visual or range inputs (e.g., Bekey et al. 1993, Borst et al. 1999 and Miller et al. 2003). The modeled geometry is used to assess the quality of many potential grasps before one is selected for execution. When the number of target objects is large, it becomes cumbersome to pre-define the association between each object shape and grasp types. In an alternative approach, one can make minimal *a priori* assumptions about the geometry of the object being grasped, and instead rely on haptic feedback to direct the tactile exploration of an object until a suitable grasp is found. Teichmann and Mishra (1994) and Coelho and Grupen (1997), introduced methods in which the local surface normal for each of several contacts was first estimated. Based on this information, contact displacements were computed that followed the negative gradient of a cost function. The cost functions were such that their minima corresponded to a quality grasp of the object. In Teichmann and Mishra (1994), this cost function was based on the area of the triangle formed by three contact points. In Coelho and Grupen (1997), two cost functions were used: one that described the net force applied to the object by the set of contacts, and another that described the net moment applied by the same contacts.

In the case of Coelho and Grupen (1997), and in subsequent work by Platt et al. (2002) and Platt (2006), the gradients of the cost functions were estimated by making simple assumptions about the object's surface properties, including local geometry. First of all, the contacts were assumed to be frictionless. In the case of the force cost function, the surface was also assumed to be locally convex, and in particular, that it

was a unit sphere. In the case of the moment cost function, the surface was assumed to be planar. The *force controller* and *moment controller* were defined such that each contact was displaced so that it followed the negative gradient of the corresponding cost function. Platt et al. combined the actions of the two controllers through a nullspace operation that favored the actions of the force controller over those of the moment controller. This work has shown promise in enabling a grasping system to interact with objects of unmodeled geometries (Platt 2006; Platt et al. 2006). However, the current approach suffers when the object's surface differs substantially from the assumption of local convexity. Specifically, when the surface is concave, the force controller will drive the contact in a direction that serves to increase the net force rather than decrease it. This behavior substantially limits the class of objects that can be addressed by this grasp search approach.

The challenge, therefore, is how to appropriately address objects that contain concavities. Park and Starr (1990) describe a soft-finger grasp synthesis method for polygons of known shape that considers as potential contact locations edges and/or convex and concave vertices. Funahashi et al. (1996) analyzed the stability of grasps involving fingers with controllable stiffness and either concave object surfaces or fingers. However, they do not address the grasp synthesis problem.

2.2 Object Recognition

As a main issue in computer vision, object recognition has to deal with three major problems: representation, detection and learning (Fergus et al. 2003). For representation, we need to find a method to represent the features that can be used to discriminate one object from another. Ideally, these features should be invariant even when scale, pose or light conditions change. There are two main categories of methods (Belongie et al. 2002): feature-based methods and color-based methods. The former suggests that the geometric configurations of features are important; the latter focuses on the distribution of gray or color values of corresponding regions. The next question is how to detect a feature in the corresponding regions of two given images, given that the same object may be rotated, translated or scaled in the different images. Ultimately, a system should be able to learn useful features automatically in an unstructured environment.

Pontil and Verri (1998) first transformed color images into gray-level images and then reshaped the image pixel matrices into vectors. Images from a pair of objects

were classified by using linear Support Vector Machines (SVMs) in the vector space without differentiating their poses. From their experimental results, SVMs were well suited for appearance based recognition. However, their approach suffered from a major problem in that the system required a large amount of memory when the number of objects increased, because it computed an *optimal separating hyperplane* for each object pair. Ardizzone et al. (2000) proposed a shape recognition algorithm for single objects contained in regions of interest (ROIs). Each object's shape was described in terms of eigenvalues of the covariance matrix computed from the pixel rows of the ROI and those eigenvalues were arranged as a vector. This method was robust to rotations or scaling, and to some extent, lighting conditions. However, the system failed in presence of complex textures. Wang et al. (2006) used an entropy-based feature detector to detect similar features in different images taken from natural scenes. They then estimated the parameters of the scale-invariant features by using expectation maximization (EM) and the images were classified by visual features in a Bayesian manner.

Piater and Grupen (2002) used appearance features to pre-shape a robot hand for tactile based grasping. In their work, primitive features were combined into more discriminative compound features. They employed two types of primitive features. An *edgel* is a small edge with intensity and orientation. A *texel* is a vector that encoded a local texture signature. A *constellation* is a compound feature combining primitive features with rigid position and orientation relationship between them. A constellation of features is rotation-invariant in the image plane. They used feedback from object recognition as well as robotic grasping performance to evaluate whether a constellation of features was useful or not. Useless features would be ultimately removed from the feature repository and replaced with other candidates. As a result, the system learned object-specific grasp parameters by itself and considerably increased the quality of a haptic-based search for a grasp by choosing an initial hand shape and pose visually.

2.3 Aspect Recognition

Schiele (1997) introduced a robust 3D object recognition algorithm. The basic idea was that by taking images of an object from different aspects, an object could be recognized with less ambiguity than it could from a single image. However, since their goal was object recognition, all features corresponding to different aspects of an

object were equivalent, that is, features were not associated with specific aspects. A framework of object pose recovery was given by Detry and Piater (2007). In their work, local 3D features were combined together into meta features hierarchically. Features corresponding to the same object pose were clustered by nonparametric probability distributions in 3D pose space. The computational advantage was that the set of observations for each feature could be directly used as a density function. Thus no cumbersome training process was needed to estimate the pdf, as is often the case with parametric distributions. However, they did not discuss the problems of how to systematically generate useful features, how to detect primitive features or how to represent symmetries of a given object.

Chapter 3

Probability Density Functions on the Unit Sphere

3.1 Object Appearance as a Function of Aspect

It's not hard to imagine that when we rotate an object in front of a camera, the appearance of an object is a function of the direction of viewing. Here, we define the *appearance* of an object as a specific (relative) geometrical configuration of a set of 2D features. Specifically, we can represent all possible viewing *aspects* as the set of points on the unit sphere (we will call it the *aspect sphere* hereafter) with the observed object as the center (Figure 3.1). Since the object appearance is represented in a 2D image which lies in the image plane of the camera, each point on the aspect sphere also uniquely corresponds to a direction of the camera axis. However, we should note that a single point on the aspect sphere does not uniquely correspond to a camera pose, since object appearance is invariant under rotation about an axis that is perpendicular to the image plane (we call this axis the *camera axis*).

The fact that every point on the aspect sphere corresponds to a 2D appearance of the observed object turns the 3D recognition problem into a 2D recognition problem. We can therefore make use of 2D object recognition techniques. In this thesis, since I consider object color as an irrelevant feature for the selection of hand pose and shape, I choose a geometric feature-based object recognition technique. Geometric feature-based object recognition algorithms generally combine primitive features into higher-level features. When a higher-level feature becomes more complex, its selectivity increases but the computational efficiency drops. In previous work (Piater and Grupen 2002), an edgel is a primitive feature that is directly generated from edges and a constellation of edgels is defined as a higher-level feature. By choosing the number of components in a constellation, one can obtain a balance between the selectivity of a feature and the computational efficiency of searching for that feature in an image.

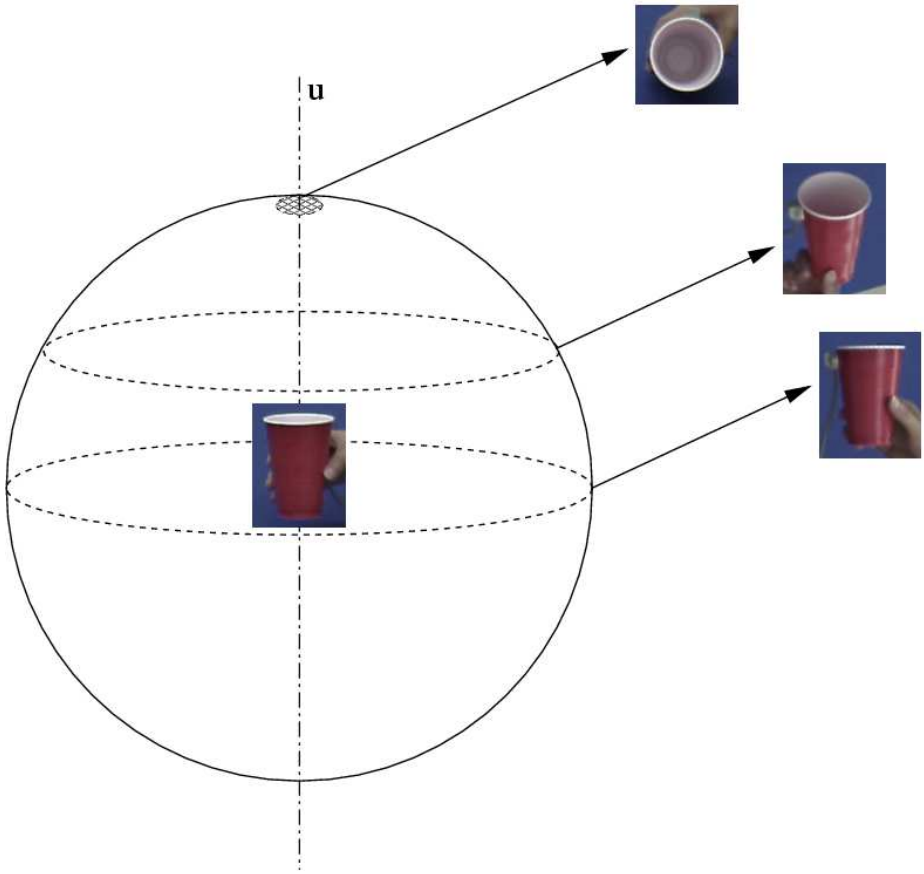


Figure 3.1: The aspect sphere of a cup with rotation symmetry about **u**.

Although we can still use 2D feature constellations for aspect recognition, the response of a single constellation will vary dramatically when the viewing angle changes. We need to define a set of constellations that cover all possible aspects of a given object. Assume that a set of feature constellations is observed on the 2D images taken by a camera when we rotate an object in front of it. The visibility of a particular feature constellation is a function of aspect, that is, there are some points on the aspect sphere where a constellation can be observed. Our interest is to find a way to describe clusters of these positive observations. One way to do this is to use a probability density function (pdf) over the aspect sphere:

$$p(a|Obj, C_i), \tag{3.1}$$

which denotes the probability p of being at an aspect a given that a particular object Obj is being observed and that constellation C_i is visible.

Normally, the aspects corresponding to a particular appearance of an object are only located within a small region of the aspect sphere. However, when object symmetries exist, those aspects can also be located along some annular region (rotational

symmetry) or regions apart from each other (reflection symmetry). We define the symmetry of an object as the property that different aspects on the aspect sphere correspond to exactly the same appearance (for example, the aspects around the equator of a cup). Therefore, Equation 3.1 should capture both Gaussian-like distributions, as well as distributions that acknowledge the rotational symmetries of an object. These desired properties give rise to the pdfs we discuss in the next section.

3.2 Small Circle Distribution

In this section, I will cover the basic ideas of the small circle distribution and the parameter estimation problem.

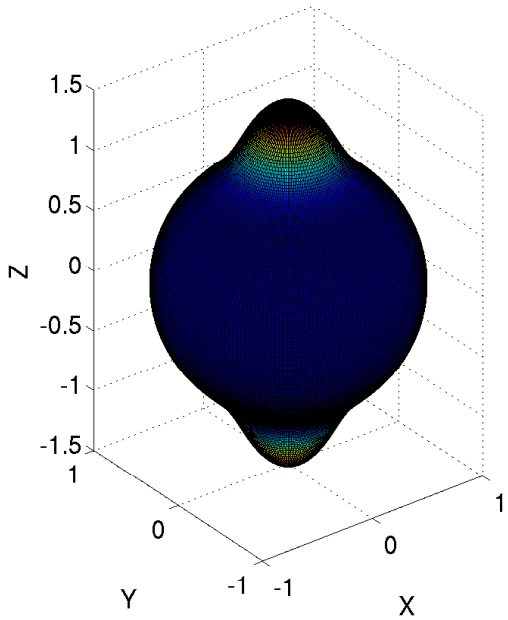
3.2.1 Gaussian-like Distribution on a Sphere

Bingham and Mardia (1978) propose the *small circle distribution*, which generalizes the well known Gaussian distribution to a unit sphere. By adjusting the parameters of this distribution, we can describe different "shapes" of clusters on our aspect sphere. The mean of a small circle distribution is a vector, which represents the mean aspect of the cluster. As with the standard deviation of a Gaussian distribution, there is a parameter that determines the *concentration* of a small circle distribution. Finally, another parameter determines the shape of the cluster, which could be annular or Gaussian-like. The pdf of the small circle distribution is:

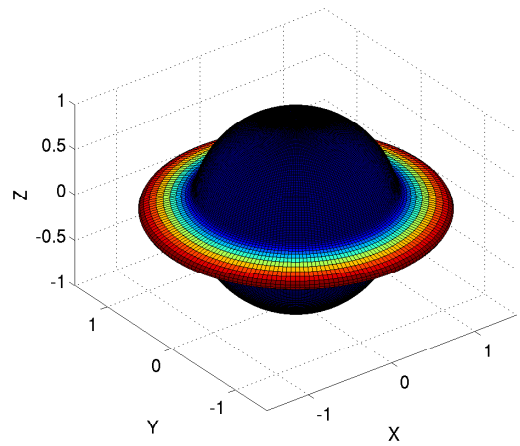
$$b(\mathbf{x}|\tau, \nu, \boldsymbol{\mu}) = \frac{1}{F(\tau, \nu)} e^{-\tau(\boldsymbol{\mu}^T \mathbf{x} - \nu)^2}, \tag{3.2}$$

where $\mathbf{x}$, $\boldsymbol{\mu}$ are unit vectors and $\boldsymbol{\mu}$ denotes the mean direction; τ is a scalar that gives the concentration of the distribution (the higher the τ, the more concentrated the distribution); ν is a scalar that determines the shape of the distribution; and $F(\tau, \nu)$ is a normalizing constant defined in Equation 3.3. Note that Equation 3.2 obtains the extreme value when $\boldsymbol{\mu}^T \mathbf{x} = \nu$, which means that the corresponding $\mathbf{x}$ can be any direction on the sphere with a fixed angle from $\boldsymbol{\mu}$. Since $\boldsymbol{\mu}$ is the mean direction, $\mathbf{x}$ will have a rotational symmetry about $\boldsymbol{\mu}$.

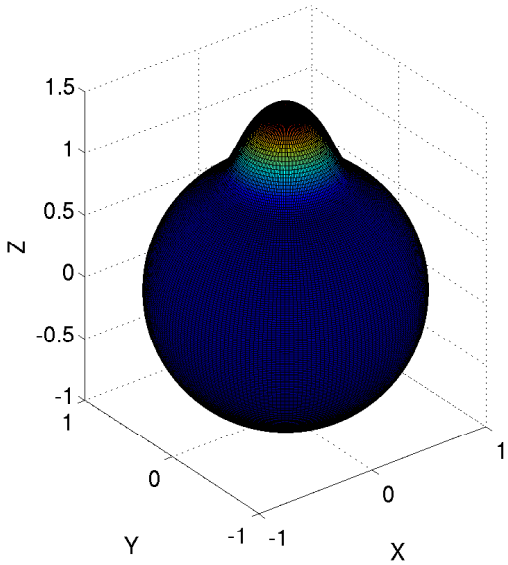
When $\boldsymbol{\mu}$, τ and ν fall in different ranges, we have different classes of distributions on the sphere. Specifically, when $\nu = 0$, b is the so-called Dimroth-Watson distribution (Mardia and Jupp 1999); particularly, when $\tau < 0$, it is a bipolar Gaussian distribution (Figure 3.2a); when $\tau > 0$, the distribution forms a girdle along the *great circle* perpendicular to $\boldsymbol{\mu}$ (Figure 3.2b). When $\nu \geq 1$ and $\tau > 0$, b has a



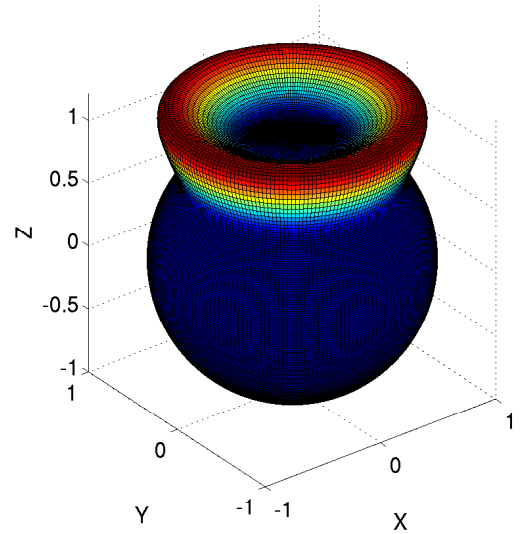
(a) bipolar Gaussian, τ=-10, ν=0 ($\gamma = 90°$)



(b) girdle, τ=100, ν=0 ($\gamma = 90°$)



(c) Fisher-von Mises, τ=50, ν=1.2



(d) small circle, τ=100, ν=0.8 ($\gamma \approx 37°$)

Figure 3.2: Gaussian-like distribution on a unit sphere with $\boldsymbol{\mu} = [0, 0, 1]$. In all cases, the surface radius is $1 + p/(2 \times \max(p))$, where p is the probability density at the corresponding aspect.

maximum in the direction of $\boldsymbol{\mu}$ and a minimum in the direction of $-\boldsymbol{\mu}$ (and no other extrema). This kind of distribution is the one-pole version of the Dimroth-Watson distribution and is also called the Fisher-von Mises distribution (Figure 3.2c). When $0 < \nu < 1$ and $\tau > 0$, b has a maximum on directions whose angular distance to $\boldsymbol{\mu}$ is $\gamma \equiv \arccos \nu$ and minima in the directions $\boldsymbol{\mu}$ and $-\boldsymbol{\mu}$. This case gives us a small circle (Figure 3.2d) about direction $\boldsymbol{\mu}$ and an angular distance γ from $\boldsymbol{\mu}$.

In Equation 3.2, $F(\tau, \nu)$ is a normalizing constant in order that the integral of b on the unit sphere equals to 1. $F(\tau, \nu)$ does not depend on $\boldsymbol{\mu}$ and by choosing $\boldsymbol{\mu}$ as $[0; 0; 1]$, it can be reduced to:

$$F(\tau, \nu) = \frac{1}{2} \int_{-1}^{1} e^{-\tau(z-\nu)^2} dz. \tag{3.3}$$

where, z denotes the z coordinate of a point on the unit sphere.

3.2.2 Maximum Likelihood Estimation

The maximum likelihood estimation of the parameters of the small circle distribution is given by Bingham and Mardia (1978). However, there is no closed form solution to this problem. One way to estimate these parameters is by using a sequential quadratic programming method (Fletcher and Powell 1963).

Let $\mathbf{x_1}, ..., \mathbf{x_n}$ represent a random sample from $b(\mathbf{x}|\tau, \nu, \boldsymbol{\mu})$ with $n \geq 4$. Define:

$$S = n^{-1} \sum_{j=1}^{n} \mathbf{x_j} \mathbf{x_j^T}, \tag{3.4}$$

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x_j} \text{ and} \tag{3.5}$$

$$\tilde{S} = n^{-1} \sum_{j=1}^{n} (\mathbf{x_j} - \bar{\mathbf{x}})(\mathbf{x_j} - \bar{\mathbf{x}})^{\mathbf{T}} = S - \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathbf{T}}. \tag{3.6}$$

Let $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ be the eigenvectors of $\tilde{S}$ with eigenvalues $t_1 > t_2 > t_3$. The suggested initial estimates are (Bingham and Mardia 1978):

$$\hat{\tau}_0 = \frac{1}{2t_3}, \tag{3.7}$$

$$\hat{\nu}_0 = \mathbf{u}_3^T \bar{\mathbf{x}}, \tag{3.8}$$

$$\hat{\boldsymbol{\mu}}_0 = \mathbf{u}_3. \tag{3.9}$$

3.3 Mixture Model of Distributions

There are cases in which a single distribution is not enough when describing the cluster of aspects corresponding to a single constellation. For example, the ellipse constellation observed on the end of a cylinder can be observed along a small circle on the aspect sphere due to rotational symmetry. This feature can be also be observed along another small circle on the aspect sphere because of the reflection symmetry between the top and the bottom of the cylinder. In this case, we use a mixture model of small circle distributions. Here, the composite density function $g(x)$ is defined as (Everitt and Hand 1981):

$$g(x|...) = \sum_{j=1}^{M} w_j b_j(x|...), \tag{3.10}$$

where

$$\sum_{j=1}^{M} w_j = 1, \tag{3.11}$$

and where M is the number of component distributions, and $b_j(x|...)$ is the density function of a small circle distribution with its own parameter set. Each element of the mixture represents a single cluster of aspects, and is weighted by w_j. Those weights are estimated by using the Expectation Maximization (EM) algorithm with multiple restarts (Dempster et al. 1977), and the parameters of individual distributions are found by maximum likelihood estimation as described above.

Chapter 4

Haptic-based Grasp Controller

Robotic grasping is traditionally approached as a pure planning problem that assumes *a priori* knowledge of the target object's geometry. However, we would like our robot to be able to robustly grasp objects with which it has no prior experience. Our approach is to use haptic information to drive a search process for appropriate finger contact locations. Given a cost function that is based on the total force and moment applied to the object by the set of contacts, and simple assumptions about the local surface geometry, this search process can be formulated as one of gradient descent on the cost function. Prior work in this area (Coelho and Grupen, 1997) has assumed that the surface of the object local to the contact is either flat or convex. However, when the surface is concave, the search process, in fact, ascends the cost function. Here, we propose a *switching controller* approach that estimates the local curvature of the object over multiple contacts. This information is then used to switch between one of two methods of estimating the gradient of the cost function. While this new approach shows comparable performance to the original when faced with objects containing only flat or convex surfaces, the new algorithm performs substantially better when objects contain concave surfaces.

4.1 Methods

The goal is to find a set of contact locations on an object such that a set of non-zero normal forces applied at the contacts result in a net force and moment applied to the object of $\vec{0}$. In the presence of soft contacts and friction, a set of contacts that satisfy these criteria ensures wrench closure, allowing the object to be squeezed at the contacts without accelerating it (Ponce et al. 1997; Platt et al. 2004).

Our approach to finding this set of contact locations involves a gradient descent search that requires the fingers to haptically explore the object. At each step of the search, contact is made with the object by each of the finger tips. Given an estimate of each of the contact locations and surface normals, the net force and moment applied to the object is estimated (assuming unit force applied at each contact). Given assumptions about the local geometry of each surface, finger (contact) displacements are estimated that are intended to reduce the net force and moment applied to the object. In the subsequent sections, we describe the process of computing these contact displacements.

4.1.1 Force Controller

The task of the *force controller* is to reduce the total force applied to the object by the set of contacts. Following Coelho and Grupen (1997), the error function for this controller is:

$$\epsilon_f = \frac{1}{2}\mathbf{f}_n{}^T\mathbf{f}_n, \tag{4.1}$$

where $\mathbf{f}_n$ is a column vector describing the net force applied by the set of contacts. This is computed as follows:

$$\mathbf{f}_n = \mathbf{f} + \mathbf{f}_e, \tag{4.2}$$

where $\mathbf{f}$ is the unit force applied by the contact of interest, and is normal to the object surface at the contact point, and $\mathbf{f}_e$ is the sum of the remaining forces, including the other contacts and any modeled (but uncontrolled) external forces. Note that one can assume a force vector of arbitrary length to express the differential roles played by various contacts in a grasp. However, we do not exploit this flexibility for the purposes of this work.

Let $\mathbf{x}$ denote the Cartesian location of the contact in $\mathbb{R}^3$. Following each probe of the object, the contact is displaced along a direction that reduces ϵ_f. This direction is computed by first estimating the gradient of the error function:

$$\frac{\partial \epsilon_f}{\partial \mathbf{x}} = \frac{\partial \epsilon_f}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}, \tag{4.3}$$

where:

$$\frac{\partial \epsilon_f}{\partial \mathbf{f}} = \mathbf{f}_n{}^T. \tag{4.4}$$

The gradient of the force with respect to contact location depends on our assumptions about the local surface geometry. For the *convex controller*, we follow

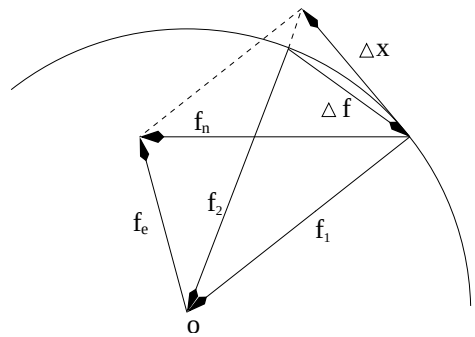


Figure 4.1: Force control for convex object. $\mathbf{f}_1$ is the current force exerted by the finger and $\mathbf{f}_2$ is the future force exerted by the finger. $\mathbf{f}_e$ is the external force on the object. $\mathbf{f}_n$ is the net force of $\mathbf{f}_1$ and $\mathbf{f}_e$. $\Delta\mathbf{x}$ is the direction of finger displacement.

the spherical assumption (Coelho and Grupen 1997). This scenario is illustrated in Figure 4.1. The current contact produces force $\mathbf{f}_1$. Given that the other contacts produce force $\mathbf{f}_e$, the total force applied to the object is shown as $\mathbf{f}_n$. The finger tip is then displaced along $\Delta\mathbf{x}$ (the tangent to the sphere at the original contact), and then translated toward the origin of the sphere. For a small $\Delta\mathbf{x}$, this latter translation is very small. Therefore:

$$\mathbf{f}_2 \approx \mathbf{f}_1 - \beta\Delta\mathbf{x}, \tag{4.5}$$

where $\beta > 0$ is a stiffness coefficient with units of N/m. The second partial derivative of Equation 4.3 can be approximated as follows:

$$\frac{\partial\mathbf{f}}{\partial\mathbf{x}} \approx \frac{\mathbf{f}_2 - \mathbf{f}_1}{\Delta\mathbf{x}} \approx -\beta\mathbf{I}, \tag{4.6}$$

where $\mathbf{I}$ is a 3×3 identity matrix. Combining Equations 4.3, 4.4, and 4.6, we have:

$$\frac{\partial\epsilon_f}{\partial\mathbf{x}} \approx -\beta{\mathbf{f}_n}^T. \tag{4.7}$$

Finger displacement will follow the negative gradient of the error function. Therefore, the displacement will be in the same direction as $\mathbf{f}_n$, subject to maintaining contact with the object's surface.

The alternative to assuming a locally convex surface is to assume a concave one. For the *concave controller*, we assume that the contact is located on the inside of a unit sphere. This scenario is illustrated in Figure 4.2. As with the previous case, we assume a small displacement in the direction of $\Delta\mathbf{x}$ (the surface tangent), and then a small translation back to the surface, toward the origin of the sphere. In this case, the following holds:

$$-\mathbf{f}_2 \approx -\mathbf{f}_1 - \beta\Delta\mathbf{x}. \tag{4.8}$$

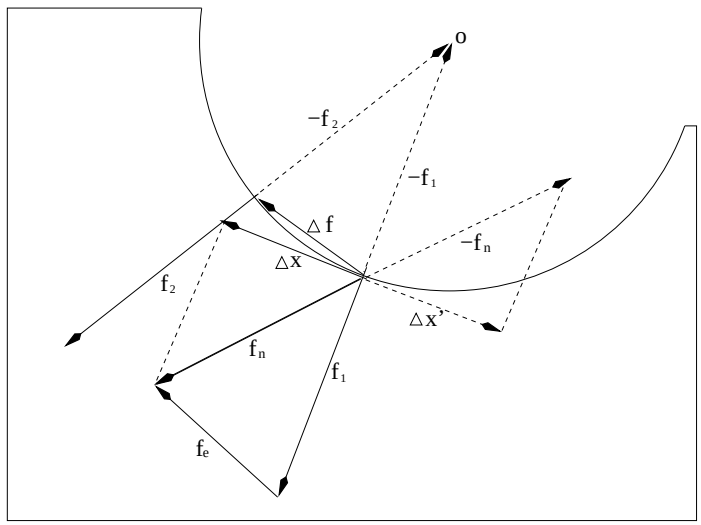


Figure 4.2: Force control for concave object. $\mathbf{f}_1$ is the current force exerted by the finger and $\mathbf{f}_2$ is the future force exerted by the finger. $\mathbf{f}_e$ is the external force on the object. $\mathbf{f}_n$ is the net force of $\mathbf{f}_1$ and $\mathbf{f}_e$. $\Delta\mathbf{x}$ is the direction of finger displacement and $\Delta\mathbf{x}' = -\Delta\mathbf{x}$.

Therefore:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \approx \frac{\mathbf{f}_2 - \mathbf{f}_1}{\Delta \mathbf{x}} \approx \beta \mathbf{I}. \tag{4.9}$$

Combining Equations 4.3, 4.4 and 4.9, we have:

$$\frac{\partial \epsilon_f}{\partial \mathbf{x}} \approx \beta {\mathbf{f}_n}^T. \tag{4.10}$$

The result is that the contact should be displaced along the direction opposite to $\mathbf{f}_n$. Note that this displacement should be done subject to maintaining some ideal force against the surface of the object (such that the object itself is not displaced). Note also that the direction of displacement given the concave assumption is in the opposite direction as when given the convex assumption.

4.1.2 Moment Controller

The role of the *moment controller* is to reduce the total moment applied to the object. The moment control error function is defined as:

$$\epsilon_m = \frac{1}{2} {\mathbf{m}_n}^T \mathbf{m}_n, \tag{4.11}$$

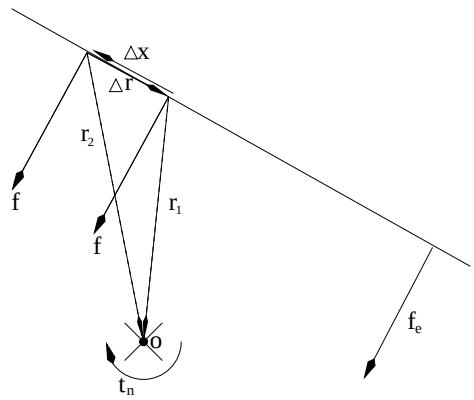


Figure 4.3: Moment control. $\mathbf{f}$ is the force exerted by the finger. $\mathbf{f}_e$ is the external force on the object. "o" is the origin about which torque is measured. $\mathbf{r}_1$ and $\mathbf{r}_2$ denote the vector from current and future contact point to the origin, respectively. In current configuration, the net torque t_n points into the page.

where $\mathbf{m}_n$ is the net moment applied to the object. Here, net moment is composed of the moment due to the contact of interest ($\mathbf{m}$), and the moment due to any other modeled forces external to the contact of interest ($\mathbf{m}_e$):

$$\mathbf{m}_n = \mathbf{m} + \mathbf{m}_e. \tag{4.12}$$

The gradient of the error function with respect to contact displacement is as follows:

$$\frac{\partial \epsilon_m}{\partial \mathbf{x}} = \frac{\partial \epsilon_m}{\partial \mathbf{m}} \frac{\partial \mathbf{m}}{\partial \mathbf{x}}, \tag{4.13}$$

where:

$$\frac{\partial \epsilon_m}{\partial \mathbf{m}} = {\mathbf{m}_n}^T. \tag{4.14}$$

One can estimate the derivative of the contact-imposed moment with respect to contact displacement by following the planar surface assumption of Coelho and Grupen (1997). This situation is illustrated in Figure 4.3. The current contact imposes force $\mathbf{f}$ at a position from which the moment reference point (o) is displaced by $\mathbf{r}_1$. Given a small contact displacement along the surface tangent of $\Delta\mathbf{x}$:

$$\begin{aligned} \frac{\partial \mathbf{m}}{\partial \mathbf{x}} &\approx \frac{\mathbf{m}_2 - \mathbf{m}_1}{\Delta \mathbf{x}}, \\ &= \frac{\mathbf{f} \times \mathbf{r}_2 - \mathbf{f} \times \mathbf{r}_1}{\Delta \mathbf{x}}, \\ &= \frac{\mathbf{f} \times (\mathbf{r}_2 - \mathbf{r}_1)}{\Delta \mathbf{x}}, \end{aligned}$$

$$
\begin{aligned}
&= -\frac{\mathbf{f} \times \Delta\mathbf{x}}{\Delta\mathbf{x}}, \\
&= -\begin{bmatrix} 0 & -f_z & f_y \\ f_z & 0 & -f_x \\ -f_y & f_x & 0 \end{bmatrix}. \qquad (4.15)
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
\frac{\partial \epsilon_m}{\partial \mathbf{x}} &= -\mathbf{m}_n{}^T \begin{bmatrix} 0 & -f_z & f_y \\ f_z & 0 & -f_x \\ -f_y & f_x & 0 \end{bmatrix} \\
&= -\mathbf{f} \times \mathbf{m}_n. \qquad (4.16)
\end{aligned}
$$

The result is that the moment controller recommends contact displacements in the direction of $\mathbf{f} \times \mathbf{m}_n$.

4.1.3 Concavity Detection

Key to making use of the appropriate force control law (the convex or the concave form) is estimating the local surface curvature. One possible approach is to recover local curvature using a tactile or image array (e.g., Jiar et al. 1994; Lee and Nicholls 1999). In our case, we assume that a 6-axis load cell is embedded within the finger tip, from which one can infer the surface normal but not the local curvature (Bicchi et al. 1993). Hence, it becomes necessary to integrate information over multiple probes of the object. At any given time, for the purposes of the force controller, a surface is assumed to be either concave or convex (with surfaces initially assumed to be convex).

By comparing Figure 4.1 and Figure 4.2, the angle between $\Delta\mathbf{f}$ and $\Delta\mathbf{x}$ depends on the local curvature, where $\Delta\mathbf{f}$ is the change the force exerted by the contact and $\Delta\mathbf{x}$ is the displacement of the contact. If the local surface is convex (Figure 4.1), the angle between $\Delta\mathbf{f}$ and $\Delta\mathbf{x}$ will be between 90° and 180°; If the local surface is concave (Figure 4.2), the angle between $\Delta\mathbf{f}$ and $\Delta\mathbf{x}$ will be between 0° and 90°. Note that $\Delta\mathbf{f}$ is zero when the local surface is flat. The angle between $\Delta\mathbf{f}$ and $\Delta\mathbf{x}$ can be represented by inner product. Given two subsequent probes of an object, the assumed curvature is determined as follows:

$$
\begin{aligned}
\Delta\mathbf{f}^T\Delta\mathbf{x} &< -\alpha_1 && \text{convex} \\
\Delta\mathbf{f}^T\Delta\mathbf{x} &> \alpha_2 && \text{concave} \\
\textit{otherwise} & && \text{no change}
\end{aligned}
$$

where $\alpha_1 > 0$ and $\alpha_2 > 0$ are switching parameters. Ideally, $\alpha_1 = \alpha_2 = 0$, which corresponds to the case we discussed above. However, in practice, these parameters are empirically selected so that the switching controller is sensitive enough to discover the concavities of the test objects, but is not distracted substantially by uncertainties in the force estimation process. Intuitively, the smaller α_1 is, the more easily a surface is considered to be convex; the smaller α_2 is, the more easily a surface is considered to be concave. In the experiment, both α_1 and α_2 are set to 0.5 Nm. This is equivalent to:

the angle between $\Delta\mathbf{f}$ and $\Delta\mathbf{x} > 120°$	assume convex
the angle between $\Delta\mathbf{f}$ and $\Delta\mathbf{x} < 60°$	assume concave
otherwise	no change

4.1.4 Combining Force and Moment Control Actions

Following Platt et al., the displacement recommendations of both the force controller and the moment controller are combined into a single displacement (Platt et al. 2002; Platt 2006). This combination gives the moment controller the opportunity to influence the motion of the contact as long as it does not interfere with the actions of the force controller (Huber and Grupen 2000; Huber 2000). Specifically, the combined control action is computed as follows:

$$\boldsymbol{\phi} \equiv \boldsymbol{\phi}_f + \mathbf{N}(\boldsymbol{\phi}_f)\boldsymbol{\phi}_m, \tag{4.17}$$

where ϕ is the composite displacement, ϕ_f and ϕ_m are the recommended displacements by the force and moment controllers, respectively, and $N(.)$ is a null space projection matrix.

Recall the result given by Equation 4.16. In the general case, the choice of the moment reference point can affect the magnitude and sign of this recommended displacement. However, since the moment controller is subject to the force controller, the force controller will at all cost reduce the net force to zero. When the net force due to the set of contacts is zero, they form a *couple*. Under this condition, the net moment (and hence, the recommended displacement) eventually becomes the same for any choice of reference point (Mason 2001). Hence, the choice of reference becomes arbitrary.

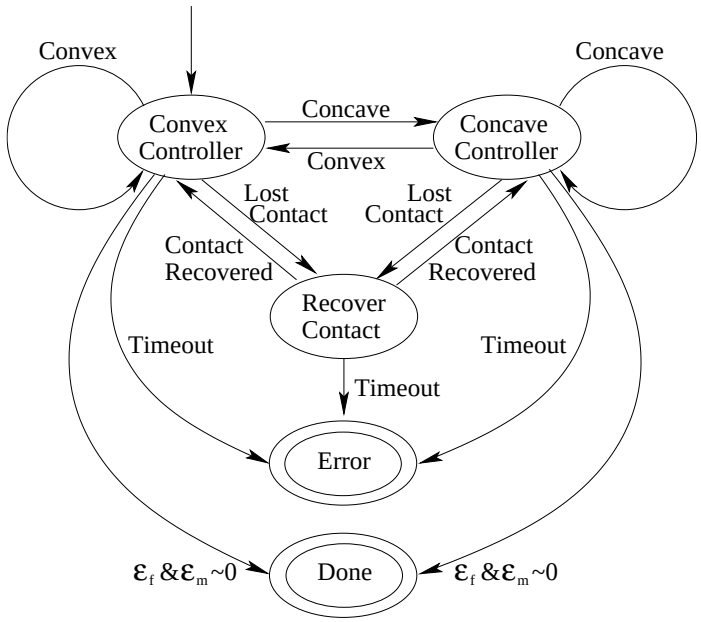


Figure 4.4: Finite state machine for the controller with concavity detection

4.1.5 Switching Controller

The sequential behavior required for the haptic search process is implemented using a finite state machine (FSM), as shown in Figure 4.4. Each finger is controlled using one such FSM. The object is initially assumed to be located such that flexing of the fingers toward the palm will bring each finger into contact with the object. At each control step, the surface normal at the contact is used to estimate the local curvature of the surface and to recommend a displacement of the contact. The latter is computed using the combined force and moment controller recommendations (as described above), where the force controller is either of the type *convex* or *concave*. Displacement of the contact is implemented by drawing the finger away from the surface along the normal, translating the finger along the recommended displacement, and then placing the finger back onto the surface along the original surface normal. Under most conditions, the finger will again contact the surface, and the process will repeat from either the concave or convex states.

The search process terminates under two conditions. First, if the net force and moment fall below a critical threshold, the search is considered to have terminated successfully (the *Done* state). Second, if the search is still ongoing after a specified period of time, the controller is considered to have terminated in an *Error* state (we say that the controller has "timed out"). One scenario in which this can happen is

when the controller has achieved an equilibrium state or is in a cycle in which the force or moment error does not satisfy the critical threshold.

There are also a number of cases in which the finger does not successfully return to the surface of the object after displacement is executed. This situation can happen if the local surface is not smooth with respect to the size of the contact displacement. For example, this can happen when the contact is displaced off the corner of an object. This situation can also happen when there is an error in the estimation of the surface normal. Here, the recommended displacement can take the finger far away from the surface. In either case, the *recover contact* state is responsible for taking several heuristic approaches to bringing the finger back into contact with the object. Both problems are sensitive to the magnitude of the contact displacement for a single control step. Therefore, contact displacement magnitude is considered a parameter that must be selected appropriately (this is addressed in the *Results* section).

4.1.6 Simulation Implementation

All experiments were conducted within a custom simulation environment that is based on the Visualization Toolkit (VTK) package (Schroeder et al. 2006). This toolkit allows for the modeling of the surfaces of the robotic fingers and of the objects of interest. The surfaces are modeled as meshes; collision between two surfaces is detected as a collision between two mesh triangles. The force direction (and hence surface normal) is then determined by the orientation of the colliding triangles. Due to this approach, estimation of applied force exhibits a certain degree of variation, depending on small changes in orientation of the colliding triangles.

The simulated robot used for the experiments described in this paper consists of two fingers (Figure 4.5). Each finger has four degrees of freedom: one adduction/abduction and three flexion. The fingers are mounted relative to one another such that they are able to oppose one-another. For these experiments, the hand is fixed in space. Hence, any objects are placed near the palm of the hand such that both fingers can initially touch the object through a simple flexion motion.

4.2 Experimental Results

The proposed controller is explicitly designed to properly explore objects with both convex and concave surfaces. Our experimental hypothesis is that when presented with an object containing a concave surface, the proposed controller should perform

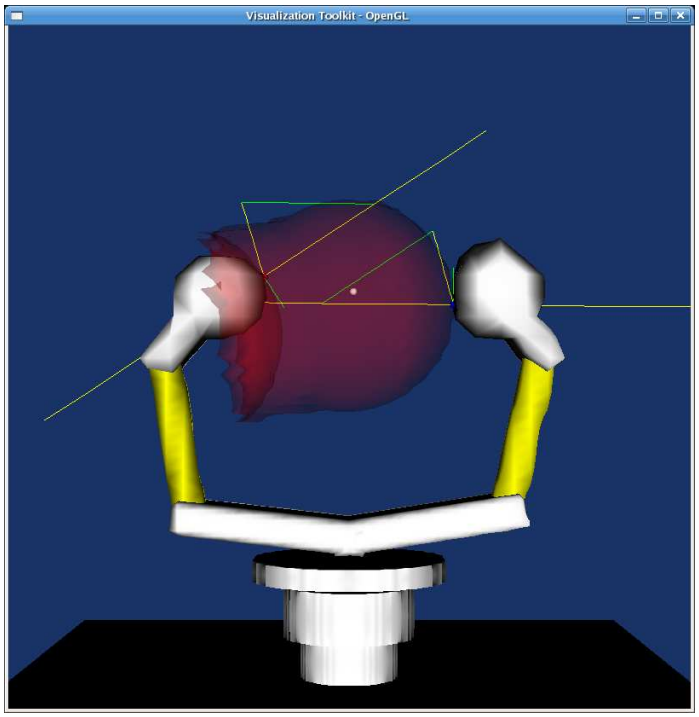


Figure 4.5: An example grasp attempt. Here, one finger is on the concave end and the other finger is on the convex end of a capped cylinder.

better than a controller that only "expects" convex surfaces. However, when an object is composed of non-concave surfaces (convex or flat), the two controllers should perform identically.

In addition to the choice of control algorithm, two other factors can statistically significantly affect the performance of a controller: the contact displacement magnitude and orientation of the object relative to the fingers. The latter factor affects which of the object's surfaces are initially exposed to the fingers. We have therefore implemented (for most of the following experiments) a 3-factor analysis of algorithm, displacement magnitude, and object orientation. For each combination of factors, a sample size of $N = 20$ is used. Although object orientation is a controlled factor, a small, random orientation is added to the object prior to initiation of the exploration process (uniform distribution: ± 5 degrees). Two measures of performance were used: the total number of successful grasps (achieving the *done* state), and the running time of the controller. A controller "times out" when a probe occurs $400sec$ into the search. By observation, a controller usually does not make much progress as far as net force and net torque are concerned after running $400sec$.

The implementation of the convex-only controller is the same as the switching controller (as described in Figure 4.4). The only difference is that the *concave controller* state has been removed (and consequently, concavity detection is not performed).

4.2.1 Convex Objects

Two convex objects are used in this study: a sphere and a cube. Because the sphere is symmetric about all rotations, we use a two-factor analysis (controllers (2) and contact displacement magnitude (16)). Figure 4.6 shows the percent of successful grasps for each algorithm and contact displacement magnitude ("step size"). Both controllers perform well when step size is less than 10, but performance degrades with larger step sizes. This degradation is due to the fact that the larger step sizes tends to lead to instabilities around the equilibrium point. In addition, the larger step sizes can result in the finger "dropping off" of the object all together. Also, when step size is larger than 10, the switching controller oscillates a little more than the convex-only controller. The reason is that the estimation of the local curvature adds additional noise to the switching controller and occasionally the noise increases or decreases the percentage of success.

Across all step sizes, the mean success for both controllers is 80%. The total number of successes does not depend on the controller type (p<1 for both χ^2 and Fisher's Exact Test) but depends on the step size (for controller with concavity detection (w/ CD), $\chi^2 = 85.78$, $p < 10^{-10}$; for controller without concavity detection (w/o CD), $\chi^2 = 94.47$, $p < 10^{-12}$).

Furthermore, a two-way ANOVA analysis on the total running time of the controller indicates a significant effect of step size ($F = 7.95$; $p < 10^{-4}$), but no significant effect on choice of controller ($F = .17$; $p < .69$). There is, however, a significant interaction effect ($F = 6.79$; $p < 10^{-4}$). This effect highlights the fact that the switching controller performs slightly worse for small step sizes. This effect is due to the controller's inability to consistently estimate the curvature of the object when displacements are small. Specifically, the "noise" in the contact normal estimation can sometimes dominate the curvature estimate in these cases.

The second convex object is a cube (Figure 4.7). Across all step sizes and object orientations, the convex-only controller is successful 76.6% of the time, while the switching controller is successful for only 73.6% of the trials. This difference is not significant according to a Fisher's Exact Test ($p < .10$). A three-way ANOVA test on running time shows a significant effect on the average running time of step size

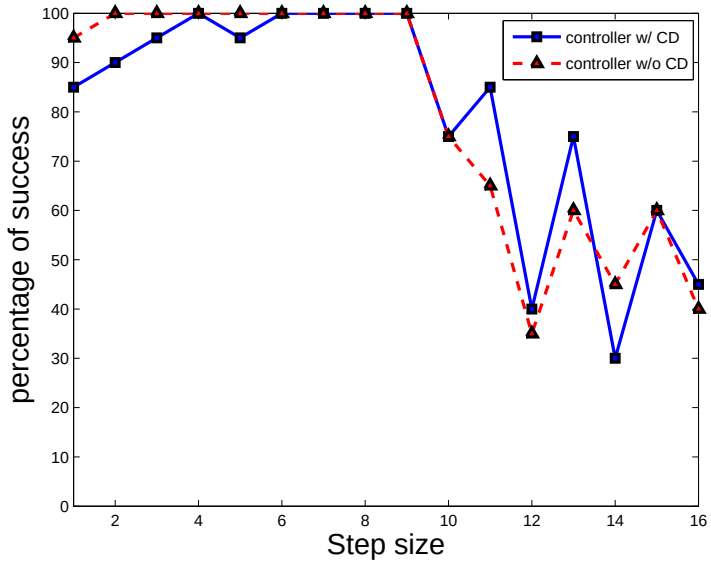


Figure 4.6: Successful grasps by the switching controller and the convex-only controller when grasping the sphere.

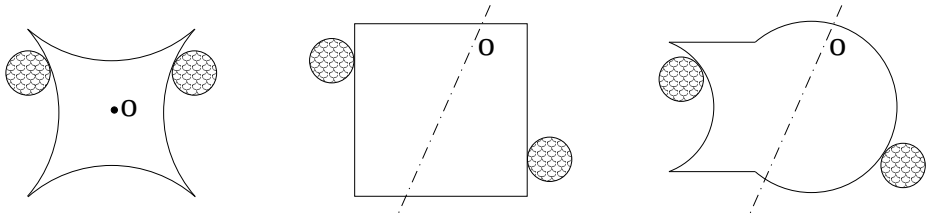


Figure 4.7: Cross-section of three of the four objects: "concave cube," cube, and "capped cylinder." (O is the rotation axis used in the experiments)

($F = 29.4$; $p < 10^{-4}$) and on choice of controller ($F = 6.78$; $p < .01$). Mean running time for the convex-only controller is $204.75sec$, while the switching controller requires a mean of $219.32sec$. The small additional cost in running time of the proposed controller (7.12%) is due to occasional misclassification of the surface curvature, which is not substantial in practice. This misclassification effect results in at least one control step moving in the incorrect direction before the surface is correctly classified, since the switching controller needs two successive probes to estimate the local curvature.

4.2.2 Concave Objects

Two objects containing concavities are used: a "concave cube," and a "capped cylinder" (see Figure 4.7). The concave cube is constructed by subtracting spherical caps

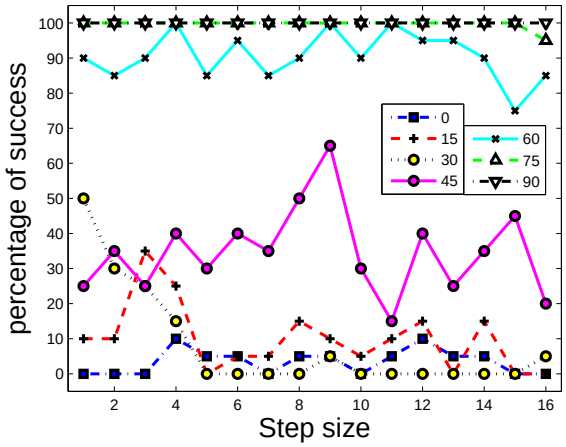
from each side of a cube. The result is that a majority of its surface is concave, with only areas around the corners maintaining a flat surface. Most of the surface of the capped cylinder is convex, but one small region is concave. Depending on the orientation of the cylinder relative to the fingers, the initial contacts may "see" either both convex surfaces or one concave and one convex surface.

For the capped cylinder, Figure 4.8 shows the performance of each controller given the step size and the object orientation. Orientations of 75° and 90° are such that both fingers are initially presented with a convex surface. For orientations of 0° and 15°, one finger falls well within the concave region. An orientation of 45° results in one finger landing on the edge between the concave and convex surface (with 30° and 60° falling to either side of this sharp edge).

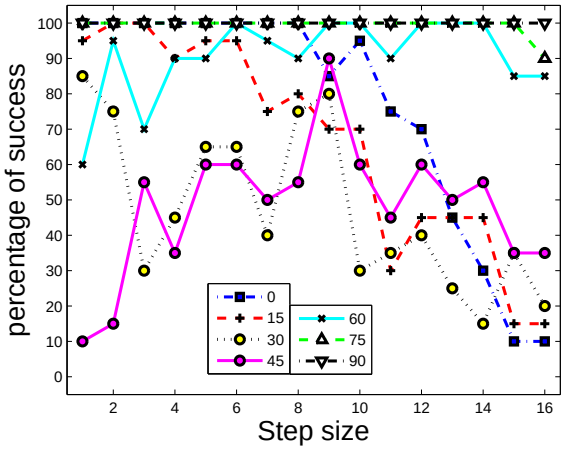
Both controllers exhibits high levels of success for orientations of 60°, 75°, and 90° (Figure 4.8a, b). However, the success of the convex-only controller suffers dramatically when presented with the concave surface (orientations of 0°, 15°, and 30°). In contrast, the switching controller (Figure 4.8b) performs well for these orientations, most dramatically for 0° and 15°. However, this performance drops off as the step size increases beyond 8.

Across all step sizes and object orientations, the switching controller is successful 75.5% of the time, while the convex-only controller is successful only 49.5%. This difference is significant according to a Fisher's Exact Test ($p < 10^{-4}$). A three-way ANOVA test on the controller running time shows a significant effect on the average running time of step size ($F = 15.3$; $p < 10^{-4}$) and of choice of controller ($F = 538$; $p < 10^{-4}$). Mean running time for the convex-only controller is $258.7sec$, while the switching controller requires a mean of $201.6sec$.

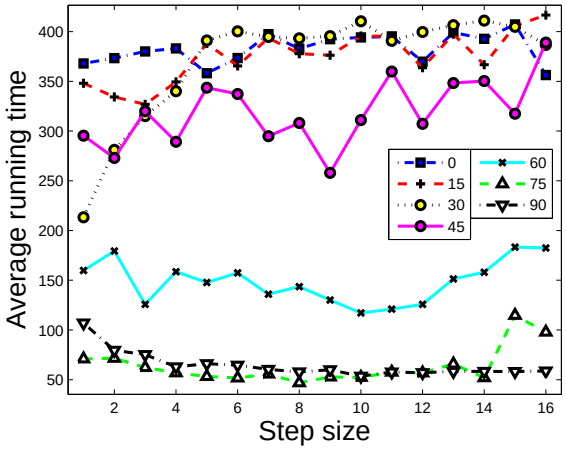
By comparing Figure 4.8c and Figure 4.8d, we can see that the controller w/o CD exhibits many timeouts ($400sec$). Also, we can see that when the orientation of the object is from 60° to 90°, for both controllers the average running time does not change very much with respect to all step sizes. When the orientation is between 0° and 30°, for both controllers the average running time has an increasing trend. This means when the local surface is concave, both controllers need more "searching" time. When step size increases, the fingers tend to lose contact with the object and take more time to recover compared to convex surfaces. When object orientation is between 60° and 90°, the average running time does not depend on the controllers ($F = 1.08, p < .302$, Two-way ANOVA) or the step size ($F = .24, p < .998$) and there is no interaction effect between step sizes and controllers ($F = .04, p < 1$).



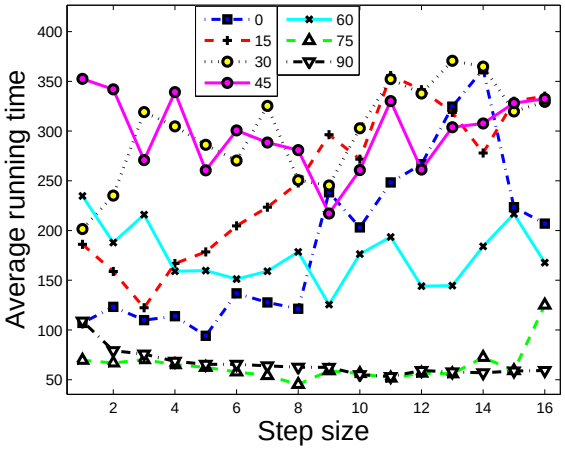
(a) Percentage of success for the convex-only controller



(b) Percentage of success for the switching controller



(c) Average running time for the convex-only controller



(d) Average running time for the switching controller

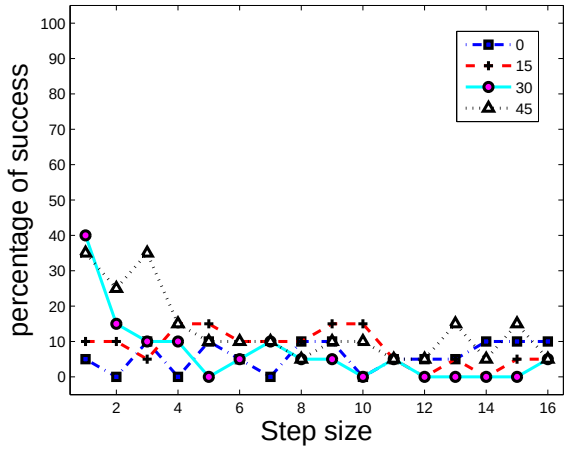
Figure 4.8: Performance of the convex-only and the switching controllers on the capped cylinder.

In the concave cube case, Figure 4.9 shows the performance of each controller given the step size and the object orientation. Only four different orientations are considered due to the symmetry property of the concave cube. When orientation is 0°, two fingers are initially presented with concave surfaces opposing with each other (Figure 4.7). When orientation is 45°, two fingers are initially presented with concave surfaces adjacent to each other. The convex-only controller achieves a very low percentage of success across all step sizes and object orientations (Figure 4.9a) and the corresponding running time is around 400*sec* (Figure 4.9c, mostly due to timeout). In contrast, the switching controller (Figure 4.9b and d) performs well for all orientations. However, this performance drops off as the step size increased beyond 9.

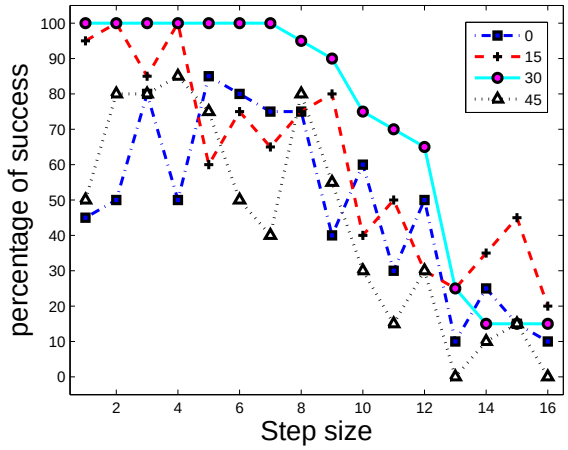
Across all step sizes and object orientations, the switching controller is successful 56.6% of the time, while the convex-only controller is successful only 8.7% of the time. This difference is significant according to a Fisher's Exact Test ($p < 10^{-4}$). The step size also has a significant influence on the success rate, according to a χ^2 test (for the switching controller, $\chi^2 = 342.4$, $p < 10^{-4}$; for the convex-only controller, $\chi^2 = 29.1$, $p < .016$).

4.2.3 Varying Object Size

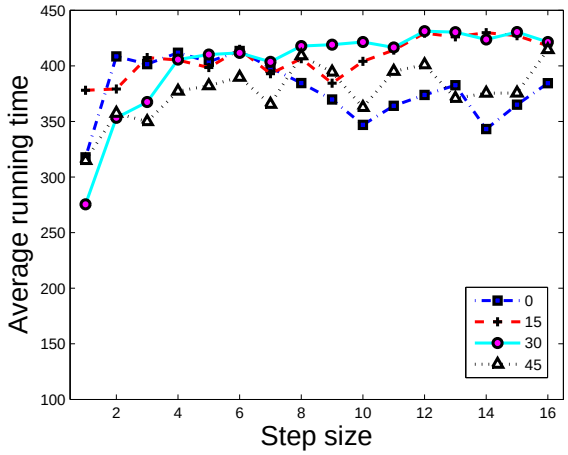
One critical question to ask is the degree to which object size influences the outcome of the search process. Since the concave cube is the most difficult object for both controllers, I use it as a representative of the four objects to address this issue. I present concave cubes that are 50% larger and 50% smaller than the original size reported above. For the large cube, across all step sizes and object orientations, the switching controller is successful 53.8% of the time, while the convex-only controller is successful only 6.4% of the time. This difference is significant according to a Fisher's Exact Test ($p < 10^{-4}$). For the switching controller, the percentage of success also depends on the step size ($\chi^2 = 155.1$, $p < 10^{-4}$); however, that is not true for the convex-only controller ($\chi^2 = 16.5$, $p < .348$). For the smaller cube, the switching controller is successful 52.7% of the time, while the convex-only controller is successful 31.6% (the difference is significant according to a Fisher's Exact Test; $p < 10^{-4}$). The latter controller demonstrates a marked improvement in performance over the other object sizes. This improvement is due in large part to a fortuitous alignment between the edge of the cube and the initial positions of the fingers. The alignment enables the convex-only search process to discover a grasp solution in which the flat



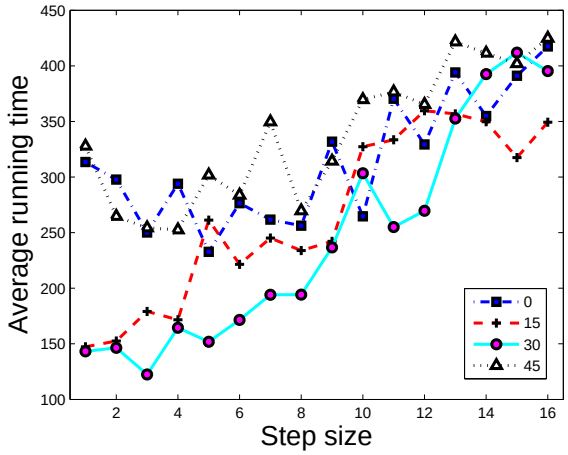
(a) Percentage of success for the convex-only controller



(b) Percentage of success for the switching controller



(c) Average running time for the convex-only controller



(d) Average running time for the switching controller

Figure 4.9: Performance of the convex-only and the switching controllers on the concave cube.

regions around the corners of the cube are used in the final solution, rather than the concavity. The step size also has a very significant influence on the success rate (for the switching controller, $\chi^2 = 481.1$, $p < 10^{-4}$; for the convex-only controller, $\chi^2 = 112.1$, $p < 10^{-4}$).

4.2.4 Randomized Object Poses

In the above experiments, each object is positioned in a fixed location and a fixed set of rotations are used (about a single axis). In order to examine the robustness of our controller over a wider range of initial conditions, we also perform a set of experiments in which the pose of the object is chosen randomly. We choose the capped cylinder and the concave cube with a range of controller step sizes (6, 7, 8 and 9). The object position is selected from a normal distribution (standard deviation: 15% of finger radius) and the orientation is selected uniformly from the possible 3D orientations.

For each of the two objects, both the convex-only and the switching controllers are employed. A total of 100 samples are taken for each of the four conditions. For the cylinder, the switching controller achieves 86% success, where the convex-only controller achieves 75%. For the concave cube, the switching controller achieves a success rate of 49.3%, while the convex-only controller achieves a rate of 9.5%. For this latter object, of the failures to grasp, more than 70% are cases in which an equilibrium is successfully reached, but either the net force or moment (or both) are above the acceptable threshold (e.g., grasping a cube using two fingers on the same corner). The difference between the two controllers is still significant ($p < 10^{-4}$ for both cases, Fisher's Exact Test).

4.3 Conclusions and Future Work

When grasping and manipulating objects in open environments where uncertainties exist in their geometry and pose, robust behavior requires the on-line integration of sensory data. In particular, one can make use of haptic information to guide the search for contact locations that will comprise a stable grasp. This search process can be formulated as one of gradient descent of a wrench-based error function. However, estimation of the gradient of this error function with respect to contact movement relies on knowledge of the curvature of the surface around the contact point.

In this chapter, I have proposed an approach in which the local curvature is estimated to be either convex or concave using a pair of spatially distributed probes

over the object. Given this estimate, the gradient following algorithm can appropriately assess the direction in which to displace the contact in order to further reduce error. For objects that include concave surfaces, our experimental results show a substantial performance increase (22.07% for the capped cylinder) by the proposed algorithm over an algorithm in which the surface is always assumed to be convex. For convex objects, although the proposed algorithm pays a small performance cost, it is not substantial (7.12% for the cube). These results hold for a range of parameter settings, including the magnitude of the search step size.

Because the search for an appropriate grasp configuration follows the gradient of the cost function, the approach suffers from the existence of local minima. This is particularly clear in case of the concave cube, in which the proposed controller fails on 43.4% of the trials, 70% of which are cases in which the two fingers are left attempting to grasp a single corner of the cube. In general, the existence of these local minima is an interaction of the geometry of the object and the number of contacts involved in the grasping process. One can address this issue simply by detecting that a local minima has been found, and then restarting the search process from a new configuration. In this restart step, one could also remove or add fingers, and hence contacts.

More generally, it is possible to introduce vision-based techniques to bias the selection of the initial configuration of the fingers. Specifically, we would want such a configuration to drop the contacts into a well of the error function whose minimum corresponds to an acceptable grasp. Along these lines, Coelho et al. (2000) have developed an approach in which visual representations are explicitly acquired that robustly predict the outcome of a grasp search process. Once the models have been learned, it becomes possible to use the visual representations in a novel situation to place the hand and fingers in a configuration that is often near to an acceptable grasp. In the next two chapters, I will extend this approach to allow grasping of three-dimensional objects by object classification and aspect recognition.

Chapter 5

Object Classification

Visual information such as object shape and color can play an important role for hand pre-shaping during human grasp. Compared to color, object shape alone can give a lot of the information that we need to shape the hand into an appropriate grasp. As the first step, I only consider object visual features that capture shape to classify the appropriate grasp type(s) for the object in a given image. First, a way to represent visual features is discussed and a method of detecting such features is given. Then, I propose an approach that can learn useful features by supervised learning. I compare two learning methods: linear function approximation and least squares support vector machines (LS-SVMs).

5.1 Problem Definition and Algorithm

In this chapter, I propose an object classification approach that can be used for estimating grasp types. The inputs of the approach are images of target objects taken by a camera (e.g., Figure 5.1). The outputs are the corresponding grasp types we should use to grasp the object in a given image (defined as in the work of de Granville, Southerland, and Fagg 2006). Essentially, this approach can be viewed as one of object classification. The classes can be either grasp types or object identities, although I focus on the former. In general, object recognition algorithms can be divided into two categories: 1) color based, which makes direct use of pixel brightness, and 2) feature based, which makes direct use of spatial arrangements of extracted features (such as edges). In our case, since the color of the objects of interest is arbitrary, we choose a feature-based recognition approach that uses edgel constellations to discriminate different geometric shapes (Piater and Grupen 2002).

(a) The block

(b) The cup

Figure 5.1: Image captured by robot vision.

I employ a supervised learning approach to establish a relationship between visual features and grasp types. Piater and Grupen (2002) used a single constellation to determine a particular grasp type. In this chapter, I make use of multiple constellations in an attempt to make a more robust decision. I first use a linear function to approximate the relationship between constellations and grasp types. However, the linear model assumes that features independently contribute evidence to the grasp classification. In order to consider combinations of individual feature responses, I also implement LS-SVMs with a polynomial map. In the experiment, I compare the linear function approximation to LS-SVMs. The hypothesis is that LS-SVMs should achieve higher accuracy on the test set than linear function approximation, and that both approaches should achieve a higher accuracy than simple heuristics.

5.2 Overall Structure

The overall structure of the algorithm is given in Figure 5.2. At the beginning, the memory is blank. First, the algorithm randomly generates a certain number of constellations (n) from training images and determines the "usefulness" of each by their performance on the training set. At each iteration, the algorithm also keeps track of the performance of these constellations on the test set, which contains a different set of images from the training set. The performance on the test set is used for evaluating the experimental result but is not used to determine the usefulness of a constellation. In the current set of constellations, useful ones are stored, and useless ones are discarded and replaced with new ones. This process repeats until all of the current constellations are considered useful. In the following sections, I will give the details of each building block in the algorithm.

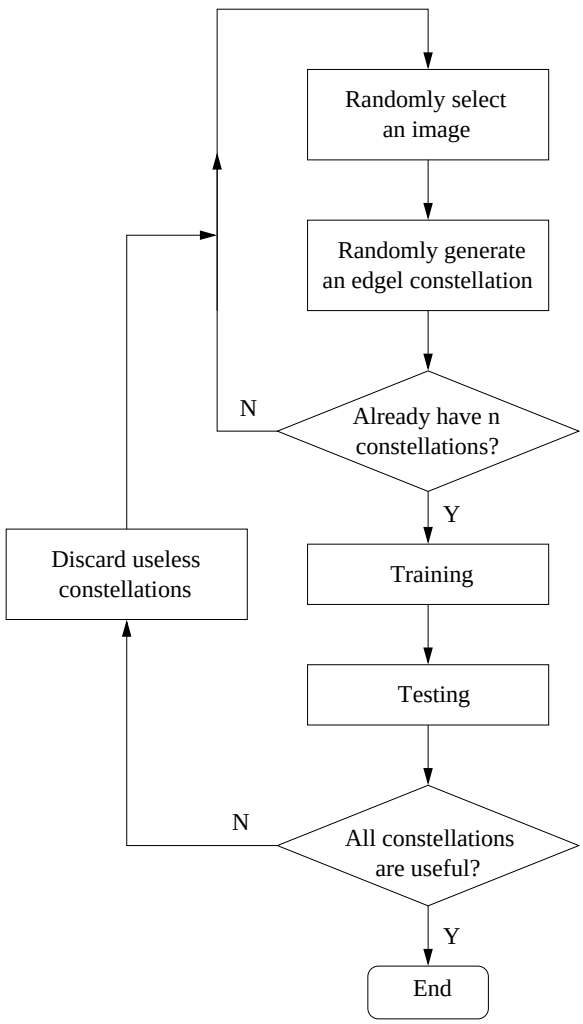


Figure 5.2: Algorithm overview.

5.3 Constellation of Edgels

Piater and Grupen (2002) used an edgel primitive in an image to differentiate geometric shapes. An *edgel* is defined as a small edge with intensity and orientation in an image. In our approach, every edgel in an image is denoted as a vector:

$$I_e = \begin{bmatrix} g_v \\ g_h \end{bmatrix}, \tag{5.1}$$

where, g_v and g_h are the results of convolving the image with vertical and horizontal first derivative masks centered at the corresponding image pixel. Intuitively, the magnitude of g_h is high if a pixel belongs to a vertical edge and the magnitude of g_v

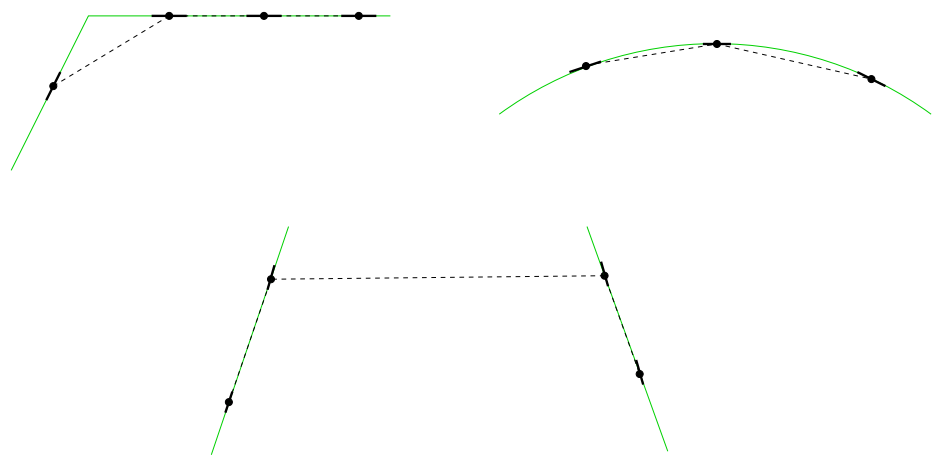


Figure 5.3: Example of edgel constellations (Piater and Grupen 2002).

is high if a pixel belongs to a horizontal edge. The orientation and the magnitude of this edgel can be computed accordingly:

$$\theta = atan2(g_h, g_v), \tag{5.2}$$

and

$$mag = \sqrt{g_h^2 + g_v^2}. \tag{5.3}$$

The orientation of an edgel is simply the direction of the edge to which it belongs. The magnitude of an edgel denotes the degree of intensity change orthogonal to the edge.

In this thesis, I use the Sobel mask for edge detection. The Sobel horizontal mask has the following form:

$$s = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{5.4}$$

The Sobel vertical mask is s^T. The edge image of a given image is calculated by Equation 5.3.

Piater and Grupen (2002) defined *constellation* as a set of edgels that satisfy some geometric configuration in 2D Euclidean space (Figure 5.3). This geometric configuration is represented by the relative position and orientation between edgels in a constellation. A constellation is rotation-invariant in the image plane. By following their idea, we use a homogeneous transform to encode the relative position

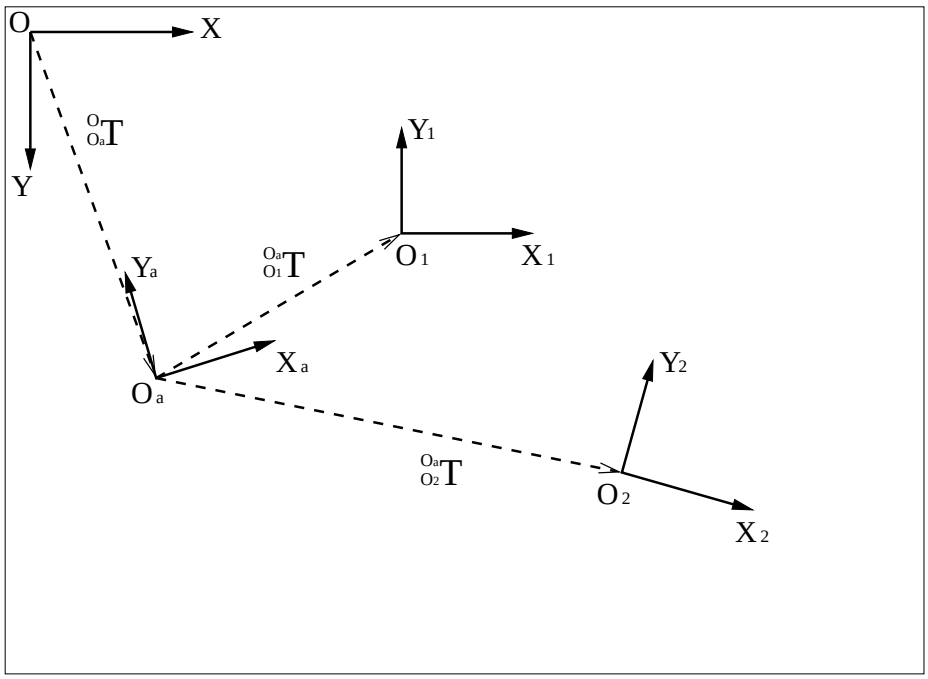


Figure 5.4: Homogeneous matrix model of an edgel constellation.

and orientation between edgels in a constellation. A 2D homogeneous transformation matrix is defined as:

$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta & x_t \\ \sin\theta & \cos\theta & y_t \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.5} $$

where ${}^{A}_{B}\mathbf{T}$ denotes the transform from *coordinate frame* B to A, which contains the information of the orientation and position of coordinate frame B relative to A. A coordinate frame is defined in the 2D plane with an origin and two directions that are orthogonal to each other. To transform coordinate frame A to B: 1) translate A by x_t along the x axis and y_t along the y axis and then 2) rotate counterclockwise by θ.

An example is given in Figure 5.4. "O" denotes the global coordinate frame, with "X" and "Y" denoting the orientation of this coordinate frame. Similarly, O_a denotes the *first* edgel (also referred to as the *anchor*) coordinate frame and O_1 and O_2 are the coordinate frames of two other edgels in this constellation. The x directions in each of the coordinate frame (except the global) are selected to coincide with the orientation of the edgels. When a constellation is generated, all the edgels in a constellation

are chosen from a training image. The pose of edgel i in this constellation can be computed as follows:

$$ {}^{O}_{O_i}T = {}^{O}_{O_a}T \, {}^{O_a}_{O_i}T, \tag{5.6} $$

where ${}^{A}_{B}T$ denotes the homogeneous matrix from coordinate frame B to A. The transform from the coordinate frame of edgel i to the anchor can be calculated by:

$$ {}^{O_a}_{O_i}T = {}^{O}_{O_a}T^{-1} \, {}^{O}_{O_i}T, \tag{5.7} $$

which gives us the orientation and position of coordinate frame O_i relative to O_a.

When we match a constellation in a given image, the transformation matrix between the global coordinate frame and the anchor edgel coordinate frame can be arbitrary since any point in the image can be chosen as the anchor edgel. Therefore, we encode a constellation relative to the anchor edgel coordinate frame instead of the global coordinate frame. This conversion is done by using Equation 5.7. During the matching phase, given an image, we will first choose an arbitrary anchor edgel above some magnitude. The key is that ${}^{O}_{O_i}T$ gives us the position of where to look for every other edgel as well as its expected orientation in the global coordinate frame (computed by using Equation 5.6). Assuming that θ_i is the expected orientation of the x direction of element i in this constellation, element i can be written in vector form:

$$ F_i = \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix}, \tag{5.8} $$

where the magnitude of element i is defined as 1. In this way, we can also calculate the positions of every element in a constellation based on the anchor edgel. Assuming element i at the corresponding position in a given image is:

$$ I_i = \begin{bmatrix} g_v \\ g_h \end{bmatrix}, \tag{5.9} $$

we can compute the degree to which an edgel in the image matches an element in a constellation by the following:

$$ f_i = \max\left(0, {I_i}^T F_i\right), \tag{5.10} $$

where $f_i = 1$ means a high-magnitude edgel in the same orientation as the constellation element i. We can compute the response for a particular constellation k with n_k elements by (given an arbitrary anchor point):

$$ \bar{C}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \log f_i^k. \tag{5.11} $$

The maximum response of constellation k in a given image can then be computed by:

$$C_k = \max_{all\ possible\ O_a's} \bar{C}_k. \tag{5.12}$$

In practice, the set of O_a's can be pruned dramatically by only focusing on pixels that exhibit an edge magnitude above some threshold. We will combine the responses of different constellations for object classification in this chapter and aspect recognition in Chapter 6.

5.4 Visual Feature Learning

Now the question is: where does a constellation come from? One possibility is to manually define constellations. For example, consider the block and cup in Figure 5.1. We can define a constellation to capture the ellipse on top of the cup. Since this kind of "ellipse" can only be observed in a image of the cup, this feature should be useful for object recognition. However, this requires *a priori* knowledge about the objects that the system is trying to recognize. As the population of objects increases, it becomes more difficult for us to understand what constellations can effectively discriminate between those objects. One solution is to automatically learn useful constellations. In our approach, we randomly select pixels in a training image from which to compose constellations following a set of constraints: the edge magnitude of the elements in a constellation must exceed some threshold and pairwise element positions must fall within a specific range. The magnitude constraint guarantees that the elements in a constellation have relatively high magnitudes. The distance constraint guarantees that most of the elements in a constellation are apart from each other.

We qualify all generated constellations through the learning process: discriminative constellations will be kept and the useless ones will be replaced by other candidates until the system obtains a certain number of useful constellations. We define a discriminative (useful) constellation as one that can be only observed in some of the training images.

At the beginning of the experiment, the algorithm has no knowledge about which edgel constellation corresponds to which kind of grasps. The algorithm uses image-grasp tuples to first learn discriminative constellations and then to associate these constellations with grasp types. The grasp types considered in our experiment belong to one or more of the following categories (de Granville et al. 2006): Top-Gaussian, Top-Girdle, Side-Gaussian, and Side-Girdle. Top or side refers to the reach approach

direction used to grasp an object. For example, one can grasp a cup (Figure 5.1b) from either top or side. Gaussian or Girdle refers to two different kinds of distributions of the hand orientation. A Gaussian distribution can be used to describe grasps for which hand orientation is restricted in all three dimensions. For example, one can grasp a block (Figure 5.1a) from the top only following two certain orientations, which can be captured by two Gaussian distributions. A girdle distribution can be used to describe grasps for which hand orientation is restricted in two dimensions but free in the remaining one. For example, one can grasp a cup (Figure 5.1b) from the top or side regardless of the hand rotation about the axis of symmetry.

In the experiment, two learning methods are considered: linear function approximation and LS-SVMs.

5.4.1 Linear Function Approximation

For the linear function approximation method, we assume that each grasp type is a linear function of responses of all constellations:

$$\mathbf{G} = \mathbf{C}\mathbf{W}, \tag{5.13}$$

where $\mathbf{G}$ is a matrix that denotes the degree to which each grasp is appropriate for every training image ($+1$ recommends a grasp while -1 rejects a grasp). $\mathbf{C}$ is a matrix that denotes the maximum responses to all constellations for all training images, and $\mathbf{W}$ is a weight matrix that gives us the degree to which each constellation suggests a particular grasp. As an example, assuming that the number of training images is n, and the number of randomly-generated constellations is m, then Equation 5.13 can be represented explicitly by:

$$\begin{bmatrix} G_{11} & G_{12} & G_{13} & G_{14} \\ G_{21} & G_{22} & G_{23} & G_{24} \\ \vdots & \vdots & \vdots & \vdots \\ G_{n1} & G_{n2} & G_{n3} & G_{n4} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1m} & 1 \\ C_{21} & C_{22} & \cdots & C_{2m} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nm} & 1 \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & w_{m3} & w_{m4} \\ w_{01} & w_{02} & w_{03} & w_{04} \end{bmatrix}, \tag{5.14}$$

where the ith row of $\mathbf{G}$ denotes which of the four grasps can be used to grasp the object in image i; the ith row of $\mathbf{C}$ denotes the maximum responses of image i to all constellations calculated by Equation 5.12 (appended with a constant 1); and w_{ij} denotes the weight of constellation i to grasp type j (w_{0j} with $j \in \{1, 2, 3, 4\}$ are

intercepts). Note that constant 1's are added into $\mathbf{C}$ since the intercepts of the linear functions contained by Equation 5.13 are not necessarily zero.

The Moore-Penrose pseudo-inverse is used to calculate the weight matrix $\mathbf{W}$ (Penrose 1955), which minimizes the sum of squared prediction errors:

$$\mathbf{W}^* = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{G}. \tag{5.15}$$

Once the weight matrix $\mathbf{W}$ is solved, the algorithm obtains a set of linear functions for all grasp types and the training process terminates.

In the testing stage, given a test image, the algorithm will first compute the responses of all constellations in this image and then evaluate the linear function associated with each grasp type:

$$\hat{\mathbf{g}} = \mathbf{c}\mathbf{W}, \tag{5.16}$$

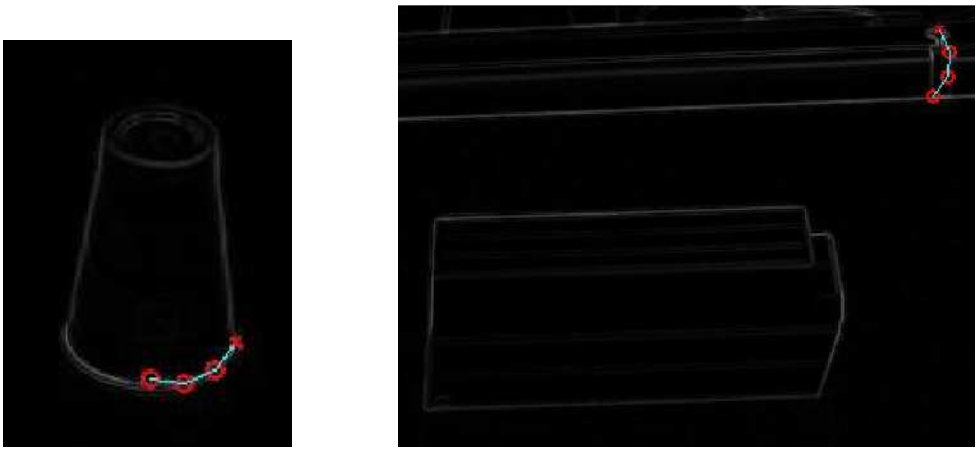
where we interpret that a positive $\hat{\mathbf{g}}_i$ recommends grasp type i and a negative value rejects this grasp type.

5.4.2 Least Squares Support Vector Machines

LS-SVMs are simplified versions of SVMs that use equality constraints instead of inequality constraints in the optimization problem (Suykens and Vandewalle 1999). In this way, a quadratic programming problem is converted into a problem of solving a set of linear equations, which can be solved more efficiently than the former. Essentially, the LS-SVM approach is trying to find a function that maps responses of constellations to grasp types just as before:

$$\hat{\mathbf{G}} = \phi(\mathbf{C})\mathbf{W}, \tag{5.17}$$

where $\phi(\mathbf{C})$ is a mapping function and $\mathbf{W}$ is a weight matrix that gives us the degree to which each feature in $\phi(\mathbf{C})$ suggests a particular grasp. I use a family of mapping functions called *polynomial maps*. In this case, each feature of $\phi(\mathbf{C})$ is a product of multiple features in $\mathbf{C}$. When using a polynomial map of degree 1, Equation 5.17 is equivalent in representational power to Equation 5.16. However, by using a polynomial map of degree larger than one, each feature response of $\phi(\mathbf{C})$ will become a product of two or more constellation responses in $\mathbf{C}$. In this case, LS-SVMs should outperform (or at least perform as well as) linear function approximation, since the former considers all possible combinations of two or more constellations as well as the responses of single constellations.



(a) A constellation found on the cup.

(b) The same constellation found on the background in an image of the block.

Figure 5.5: Edgel constellations found by heuristic method.

5.5 Experimental Evaluation

5.5.1 Methodology

In the experiment, we use ten images each of a cup and a block from slightly different perspectives as a training set (see Figure 5.1). At any one time, eight constellations are considered. In the testing stage, another 20 images containing cups and blocks (ten of each again) from different perspectives are used as the test set. For every iteration in Figure 5.2, the prediction accuracy of both linear function approximation and LS-SVMs is recorded for comparison (with the same set of constellations). The prediction accuracy is measured by the percentage of correct predictions in the test set. A prediction is considered to be correct if it is consistent with the true grasp types associated with the test image. For these experiments, we use a polynomial map of degree three.

5.5.2 Results and Discussion

The manually defined constellations work as heuristics. Since the task is to separate the cup from the block, one constellation is manually defined on the curve of the cup (Figure 5.5a). The accuracy is 85% with 20 test images. Most of the errors come from the situation in Figure 5.5b, because of a curve-like shape that is visible in the background.

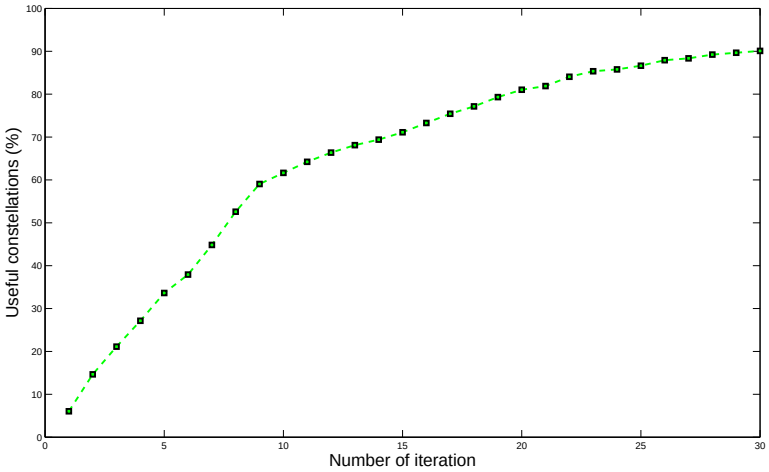
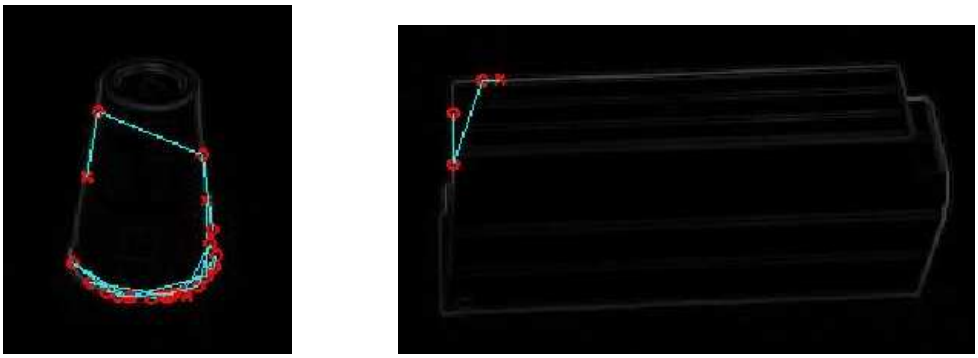


Figure 5.6: The average percentage of useful constellations found during the learning process.



(a) A constellation found on the cup.

(b) A constellation found on the block.

Figure 5.7: Edgel constellations found by linear function approximation method.

In total, 29 independent experiments are done by using the same set of training and testing images. The percentage of "useful" constellations versus the number of iterations is shown in Figure 5.6 (mean of 29 experiments). Since different experiments may take a different number of iterations to terminate (when all constellations are useful), in order to calculate the mean performance of 29 experiments, I cut off the number of iterations at 30. From this figure we can see that after 30 iterations, about 90% of the constellations are considered useful. The curve increases monotonically because only useless constellations are regenerated (the number of useful constellations won't decrease). Some discriminative features such as the curve (Figure 5.7a) and the right angle (Figure 5.7b) are found.

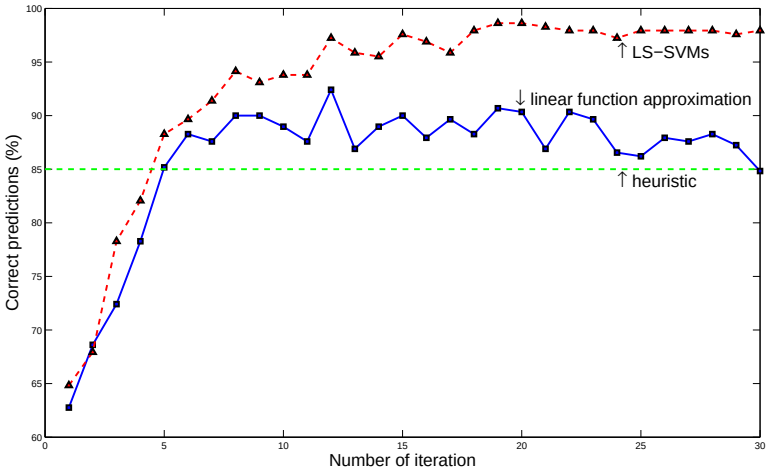


Figure 5.8: Learning curve: linear function approximation vs. LS-SVMs.

The average performance of each method for 29 experiments versus the number of iterations is shown in Figure 5.8. The LS-SVM method outperforms the pseudo-inverse method all the time except for iteration 2. At iteration 2, the average number of useful constellations is 15% (according to Figure 5.6), equivalently, 1 constellation. As we know, the advantage of the LS-SVM method is that it considers the combinations of several features. However, if only one constellation is useful, the LS-SVM method may even perform worse since it considers combinations containing useless features.

Both of the learning methods perform better than the heuristic method from 5 to 29 iterations. An immediate question is whether the difference between the learning performance of these two methods is statistically significant. One way to test this is to use randomized analysis of variance (ANOVA)(Piater et al. 1998). The null hypotheses here are: 1) The mean performance of the two algorithms are the same 2) The relation between the number of iterations and performance (learning effect) doesn't depend on the algorithm. The data set is 58 learning curves (29 for each algorithm) from iteration 1 to iteration 20 (in order to maximize the performance of the linear function approximation method). First, a conventional two-way ANOVA is performed on this data set and gives F_{alg}=38.3990, F_{int}=0.5270 respectively (*alg* means algorithm, which corresponds to hypothesis 1 and *int* means interaction, which corresponds to hypothesis 2). Then, this data set is shuffled and ANOVA is performed 1000 times, which gives us an estimate of the sampling distribution. By comparing F_{alg} and F_{int} with the distribution of F^*_{alg} and F^*_{int}, we obtain $p_{alg} < 0.009$ and $p_{int} <$

0.685 which means that we can safely reject null hypothesis 1 at the .01 level; however, we cannot reject null hypothesis 2 safely. As a conclusion, the difference between the mean performance of linear function approximation and LS-SVM classifiers is statistically significant; but there is no statistically significant difference between the learning effects of these two algorithms, that is, these two algorithms learn at similar speeds.

5.6 Parameter Selection

In the experiment, there are several parameters which one can choose independently. These parameters are summarized below (the chosen values are given in parentheses):

1) The number of edgels in a randomly generated constellation (four). The bigger this number is, the more selective the constellation is. Since I only consider two objects in the experiment, a constellation of four elements is good enough to discriminate them (such as the constellation in Figure 5.5a). Also, I deliberately choose the value of this parameter to be the same as the heuristic method for performance comparison.

2) The pair-wise distance between elements in a constellation. This parameter depends on the scale of the object. In our case, it is randomly sampled from a Gaussian distribution with a mean of 20 pixels and a standard deviation of 10 pixels (the same parameters are used for all objects). I choose the above values by observing samples of constellations generated by different Gaussians. If the mean of the Gaussian is too small, a lot of elements in a constellation will cluster together to describe useless features (such as straight lines or small corners); if the mean is too large, some pixels out of the object will be selected as part of a constellation. The standard deviation of the Gaussian is selected in order to cover the scale changes due to different objects or different distances between objects and the camera.

3) The lower bound of edge magnitude included in a constellation during the sampling step (0.4 with the maximum magnitude normalized to 1). The edges with magnitudes smaller than this value in an image are too weak to be seen (such as shadows) and a constellation including such edges tends to be useless.

4) The lower bound of magnitude for a pixel being considered as the anchor point of a constellation during the query step (0.16 with the maximum magnitude normalized to 1). The higher this value is, the fewer the pixels in an image are left as candidate anchors. In this way, we can prune the search in Equation 5.12. However,

this parameter should not be too large, otherwise, some possible good matches of a given constellation will be missed. For example, the response to a constellation can still be high if the magnitude of the anchor is low, but the match values of the other elements are high.

5.7 Conclusion

In this chapter, we have developed a learning algorithm that can use image input to predict appropriate grasp types. Edgel constellations, which are sets of pixels with fixed relative positions and orientations, are defined in order to differentiate geometric shapes in an image. Candidate constellations are sampled from training images until a good performance is achieved. Two learning methods, linear function approximation and support vector machines, are used to construct classifiers of grasp type from the observed set of constellations. Experimental results showed that the mean performance of the LS-SVM method is statistically significantly better than the linear function approximation method; however, there is no statistically significant difference between the learning speeds of these two approaches.

Chapter 6

Aspect Recognition

In the previous chapter, I discussed the object/grasp recognition problem, particularly the mapping between observed visual features and grasp types. In this chapter, I will assume that the identity of the object in a given image is known and focus on the problem of recognizing a particular object viewpoint. Experimental results show that given examples of image/aspect pairs that cover all possible aspects well, the aspect of an object in a novel image can be recognized.

6.1 Problem Definition and Methods

The object recognition approach presented in the last chapter is lacking in three ways: First, the response of a feature constellation can vary dramatically as the object is rotated out of the image plane. We can amend this by considering sets of constellations that cover all object views and then utilize this to recognize the aspect of a given object. Second, it was not uncommon for some learned constellations to incorporate background or shadow features. Third, the evaluation of the usefulness of a constellation involved searching for that constellation across the entire training data set, which can be a very expensive process.

In designing a solution to these problems, I first observe that background and shadow features tend not to be robust to perspective or lighting changes. I therefore hypothesize that useful constellations should be *locally robust*. A constellation is defined to be locally robust if it can be found in no less than a certain percentage of the total number of *neighborhood images*. Neighborhood images are those training images with similar aspects. However, in order to make aspect recognition feasible, constellations must also be globally discriminative. I propose to improve the constellation learning process by using a two-step filtering approach. The first step is to only use

constellations that can be robustly found within neighborhoods of aspects. The second step measures the selectivity of a constellation by comparing its response to the neighborhood images and the entire set of training images. Only constellations with higher selectivity will be retained. The hypothesis is that the algorithm with these two steps of filtering should perform statistically significantly better in recognizing correct aspects than an algorithm without such filtering.

6.2 Object Model Learning

As discussed in the background chapter, a particular feature constellation can be observed over a set of aspects on the aspect sphere. This set of aspects can be summarized by a mixture model of probability density functions (pdfs) on the aspect sphere. In turn, this mixture model of pdfs gives us a principled way of inferring aspect given the observation of a set of constellations.

The overall structure of the algorithm is outlined in Figure 6.1. During the training process, the algorithm randomly generates one constellation at a time from the training images and then uses the two steps of filtering to determine the usefulness of this constellation. If this constellation is determined to be useful, the algorithm then constructs a mixture model of pdfs for this constellation; if not, this constellation is discarded and a new constellation is generated. Note that in Figure 6.1, the two conditional tests corresponding to the local and global filters lead to different places. This is because there are some "bad" images in the training set (such as ones that do not include the target object). Empirically, a constellation usually fails to pass the local filter because it focuses on some unstable features (such as shadows); however, a constellation usually fails to pass the global filter because it is generated from a bad image. Therefore, the algorithm randomly chooses a new image in the latter case instead of randomly generating a new constellation from the same image. This training process continues iteratively until the entire set of generated constellations cover most of the training images. During the testing process, the model will use learned constellations to recognize the aspects of an independent set of testing images.

6.2.1 Data Collection and Preprocessing

In our experiments, each element in the data set is a *tuple* of image and object pose. A Polhemus Patriot (Colchester, VT) is attached to the object so that the 3D position and orientation of the object can be paired with each image. Tuples are gathered

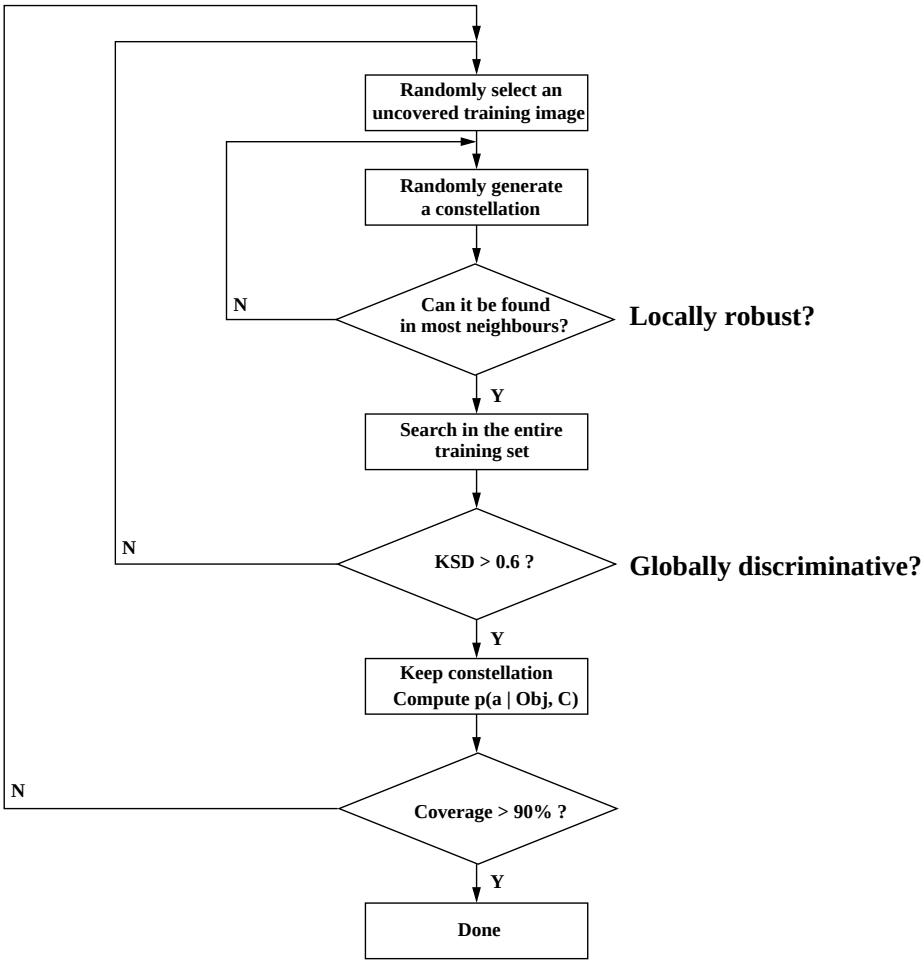


Figure 6.1: Overall structure of the aspect recognition algorithm.

continuously as the object is rotated in front of the camera. In all, a data set will contain about 2000 tuples.

The next question is how to focus the algorithm's attention to the region where the object is located. A region of interest (ROI) is a particular region in a given image that ideally contains only the object. Since the training and testing images we are dealing with in our experiments are complicated (in the sense of the object and background), we need a systematic way to extract the object out of a scene. An example of image preprocessing is given in Figure 6.2. First, we use background differencing to remove the still objects such as the table and walls (Figure 6.2b). However, some objects (such as the hand) that move together with the manipulated object cannot be removed in this way. These kinds of objects are removed using color model filtering (Figure 6.2c). A color model is a mixture of Gaussians that describes the color distribution of individual pixels taken from some image region. In

(a) Original image

(b) Background removed

(c) Skin removed

(d) Leftmost blob

(e) Grayscale image

(f) Edge image

Figure 6.2: An example of image preprocessing.

our case, these image regions are selected on the objects that we want to remove. The mixture models are trained for these regions using Expectation Maximization (EM). In novel images, regions that can be explained by the learned model are removed. Next, blob detection is performed (Horn 1986). Blob detection is a means of finding contiguous regions of pixels of interest in a binary image (with 1 as white and 0 as black). In our case, these pixels correspond to those that 1) are different from the background, and 2) are not skin color. After blob detection, the rightmost (or the leftmost, depending on how the object is being held) blob with a number of pixels larger than some threshold is extracted as a sub-image from the original image according to the bounding box. Normally only the object itself is contained in this sub-image after these steps (Figure 6.2d). The color image in Figure 6.2d is converted to a grayscale image in Figure 6.2e. Then the grayscale image is convolved with vertical and horizontal masks and the edge image (Figure 6.2f) is calculated using Equation 5.3.

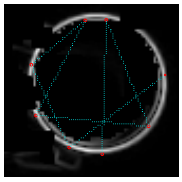
6.2.2 Local Filter

The first step in the filtering process requires that constellations be locally robust. First, we sample a random constellation from the given image and then compute the constellation responses in the 10 images that have the closest aspect (we call them neighborhood images). Only constellations that are found in as least 70% of the neighborhood images are kept. This step of filtering is computationally inexpensive because the search involves a small number of images.

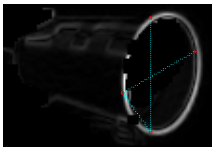
6.2.3 Global Filter and Threshold Selection

Constellations should also be globally discriminative. We will take Figure 6.3 as an illustration. Note that the rotation axis of the cup hereafter is approximately aligned with the x axis and the top of the cup is on the side of $+x$ in the coordinates of an aspect sphere. This is determined by the orientation of the Polhemus sensor attached to the cup and is different from the setup in Figure 3.1.

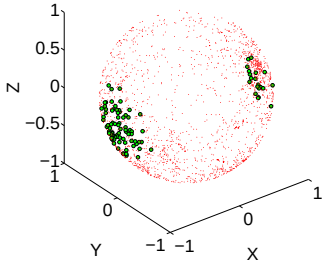
In Figure 6.3a, a locally robust constellation is generated on the top view of the cup. If we search for this constellation in the entire training set, the number of images in which this constellation can be observed depends on the constellation response threshold. The higher the threshold, the smaller the number of images. This trend is shown in Figure 6.3c, e and g. In each panel, the aspects corresponding to the top



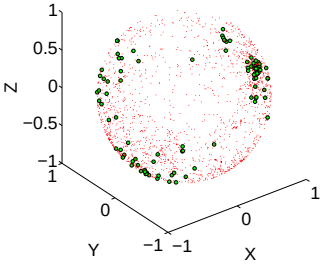
(a) Constellations found on the top of the cup



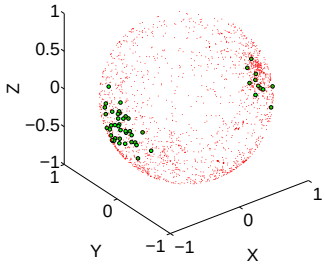
(b) Constellations found on the upper side of the cup



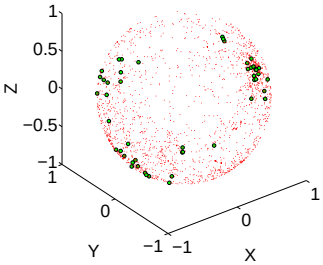
(c) Aspects for top 100 responses



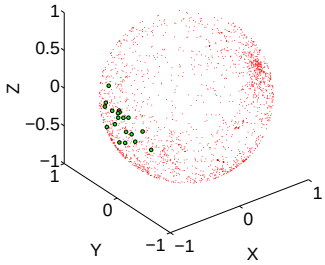
(d) Aspects for top 100 responses



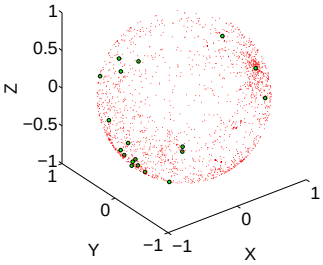
(e) Aspects for top 50 responses



(f) Aspects for top 50 responses

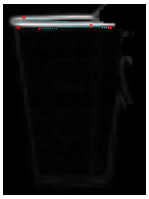


(g) Aspects for top 20 responses

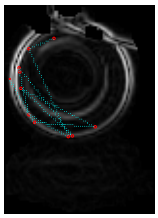


(h) Aspects for top 20 responses

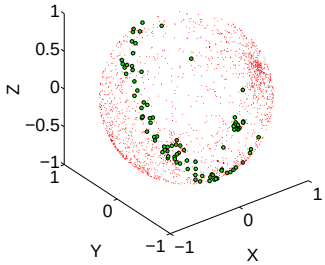
Figure 6.3: Constellation responses for the cup-top & upper side view.



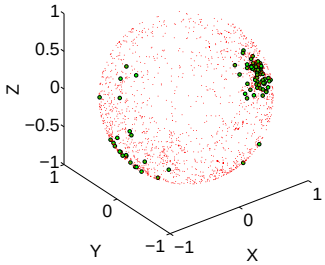
(a) Constellations found on the middle side of the cup



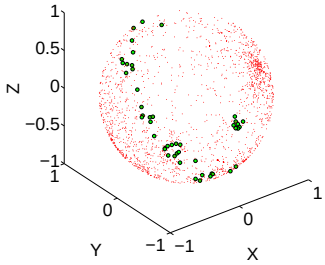
(b) Constellations found on the bottom of the cup



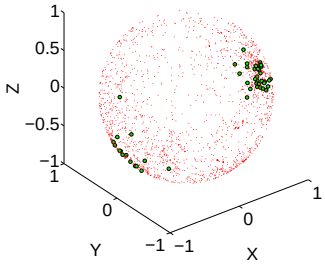
(c) Aspects for top 100 responses



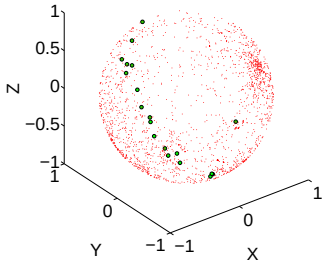
(d) Aspects for top 100 responses



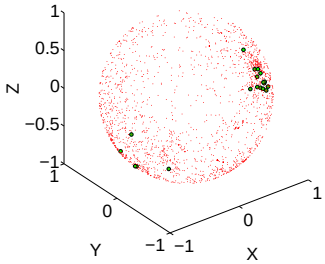
(e) Aspects for top 50 responses



(f) Aspects for top 50 responses

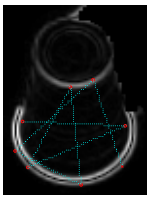


(g) Aspects for top 20 responses

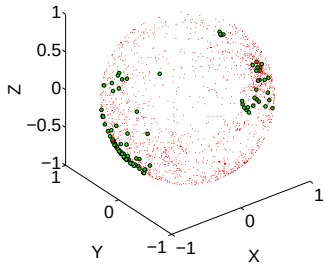


(h) Aspects for top 20 responses

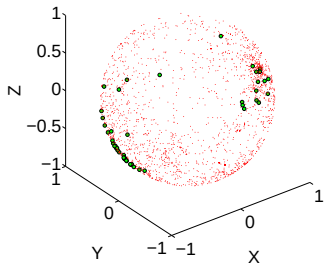
Figure 6.4: Constellation responses for the cup-middle side & bottom view.



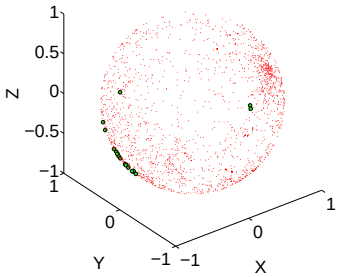
(a) Constellations found on the lower side of the cup



(b) Aspects for top 100 responses



(c) Aspects for top 50 responses



(d) Aspects for top 20 responses

Figure 6.5: Constellation responses for the cup-lower side view.

n responses are highlighted in green, where n determines the threshold. The highest $(n+1)^{th}$ response is used as the corresponding threshold so that there are n responses above this value. Ideally, the constellation in Figure 6.3a should only be observed in some aspects as shown in Figure 6.3g. However, when the threshold becomes lower, this constellation can also be observed in some aspects on the other side of the aspect sphere, which corresponds to the bottom of the cup. This seems reasonable since a circle (with a slightly smaller radius) can be also seen on the bottom view of the cup.

In Figure 6.3b, a locally robust constellation is generated from the upper side view of the cup. In the corresponding aspect figures, there is an annular cluster on the $-x$ side and a regional cluster on the $+x$ side. Even with the aspects for top 20 responses (Figure 6.3h), there are still a few aspects corresponding to the lower side view of the cup.

In Figure 6.4a, a locally robust constellation is generated on the middle side view of the cup. This constellation is very selective and can only be observed in the aspects along a great circle of the aspect sphere. In this case, the resulting distribution does not depend on the choice of the threshold very much.

In Figure 6.4b, a locally robust constellation is generated on the bottom view of the cup. This case resembles the top view case. However, even with the aspects for top 20 responses (Figure 6.4h), there are still a few aspects corresponding to the top view of the cup.

In Figure 6.5a, a locally robust constellation is generated from the lower side view of the cup. In the corresponding aspect figures, there is a bigger annular cluster on the $-x$ side and a smaller annular cluster on the $+x$ side. In Figure 6.5d, the threshold is too high for the aspects corresponding to the lower side view and the centroid of these 20 aspects shifts to the $-x$ side. Since the shapes of these clusters vary substantially with different thresholds, the immediate challenges are: 1) how to systematically select a threshold for each constellation, and 2) how to determine whether there is an obvious cluster or not?

One way to find the maximally discriminative threshold between the local neighborhood and the rest of the training set is to use the Kolmogorov-Smirnoff distance (KSD) between the two populations. The KSD is defined as the maximum distance between two cumulative probability distributions. Recall that the value of the response of a constellation is calculated by Equation 5.11, which is a real number between 0 and $-\infty$, with 0 corresponding to the highest match.

By intuition, the threshold we choose should be low enough to allow a constellation to be found in most of the neighborhood images where this constellation is generated, at the same time it should be high enough in order not to allow the same constellation to be found in too many of the other training images. However, the response of this constellation may still be high in some other aspects, since different aspects may look similar due to symmetry. The optimal threshold should maximize the difference between the neighborhood images and the ensemble with respect to the response of a constellation. Such thresholds, specific to each constellation, can be determined optimally by calculating the KSD between the response of the 10 neighborhood images and the entire set of training images (Piater and Grupen 2002).

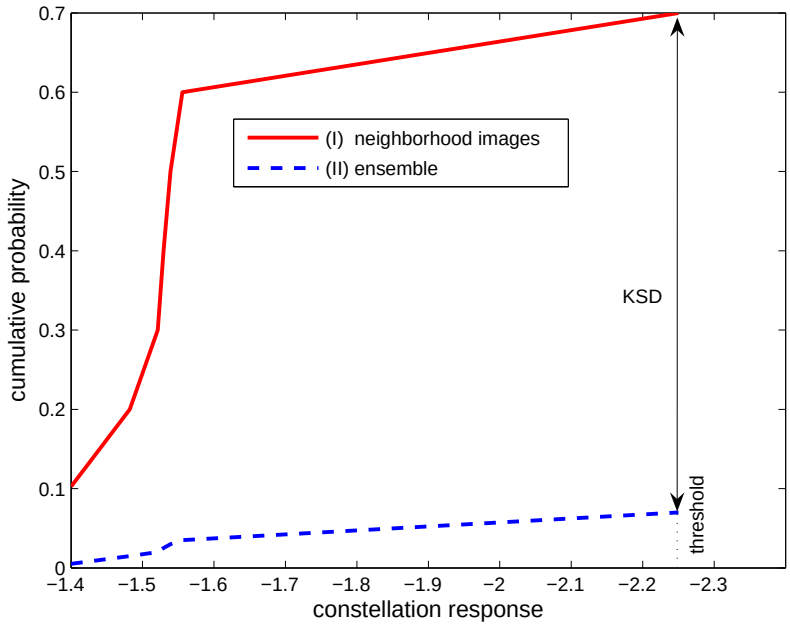
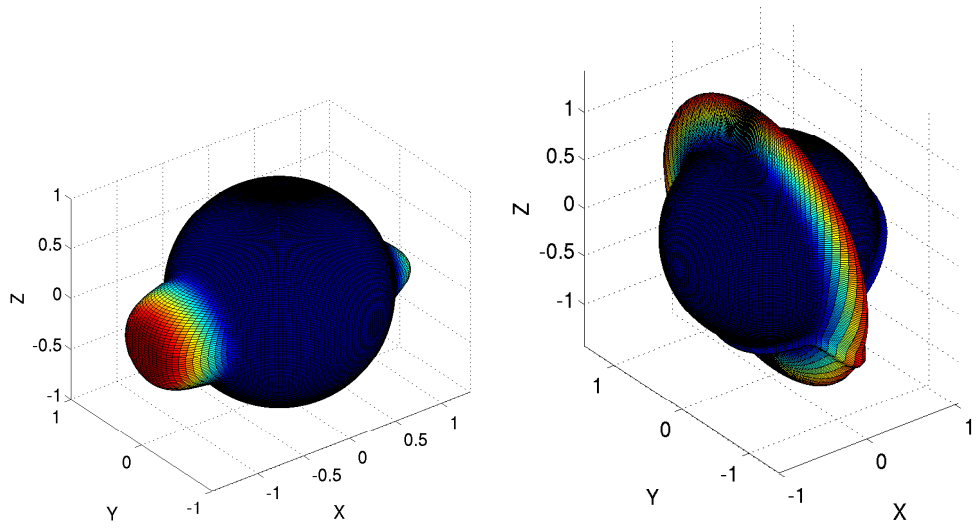


Figure 6.6: Use KSD to determine the threshold for a constellation.

The responses corresponding to the neighborhood images and the entire training set are compared as illustrated in Figure 6.6. The horizontal axis is the different threshold values that are considered and the vertical axis is the fraction of images in which a constellation can be found given a particular threshold value. The two curves correspond to the neighborhood images (I) and the ensemble (II), respectively. Each curve represents cumulative probability of a constellation being observed in the set of images given the different threshold values. However, we only consider a certain set of discrete thresholds here. We first sort the response values of the neighborhood images in descending order (the lowest 3 are usually too small and are



(a) Mixture model corresponds to Figure 6.3c

(b) Mixture model corresponds to Figure 6.4c

Figure 6.7: Examples of mixture models.

not considered). Then, we calculate the midpoints between each pair of the adjacent response values and use these midpoints as the candidate thresholds. The optimal threshold is selected such that the distance between the two curves is maximized (KSD). However, if the KSD is too small (less than 0.6), a constellation is considered as not discriminating the neighborhood images from the entire set of training images. In this case, this constellation is discarded.

6.2.4 PDF: Mixture Models

Once the constellation threshold is determined, the training set images can be partitioned into those in which the constellation is recognized and those in which it is not. The aspects of these images in which constellation i is recognized are used to construct a mixture model $p(\mathbf{a}|Obj, C_i)$ as discussed in Section 3.3. Some examples of such mixture models are given in Figure 6.7. Two pdfs are used in each of these mixture models. In Figure 6.7a, there are two Fisher-von Mises distributions on each side, which correspond to the two clusters in Figure 6.3c. In Figure 6.7b, there is a girdle distribution intersected by a small circle distribution, which corresponds to the aspects shown in Figure 6.4c. Once the algorithm learns the mixture model $p(\mathbf{a}|Obj, C_i)$ by EM, the training process for this particular constellation is over.

6.2.5 Covering the Aspect Sphere

The algorithm iteratively generates constellations from the uncovered training images until a certain coverage rate is obtained. We define a training image as being covered if at least one constellation in the current repository has a response higher than the threshold for the image.

6.3 Recognizing Novel Aspects

In this section, we assume that a sufficient number of useful constellations have been acquired as well as the mixture distribution associated with each constellation. Here, "sufficient" means that 90% of the training images are covered by constellations. During the testing stage, for a novel scene containing a known object, we search for all the constellations corresponding to this object in the repository. The mixture distributions associated with each found constellation are used to estimate the maximum likelihood estimate of the viewing aspect.

Given a novel image, we would like to accurately estimate the viewing aspect. More specifically, assuming that a set of constellations $C_1, C_2, ..., C_N$ are observed in an image, we solve this problem by first estimating $p(a|Obj, C_1, ..., C_N)$ and then finding the maximum likelihood aspect. Assuming that the constellations are independent of each other (the naïve Bayes assumption), we can estimate the probability density of the object aspect accordingly:

$$p(a|Obj, C_1, ..., C_N) = \prod_{i=1}^{N} p(a|Obj, C_i). \tag{6.1}$$

In general, this pdf contains local maxima. We find the global maxima by first randomly selecting a set of aspects from a gridded sample of the aspect space according to the pdf. Then, for each sample, the algorithm locally ascends the pdf until a maximum is found. Finally, the algorithm selects those samples whose densities are higher than 90% of the highest density of all samples as the recognized aspects. Usually these recognized aspects are very close to each other, in which case there is a global maximum of the combined mixture density whose density is much higher than the local maxima. However, sometimes there is a local maximum whose density is close to that of the global maximum. In this case, recognized aspects are sampled from both the local and global maxima and may be apart from each other. In the real situation, since a robot can only choose one recognized aspect, we can adjust the algorithm by choosing only the one sample with highest density.

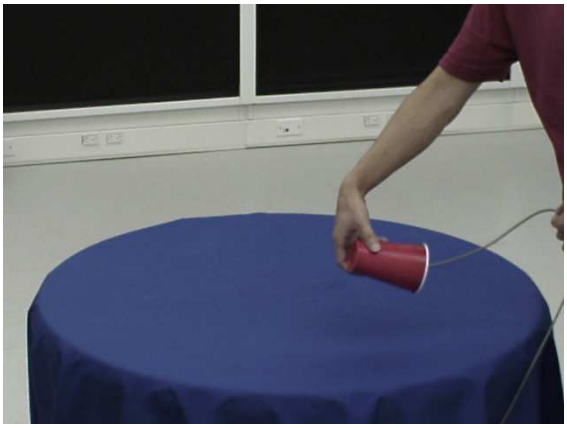
Three examples of aspect recognition are given in Figures 6.8, 6.9 and 6.10. In the first, a testing image of the cup is shown in Figure 6.8a and the corresponding (true) aspect is shown on the aspect sphere in Figure 6.8b. For this particular image, two constellations are observed with responses above their respective thresholds, which are shown in Figure 6.8c and Figure 6.8e. The mixture models of pdfs correspond to these two constellations are shown in Figure 6.8d and Figure 6.8f. The maximum likelihood aspects are sampled in the region with higher combined mixture probability density (Figure 6.8g). These maximum likelihood aspects are very close to each other and one of them is shown in Figure 6.8h. The training image which has the nearest aspect to this point is shown in Figure 6.8i. In this case, the angle between the estimated aspect and the true aspect is 0.31°.

In the second sample, the same recognition process of an image of the spray bottle is given. In this case, there are also two constellations observed but none of them can be used to estimate the aspect of the testing image accurately. In Figure 6.9c and Figure 6.9e, both of the two constellations have best matches on the label, due to the complicated texture. In this case, the angle between the estimated aspect and the true aspect is as large as 56.32°.

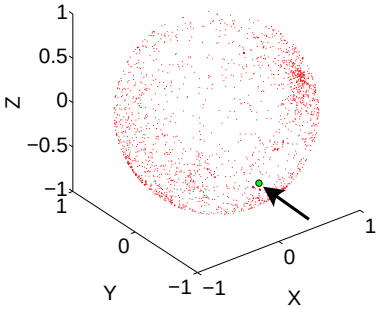
The third example is of the mug. Two constellations are observed in the test image and both of them incorporate part of a finger (Figure 6.10c, e). The mixture model corresponding to the first constellation (Figure 6.10d) has a much lower concentration than the one corresponding to the second constellation (Figure 6.10f). However, the combination of these two mixture models still gives a concentrated distribution (Figure 6.10g). The maximum likelihood aspects concentrate at two areas as shown in Figure 6.10h. The training image which has the nearest aspect to the point on the left is shown in Figure 6.10i. In this case, the mean of the angles between the estimated aspects and the true aspect is 19.78°.

6.4 Experimental Results

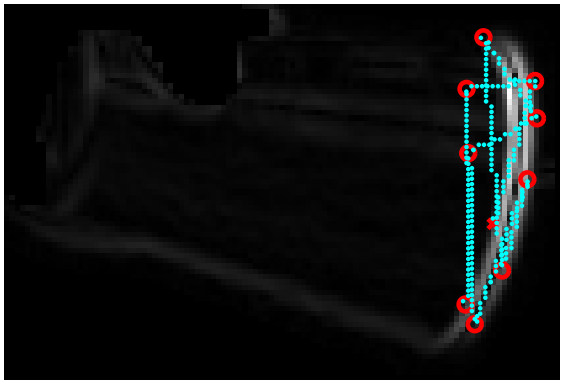
Both symmetric (a cup and a block) and asymmetric (a mug and a spray bottle) objects are used in the experiment. For each object, about 2000 images are taken uniformly in order to cover the aspect sphere as well as possible. In addition, for each object, we performed 10 independent experiments and for each experiment a different set of 100 images are randomly selected and reserved as the test data set. Error is measured for each test image as the angle between the estimated and true aspects



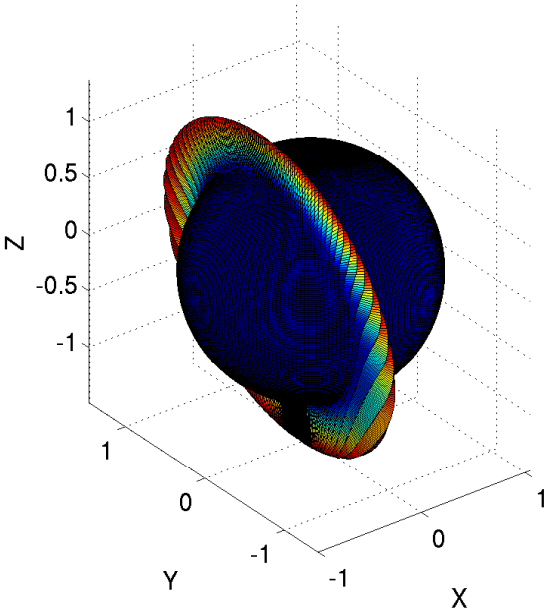
(a) A testing image of the cup



(b) The true aspect (the other points are surface texture)

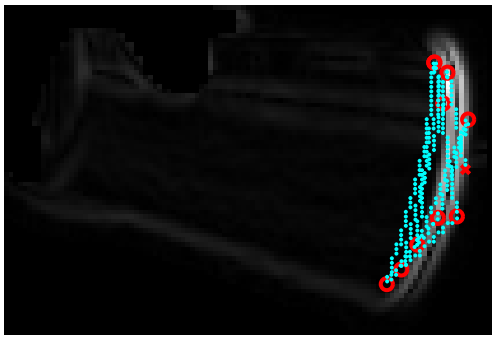


(c) Constellation one is observed

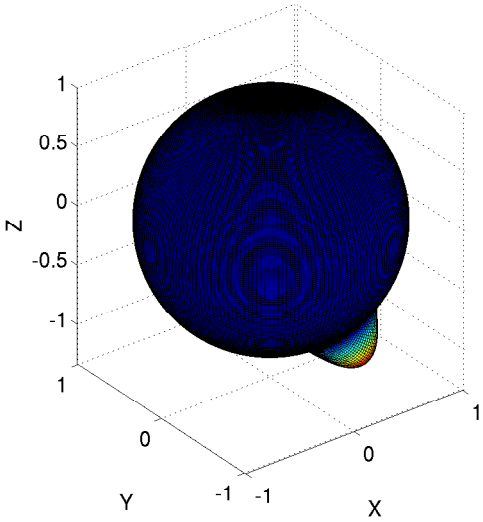


(d) The mixture density of constellation one

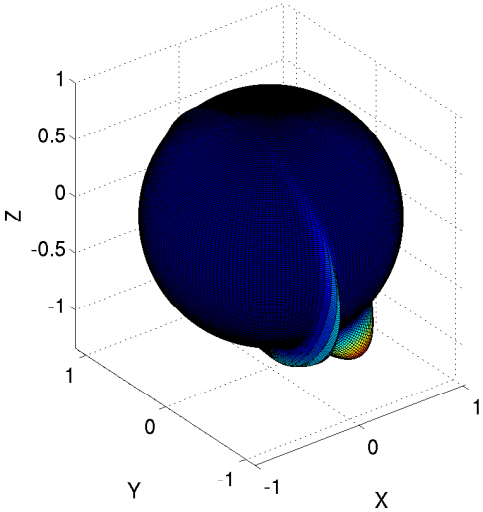
Figure 6.8: A sample recognition process of the cup.



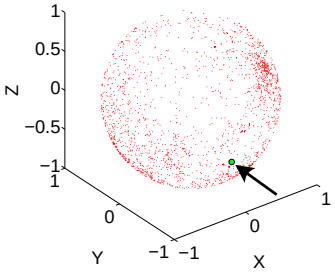
(e) Constellation two is observed



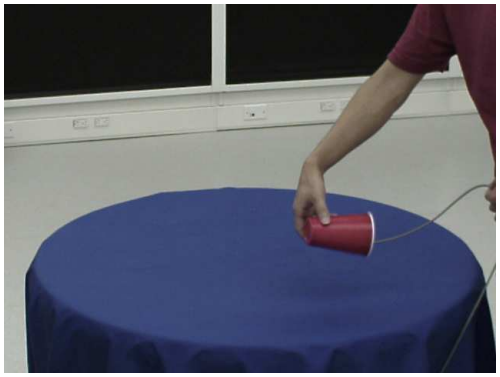
(f) The mixture density of constellation two



(g) The combined mixture density

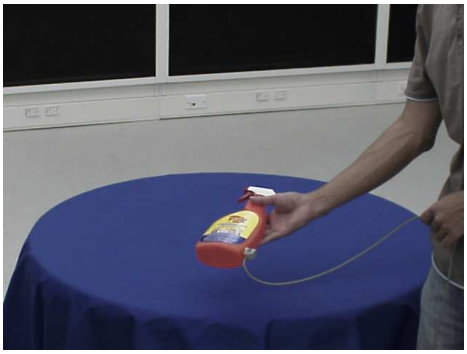


(h) The estimated aspect

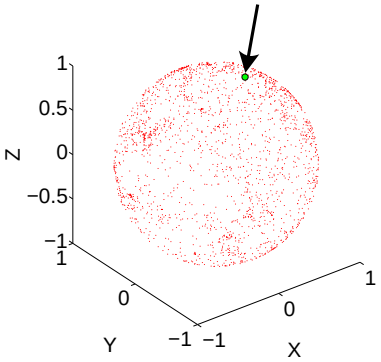


(i) The nearest training image

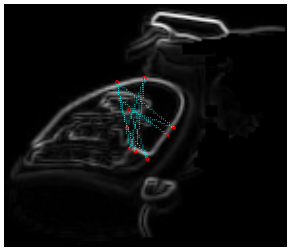
Figure 6.8: Continued.



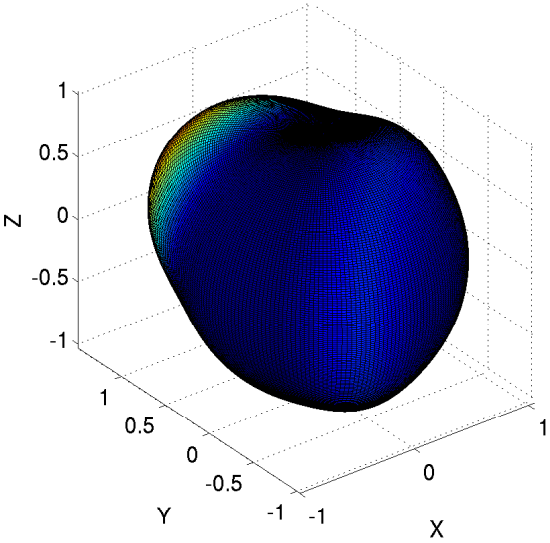
(a) A testing image of the spray bottle



(b) The true aspect

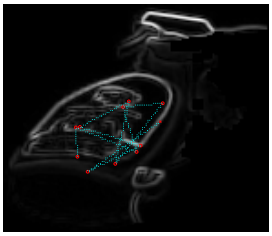


(c) Constellation one is observed

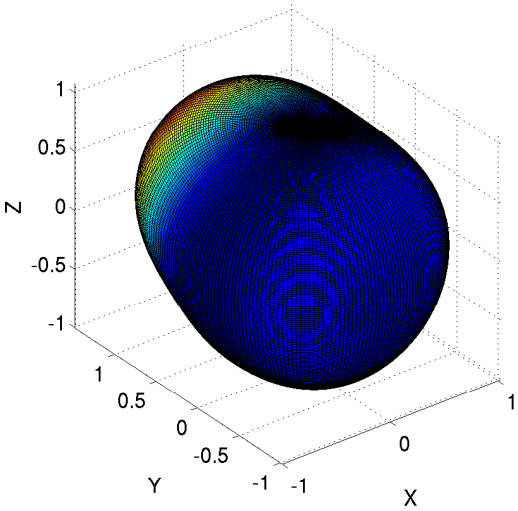


(d) The mixture density of constellation one

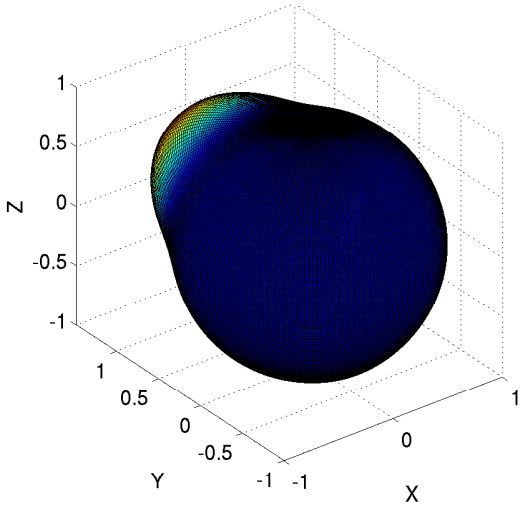
Figure 6.9: A sample recognition process of the spray bottle.



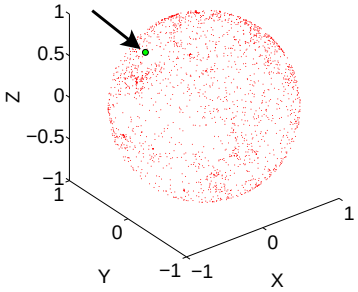
(e) Constellation two is observed



(f) The mixture density of constellation two



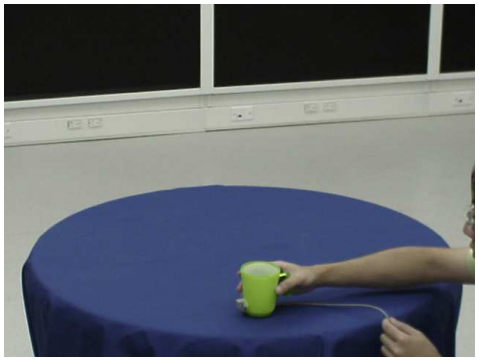
(g) The combined mixture density



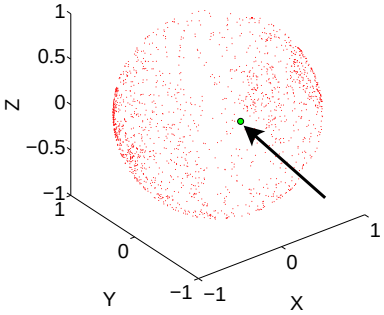
(h) The estimated aspect

(i) The nearest training image

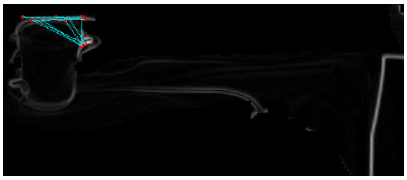
Figure 6.9: Continued.



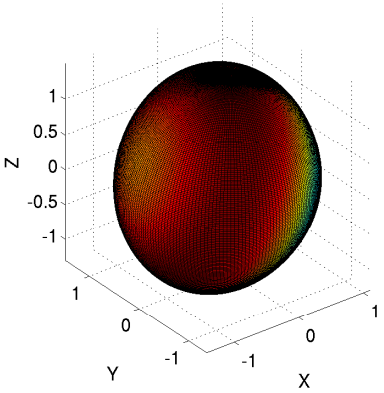
(a) A testing image of the mug



(b) The true aspect

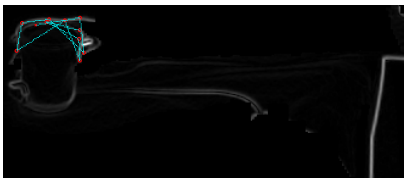


(c) Constellation one is observed

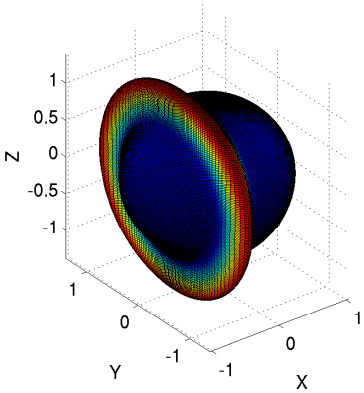


(d) The mixture density of constellation one

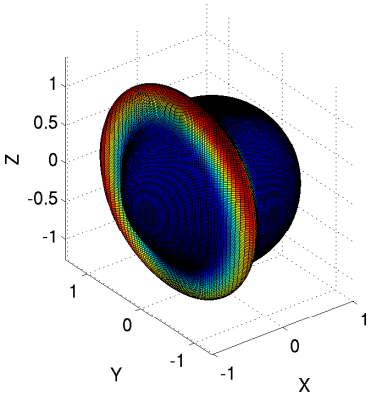
Figure 6.10: A sample recognition process of the mug.



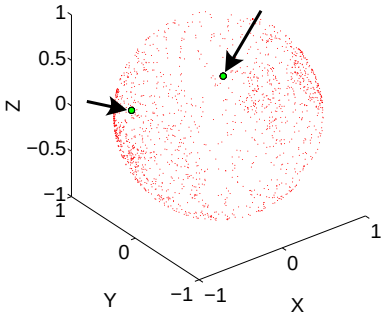
(e) Constellation two is observed



(f) The mixture density of constellation two



(g) The combined mixture density

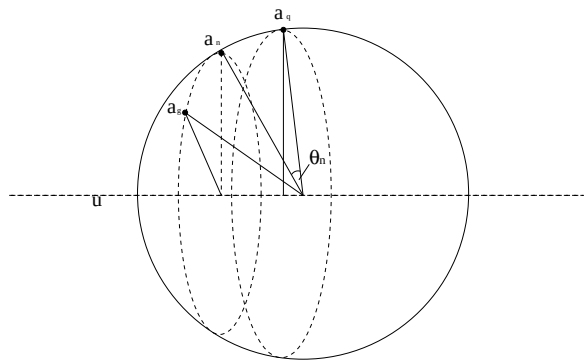


(h) The estimated aspects

(i) The nearest training image corresponding to the aspect on the left

Figure 6.10: Continued.

(a) A cup with rotational symmetry



(b) a_n is the point on the circle defined by $\mathbf{u}$ and a_g that is closest to a_q

Figure 6.11: Find the nearest ground truth aspect down to symmetry.

(down to the symmetry of the objects). When there are multiple estimated aspects, the mean of error is calculated for a single test image. We report the mean error over 100 images and 10 experiments. We compare the proposed approach with one in which no filtering is performed and with a method that guesses aspects randomly.

The recognition error is measured by the angle between the true aspects (read from the Polhemus sensor) and the estimated aspects. We can calculate this angle by:

$$\theta = \arccos(\mathbf{a}_g{}^T \mathbf{a}_q), \tag{6.2}$$

where $\mathbf{a}_g$ is the ground truth, $\mathbf{a}_q$ is the recognized aspect of the object. However, we cannot directly use Equation 6.2 to calculate the error when there are symmetries in the objects. In particular, object symmetries should not contribute to the error measure. When a symmetry exists, it is a matter of finding which of the many aspects equivalent to $\mathbf{a}_g$ is closest to $\mathbf{a}_q$.

For the cup with a symmetry about the major axis $\mathbf{u}$ in Figure 6.11a, $\mathbf{a}_n$ is the point on the circle defined by $\mathbf{u}$ and $\mathbf{a}_g$ that is closest to $\mathbf{a}_q$ (Figure 6.11b). When the top and bottom of the cup are symmetric, as well as the rotational symmetry, two circles are considered: the one defined by $\mathbf{a}_g$ and the other defined by $-\mathbf{a}_g$

For the four objects in our experiment, we assume the following symmetries:

- cup: rotation about major axis, top-bottom,
- mug (when posed with the handle on the right): top-bottom, left-right, front-back,
- block: top-bottom, left-right-front-back (all four surfaces are the same), and

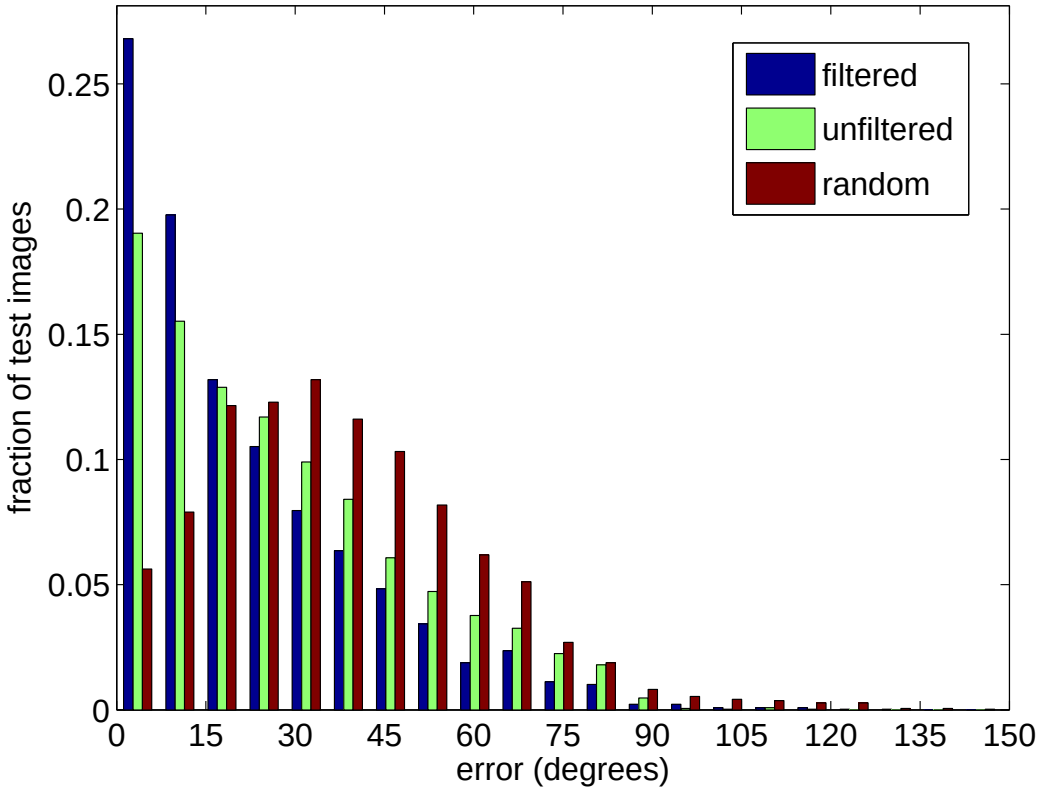


Figure 6.12: Error histograms of three methods by integrating 10 experiments and 4 objects.

- spray bottle: left-right, front-back.

Both the filtered and unfiltered methods cover 3556 out of 4000 testing images (4 objects, 100 test images and 10 experiments). However, the random method "covers" all 4000 testing images. For comparison purposes, the estimation errors of 3556 testing images are randomly selected out of 4000 for the random case, since the two sets of 3556 images are different for the filtered and unfiltered method. The error histograms for the three methods are given in Figure 6.12 by integrating all ten experiments and four objects. In this figure, we can see the distributions of three cases more clearly. For the random case, the errors distribute more like a Gaussian, with a mean around 30 degrees. For the other two cases, the trends decrease exponentially and the filtered method decreases faster than the unfiltered method.

The mean errors and the standard deviations (in degrees) for each object of the 10 experiments are shown in Table 6.1. The corresponding bar plots of the mean errors with the standard deviations are shown in Figure 6.13.

Table 6.1: Aspect recognition results

Object	Filtered		Unfiltered		Random	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Block	21.4957	1.7418	23.1824	1.1283	32.2865	1.4450
Cup	10.8053	1.0333	15.8922	3.0194	29.1538	1.8396
Mug	24.2612	2.1650	29.7212	2.4481	36.2371	1.7643
Spray bottle	30.8753	2.4017	40.1429	3.1165	54.3870	1.4899

For both methods, we can see that the errors for the spray bottle are relatively large compared to that for the other objects. The reason is as discussed before: since the shape and texture of the spray bottle are more complex than the other objects, many constellations can match fairly well on the "texture" of the labels even though they are not originally generated from those regions.

We can also see that filtering becomes useful especially for the more complicated objects. The reason is that for the simpler objects such as the cup, the pixels available for generating constellations are quite limited. This is the case because we eliminated most of the background when we selected the ROI. In this case, even without filtering, most of the constellations generated from the neighborhood images are already good enough. As the objects become more complicated, such as the spray bottle, filtering becomes necessary since the probability that a constellation is generated on the background or on the label is high. The filtering method only keeps the locally robust and globally discriminative constellations, while the other method keeps all constellations, including the ones generated on the shadows and the ones that capture straight lines.

By examining the bar plot in Figure 6.13, we can see the trend in Table 6.1 more clearly. The performance difference between the filtered and unfiltered methods for the block is small. This is because the pairwise distance between elements in a constellation that we choose is not long enough for generating discriminative features, since the block is much longer than the other objects. As a result, the constellations that pass the filter by luck are comparable in utility to the ones that do not. The performance difference between the filtered and unfiltered methods is significant for all four objects by a two-tail, paired t-test (block: $p < 0.05$; cup: $p < 10^{-3}$; mug: $p < 10^{-4}$; spray bottle: $p < 10^{-4}$). We should also note that the random guess algorithm does not perform as poorly as we may expect. However, these errors have

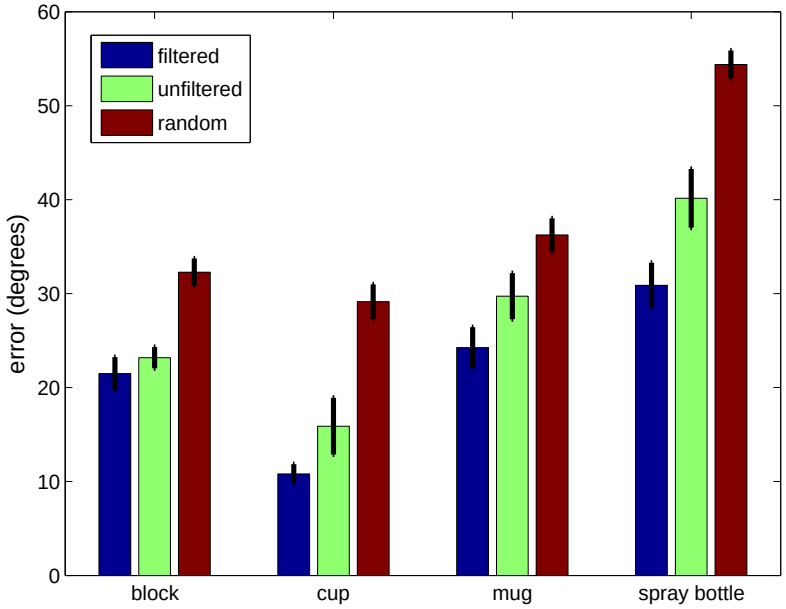


Figure 6.13: Performance comparison between the filtered and unfiltered methods. Bars represent mean performance, whiskers represent standard deviation.

also been adjusted according to the symmetric properties of the objects. In other words, the largest possible errors are bounded above.

The mean and standard deviation of the number of constellations for each object of the 10 experiments are shown in Table 6.2. Since the number of training images for each object is different, these numbers have been normalized by the total number of training images. The corresponding bar plot is shown in Figure 6.14. The number of constellations that are generated reflects the selectivity of constellations to training images. The larger the number, the more selective the constellations. This number depends on the complexity and symmetry properties of the object. The spray bottle has fewer symmetries than the other objects and the labels on each side have a complicated texture. A constellation generated on the label is very selective and only covers a small number of training images. As a result, the algorithm needs to generate more constellations to cover most of the training images. The mug also has fewer symmetries than the block and the cup. A constellation generated in an aspect tends to be observed in all of its symmetrical aspects. As a result, this constellation covers more training images than a constellation generated from an object with fewer symmetries.

Table 6.2: The number of constellations per training image

Object	Filtered		Unfiltered	
	Mean	S.D.	Mean	S.D.
Block	0.0311	0.0041	0.0342	0.0030
Cup	0.0351	0.0037	0.0344	0.0022
Mug	0.0403	0.0027	0.0341	0.0033
Spray bottle	0.0402	0.0020	0.0409	0.0027

6.5 Parameter Selection

In the experiment, there are several parameters that one can choose independently. These parameters are summarized below (the settings in the current experiment are given in parentheses):

1) The number of neighbors of an aspect used for local filtering (10). Usually the bigger this number is, the more general the constellation generated will be. However, if this number is too big, the constellation may capture some meaningless features, such as straight lines. This number should not be too small either, because we want locally robust features that can be found consistently in some number of images.

2) The percentage of the neighborhood images (70%) in which a constellation should be observed in order to pass the local filter. Ideally this parameter should be 100%, however, in reality 100% is hard to achieve due to noise. A value slightly less than 100% can speed up the training process and meet our purpose.

3) The number of edgels in a constellation, which is randomly generated from a uniform distribution between 10 and 14. The bigger this number is, the more selective the constellation is. I have done some exploratory experiments to select this parameter. When this parameter is larger than 14, it takes a relatively long time to find a constellation that can pass the local filter; when this parameter is less than 10, it takes relatively a long time to find a constellation that can pass the global filter. Since we want more variety in the set of generated constellations, I choose a range rather than a fixed value.

4) The pair-wise distance between elements in a constellation, which is randomly sampled from a Gaussian distribution with a mean of 35 pixels and a standard deviation of 10 pixels (the same parameters are used for all objects). The above values are chosen in the same way as in Chapter 5. Here, note that the block is much longer

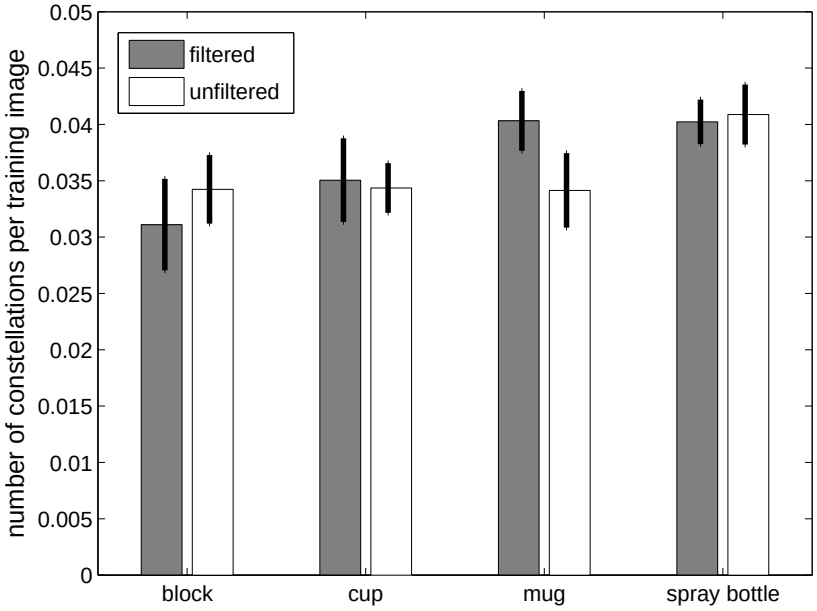


Figure 6.14: The number of constellations generated per training image for the filtered and unfiltered methods. Bars represent mean and whiskers represent standard deviation.

than the other objects in one dimension. As a result, the parameter used in the experiment does not fit the block very well and all the constellations generated for the block only cover a small corner. This in part explains the smaller difference between the filtered and unfiltered method for the block compared to the other objects, since both of these two methods generate a lot of useless features.

5) The minimal KSD with which a constellation is considered to be globally discriminative (0.6). Since we only use the top 7 thresholds (another parameter) generated from the 10 neighborhood images, the largest possible value of curve I in Figure 6.6 at the selected threshold is 0.7, which means the largest possible value of curve II at the selected threshold is 0.1. Since we use all of the training images above this threshold to train the mixture distribution, the maximum number of images that can be used is 10% of the entire training set. By observation, the biggest cluster (as far as the number of aspects is concerned) which can be clearly described by a mixture model usually contains a number of aspects no bigger than 10% of the number of training images. Note that the number of images used to train a mixture model should also be lower-bounded, since too few images can result in numerical instabilities in the EM algorithm for mixture model learning. Since the ideal case is

that a constellation can be only observed in 7 out of 10 neighborhood images and in none of the other images in the training set, this lower bound is set to 7. For the unfiltered method, the threshold for each constellation is uniformly selected so that 7 to 200 training images (about 10% of the training data) are used for the training of a mixture distribution. The above values are chosen in order to compare the performance between the filtered and the unfiltered method.

6) The number of small circle distributions in a mixture model (2) and parameter constraints for the small circle distribution ($0 \leq \tau \leq 500$, $0 \leq \nu \leq 1.3$, empirically selected). However, we are not exactly sure how many component distributions we need for different objects. There are two kinds of symmetries in the objects: rotational symmetry and reflection symmetry. The rotational symmetry has already been captured intrinsically by the small circle distribution, as well as some bipolar reflection symmetries. Empirically, it seems reasonable to use 2 small circle distributions to capture both rotational and reflection symmetries in a given object. The idea is that mixture models of similar constellations may capture more symmetries than a single mixture model. That is, in some sense, using more constellations is equivalent to using more component distributions in a mixture model. By observation, a small circle distribution with $\tau = 500$ can be used to describe the most concentrated cluster; a small circle distribution with $0 \leq \nu \leq 1.3$ can cover all the different distributions that we are interested on a unit sphere. The upper bounds of ν and τ together influence how concentrated a small circle distribution could be. I upper bound these two parameters in order to accelerate the search for the maximum likelihood parameter set.

7) The lower bound of edge magnitude included in a constellation (the same as in Chapter 5).

8) The lower bound of magnitude for a pixel being considered as the anchor point of a constellation (the same as in Chapter 5).

6.6 Conclusion and Discussion

In this chapter, I focus on the aspect recognition problem. Supervised learning is used to build a relationship between an image of a known object and the corresponding aspect from which the object is viewed. A separate test set is used to measure the performance of the algorithm. Two methods–one with filtering and one without filtering of useful constellations–are compared. The mean result of 10 experiments

shows that the proposed filtered method out-performs the method without filtering, in particular when the object shape and texture are complicated.

Compared to the other objects, the spray bottle has the highest recognition error. The difficulties for the spray bottle are the complex shape and the labels on both the front and back. For example, a discriminative constellation is hard to find to tell the difference between the top view of the spray nozzle and the narrow side view, since there are two parallel lines in both views. Also, the labels are a rich source of edges of many orientations and many constellations can be matched fairly well. One way to combat this is to increase the selectivity of constellations, for example, by increasing the total number of elements. However, when the constellations become too complicated, the efficiency of the proposed approach is reduced. The main reason is because of the trade-off between selectivity and robustness: a constellation can hardly pass the local filtering if it's too selective. Also, if constellations are too selective so that each of them can be only observed in a small subset of the training images, the proposed approach needs to generate more constellations in order to obtain a certain coverage of the training set. In addition, the matching of more complicated constellations is more computationally expensive.

By the naïve Bayes assumption, we assume that all constellations are independent of each other. This gives us a simple way to estimate the maximum likelihood aspects given that a set of constellations is observed. However, this assumption can be violated when the number of constellations becomes large. In this thesis, this issue is not addressed and is left as future work.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, I take several steps to bridge the gap between robotic grasping and vision. The contributions of this thesis are as follows:

1) A haptic-based grasp controller that incrementally displaces the fingers until a quality grasp is found. The proposed method expands the set of applicable objects to include those with concave surfaces.

2) A feature-based object classification and aspect recognition algorithm. The object classification algorithm combines the responses of multiple constellations to achieve a more robust decision. The object aspect recognition approach systematically generates locally robust and globally discriminative constellations by a two-step filtering process. These constellations are associated with mixture models of distributions, which are used for aspect recognition later on.

In conclusion, this thesis has shown that aspect can be recognized by randomly sampling constellations from example images and that the idea of maintaining only locally robust and globally discriminative constellations can lead to higher aspect recognition performance.

7.2 Future Work

The future work related to this thesis is as follows:

1) When the texture of the objects becomes complicated (such as the spray bottle), constellations of edgels introduce ambiguity between images of different aspects. We need to improve the current object recognition technique to make the recognition of a single aspect more accurate and robust. Also, constellations of edgels are

subject to scale change. One way to scale a given constellation is by scaling the relative distances between its components. Independent of the object recognition algorithm, the clustering method that uses small circle distributions can still be used for summarizing aspects with similar appearance.

2) There are several downsides to the parametric models of probability distributions that we used in this work. First, sometimes it is rather difficult to find the proper parameters of a single distribution and to find how many distributions are appropriate for a mixture model. Second, when the number of distributions in a mixture model and the number of total constellations become large, both the expectation-maximization and maximum likelihood estimation become time-consuming. Under these conditions, it is also harder to avoid local maxima. One possible solution is to begin to use nonparametric representations for these distributions.

3) The algorithm can only recognize the aspect of a given object. In order to estimate 3D orientation, we must also model the orientation of the objects within the image plane.

4) During the aspect recognition process, we make the naïve Bayes assumption. However, this assumption can be violated when the number of constellations becomes large. As a result, we need to revise our algorithm to find the maximum likelihood aspects in Equation 6.1.

5) Object identity recognition and aspect recognition can be related to grasping in several ways: constellations can suggest different grasp types and hand orientations. In addition, the act of executing a grasp can also contribute evidence for recognition and pose estimation.

Bibliography

–, 2006: Polhemus. Colchester, VT.

Ardizzone, E., A. Chella, and R. Pirrone, 2000: Feature-based shape recognition by support vector machines. *Proceedings of European Conference on Artificial Intelligence 2000 Workshop on Machine Learning in Computer Vision*, electronically published.

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

Belongie, S., J. Malik, and J. Puzicha, 2002: Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 509–522.

Bicchi, A., J. Salisbury, and D. Brock, 1993: Contact sensing from force measurements. *International Journal of Robotics Research*, **12**, 249–262.

Bingham, C. and K. V. Mardia, 1978: A small circle distribution on the sphere. *Biometrika*, **65**, 379–389.

Borst, C., M. Fischer, and G. Hirzinger, 1999: A fast and robust grasp planner for arbitrary 3D objects. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1890–1896.

Coelho, Jr., J. A. and R. A. Grupen, 1997: A control basis for learning multifingered grasps. *Journal of Robotic Systems*, **14**, 545–557.

Coelho, Jr., J. A., J. Piater, and R. A. Grupen, 2000: Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems Journal, special issue on Humanoid Robots*, **37**, 195–219.

de Granville, C., J. Southerland, and A. H. Fagg, 2006: Learning grasp affordances through human demonstration. *Proceedings of the International Conference on Development and Learning*, electronically published.

Dempster, A., N. Laird, and D. Rubin, 1977: Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, **39**, 1–38.

Detry, R. and J. H. Piater, 2007: Hierarchical integration of local 3D features for probabilistic pose recovery. *Proceedings of the Robotics: Science & Systems 2007 Workshop - Sensing and Adapting to the Real World*, electronically published.

Everitt, B. S. and D. J. Hand, 1981: *Finite Mixture Distributions*. Chapman and Hall, London.

Fergus, R., P. Perona, and A. Zisserman, 2003: Object class recognition by unsupervised scale-invariant learning. *Proceedings of Computer Vision and Pattern Recognition*, 264–271.

Fletcher, R. and M. J. D. Powell, 1963: A rapidly convergent descent method for minimization. *Computer Journal*, **6**, 163–168.

Funahashi, Y., T. Yamada, M. Tate, and Y. Suzuki, 1996: Grasp stability analysis considering the curvatures at contact points. *Proceedings of the IEEE International Conference on Robotics and Automation*, 3040–3046.

Horn, B. K. P., 1986: *Robot Vision*. MIT Press, Cambridge, MA.

Huber, M., 2000: *A Hybrid Control Architecture for Adaptive Robot Control*. Ph.D. thesis, Department of Computer Science, University of Massachusetts, Amherst.

Huber, M. and R. A. Grupen, 2000: A hybrid architecture for learning robot control tasks. *Robotics Today*, **13**, unnumbered.

Jiar, Y., X. Wu, Z. Zhao, and K. Li, 1994: A sensory robotic grasping system for object recognition and shape recovery. *Proceedings of the IEEE International Conference on Industrial Technology*, 333–337.

Lee, M. and H. Nicholls, 1999: Tactile sensing for mechatronics: A state of the art survey. *Mechatronics*, **9**, 1–31.

Mardia, K. V. and P. E. Jupp, 1999: *Directional Statistics*. Wiley Series in Probability and Statistics, Wiley, Chichester, West Sussex, England.

Mason, M. T., 2001: *Mechanics of Robotic Manipulation*. MIT Press.

Miller, A., S. Knoop, H. Christensen, and P. Allen, 2003: Automatic grasp planning using shape primitives. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1824–2829.

Park, Y. C. and G. P. Starr, 1990: Grasp synthesis of polygonal objects. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1574–1580.

Penrose, R., 1955: A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, volume 51, 406–413.

Piater, J. H., 2001: *Visual Feature Learning*. Ph.D. thesis, Department of Computer Science, University of Massachusetts, Amherst.

Piater, J. H., P. R. Cohen, X. Zhang, and M. Atighetchi, 1998: A randomized ANOVA procedure for comparing performance curves. *Proceedings of the Fifteenth International Conference on Machine Learning*, 430–438.

Piater, J. H. and R. A. Grupen, 2002: Learning appearance features to support robotic manipulation. *Proceedings of the Cognitive Vision Workshop*, electronically published.

Platt, R., A. H. Fagg, and R. A. Grupen, 2004: Manipulation gaits: Sequences of grasp control tasks. *Proceedings of the IEEE International Conference on Robotics and Automation*, 801–806.

Platt, Jr., R., 2006: *Learning and Generalizing Control-Based Grasping and Manipulation Skills*. Ph.D. thesis, Department of Computer Science, University of Massachusetts, Amherst.

Platt, Jr., R., A. H. Fagg, and R. A. Grupen, 2002: Nullspace composition of control laws for grasping. *Proceedings of the International Conference on Intelligent Robots and Systems*, 1717–1723.

Platt, Jr., R., R. A. Grupen, and A. H. Fagg, 2006: Learning grasp context distinctions that generalize. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, electronically published.

Ponce, J., S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet, 1997: On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research*, **16**, 11–35.

Pontil, M. and A. Verri, 1998: Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 637–646.

Schiele, B., 1997: *Object Recognition using Multidimensional Receptive Field Histograms*. Ph.D. thesis, Department of Computer Science, INP Grenoble, Grenoble, France.

Schroeder, W., K. Martin, and B. Lorensen, 2006: *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, Inc., 4th edition.

Smith, L. B., C. Yu, and A. Pereira, 2007: From the outside-in: embodied attention in toddlers. *Advances in Artificial Life*, Springer Berlin, volume 4648, 445–454.

Suykens, J. A. K. and J. Vandewalle, 1999: Least squares support vector machine classifiers. *Neural Processing Letters*, **9**, 293–300.

Teichmann, M. and B. Mishra, 1994: Reactive algorithms for 2 and 3 finger grasping. *Proceedings of the 1994 International Workshop on Intelligent Robots and Systems*.

Wang, D., B. T. Watson, and A. H. Fagg, 2007: A switching control approach to haptic exploration for quality grasps. *Proceedings of the Robotics: Science & Systems 2007 Workshop - Sensing and Adapting to the Real World*, electronically published.

Wang, G., Y. Zhang, and F. Li, 2006: Using dependent regions for object categorization in a generative framework. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1597–1604.