UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LEARNING VISUAL FEATURES FOR GRASP SELECTION AND CONTROL

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

DI WANG

Norman, Oklahoma

2012

LEARNING VISUAL FEATURES
FOR GRASP SELECTION AND CONTROL

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Andrew H. Fagg, Chair

Dr. Dean F. Hougen

Dr. Amy McGovern

Dr. Sesh Commuri

Dr. Joseph P. Havlicek

This dissertation is dedicated to my father, Aijun Wang, and my mother, Huiping Zheng, for encouraging my enthusiasm for robotics since I was young.

Acknowledgements

First, I would like to thank Kim Houck, Tom Palmer, and Josh Southerland for all their help and support in all these years when we were working together in the Symbiotic Computing Laboratory. I would like to thank Elizabeth Craig for lending me a hammer for data collection. I also would like to thank Dr. Vittorio Ferrari at the University of Edinburgh for sharing his visual learning code with me, although we have never met in person before. I would like to thank Dr. Dean Hougen, Dr. Amy McGovern, Dr. Sesh Commuri and Dr. Joseph Havlicek for their valuable suggestions during and after my PhD defense.

I would like to give my special thanks to my father, Aijun Wang, not only for raising me up, but also for helping me with the data collection, and for visiting me in the US even with a fracture in his leg. I would like to give my special thanks to my mother, Huiping Zheng, not only for encouraging my enthusiasm for robotics since I was young, but also for visiting me and cooking delicious food for me in the US.

Finally, I would like to give my special thanks to Dr. Andrew Fagg for mentoring, teaching, guiding, and encouraging this endeavor.

This research was supported by the Foundation Fellowship through the Graduate College of the University of Oklahoma.

Table of Contents

List of Tables

List of Figures

Abstract

J. J. Gibson suggested that objects in our environment can be represented by an agent in terms of the types of actions that the agent may perform on or with that object. This *affordance* representation allows the agent to make the connection between the perception of key properties of an object and these actions. In this dissertation, I explore the automatic construction of visual representations that are associated with components of objects that afford certain types of grasping actions. I propose that the type of grasp used on a class of objects should form the basis of these visual representations. The visual categories are driven by grasp types. A grasp type is defined as a cluster of grasp samples in the 6D hand position and orientation space relative to the object. Specifically, for each grasp type, a set of view-dependent visual operators can be learned that match the appearance of the part of the object that is to be grasped. By focusing on object parts, as opposed to entire objects, the resulting visual operators can generalize across different object types that exhibit some similarities in 3D shape. In this dissertation, the training/testing data set is composed of a large set of example grasps made by a human teacher, and includes a set of fifty unique objects. Each grasp example consists of a stereo image pair of the object alone, a stereo image pair of the object being grasped, and information about the 3D pose of the hand relative to the object. The grasp regions in a training/testing image that correspond to locations at which certain grasp types could be applied to the object are automatically estimated. First, I show that classes of objects can be formed on the basis of how the individual objects are grasped. Second, I show that visual models based on Pair of Adjacent Segments (PAS) features can capture view-dependent similarities in object part appearance for different objects of the same class.

Third, I show that these visual operators can suggest grasp types and hand locations and orientations for novel objects in novel scenarios. Given a novel image of a novel object, the proposed algorithm matches the learned shape models to this image. A match of the shape model in a novel image is interpreted as that the corresponding component of the image affords a particular grasp action. Experimental results show that the proposed algorithm is capable of identifying the occurrence of learned grasp options in images containing novel objects.

Chapter 1

Introduction

J. J. Gibson (1977) suggested that objects in our environment can be represented by an agent in terms of the types of actions that the agent may perform on or with the object. This *affordance* representation allows the agent to make a connection between the perception of key properties of an object and these actions. The key to the affordance idea is that it simultaneously captures properties of an object, as well as the morphology and capabilities of an agent. For example, a chair affords sitting to a human, but not to a robot with wheels. How does an agent come to represent such affordances? One can either curiously explore the world, or observe other agents with similar capabilities interacting with the world. Smith et al. (2007) have shown that during early development, infants not only curiously explore the world by themselves, but also observe their parents interacting with the world. In particular, they pay more visual attention to their parents' hands than the other parts of the body. One possible explanation is that this kind of observation helps infants to form representations of how objects and hands interact during grasping and manipulation.

Computationally, affordances can be seen as providing a menu of actions that can be applied by an agent to a target object (Gibson, 1966, 1977). These actions are task neutral; an agent can further prune this set of actions given the context of a larger task. Once the agent has determined which affordance to use, it is important to properly instantiate this affordance with a real world object. One solution is to use available visual cues to help with the selection and instantiation of an affordance before any actions are made. These visual cues may indicate the possible contact

locations and other relevant properties of the object such as pose, material and weight.

How does an agent learn the association between visual cues and affordances? The work of Pereira et al. (2010) suggests that the growth of visual and grasping skills should be intertwined from the beginning of an agent's "life." Their experiments also show that when infants are confronted with novel objects, they tend to grasp and rotate these objects in order to observe them from different viewing angles. However, not all viewing angles are equally important: the infants tend to pay more visual attention to the views of an object that contain planar surfaces. One possible explanation is that these "planar views" afford more stable visual features as the object is rotated within a small range.

Is it possible for a robot to learn grasp affordances and associated visual representations automatically as what we have done as infants? In this dissertation, I provide a computational account of how such a representation for grasp affordances may arise. Specifically, I propose an algorithm that associates grasps directly with object parts without higher-level object recognition. In this way, a learned affordance can possibly be generalized to novel object that shares some parts with a known object. For example, the top parts of a mug and a glass are visually similar to each other, and both of them afford a ball grasp around their rims. If we can associate the set of visual features generated around the rims with this top grasp, we will achieve a more compact and generalizable model. Based on the above idea, I propose a learning approach that allows a novice robot to automatically learn grasp affordances and associated visual features by observing a human teacher and to apply learned models to real-life grasping tasks. The association between grasps and visual features are learned simultaneously using both grasping and visual experience. In particular, this algorithm uses learned categories in the grasping domain to drive the learning of categories in the visual domain. The training/testing data set is composed of a large set of example grasps made by a human teacher, and includes a set of fifty unique ob-

jects. Each grasp example consists of a stereo image pair of the background, a stereo image pair of the object alone, a stereo image pair of the object being grasped, and information about the 3D pose of the hand relative to the object. The grasp regions in a training/testing image that correspond to locations at which certain grasp types could be applied to the object are automatically estimated.

The key steps in this approach are as follows. First, the algorithm clusters the example grasps into a compact set of *proto-grasps* using the approach of de Granville et al. (2006). The entire set of proto-grasps defines a *grasp affordance model*. This process takes into consideration both the hand positions and orientations of these example grasps relative to the object. Second, the set of individual objects are partitioned based on the similarity between their grasp affordance models. These partitions define *object categories*. Third, for each object category, the algorithm learns a unified grasp affordance model by using example grasps from all objects in this category. Each cluster in this unified grasp affordance model defines a *grasp type*. Fourth, the algorithm learns visual features that are predictive of these grasp types. For each grasp type, the proposed algorithm extracts a set of image fragments from example images based on locations at which object-hand contacts occur. These image fragments contain similar looking object components corresponding to different view angles. Then, for each group of image fragments, the proposed algorithm learns a set of view-dependent visual features. Fifth, given a query image that contains a novel object, the proposed algorithm identifies graspable locations by matching visual features corresponding to different grasp types and viewing angles. Finally, the hand orientation associated with the best matching visual feature is used together with the identified graspable location to cue a grasp.

The experimental results show that objects with difference appearances can be robustly categorized by their grasp affordance models. The different grasp types in an object category can be used to drive the learning of meaningful visual models

that capture partial object shapes. These visual models, which describe object parts, generalize well to novel objects with similar partial shapes. A matched visual model can suggest an approximation to how the hand should be positioned and oriented relative to the object in novel scenarios containing novel objects. In many situations, the grasp types and corresponding hand poses are estimated accurately by matching 2D shape models. However, for certain visual aspects, the hand orientations are not estimated reliably.

Chapter 2

Related Work

2.1 Robotic Grasping

In order to manipulate objects in the world, a robot needs to first grasp these objects. The goal of robotic grasping is to find a set of contact points on the object in order to achieve a grasp that satisfies certain quality measures. The quality of a grasp can be measured by its stability (force/form closure, Mason and Salisbury, 1985; Bicchi, 1995), task compatibility and adaptability to novel objects (Sahbani et al., 2012). Robotic grasping approaches can be divided into analytical and empirical approaches.

Analytical approaches aim to find a set of quality grasps by considering the geometric, kinematic and dynamic formulations of the hand and the object to be grasped. These 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; Miller et al., 2003). Bekey et al. (1993) develop a task-oriented grasp planner. This planner is based on the assumption that the robot has knowledge about its own hand, the target object geometry, the current task, and human grasps. In this approach, a set of canonical grasp types and a set of geometric primitives are first defined. Then, given the current task description and some geometric primitives contained in a target object, the algorithm selects a subset of the canonical grasp types that is appropriate for the current task. Borst et al. (1999) propose a grasp planning algorithm that applies to arbitrarily shaped

3D objects. Assuming the 3D model of an object and a large set of possible grasp locations, their algorithm quickly finds a small set of feasible candidate grasp locations on the object, based on a static grasp stability measure (Ferrari and Canny, 1992). These candidate grasps are further filtered by a computationally efficient hierarchical approach, which is composed of simple heuristics, such as kinematic constraints of the fingers. In the work of Miller et al. (2003), objects are simplified into a collection of geometric primitives, such as spheres, cylinders, cones and boxes, based on their exact 3D mesh models. Each of the primitive shapes is associated with a set of pre-defined hand poses and finger configurations. These possible grasps are further evaluated using a simulation environment.

In the above algorithms, the modeled object geometry is used to assess the quality of many potential grasps before one is selected for execution. In an alternative branch of the analytical approaches, one could 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 subsequently Coelho and Grupen (1997), introduced methods in which the local surface normal for each of several contacts is first estimated. Based on this information, contact displacements are computed that followed the negative gradient of a cost function. The cost functions are such that their minima correspond to a quality grasp of the object. In the case of Teichmann and Mishra (1994), this cost function is based on the area of the triangle formed by three contact points. In the case of Coelho and Grupen (1997), two cost functions are used: one that describes the net force applied to the object by the set of contacts, and another that describes the net moment applied by the same contacts. Then, the algorithm switches between two controllers that iteratively minimize these two cost functions, respectively. Platt et al. (2002) combine the actions of the two controllers through a nullspace operation that favors the actions of the force controller over those of the

moment controller. This work has showed 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 assumption of local convexity. Wang et al. (2007) address the concavity issue by introducing a second force controller that makes the assumption of local concavity. A meta-level controller then switches, on a per-contact basis, between the convex and concave control actions as a function of the estimated curvature of the object.

The haptic-based approaches are local in the sense that they do not assume global geometry of the object. Due to this fact, these approaches inevitably suffer from the problem of local optima. As a compensation, a robot can use vision or range data to preshape its hand into a position and orientation in which the grasp controller is more likely to succeed. Jonquières et al. (1999) propose an object-centered approach for vision-guided robotic grasping. The algorithm matches a 2D image against the projection of the CAD model of an object in order to achieve object recognition and pose estimation. Given the recognized object location and orientation, a grasp can be planned accordingly. Kragic and Christensen (2002) extend this approach by using multiple images, which results in more robust object recognition and pose estimation results. This extension falls within the framework of visual tracking. First, the authors propose a 2D visual tracking approach based on some visual cues of the object, such as color, texture, correlation and motion between image frames. A consensus of these visual cues is achieved through a voting scheme. Second, the authors propose a 3D visual tracking algorithm that assumes the 3D model of an object, which is in spirit similar to the approach of Jonquières et al. (1999). Finally, the authors argue that the 2D tracking algorithm can be used to initialize and enhance the 3D tracking process. Since the target object being grasped is recognized and aligned with a predefined 3D model as a whole, the above approaches will not generalize to novel objects.

An alternative approach is to cue grasps with partial visual information, and without higher-level object recognition. Stanley et al. (2000) use 2D vision to guide a parallel gripper for grasping planar objects. The algorithm first efficiently samples a set of candidate grasp points on the object boundary by incrementally increasing the sampling resolution. Then, the algorithm evaluates the quality of these candidate grasp points using a cost function. For a pair of grasp points (for a parallel gripper), the cost function measures the curvature near the grasp points, the parallelism between the two grasp points, and the distance between the grasp points to the center of mass. Blake et al. (1993) propose a similar approach. In addition, their algorithm exploits the symmetry property within 2D object contours. The proposed algorithm can efficiently find these symmetries in 2D object silhouettes, and thus, a set of candidate grasp points is calculated based on these symmetries accordingly. In the work of Ikeuchi et al. (1986), a set of quality grasp points is calculated based on both photometric stereo and range data. Given the surface normals of an object that are estimated by photometric stereo, quality grasp points are selected as those on the object contours that oppose each other with overlapping surface normals.

Rao et al. (1988) aim to develop an algorithm that can grasp 3D objects in more generic scenarios. First, the 3D volumetric approximation of an object is obtained by 3D range data. Then, this 3D volumetric model is decomposed into some predefined primitive shapes. For each primitive shape, a set of candidate grasps is selected accordingly from a list of heuristically defined grasp types. Dufournaud et al. (1998) propose a grasp synthesis algorithm based on stereo vision. In this work, the shape of an object part being grasped is approximated by conics. This is based on the observation that many objects in our environment contain curves and rotational symmetries. Then, an ellipsoid is fitted to each conic, whose grasps can be calculated efficiently.

One disadvantage of the analytical approaches is the computational complexity. This is due to the fact that a large set of grasps are sampled and evaluated before

a feasible grasp is actually selected for execution. In order to reduce computational complexity, some of these algorithms make simplifying assumptions about the geometries of the robotic hand and the object being grasped, and are based on heuristics that are elaborated by the designer. Also, these approaches usually treat each object individually, which makes the previously found grasp solutions difficult to be reused on similar objects. Instead, some recent works on robotic grasping use empirical approaches (e.g., Ying et al., 2007; Saxena et al., 2008; Goldfeder et al., 2009; Kjellström et al., 2011; Aleotti and Caselli, 2011). Empirical approaches construct models from data gathered in the context of grasping. These approaches formulate the grasp synthesis problem as a learning problem, that is, a robot learns how to grasp a certain object by observing either image features directly (Ying et al., 2007; Saxena et al., 2008; Goldfeder et al., 2009), or human demonstration (Kjellström et al., 2011; Aleotti and Caselli, 2011).

The work of Saxena et al. (2008) aims to grasp novel objects that are seen by a robot for the first time. The proposed algorithm classifies each pixel in a given image as either graspable or not. In order to generalize to novel objects, this classifier is trained on a large number of synthetic images of different object classes and image conditions (such as object pose and lighting condition). In the testing process, their algorithm takes two or more images of a novel object by using a camera mounted on a robot arm. For each image, their algorithm first predicts the graspable points. In order to triangulate the 3D grasp location by using two images, their algorithm also finds the correspondence between the grasp points in each of the image. Given the uncertainty in their system (for example, the modeled camera/arm position may not be accurate, and the grasp points may be mismatched to each other), the authors propose a probabilistic model that maximizes the probability of a location in 3D space being a graspable point given the set of grasp points in each 2D image. Once the best grasp point in 3D is found, the robot will move its hand (in their case, a

parallel gripper) to that location and then close the hand to perform a grasp. Their experiments on a physical robot show promising results that novel objects can often be grasped, even in cluttered scenes without pre-defined 3D models of the objects. This algorithm is further improved by Balaguer and Carpin (2010a). The improved algorithm uses principal component analysis (PCA) to reduce the dimensionality of the visual descriptor for each pixel in the image, which results in a much more efficient algorithm for both training and testing. Balaguer and Carpin (2010b) further propose an algorithm to estimate the orientation of the hand that can be used to grasp an object assuming that the grasp points have already been estimated. However, they only consider a limited set of possible object orientations and the different grasp points are not differentiated based on their functions.

Goldfeder et al. (2009) propose an algorithm that aims to solve the problem of grasping novel objects by using partial 3D scans. A *view sphere* is a unit sphere in the object-centered coordinate frame, and each point on the view sphere corresponds to a particular viewing angle of the object. Instead of using a single view of an object for matching, their algorithm combines visual features from a set of depth images within a spherical cap on the view sphere of an object into a single feature (so-called *cap descriptor*). By using a set of images surrounding a view point to construct a feature descriptor (instead of just one image), the matching between objects is more robust and less ambiguous. In this approach, partial matching is allowed for a given object, since the cap descriptors only capture partial views of an object on the view sphere. The authors test this method on the Columbia Grasp Database (CGDB) in simulation. The CGDB is a collection of form closure grasps on the object models contained in the Princeton Shape Benchmark (PSB). First, the algorithm calculates cap descriptors on some canonical viewpoints of a set of training objects in CGDB and stores them as a database. Then, for a test object, the algorithm matches its cap descriptor corresponding to a single viewpoint to the ones in the database. The

best matched cap descriptors in the database give rise to the object identity for the test image. Next, the algorithm aligns the partial model of the test object to the model of the matched training object. This gives the pose of the test object relative to the object models in the database. Given the N best matching objects in the database, the candidate grasps associated with these objects are ranked based on their generalizability across these N objects. The top ranking candidate grasp is selected as the recognized grasp for the test object. The simulation experiments show promising results. However, the authors only test this method in simulation and they do not deal with background clutter in natural scenes.

Aleotti and Caselli (2011) propose an empirical approach for part-based semantic robot grasping, which aims to facilitate task-oriented manipulation. They first decompose the 3D mesh model of an object into parts topologically by using a *Reeb graph*, which is a data structure that represents the skeleton of a geometric model. Given the 3D mesh model of an object, the set of feasible grasps on this object is demonstrated by a human teacher in an interactive 3D virtual environment with a data glove. For a given grasp, the corresponding object part is detected as the one that contains most of the contact points during human demonstration (by collision detection). Given the 3D mesh model of a novel object, they first compute its Reeb graph. Then, the object recognition process is a graph matching process. An object is recognized if its Reeb graph matches one of those in the database in structure. Once a good match of the test object is found in the database, the corresponding grasps are recovered accordingly.

2.2 Human Visual Processing for Grasping

Humans are capable of recognizing objects robustly. The human visual processing systems can be divided into two major parts: the dorsal pathway and the ventral

pathway (Milner and Goodale, 1995, 2008). The dorsal pathway is responsible for extracting useful visual information to facilitate action, such as grasping (vision for action); while the ventral pathway is responsible for recognizing the identity of objects (vision for recognition). Although these are two distinct processes, the dorsal and ventral pathways must work together to solve many grasping problems (Jeannerod, 1997; Graf, 2006). Some of the previous works focus on the separate development trends of these two systems in the early ages of children, and how these two systems cooperate with each other (Johnson et al., 2001; Mareschal and Johnson, 2003; Street et al., 2011).

Street et al. (2011) examine toddlers' development in visual and motor skills through a simplified version of the *posting task*. In this task, a toddler is required to grasp and insert a planar object into a slot. The experimental results show that only toddlers older than a certain age can accomplish this task (about 24 months old). However, even 18-month-olds can successfully orient their empty hands to align with the slot. This suggests that the observed developmental change is a reflection of the ability of aligning object with object, instead of aligning hand with object (the slot). In the task of inserting an object into the slot, the toddler not only needs to align the hand with the object, but also needs to relate this object with a second object (the slot). The hand-object task is generally believed to involve the dorsal pathway; while the object-object task involves the ventral pathway, since the properties of both objects need to be recognized in order to do the alignment. Therefore, these results suggest that the developmental changes in the dorsal and ventral pathways are intertwined. These results also suggest that one of the important skills that children develop during their early age is to preshape the hand and to adjust the hand orientation relative to the object to be grasped before actual contact. One interesting question during this developmental process is how children filter out irrelevant visual information and only focus on object properties that are useful for action.

Smith et al. (2007) have done some experiments on visual attention in infants. Unlike most of the other work with third person cameras tracking the actions of infants, they use a camera attached to the forehead of an infant to see what the infant sees. In their experiment, the parent and the infant interact with each other by playing with some toys on a table. In addition to the infant mounted camera, a forehead camera is also mounted to the caregiver. The field of view of a forehead-mounted camera is used to approximate what the parent or the infant sees, giving a first-person view of the interaction. From their experimental results, the first-person views of the parent and the infant are very different. Compared to the parent, the infant's view tends to focus narrowly on the current object of interest. This is not only because the infant is closer to the target object due to the length of the arm, but the infant also actively moves the object of interest closer to their eyes. As a result, the object of interest occupies a large portion of the field of view, and therefore, background visual information is filtered out. The infant also changes the view more dynamically than the parent, in order to track the object of interest closely. All the above results suggest that the infant actively constrains the amount of information that is extracted at a time. This property could simplify the process of learning of visual and motor skills. These results provoke the question that whether a robot can learn the association between vision and grasp actions in a similar manner. Particularly, a robot should "zoom in" on the part of the object that is related to the current grasp action, which may be made by either the robot itself or a human teacher.

One reason that young children can develop their visual and motor skills rapidly in a short period of time is their ability to generalize to novel objects with similar properties. Human adults have the ability to recognize novel objects that share a few common geometric components with previously known objects (Biederman, 1987; Hummel and Biederman, 1992). The representation of objects in the human brain

can be separated into two key components: the shape of individual parts and the geometric configuration between these parts. Kotovsky and Gentner (1996) show that younger children tend to recognize objects based on the shape of a few object parts, while older children tend to pay more visual attention to geometric relations between these parts. Augustine et al. (2011) further shows that these abilities develop in the early stage of infancy in a very short period of time, and are separable in the infants' early development.

So how does an infant learn the association between visual cues and actions? Pereira et al. (2010) suggest that the growth of visual and grasping skills are intertwined during the early ages of infants. Their experiments show that when infants are confronted with novel objects, they tend to grasp and rotate these objects in order to observe them from different viewing angles. However, not all viewing angles are equally important: the infants pay more visual attention to the views of an object in an upright orientation that contain planar surfaces. One possible explanation is that these "planar views" afford more stable visual features as the object is rotated within a small range. The planar view of an object in an upright orientation provides a stable reference that helps an infant with relating other views of the object and constructing a coherent multi-view model.

2.3 Developmental Robotics

Developmental robotics is a branch of robotics research that focuses on the problem of how a robot can develop its sensory and motor skills incrementally through experience with the world. This experience may come from its own, or by observing other agents (Asada et al., 2009; Argall et al., 2009). The concept of developmental robotics is explained in Weng et al. (2001). In this paper, the authors discussed the distinction between machine learning and robotics mental development. In "machine

learning," the human designer manually programs an algorithm whose parameters can be changed during execution. In developmental robotics, the human designer designs an algorithm that allows the robot to improve itself on-the-fly, either through its own experience or by interacting with other agents. This autonomous mental development process from "infancy" to "adulthood" resembles that of human development. A developmental robot is more generalizable to new environment and tasks, since the specific tasks are not defined in advance. They also explicitly argue that this development should be from simple to complex. For example, in order to manipulate an object in a larger task, the robot should learn how to grasp it first. Finally, the authors argue that developmental robotics and cognitive psychology can benefit from each other: the latter inspires the former (Brooks et al., 1998; Asada et al., 2001), while the former provides a computational testbed for verifying principles of the former (Sporns, 2003).

Developmental robotics itself covers a wide range of robotics research areas, such as social interaction, sensorimotor control, categorization, value system, developmental plasticity, motor skill acquisition and morphological changes (Lungarella and Metta, 2003). Given the scope of this dissertation, I focus on works that involve human demonstration and the development of grasping skills. A more comprehensive survey of developmental robotics can be found by Lungarella et al. (2003). Some of the previous works explore developmental robotics in the domain of robotics grasping (Metta and Fitzpatrick, 2003; Stoytchev, 2005; de Granville et al., 2006, 2009).

Instead of using heuristics or detailed models that pre-define a set of "good" grasp points on an object, these approaches ground behavior in the real experiences of the agent or nearby agent. Particularly, an agent curiously explores the world or observes other agents interacting with objects. Metta and Fitzpatrick (2003) propose an approach that allows a robot to learn visual and motor skills simultaneously. In this work, the robot learns a vision-action association by curiously exploring the

world with simple actions and observing the consequences. Similarly, in the work of Stoytchev (2005), a robot learns a set of affordances during a babbling stage by exploring a random series of actions on an object and observing the result. The set of actions available to the robot is pre-programmed by the author. As a result, the robot learns a set of so-called *binding affordances*). The term "binding" is defined as a series of actions that allow a robot to attach objects to its body and to control them as an extension of its body. The author uses this term in order to avoid from committing to the purposeful act of grasping. Simulation experiments show that this behavior-grounded approach can be used by a robot to autonomously learn the binding affordances of different objects. One shortcoming of this approach is that the set of actions available to a robot is pre-programmed and may not be thorough enough to discover all possible binding affordances. Also, the learned binding affordances are object specific and can not be generalized between objects with similar parts.

Instead of learning grasp affordances from self-experience, an agent can also learn by observing other agents grasping objects. De Granville et al. (2006, 2009) propose an algorithm that allows an agent to learn object specific grasp affordances by observing a human teacher grasping an object in some canonical ways. Each grasp sample is a point in 6D pose space in an object-centered coordinate frame. The set of grasp samples is explained by a mixture of probability distributions, with each individual probability distribution corresponding to a single cluster. An example is shown in Fig. 2.1. The object used in this example is a cylinder, as shown in Fig. 2.1(a). Given this object, a human teacher wearing a dataglove demonstrates four types of grasps: one for each end, and two along the major axis of the cylinder (corresponding to "overhand" and "underhand" configurations). The grasp positions and orientations of the grasp samples are shown in Fig. 2.1(b) and (c) respectively. The position of a grasp is chosen to be roughly a point between the thumb and index fingers of the human teacher, which is a fixed translation from the sensor mounted on the data

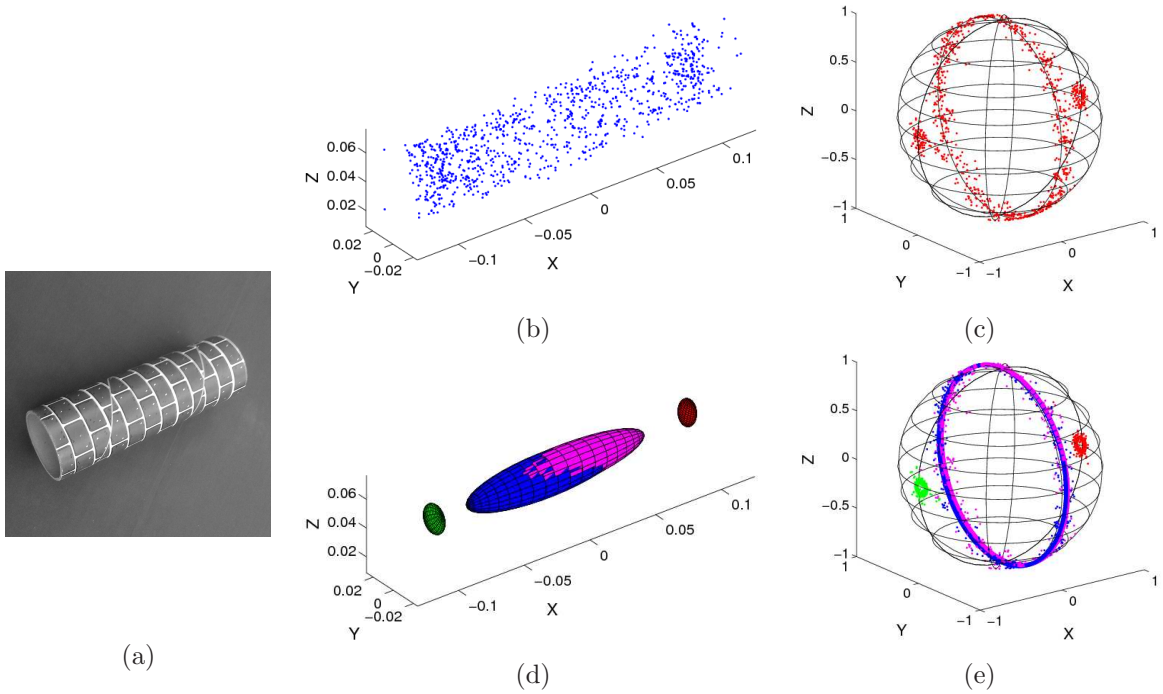


Figure 2.1: The training examples and learned affordance model for the cylinder within an object-centered coordinate frame (a figure from de Granville, 2008). (a) The cylinder; (b) The position of the hand; (c) The orientation of the hand; (d) The position component of the learned affordance model; (e) The orientation component of the learned affordance model.

glove. The points that have extreme x coordinates in Fig. 2.1(b) correspond to the grasps on the two ends of the cylinder with palms perpendicular to the major axis of the cylinder.

The hand orientations that correspond to the hand positions in Fig. 2.1(b) are shown as the two concentrated small circles of points in Fig. 2.1(c). In Fig. 2.1(c), the center of the unit sphere corresponds to the center of the object, and the vector connecting a sample point and center corresponds to an orientation perpendicular to the palm. This visual representation aliases the hand rotations about the line perpendicular to the sphere surface. We lose one degree of freedom here for 3D rotation since we only use S2 (3D unit sphere) for illustration. For the grasps on the ends of the cylinder, the only degree of freedom of the hand is exactly the rotation about

the axis perpendicular to the palm. This explains why the orientations of these grasp samples apparently collapse into two concentrated clusters in Fig. 2.1(c). The majority of points in the middle of the elongated shape in Fig. 2.1(b) correspond to the grasp samples along the major axis of the cylinder, in either the overhand or underhand configurations. The orientations for these grasps correspond to the cluster of points around a great circle of the sphere in Fig. 2.1(c). Again, the two clusters that correspond to overhand and underhand configurations appear to collapse together since these two grasps are related by a rotation about the axis perpendicular to the palm. In Fig. 2.1(d) and 2.1(e), the above four clusters of grasp samples are captured by four probability distributions. A distribution in the position space and a distribution in the orientation space (shown in the same color) together give a joint distribution in the pose space. For the position component, the authors use multivariate Gaussian distributions. Intuitively, the volumes of the ellipsoids in Fig. 2.1(d) describe the variations for each cluster. For the orientation component, they use distributions that capture the symmetry of rotations. Specifically, orientations with a unidirectional mean direction is captured by a Dimroth-Watson (DW) distribution; orientations with a rotational symmetry is captured by a girdle distribution (Mardia and Jupp, 1999; Rancourt et al., 2000; Rivest, 2001). In this example, all four grasp types are captured by girdle distributions since they are rotationally symmetric about the length of the cylinder. Although the hand positions that correspond to the overhand and underhand grasps coincide, their orientations are very different (180 degrees rotation about the axis that is perpendicular to the palm). This explains why the algorithm chooses two distributions in Fig. 2.1(e) (blue and magenta), although the visualization aliases them. Once the object position and orientation are known, each of these grasp clusters can be used to parameterize a reach controller. This reach controller brings the robot hand into a spatial relationship with the object, which is specified by the learned grasp affordance model.

2.4 Object Recognition

Object recognition deals 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. Such a representation can be the geometric configurations of individual features (such as pixel gradients), or the distribution of gray or color values of sub-regions of a given image. Ideally, these features should be invariant even when scale, pose or light conditions change. For detection, the 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. For learning, a system should be able to learn useful features automatically in an unstructured environment.

There are many different ways to categorize object recognition algorithms. One way is to divide these algorithms into texture-based[1] recognition and shape-based recognition (Opelt et al., 2006). Shape-based object detection and texture-based object detection can be considered as two methods that operate at different frequencies on a given image. The former focuses on the gross shape of a target object, while the latter pays more attention to the finer details. For robotics applications such as grasping, shape-based features are more relevant since the grasp types afforded by an object depend more on its shape than its texture. However, as far as object identification, texture-based features are useful, since they usually provide some unique features that differentiate one object from the others.

One way to recognize a known object is to match individual visual features. However, this method usually does not work well in reality since the images corresponding to the same object may vary and it is difficult to find exact matches for individual visual features. One way to compensate for this is to model an object as a class of

[1] In the scope of this dissertation, "texture" refers to the appearance texture.

visual features and the object recognition problem as a classification problem. The key idea behind this method is that a classifier allows us to represent volumes in the feature space that correspond to a target concept (in this case, a particular object). Pontil and Verri (1998) use classifiers to recognize 3D objects from 2D appearance. They first transform color images into gray-level images and then reduce the image resolution to 32 by 32 by averaging neighborhood pixels. They then reshape the 32 by 32 image pixel matrices into vectors. As a result, each image is represented as a point in high dimensional Euclidean space. A set of images is taken for each object from different viewing angles. Therefore, each object is modeled as a set of feature vectors. Images from a pair of objects are classified by using linear Support Vector Machines (SVMs) in the vector space without differentiating their poses. From their experimental results, SVMs are well suited for appearance based recognition. However, their approach suffers due to requiring a large amount of memory when the number of objects increases. This is because their algorithm needs to compute one SVM to distinguish each pair of objects.

2.4.1 Texture-based Object Recognition

A family of robust object detection algorithms makes use of distributions of intensity gradients around small local areas in a given image. These distributions represent the overall structure of intensity gradients, which are more robust than gradients of individual pixels. One representative algorithm is the *scale invariant feature transform* (SIFT, Lowe, 2004). SIFT first detects a set of salient points (so-called *keypoints*) and then calculates feature descriptors around each keypoint.

The goal of keypoint detection is to find the locations in a given image that are robustly identifiable in other images of similar perspectives of the same or similar objects. Specifically, SIFT aims to identify keypoints that are repeatable when scaling and in-image rotation exist. First, scale space images are generated from the original

image. Then, the Difference of Gaussian (DOG) responses are computed from the two nearby scales in this scale space. These DOGs respond most strongly to areas of high contrast (dark surrounded by bright, or vice-versa). The most stable locations are identified by the spatial and scale extrema in the DOG images. These stable locations are termed "keypoints" by (Lowe, 2004). The gradient (orientation) of a pixel in an image is defined as the direction in which the image intensity changes most quickly. A canonical orientation is assigned to each keypoint by using the dominant orientation of pixels within a surrounding region. In order to achieve rotation invariance, the orientation of the surrounding features are relativized to this canonical orientation.

Each keypoint corresponds to a 2D location in a given image. This location defines the origin of a 2D coordinate frame, whose orientation and scale are determined by the canonical orientation and scale of the keypoint. The SIFT descriptor is a vector that represents the appearance of the patch surrounding the keypoint. First, a patch is divided into subregions, and for each subregion, a histogram of local orientations is computed. Each bin of this histogram counts the number of pixels with gradients in a particular range of orientations. In practice, a shift by an individual pixel in either direction does not substantially change the orientation histograms. This property makes SIFT less sensitive to variations in registration of the patch location during the matching process. Finally, the values of all bins for each histogram are appended together into a single feature vector. Lowe shows experimentally that a descriptor of length 128 gives high performance for feature matching. This corresponds to 4×4 patches and 8 bins in each patch. In order to reduce the effect of illumination changes, the descriptor vector is normalized to unit length.

SIFT features are capable of identifying fine details within images, including both shapes and textures. This type of visual feature is considered to be very discriminative (Mikolajczyk and Schmid, 2005) and is widely used in object recognition related tasks (Rothganger et al., 2006; Romea et al., 2009). SIFT features are also robust to

orientation, scale and lighting changes. However, since SIFT features are calculated based on local appearance of an image, it is sensitive to object texture. Also, not all objects have distinguished keypoints (for example, all the keypoints on a circle give rise to very similar SIFT descriptors), and using keypoints alone sacrifices the shape information available in smooth portions of object contours Belongie et al. (2002).

2.4.2 Shape-based Object Recognition

As a contrast, another family of object detection algorithms pays more attention to object shape. This is more appealing to robotics grasping applications, since one would like a learned model that is generalizable to objects with similar shapes. Ardizzone et al. (2000) proposed a feature-based shape recognition algorithm. The training set of images contains single objects of some basic geometric shapes (such as the cube, cylinder and cone). The training images are first converted into grey-scale images. Then, each grey-scale image is represented as a set of vectors with each of the vector corresponding to a single row of pixel values of the image. Then, this set of vectors is transformed into a new set of uncorrelated zero-mean vectors by using a Karhunen-Loève transform. The shape of each object is described as a feature vector that consists of eigenvalues of the covariance matrix computed from the new set of vectors. This *eigenvalue vector* is directly related to the change of surface normals of a given object and robust to rotations or scaling, and to some extent, change in lighting conditions. Furthermore, the eigenvalue vectors calculated from objects with similar shapes tend to form clusters in the feature space. SVMs are trained in order to classify feature vectors that correspond to different shapes. Given a test image that contains similar geometric shapes as the ones in the training set, the algorithm first scans the image with a fixed window. The region of interest (ROI) is determined by using the window location that maximizes the correlation between the feature vector calculated on this window and one of the prototype feature vectors.

Then, the object contained in the ROI is classified by using the learned SVMs. The experimental results are promising and some known shapes can even be recognized in natural scenes with partial occlusion. However, since this algorithm is trained on single object views, there are some ambiguities in recognition (such as the side view of a cone and a triangle). This algorithm fails in presence of complex textures on the object surfaces.

Belongie et al. (2002) propose a method that aims to measure the similarity between shapes and use it for object recognition. Assuming there are some similarity between two shapes, they first propose a method that automatically finds the corresponding points on these two shapes. Given these corresponding points, they then calculate an aligning transform between these two shapes. In order to solve the correspondence problem, they propose a novel descriptor, namely, the *shape context*. This descriptor first randomly samples a set of n points that lie on the edges exhibited by an image. For each point in this set, the shape context descriptor of that point captures the distribution of the remaining points relative to it. More specifically, for each sampled point, they first calculate a set of $n-1$ vectors pointing from this point to all the other sampled points. Then, a coarse histogram is calculated to describe the distribution of this set of $n-1$ vectors in log-polar space. The corresponding pixels of two images are those that have similar shape contexts. Then, a non-rigid transformation that relates the shapes in two images is estimated by using these corresponding pixels. The dissimilarity between the two shapes is measured by the matching errors of all points sampled on these two shapes, together with a term that reflects how much deformation is needed to align these two shapes (which is measured by the bending energy contained in the aligning transformation).

During the testing process, the dissimilarities between a novel image of a known shape and all the images in the database are calculated, and the one with the least dissimilarity is considered to be the best matched image that contains the same

shape. The authors evaluate the performance of the shape context descriptor in object recognition extensively on several data sets, including silhouettes, trademarks, handwritten digits, and 3D objects. The proposed method performs well as far as retrieving the most similar shape from the database for a given test shape. However, the proposed algorithm assumes that the shape to be matched is perfectly segmented from the background clutter. This assumption is generally not true in natural image matching tasks, such as recognizing an object in order for a robot to grasp. The shape context descriptor is invariant to in-image translation and scaling. However, the descriptor used in the experiments is not invariant to in-image rotation since it uses absolute coordinate frame for computing the shape context at each point. One can compensate for this by using local coordinate frame associated with each sample point.

Forssén and Lowe (2007) use an affine invariant shape descriptor for maximally stable extremal regions (MSERs). A set of binary images are generated by applying different thresholds to the original image. The set of extremal regions are the connected black or white regions in these binary images, and MSERs correspond to the regions that are stable across a set of thresholds. These MSERs are used to define SIFT key points. Instead of using grey-scale images to calculate feature vectors, as in SIFT, MSER-SIFT uses the binary MSER itself to calculate the SIFT descriptors. Since MSERs usually capture the shape information of an image patch, the SIFT features calculated on these regions depend more on the shape of the patch rather than its texture. Experimental results show that MSER-SIFT features are often more robust for feature matching on 3D scenes and images with large illumination changes than SIFT features. However, SIFT features perform better for images that contain planar and parallax-free scenes. The reason is that SIFT features incorporate more context of an interest point than MSER-SIFT, and this context is preserved only when the scene is planar. The results also show that when large scale changes exist,

the MSER-SIFT features calculated over multiple scales increase the feature matching performance.

Ferrari et al. (2010) employ a novel visual feature known as Pair of Adjacent Segments (PAS) for shape matching. Given an image, the PAS features are found as follows. First, the edges contained in this image are found by an edge detector. Then, these edges are segmented into approximately straight segments. A PAS feature is generated from each pair of adjacent edge segments. Intuitively, two segments are adjacent if they belong to the same edge or if one segment is at the end of one edge directing towards the other segment. Each PAS feature has a location (the mean of the two segment centers), a scale (the distance between the two segment centers), a strength (the mean of edge strengths), and a descriptor. The descriptor is designed to be invariant with translation and scale changes, however, it is not invariant to in-plane rotations. This descriptor depends on the order the two segments. The first segment is determined as the leftmost segment of the two. If the two segments have similar x coordinates, they are ordered from top to bottom. Given this ordering, the descriptor is a 6-element vector:

$$\left(\frac{r_2^x}{N_d}, \frac{r_2^y}{N_d}, \theta_1, \theta_2, \frac{l_1}{N_d}, \frac{l_2}{N_d}\right),$$

where $\left(\frac{r_2^x}{N_d}, \frac{r_2^y}{N_d}\right)$ denotes a vector pointing from the midpoint of the first segment to the midpoint of the second segment, normalized by the distance between the midpoints of these two segments, N_d. θ_1 and θ_2 are the orientations of the two segments respectively. $\frac{l_1}{N_d}$ and $\frac{l_2}{N_d}$ are the lengths of the two segments, normalized by N_d.

Given a set of training images that have been labeled as containing the target shape within an identified bounding box (as shown in Fig. 2.2-2.4), the algorithm first searches for a common set of PAS features within these bounding boxes across all images. During this process, most of the PAS features corresponding to the back-

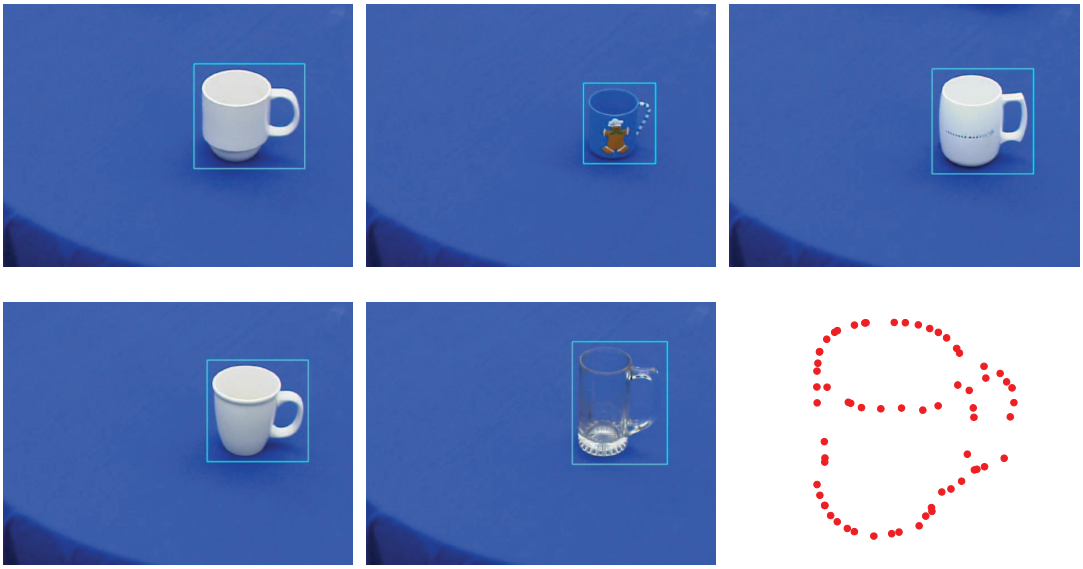


Figure 2.2: The set of objects used in our experiments. The rectangular regions in each image correspond to the target objects, which are manually selected. The last panel shows a learned model of the mug.

ground will be eliminated, because they tend to only exist in a small subset of the training images.

The algorithm then constructs a *codebook* of prototype descriptors by clustering the descriptors of these common PAS features. The codebook C is a collection of the centermost PAS of each cluster, which is called a *PAS type* T_i. The set of common PAS features is reduced to a small number of PAS types, which makes the future feature matching process computationally efficient. The PAS types that frequently occur at similar locations and scales relative to bounding boxes are selected as *model parts*. Based on the idea that a good model should consist of naturally connected model parts coming from a small number of training images, the model parts are *assembled* together to form an initial shape model. The initial shape model grossly describes the shape of the target concept. However, across the training set images, some variation exists in the relative positioning of the PAS features. The model is further refined by matching it back to the training images through the Thin Plate Spline Robust Point Matching algorithm (TPS-RPM, Chui and Rangarajan, 2003).

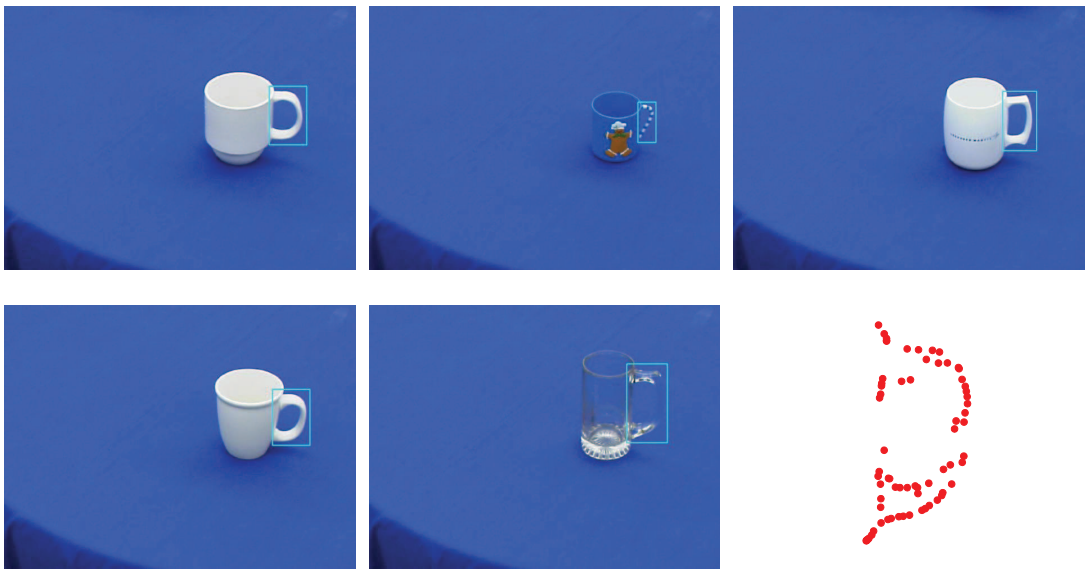


Figure 2.3: The set of objects used in our experiments. The rectangular regions in each image correspond to the grasp regions for handle grasps, which are manually selected. The last panel shows a learned model of the handle.

The TPS-RPM algorithm can robustly match two sets of points with a non-rigid TPS mapping, which consists of an affine component d and a non-rigid warp w. It estimates the correspondence between the two sets of points and searches for a set of d/w parameters given the initial solution provided by aligning the bounding box of the model and the one in the image. The dissimilarity between the shapes formed by the two sets of points can be measured by the energy contained in the non-rigid warping.

The TPS mappings between the shape model and each training image define a subspace in the full parameter space of TPS mappings. This subspace is used as a deformation model that describes how much within-class variation of the model shape are observed in the training images. This deformation model is further used in the testing process to constrain the matching between the model shape and a shape in the test image. With this constraint, the model shape can only match to a shape in the test image with a TPS mapping in the part of parameter space that is exhibited by the training images. Since the training images may not cover all possible variations

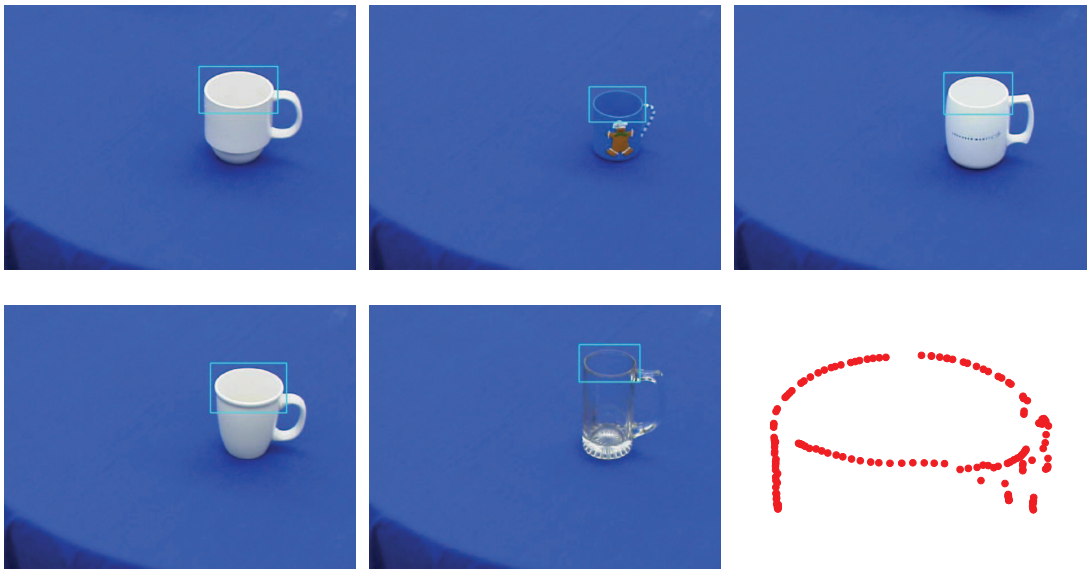


Figure 2.4: The set of objects used in our experiments. The rectangular regions in each image correspond to the grasp regions for top grasps, which are manually selected. The last panel shows a learned model for the top of the mug.

of the model shape, one can relax this constraint to allow some shape matches that are slightly different from those in the training images. The learned shape models are represented as sets of *edgels* (short for *edge pixel*, a pixel in an image that has the characteristics of an edge, i.e., orientation and/or magnitude) sampled on the model edges as shown in the last panels in Fig. 2.2-2.4. From these figures, we can see that the learned shape models capture the common edge segments across different training image fragments (the sub-images defined by the bounding boxes). These shape models do not capture finer object details that only exhibit in individual objects (such as the label on the mug in Fig. 2.2).

During the testing phase, the algorithm tries to find a match between a learned shape model and a test image. First, the algorithm matches PAS features observed in a test image to the model PAS types by using their descriptors. Since a PAS type in the shape model is associated with a location (relative to the model centroid) and a scale, each match votes for a particular center location and scale of the model in the test image by using a Hough-style voting scheme. The local maxima in the 3D

Hough voting space (x/y translation and scale) indicate potential good matches of the model. The Hough voting process greatly reduces the complexity of the problem, since the entire space of all possible translations and scales of the model in a given image is huge.

The initial matches found by the Hough voting process are further refined by the TPS-RPM algorithm. An initial condition from Hough voting specifies a center location l and scale s of the shape model to be matched. The algorithm then "superimposes" the edgels in the model onto the test image by centering and scaling the model edgels according to l and s. The transformed model edgels form a point set V. The bounding box of the superimposed model cut out a region of the test image. The set of edgels in the test image that fall in this region form another point set S. Notice that S may include edgels that do not belong to the target shape. The goal of the TPS-RPM algorithm is to find the correspondence between S and V, while rejecting points in S that do not correspond to any model points.

The two point sets $V = \{v_a\}_{a=1..K}$ and $S = \{s_i\}_{i=1..N}$ are matched with a non-rigid TPS mapping $\{d, w\}$. Here, d is an affine component that captures the translation, rotation, scaling, and shearing between these two point sets; w is a non-rigid warp that captures the shape deformation between these two point sets. TPS-RPM estimates the correspondence matrix $M = \{m_{ai}\}$ between V and S, and searches for a set of d/w parameters that minimize a cost function. The cost function is composed of 1) the distance between points in S and the corresponding points in V after mapping them by $\{d, w\}$, 2) the energy contained in $\{d, w\}$, which is measured by local warpings w and deviations of d from the identity matrix, 3) the orientation difference between corresponding points, and 4) a term which is inversely proportional to the edgel strength of matched image points.

Since both M and $\{d, w\}$ are unknown at the beginning, the TPS-RPM iterates between the following two steps.

1) Given the current TPS mapping $\{d, w\}$, update M. M is a continuous-valued soft-assign matrix, which evolves through a continuous correspondence space.

2) While fixing the correspondence matrix M, update the TPS mapping $\{d, w\}$ between V and a point set $U = \{u_a\}_{a=1..K}$, where each point u_a in U is a linear combination of all points in S weighted by the soft assignment m_{ai}:

$$u_a = \sum_{i=1}^{N} m_{ai} s_i.$$

The TPS-RPM iterates between these two steps until a certain number of iterations is reached, which is manually chosen.

The soft-assign matrix M is updated by setting its entry m_{ai} as a function of the distance between s_i and v_a, after mapping by the TPS:

$$m_{ai} = \frac{1}{\Gamma} \exp\left(-\frac{(s_i - f(v_a, d, w))^T (s_i - f(v_a, d, w))}{2\Gamma} \right), \tag{2.1}$$

where $f(v_a, d, w)$ is the mapping of model point v_a by the TPS $\{d, w\}$. M is normalized to ensure the rows and columns sum to 1. Γ is a temperature parameter, which is largest at the beginning. As the iterative process continues, Γ decreases, and the algorithm is more certain about the correspondence between model points and image points. At the same time, the TPS mapping between the model and image points is less constrained, so that the matched shape can fit better to local object contours in the test image.

Once the TPS-RPM converges for a given initial condition, a matching score is computed. The matching score of a match is calculated by a weighted sum of the following terms:

1) The number of model points that have been found as good matches to some image points. According to Chui and Rangarajan (2003), this is measured by the

number of points v_a with $\max_{i=1..N}(m_{ai}) > 1/N$.

2) The sum of squared distances between the TPS-mapped model points and their corresponding image points:

$$\sum_{a \in P} |f(v_a, d, w) - u_a|^2 / r^2,$$

where P is the set of indices of matched points from step 1. This measure is normalized by the size of the matched shape model, r.

3) The deviation of the affine component d from the identity matrix I:

$$\sum_{i,j \in [1,2,3]} \left(I(i,j) - d(i,j) / \sqrt{\det(d)} \right)^2.$$

The normalization term $\sqrt{\det(d)}$ eliminates deviations due to scale change. This would prefer matches with the same orientation as the model.

4) The energy contained in the non-rigid warp w:

$$trace(w^T \Phi w) / r^2,$$

where Φ is the TPS kernel matrix (Chui and Rangarajan, 2003).

If the highest matching score among all initial conditions is greater than a experimentally determined threshold, the corresponding match will be selected as the best match of the model in the test image. Otherwise, the algorithm claims that the given shape model is not observed.

One of the novelties of the PAS based shape model is that the TPS-RPM provides not only a matching score between the model and part of the test image, but also the boundary of the matched model in the test image. In most of the other work, the shape model is only recognized up to a bounding box. Experimental results show that a shape model of the target shape can be learned from cluttered images with only

the bounding boxes of the target shape provided. The learned shape model can be matched robustly to test images that contain similar shapes in a cluttered background. This algorithm is also robust to scale change and small affine distortion. The PAS feature can be modified to be rotationally invariant by using coordinate frames relative to each feature (detailed in Appendix A). However, the PAS feature loses part of its discrimination power when relative coordinate frames are used according to our exploratory experiments.

2.4.3 Constellation-based Methods vs. Bag of Features

Above, we have discussed the categorization of object recognition methods based on whether object texture or shape is used to construct discriminative visual features. Depending upon whether the spatial relations between individual features are used for recognition or not, object recognition methods can also be categorized to constellation-based methods and orderless methods (Marszalek, 2008). Given a set of local features that are extracted from training images, the matching of individual features to a test image can be ambiguous. Constellation-based methods reduce the probability of false matches by taking into account the relative spatial information between individual features (Piater and Grupen, 2002; Wang, 2007). However, constellation-based methods are computationally expensive and can only be applied to rigid and piece-wise planar objects. As a contrast, the orderless bag-of-features method assumes no or only weak spatial relationships between individual features. In this approach, a set of learned visual features is orderlessly put into a bag to describe a target object. During the matching phase, the identified features in a test image are matched to the ones in the bag. If the number of the matched features exceeds a certain threshold, the test image will be recognized as containing the object described by the bag. More generally, one can construct meta features that capture the distribution of individual features in the bag. In this case, if the distribution calculated

from a test image matches well with the one calculated from the bag, the test image will be recognized as containing the object described by the bag.

Ohbuchi et al. (2008) propose a method for 3D object retrieval by using bag-of-features and SIFT (called BF-SIFT). A feature vector is learned for each 3D object model. Given the 3D mesh model of an object, they first render a set of depth images by placing the camera uniformly on the aspect sphere. The location and scale of the object are normalized to be consistent across all depth images. Then, they calculate SIFT features on these depth images and cluster them by k-mean clustering. Each cluster center corresponds to a prototype SIFT feature and all the prototype features compose a codebook. The codebook serves as a look-up table. An arbitrary SIFT feature can be approximated by a prototype feature in the codebook that has the smallest distance to it in the 128 dimensional feature space. The SIFT features calculated from all depth images are translated into prototype features according to the codebook. Then, a histogram is composed by counting the frequency of each prototype feature observed in all depth images. Since these depth images cover different aspects of a single object, this histogram is used as a feature vector that captures the 3D appearance of the object.

By using a codebook of prototype features instead of raw SIFT features, a 3D object model is described more concisely, which saves space for feature storage and accelerates the feature matching process. Since the relative locations between prototype features are not encoded during the feature generation, BF-SIFT works especially well for articulated object retrieval. The authors test BF-SIFT against the other state-of-the-art shape based 3D object retrieval methods (such as Light Field Descriptors and Spherical Harmonics Descriptors) by using the McGill Shape Benchmark (MSB) of articulated 3D models and Princeton Shape Benchmark (PSB) of the rigid 3D models. Experimental results indicate that the BF-SIFT performs substantially better than the other methods for articulated 3D objects and comparably well

for generic rigid 3D objects.

2.4.4 Mean Shift Clustering

The mean shift algorithm is a nonparametric clustering technique originally proposed by Fukunaga and Hostetler (1975). Comaniciu and Meer (2002) revisit the mean shift algorithm and explore its broad usability for a variety of vision tasks, such as discontinuity preserving filtering and image segmentation. The mean shift algorithm has some excellent qualities for computer vision tasks since it does not assume *a priori* the number and the shape of clusters.

The mean shift procedure is a kernel based adaptive gradient ascent method that converges on the modes (local maxima) of an underlying density function in a feature space. Once converged, the basin of attraction of a mode, i.e., the data points visited by all the mean shift procedures converging to that mode, automatically defines a cluster of arbitrary shape. A Gaussian kernel is a common choice of kernel and is usually sufficient for most vision tasks.

The only tuning parameter to mean shift is the resolution of the clustering process, which is determined by the kernel bandwidth. This parameter can be selected based on the stability and repeatability of the clustering results across different parameter values, or by task-related domain knowledge.

Based on their experimental results, the mean shift algorithm achieves superior performance on a variety of vision tasks. However, the authors point out that attention should be paid when using the mean shift approach on high dimensional data (more than 6D).

2.4.5 3D Pose Estimation

The full pose (6D) of an object includes both its position (3D) and orientation (3D). While the position of an object can usually be estimated accurately and robustly

(usually solved simultaneously with object recognition), the problem of estimating the orientation of an object is a hard one. This is partially because the space of orientation is non-Euclidean and non-linear (Saxena et al., 2009). Furthermore, different orientations of an object may give the same appearance when symmetries exist. For example, a cylinder looks exactly the same when it rotates about its major axis. One possible way to solve this problem is to build a 3D model of a given object that acknowledges the fact that its appearance is a function of orientation. Schiele (1997) introduces a robust 3D object recognition algorithm. The basic idea is 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 is object recognition, all features corresponding to different aspects of an object are equivalent, that is, features are not associated with specific aspects.

Rothganger et al. (2006) use several images of an object taken from different viewing angles to build a 3D model of the object. SIFT features are calculated from each image, and these features are related together by matching images that are closest to each other in viewing angle. The 3D coordinates of each SIFT feature are calculated by using bundle adjustment and a 3D Euclidean model of the object is built by using the patches that correspond to SIFT features. In their experiments, they test their algorithm by taking novel images of the same object in cluttered scenes. Their algorithm is robust in the sense that an object in the training set can be recognized most of the time with arbitrary rotation and heavy occlusion. Although their goal is object recognition, their algorithm has the potential to be used for pose recognition purpose.

Based on this idea, Gordon and Lowe (2006) propose an algorithm for augmented reality. In their algorithm, the pose of a known object in the current scene is recognized accurately in real time, by extracting SIFT features and matching them to a trained 3D model of the object. Given a set of putative 2D-to-3D matches of fea-

tures, the camera pose can be solved accurately by minimizing the summed projection errors of all matched features. Since the position and orientation of the camera is known, the position and orientation of the object can be solved accordingly. They then evaluate their approach by inserting some animated objects into the real scenes according to the pose of a recognized object. Even though their algorithm is visually confirmed to work well in the above application, no quantitative measurements of the pose estimation error are given. One limitation is that this algorithm depends on the texture of the objects to be recognized, since the SIFT features are sensitive to object texture. As a result, the trained 3D models are specific to particular objects.

Subsequently, Romea et al. (2009) use Gordon and Lowe's algorithm in the robotics grasping domain. Particularly, they use a clustering algorithm to deal with the problem of registering multiple instances of the same object in cluttered and partial occluded scenes. In their experiment, quantitative results of pose estimation errors are given. Across all test objects, the average translation error is 0.67 cm and the average rotation error is 3.81 degrees. Given that the pose of an object is estimated, they use a trajectory planner to move the wrist of a robotic arm into the vicinity of the object for grasping. They show that the pose estimate of the object is accurate enough to help the robotic arm move into narrow openings and grasp the object in clutter.

All the above methods estimate object pose by building a 3D model of the object without explicitly dealing with the symmetries in a given object. The idea is that by explicitly modeling the symmetries that exist in an object, one may end up with a more compact and simple model. This model can usually facilitate robotics grasping. For example, recognizing the major axis of a cylinder is sufficient for a robot to grasp it.

2.5 Relating Vision and Grasping

2.5.1 Affordance-based Object Categorization

Object categorization is usually approached as a computer vision problem (Leibe et al., 2008; Griffin and Perona, 2008). In these methods, objects are categorized based on their visual appearance. However, for certain applications, such as robotics grasping, we care more about object affordances rather than their visual appearances. Objects that look visually different may share similar affordances. For example, both a stool and a chair afford sitting. Also, vision-based object classification or recognition usually requires a large amount of training data to cover within-class variation, different lighting conditions and aspects. Due to the above reasons, affordance-based object categorization has brought great interest recently in the robotics community. The idea of affordance based object categorization is supported by neurological evidence (Gallese et al., 1996; Rizzolatti and Craighero, 2004). The mirror neurons theory (Umiltà et al., 2001) suggests using resources that are connected to actions during perceptual analysis. In addition to the original findings in the monkey brain, recent research has shown similar structures in the human brain (Kilner et al., 2009).

A group of works focus on affordance-based object categorization. Bar-Aviv and Rivlin (2006) proposed a method to categorize objects based on their functionalities and properties instead of appearance. These functionalities are defined as a set of configurations of an agent relative to an object, which are essentially affordances. Given a CAD model or a real 3D scan of an object, these functionalities are verified by using a virtual agent interacting with the object in a simulation environment. A certain function is claimed to be found if the corresponding configuration of the agent relative to the object can be obtained.

Castellini et al. (2011) propose an object classification algorithm that combines grasp affordances with visual features. The grasp affordances of an object are encoded

as the finger configuration of the human hand when object-hand contact occurs. During the data collection process, a human teacher demonstrates a set of grasps for a given object. During the grasping process, the algorithm couples the hand configuration with images in which the object clearly shows up. For a given object, their algorithm learns a mapping function between hand configuration and visual features extracted from corresponding example images. An object is captured by an augmented set of features combining both visual features and hand configuration data. Then, they train a classifier for each pair of objects on these augmented features. In the experiments, they compare the object classification performance by using different algorithms: visual feature only, visual feature with hand configuration, and visual feature with hand configuration reconstructed by the mapping function. The results show that the approach using both visual and affordance cues outperforms the one using visual only, even for the reconstructed hand configuration.

2.5.2 Affordance-based Object Perception

Another advantage of affordance-based object categorization is the possibility for affordance-based perception. Woods et al. (1995) have done some early work on function based object categorization. Particularly, they propose the GRUFF (Generic Recognition Using Form and Function) system. During object classification, the functionalities of object parts are verified partially by vision (a range image) and partially by interaction (with a robot arm). This system allows reasoning about the function of novel objects when the goal is task-oriented in an unknown environment. For example, if the goal is sitting, it is reasonable to match an upside-down trash can to the class of chairs. The GRUFF object recognition system reasons about and generates plans for understanding 3D scenes of objects by performing a function-based labeling process. One advantage over model-based recognition systems is the possibility for action provided by the GRUFF system (perception for action). This

is consistent with Gibson's affordance idea.

In the work of Detry et al. (2009), vision and action are connected by an object-centered representation. The visual model and the grasp affordance model of an object are learned separately. The identity and pose of an object serve as an intermediate step that bridges the gap between vision and action. Once the identity and pose of an object is recognized and estimated, a robot can plan its grasp action accordingly relative to the object.

The visual model learned by their algorithm aims to recover the object pose. The authors use local 3D features to capture the appearance of image patches along object contours. Then, they use a Markov tree to combine these local features together into meta features hierarchically. The relative spatial configuration between a meta feature and its parts is captured by nonparametric probability distributions. This approach provides a method to parse an object into parts automatically. Given a novel image of a known object, this approach first matches local features and then propagates their belief of the object pose up the hierarchy.

The grasp affordance model aims to capture the distribution of a set of feasible grasps relative to the object pose. The initial set of grasp hypotheses that is afforded by an object is directly suggested by visual features. For example, the boundary between a cup and background with certain curvature may suggest a grasp on the rim of the cup. The grasp hypotheses generated by using the above method are hand poses (position and orientation) relative to the object such that closing the hand will result in stable grips of the object. These grasp samples are points in 6D pose (3D position and 3D orientation) space defined in an object-centered coordinate frame. The authors then use a probability distribution to capture the clusters of these grasp samples. Rather than using a mixture distribution as in the work of de Granville et al. (2006, 2009), they use nonparametric particle representation of probability density functions composed of a large number of small Gaussian kernels. Each sample itself

can be considered as an individual kernel and the overall distribution is simply a summation of all these individual kernels.

The initial set of grasp hypotheses is further verified by a robot through actual grasp experience. The nonparametric distribution can capture distributions of arbitrary shapes and it allows importance sampling during the grasp verification process. Only the grasp hypotheses with a high success rate in practice are kept for future use. This grasp verification process effectively increases the grasp success rate in the real environment. However, this process eliminates some useful grasp types which are more difficult to execute than the others. This is a drawback, since the goal of grasp affordance learning is to discover all useful grasps on a given object.

In their testing process, given a known object, a robot first recognizes the identity and pose of the object and aligns the learned grasp affordance model according to the object pose. Then, given the object with recognized pose, a robot can execute a learned grasp accordingly. This method is object specific since it requires the recognition of a learned object as a whole. The algorithm needs to learn unique models for each object to be manipulated, even though these objects may share the same set of grasp affordances.

A group of works concentrate on mapping partial visual features to grasp actions directly without high level semantics and object recognition (Piater and Grupen, 2002; Varadarajan and Vincze, 2011; Stark et al., 2008; Aleotti and Caselli, 2011). Piater and Grupen (2002) use appearance features to preshape a robot hand for tactile based grasping. In their work, primitive features are combined into more discriminative compound features. They employ two types of primitive features. An *edgel* is a pixel in an image that has the characteristics of an edge, i.e., orientation and intensity. A *texel* is a vector that encoded a local texture signature. Then, 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 use feedback from object recognition, as well as robotic grasping performance, to evaluate whether a constellation of features is useful or not. Useless features are eventually removed from the feature repository and replaced with other candidates. As a result, the system learns a set of useful visual features and associated hand shapes. By choosing an initial hand shape and pose visually, the quality of a haptic-based search for a grasp is considerably increased. Since this approach does not require the recognition of specific objects and the visual features learned are associated with local object parts, it has the potential to generalize to objects with similar parts.

Varadarajan and Vincze (2011) introduce the idea of *Conceptual Equivalence Classes*, which aims to solve the problem of goal-directed object recognition when the exact object cannot be found. In the case where a target object is not found, their algorithm will return an object with a similar set of affordances. However, they need to pre-define a set of object categories and manually map the object parts to part affordances. A possible solution is to map the object parts to affordances from human demonstration.

Stark et al. (2008) demonstrate the feasibility to detect certain object parts that afford certain grasp types in novel images. During the training process, a human teacher demonstrates a certain grasp for a given object. A single image of the object is taken, and the object part that is grasped by the human teacher is approximated by the overlapping region between the hand and the object. Then, a set of visual features (known as affordance cues) is calculated from this overlapping region, which captures the partial object shape. During the testing phase, a match of these affordance cues in a test image is interpreted as that the corresponding affordance is found. In their experiments, the algorithm learns two sets of visual features that correspond to two grasp types: a handle grasp and a side grasp on mugs. Since these visual features only capture partial object shapes, they can generalize to novel objects with similar

partial shapes. However, these visual features are only learned from a single image in which the target object part is clearly visible (i.e., the frontal view). Therefore, these visual features are limited in that they do not capture with-in class shape variation and aspect change.

Chapter 3

General Approach

The general goal of a robotic grasping algorithm is to find a set of points of contact between the object and the hand that allows the object to be grasped reliably. Napier (1993) and others (c.f., Cutkosky, 1989; MacKenzie and Iberall, 1994) define a grasp type, or category, in terms of the set of contacts made by the hand, the possible forces at each contact, and the ability of the contacts to resist perturbations. Our goal is for a robot to learn the mapping between images of objects and possible contact points by observing examples of successful grasps. Our hypothesis is that observed grasp type can provide meaningful labels to drive the learning of visual models that capture partial object shapes. Furthermore, a visual model learned by our algorithm will suggest an approximation to how the hand should be positioned relative to the object in different scenarios containing novel objects.

Visual feature descriptors such as SIFT are very discriminative and have been shown to perform well at recognizing specific objects. However, SIFT features tend not to generalize across small differences in objects because they capture very specific arrangements of edges and texture. Our goal is to learn visual representations that can be linked to grasping actions, and that do not rely on high-fidelity volumetric or mesh-based models of specific objects. Our approach is to employ visual features that 1) can recognize common components of objects, 2) are locally robust to appearance deformations due to rotations in 3D, and 3) are robust to common variations in geometries across objects (e.g., the shape of a mug handle). Furthermore, our approach is to define the visual feature categories by clustering examples based on how the

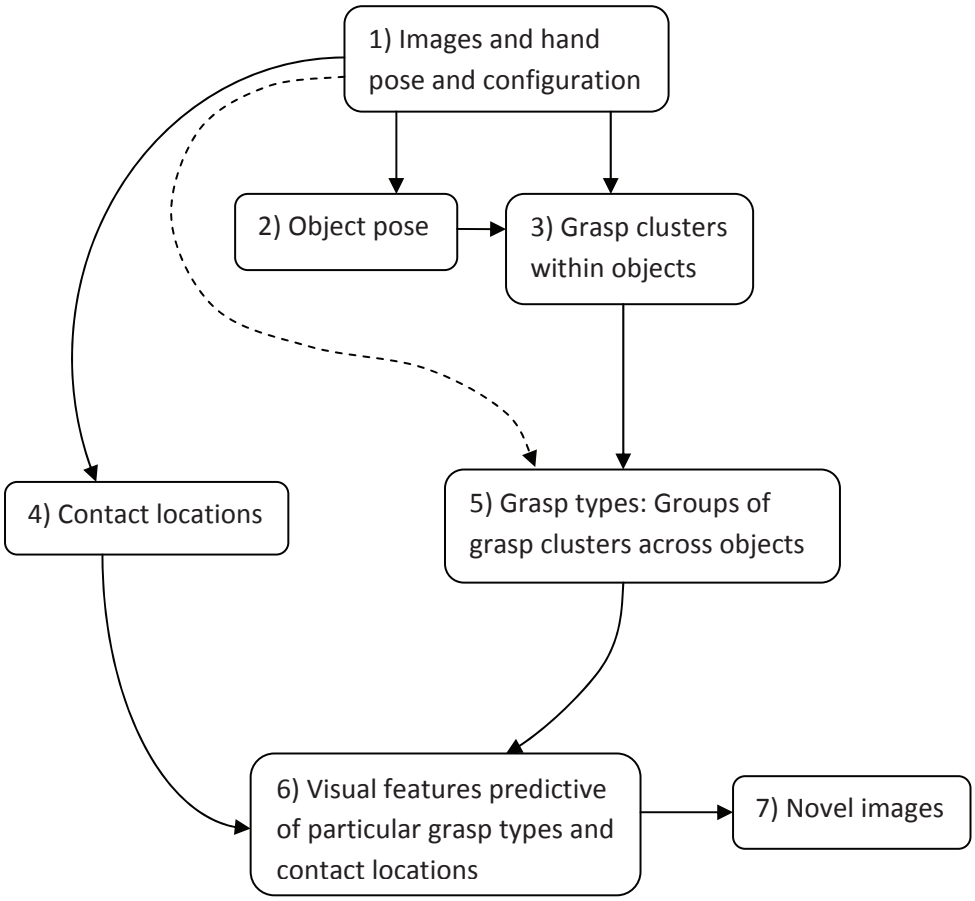


Figure 3.1: Outline of the grasp-driven visual model learning approach.

object component is actually grasped by a hand.

Given tuples of an example grasp (hand pose and configuration, relative to the object) and the object image during this grasp, our proposed algorithm will learn a set of visual features that will recognize object components in a novel image and suggest appropriate grasps. We assume that tuples corresponding to a single object are labeled as such (i.e., a set of tuples is sampled as an object is manipulated). Although there will be several examples of each object class within the training data set, we make no assumptions about the relationships between these different objects. Furthermore, we assume that individual training images may contain multiple objects and that the object in question may be occluded by the hand.

The outline of the proposed approach is shown in Figure 3.1. I propose a specific approach in the following sections. The training data to this learning algorithm are tuples of an image and a grasp example (Box 1). The grasp examples can be either provided by a human teacher or by the robot itself. Each grasp example includes information about the hand pose and configuration. For each tuple, the object pose is estimated (Box 2). Pose can be recovered using object-specific visual models (e.g., Romea et al., 2009) or by explicitly instrumenting the object with a pose sensor.

Using the work of de Granville et al. (2006) and Palmer and Fagg (2009), these hand poses are clustered into compact sets of grasps (Box 3). We refer to these as *proto-grasp*. Recall the example that we discussed in Figure 2.1. For this particular cylinder, the algorithm finds four proto-grasps: two ball grasps on either end of the cylinder and two power grasps about the major axis of the cylinder. However, these proto-grasps are tied to the coordinate frame of the specific object and will not directly apply to other objects because the transformation from one object coordinate frame to another is unknown.

In order for the learned model to incorporate some degree of variation of object shape and appearance, we further group similar proto-grasps across different objects into a single *grasp type* (Box 5). For example, handle grasps on different mugs could be grouped together as a single grasp type. Likewise, ball grasps that approach mugs and cans from the top could be grouped together as another grasp type. The dotted line from Box 1 to Box 5 indicates that grouping proto-grasps across objects into grasp types can be informed by visual information describing the shape of the object.

The grasp types induce a partition on the training set of images. For each grasp type, we identify the common visual features across sample images that suggest the location of the contacts between the hand and the object. This process requires the recognition of contact locations in the image coordinates frame. In Box 4, the contact locations are the locations at which the hand contacts the object. With kinematics

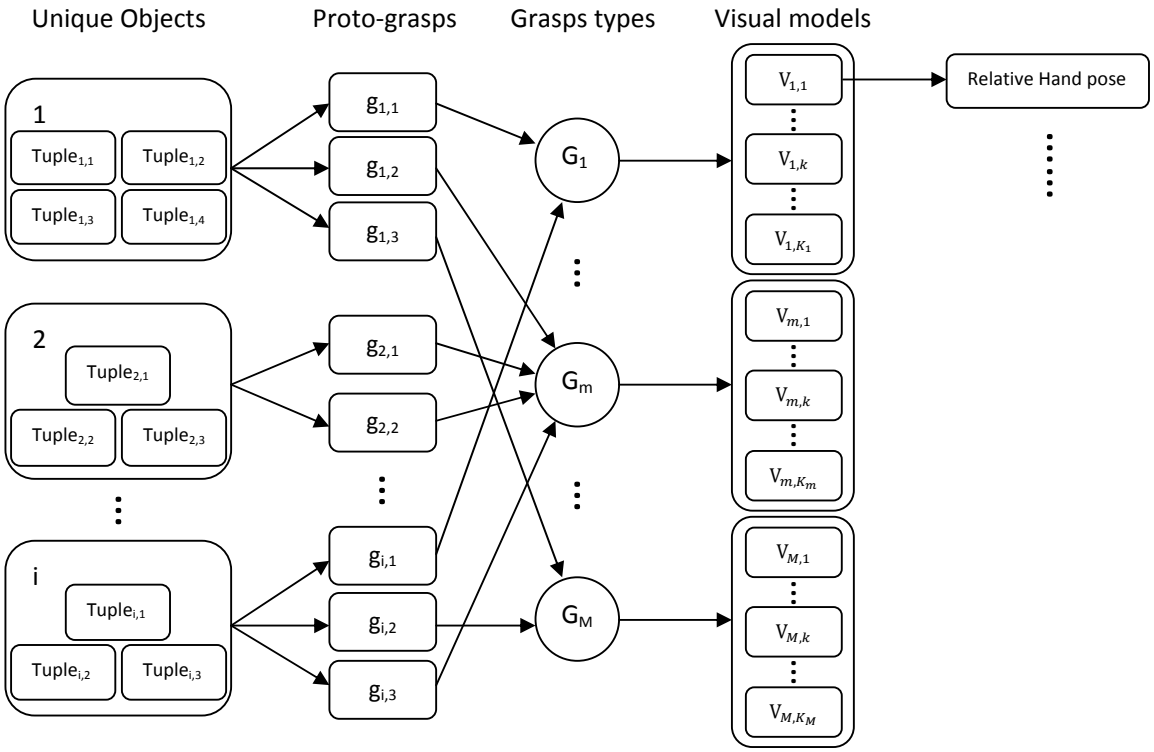


Figure 3.2: Data flow of the grasp-driven visual model learning approach.

and tactile sensors, a robot can estimate the 3D location of these contacts.

Box 4 and 5 meet at Box 6. The grasp contact locations define a region of interest (ROI) in the image that putatively contains key geometrical features that can be used to cue a grasp. For each grasp type, the algorithm of Box 6 attempts to identify a small set of visual features that are likely to occur within the ROI. Because the appearance of the object varies with aspect, we expect that a single object will require a set of visual models to cover all aspects. Our algorithm then matches these visual models to novel images (Box 7) in order to detect the grasp region for each grasp type.

3.1 Data Flow Structure

The data flow structure is illustrated in Figure 3.2. In this diagram, the leftmost column of boxes represents a set of samples collected for some unique objects during the data collection process. This data collection process is detailed in Chapter 4. Specifically, we denote the j^{th} tuple for object i as $Tuple_{i,j} =< I,\ {}^{o}_{g}T >_{i,j}$, where I is an image, and ${}^{o}_{g}T$ is the hand pose in the object coordinate frame. Each tuple is called a *grasp example*.

For each unique object i, the corresponding grasp examples are clustered into a compact set of proto-grasps. We denote the p^{th} proto-grasp for object i as $g_{i,p}$. We further group similar proto-grasps across different objects into a single grasp type G_m. We denote the entire set of grasp types as $\mathcal{G} = \{G_m\}_{m=1..M}$, with M denoting the total number of grasp types for a given training set. During this process, the entire set of grasp examples are partitioned by grasp types. The proto-grasp learning and grouping process is discussed in Chapter 5, and the experimental results are given in Chapter 6.

For each grasp type, G_m, we have a set of sample images corresponding to the set of grasp examples, $\mathcal{I}_m$, which contains similar object components. We would like to identify the common visual models $V_{m,k}$ across $\mathcal{I}_m$ that suggest the location of the contacts between the hand and the object. Here, the subscript k in $V_{m,k}$ denotes the k^{th} visual model for grasp type G_m when it is viewed from a particular viewing angle (aspect). To do this, we further cluster the entire set of sample images by aspects. A visual model, $V_{m,k}$, is learned from the k^{th} set of aspect-clustered sample images. This aspect clustering process is discussed in Chapter 7. Assuming a set of K_m visual models is learned for grasp type G_m, our algorithm then matches these visual models $\{V_{m,k}\}_{k=1..K_m}$ to novel images in order to detect the grasp region for each grasp type G_m. The visual learning and matching algorithms are detailed in Chapter 7, and the

experimental results are given in Chapter 8.

When the robot is confronted with a novel object, it will detect possible contact locations by using each learned visual model $V_{m,k}$ for each grasp type G_m. These contact locations are grasp type specific. If a visual model corresponding to a certain grasp type matches a test image well enough, we consider that this grasp type can be applied to the object contained in the test image. The 3D position of the matched visual model can be estimated through stereo triangulation (e.g., Hartley and Zisserman, 2004; Gordon and Lowe, 2006). Given the 3D position of the matched visual model, the robot can then use the relative 3D hand pose suggested by the visual model for grasping the object.

Chapter 4

Data Collection

4.1 University of Oklahoma Grasp Data set

Part of the contribution of this dissertation is the OUGD (University of Oklahoma Grasp Data set[1]). In the OUGD, we have collected about 22,000 grasp samples with corresponding stereo image pairs for ten different object categories and five objects in each category. These ten object categories are: mugs, glasses, spray bottles, hammers, hand drills, wine bottles, detergent/softener bottles, bowls, spatulas and spheres. For the first three object categories, we have also collected the corresponding range images using the Microsoft $Kinect^{TM}$, although the use of these range images is out of the scope of this dissertation. The entire set of 50 objects in the OUGD are shown in Fig. 4.1. In this figure, we show an example image for each object that corresponds to the *frontal view*. This frontal view is used as the anchor aspect for the following multi-aspect data collection process. Images of objects that belong to the same categories are arranged next to each other. We can see that the objects in the same category have similar gross shapes. However, these objects also have substantial within-class shape variations and differ significantly in color and texture. Due to this reason, color/texture-based visual features usually generalize poorly across different objects in the same category. The number of grasp types that we demonstrated varies for each object category. On average, about 120 samples were collected for each grasp type, corresponding to viewing angles that are approximately evenly distributed on the aspect sphere (about 30 degrees apart).

[1] http://www-symbiotic.cs.ou.edu/projects/OUGD

We employ a calibrated stereo camera system to collect the images. This system is capable of localizing uniquely identifiable visual features to within a few centimeters in 3D. In addition, we use a Polhemus $Patriot^{TM}$[†] to track the position and orientation (we use *pose* to indicate both variables) of both the right hand of a human teacher and the object being manipulated. The Polhemus sensor uses three perpendicular magnetic fields to estimate both position and orientation. The pose of the camera is measured in the global coordinate frame and remains the same during the entire data collection process. Given the pose of the Polhemus frame relative to the global coordinate frame, we derive a transformation from the camera frame of reference to the Polhemus frame. The Polhemus sensor is affixed to each object; the position does not change during the collection process with that object. However, from one object to the next, there are no guarantees about the relationships between the sensor locations. This is intended to model an agent's ability to track the relative pose of an object as it manipulates that object. However, it does not make any assumptions about how different objects might relate to one-another.

In our experiment, we estimate approximate contact locations of an object in a given image as the part of object occluded by the hand. Note that this only gives us an approximation of the contact locations. Accurate contact location estimation would require instrumentation of either the object or the teacher's hand.

Given that we want to differentiate the object from the background and obtain a gross approximation of where the contacts are on the object, in the data collection process, we associate a training grasp sample with a triplet of images: a background image, an object-only image and an object-with-hand image. Specifically, we collect data using the following procedure: first, the stereo camera pair is oriented with their field of view (FOV) covering the table. Then, we take a stereo image pair, which we call the background. An individual sample is composed of the hand pose relative to

[†]http://www.polhemus.com

Figure 4.1: The set of objects in the OUGD.

the object when grasping happens and a triplet of stereo image pairs: background, object-before-grasp and object-after-grasp. We collect multiple samples in order to cover different aspects and grasp types. To do this, we put the object on top of the table, and rotate the object about the z axis (perpendicular to the table), so that we can see the *frontal view* of the object from the left image. For example, the frontal view of a mug is a view such that the mug is placed upright on the table with its handle facing either left or right. We use the frontal view as the starting angle for each object. The frontal view of each object is shown in Fig. 4.1. From the starting angle, we tilt the object towards the camera until a top view is achieved, and tilt the object away from the camera until a bottom view is achieved, for about every 20

degrees. Also, for each tilt angle, we rotate the object about its z axis 360 degrees, with about 30 degrees interval. The z axis of each object is defined to be coincident with the z axis of the table in Fig. 4.1. Generally, the z axis is consistent with the object's rotational symmetry axis, if such a symmetry exists (though our algorithm does not make use of this information). We repeat the above data collection process for each grasp type on each object. For convenience, for the same object, some grasp types are demonstrated when the object is in a different initial configuration. For example, when we demonstrate the grasps on the bottom of a glass, the glass is initially put on the table up-side down, as opposed to upright in Fig. 4.1. This new initial configuration serves as the starting view point (frontal view), and the data collection process is all the same as above.

For convenience, in the above process, we choose to rotate the object about its rotational symmetry axis, if such a symmetry exists. This is equivalent to rotating the camera about the object's rotational symmetry axis. This choice makes the data collection process easier to keep track of, but is not necessary. All grasp samples are ultimately represented within the coordinate frame of the object.

The set of grasp types demonstrated for each object is summarized in Table 4.1. Note that these grasp types are unlabeled during the data collection process and we rely on the grasp affordance learning algorithm to recover these classes automatically. The number in the parentheses denotes the number of such grasp types, which are usually symmetric. For example, we can grasp from the side of a glass with either the overhand or underhand hand orientations.

4.2 Contact Location Extraction

For each grasp sample, the goal of this step is to identify the area of the image that contains any visible object contacts. We use each training image tuple to visually

Table 4.1: Grasp types demonstrated for each object category

	Rotationally symmetric	**Unidirectional**
Mug	Top	Handle
Glass	Top, Bottom[1], Side, Side[1]	
Spray bottle		Nozzle (2), Trigger, Side (2), Bottom[1] (2)
Hammer	Handle Precision, Handle Power	Top (2)
Hand drill	Handle, Barrel[2] (2)	Trigger
Wine bottle	Neck, Neck[1], Side, Side[1], Bottom[1]	
Detergent bottle	Cap	Handle, Side, Bottom[1] (2)
Bowl	Rim (2)	
Spatula	Handle (2)	
Ball	Ball Grasp	

[1]As the starting aspect, the object is held upside-down
[2]As the starting aspect, the object is rotated 90 degrees counter-clockwise

identify the locations of the object that are occluded by the hand, and hence contain the visible set of contact locations (as shown in Figure 4.2). A grasp sample is composed of a tuple of three images: 1) background only, 2) object only and 3) object grasped by a hand. We define the location in an image at which the hand contacts the object as a *grasp region*. These grasp regions are estimated in a given image by the area "covered" by the hand (Stark et al., 2008). The grasp regions in each training image tuple can be estimated by a sequence of image differencing techniques (Algorithm 1): 1) we remove irrelevant pixels from the object image (I_{obj}), such as those belonging to the holding hand and shadows, based on color cues. 2) We subtract the object image (I_{obj}) from the grasp image (I_{gsp}), which gives us an image of the grasping hand and part of the arm. 3) We subtract the background image (I_{bk}) from the object image (I_{obj}), which gives us an image of the object without background. 4) In the above two steps, we obtain corresponding binary images by thresholding. Then, we remove the extraneous pixels in these binary images by a set of erosion and

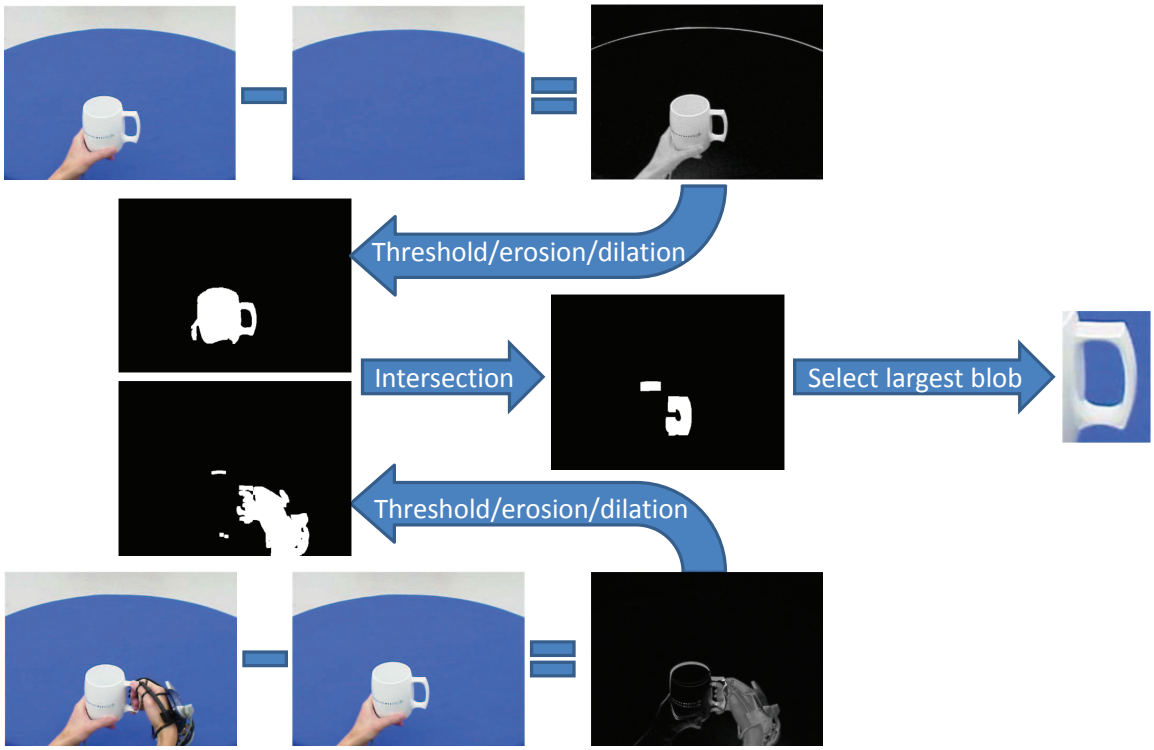


Figure 4.2: Simple steps to extract the grasp location from a tuple of training images.

dilation operations. 5) The intersection of the resultant images I_{gsp} and I_{obj} gives us an image region (the grasp region) that corresponds to the part of the object occluded by the hand.

Some of the parameters in this process require separate tuning, given that some of the objects in OUGD have colors that are close to skin, the table, or the data glove. Since our goal is to extract object parts that are predictive to certain grasps, we would like to remove pixels belonging to other extraneous objects (such as the supporting hand and the cable) as much as possible. In order to do this, we calculate a color model, M_{skin}, by using pixels manually selected on the hand. The likelihood of a pixel being the skin color, given the skin color model, is $p(pixel|M_{skin})$. Then, for a given image, we remove pixels that can be well described by the skin color model, M_{skin}, by selecting a likelihood threshold value. If we set this threshold too low, both the pixels that correspond to skin and object will be removed from the image. If we

Algorithm 1 An algorithm that extracts contact location

procedure FINDGSPLOC($I_{bk}, I_{obj}, I_{gsp}, M_{skin}, M_{shadow}$)
  I_{obj} =REMOVE_COLOR(I_{obj}, M_{skin})
  I_{obj} =REMOVE_COLOR(I_{obj}, M_{shadow})
  $I_{gsp} = I_{gsp} - I_{obj}$ ▷ locate the grasping hand
  I_{gsp} =THRESHOLD(I_{gsp})
  I_{gsp} =EROSION(I_{gsp})
  I_{gsp} =DILATION(I_{gsp})
  $I_{obj} = I_{obj} - I_{bk}$ ▷ locate the object
  I_{obj} =THRESHOLD(I_{obj})
  I_{obj} =EROSION(I_{obj})
  I_{obj} =DILATION(I_{obj})
  $I_{contact} = I_{gsp} \cap I_{obj}$ ▷ locate contact
  return the largest blob in $I_{contact}$
end procedure

set this threshold too high, the extracted grasp region will contain a large part of the hand and arm. In the scope of this dissertation, we hand-select the color model threshold on a per-object basis to achieve reasonable separation of object and skin.

4.3 Synthesizing Ground Truth Bounding Boxes

Note that for each tuple of images, only one ground truth grasp region can be generated in this way, since only one grasp type is demonstrated for a tuple of images. In the testing phase, we use the object-only image in each tuple of images as a test image. However, for a given test image, other possible grasp regions may exist besides the one that is demonstrated during data collection. For example, in Fig. 4.3, the bottom row corresponds to a stereo image pair in which a top grasp is demonstrated. However, one can still observe the grasp regions for a handle grasp (although part of the handle is occluded by the supporting hand in this view). In the testing phase, we would like a visual matching algorithm to detect all of the grasp regions associated with a test image and properly measure its performance. Therefore, we need to infer the ground truth bounding boxes for all grasps that are afforded by the object

contained in an image. We solve this problem by finding the nearest neighbor of this image on the aspect sphere in which another grasp type is demonstrated. We then overlay the ground truth bounding box of the nearest neighbor image onto the image to be synthesized and translate and scale the bounding box accordingly. We calculate the translation and scaling based on the 3D locations of the object relative to the camera associated with this image and its nearest neighbor. However, sometimes, the nearest neighbor image may not be close enough to the image in question. In this case, we do not provide a ground truth bounding box for this particular grasp in this image, since the neighbor image may look very different from the given image.

Examples of synthesized bounding boxes in a stereo image pair are shown in Fig 4.3. In the stereo image pair on the bottom row, a grasp on the top of the mug is demonstrated during the training process. Our goal is to find the bounding boxes of the handles in this image pair. In order to do so, we first find all the samples with a demonstrated handle grasp. Then, from these samples, we select the sample that has the closest viewing angle to the given image. The stereo image pair corresponding to the closest neighbor is shown in the top row in Fig. 4.3. If the viewing angle of the closest neighbor is less than α_{th} from that of the given image, and the camera in-plane rotation difference is less than β_{th}, we will consider these two samples as being close enough. Here, $\alpha_{th} = 10^\circ$ and $\beta_{th} = 25^\circ$ are selected and fixed based on exploratory experiments. Since we also check the in-plane rotations, a sample with an upside-down object will never be considered as the neighbor of a sample with an upright object. This choice insures the correctness of the ground truth bounding boxes that we extract. For an image with an upright object, the ground truth bounding boxes corresponding to grasps learned on an image with an upside-down object will be set to empty, and vice versa.

Since the object position may be shifted in the nearest neighbor image relative to the image in question, we can not just superimpose the bounding boxes of the handle

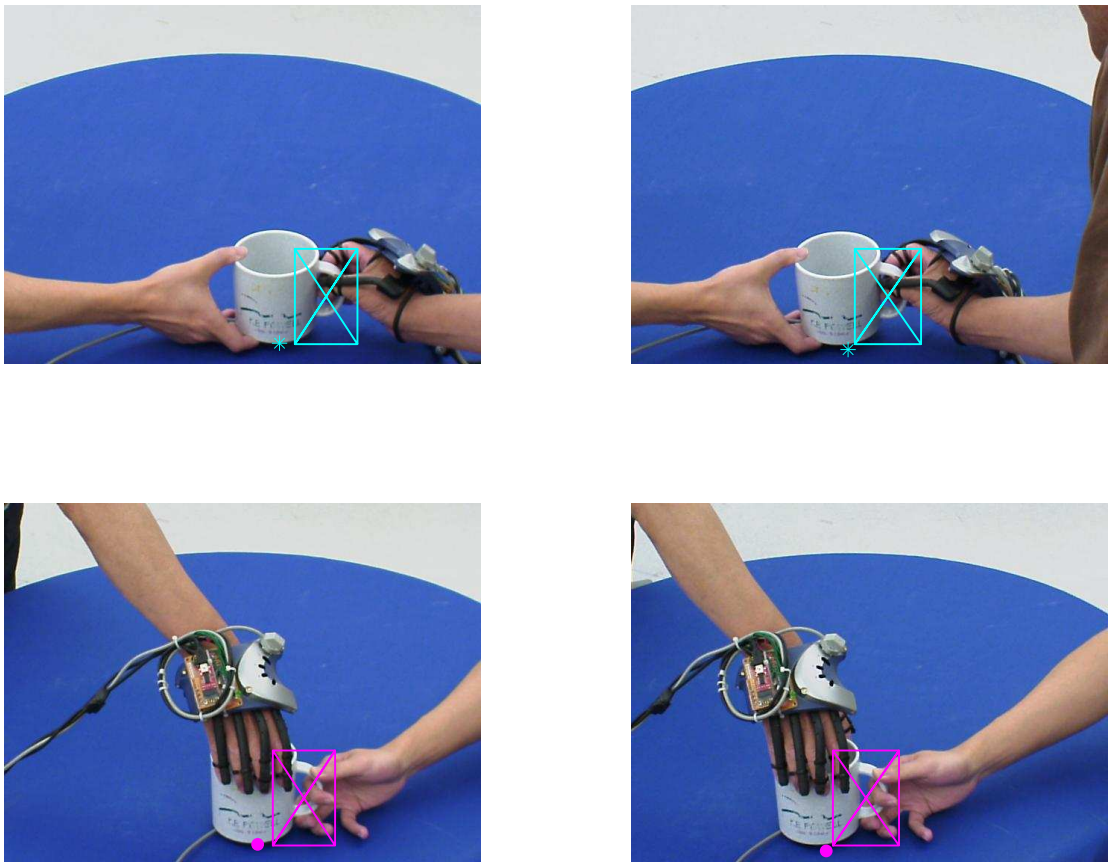


Figure 4.3: An example of synthesized ground truth bounding boxes from a neighbor image. Top row: nearest neighbor stereo image pair with ground truth bounding boxes generated by image differencing; bottom row: a stereo image pair with synthesized bounding boxes.

on top of the image in question. Since we know the 3D position of the Polhemus sensor in all images, we can shift the bounding boxes relative to the sensor position. In the nearest neighbor image, we project the 3D position of the sensor, $[X, Y, Z]$, onto the image plane by using the camera projection matrix, C (assumed to be fixed during the entire data collection process):

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = C \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix},$$

where $p = [x_p; y_p]$ is the projected sensor location in the image coordinate frame. Assume that matrix ${}^1B$ contains the homogeneous coordinates of the four corners of the ground truth bounding box in image 1's coordinate frame (the nearest neighbor image), we calculate the corresponding homogeneous coordinates in image 2's coordinate frame, ${}^2B$, by:

$$ {}^2B = s \, {}^2_{p_2}T \, {}^{p_1}_1T \, {}^1B, $$

where, s is a scalar that compensates for the scale difference of these two bounding boxes. $s = Z_1/Z_2$, where Z_1 is the depth of the sensor origin in image 1 and Z_2 is the depth of the sensor origin in image 2. p_1 (denoted as an asterisk in Fig. 4.3) and p_2 (denoted as a dot in Fig. 4.3) are the projected sensor locations in image 1 and image 2, respectively. We first translate the corners of the ground truth bounding box from image 1's coordinate frame into p_1's coordinate frame: ${}^{p_1}_1T \, {}^1B$. Since the relative position and orientation between the projected sensor location and the bounding box is fixed (down to scaling), the coordinates of the corners in p_2's coordinate frame are also ${}^{p_1}_1T \, {}^1B$. Next, we translate the coordinates of the corners back into image 2's coordinate frame: ${}^2_{p_2}T \, {}^{p_1}_1T \, {}^1B$. Finally, the corners of the bounding box are properly scaled with respect to the origin of image 2: $s \, {}^2_{p_2}T \, {}^{p_1}_1T \, {}^1B$, which accounts for the distance change between the sensor and camera in images 1 and 2.

Special cases exist for some of the object categories. For the spray bottle, the top and bottom grasps have two hand orientations that are related by a 180-degree rotation about the object's major axis. These grasp types share the same grasp regions. If one of these ground truth bounding boxes is missing, we will use the other one to replace it. The same is true for the two bottom grasps on the detergent bottles, the two grasps on the rims of bowls, the two handle grasps on the spatulas, and the two top grasps on the hammers.

Chapter 5

Object Clustering

In this chapter, we detail a particular instantiation of the approach proposed in Chapter 3. Given the grasp examples of different objects in the OUGD, the proposed algorithm first learns a set of proto-grasps for each unique object. Then, the algorithm clusters the entire set of object in the OUGD into object categories based solely on the different ways that these objects are grasped. Specifically, we cluster objects based on the similarity of their grasp affordance models. In particular, an object is considered a good match if, after alignment of the two objects, that its model explains the second object's grasp data well. Alignment is necessary because in the data collection process, we do not make any assumptions about how different objects might relate to one-another. Finally, the proto-grasps are grouped into grasp types for each object category.

5.1 Proto-grasps Learning

During the data collection process, the object and hand poses are specified in the global coordinate frame. However, we would like to represent the hand pose in the object coordinate frame, so that the learned grasp affordance model is independent of the object pose. Specifically, we change the reference frame of the hand pose from global coordinate frame to the object coordinate frame. Given the hand poses of grasp samples in the object coordinate frame, a grasp affordance model is learned according to de Granville et al. (2006) and Palmer and Fagg (2009) for each individual object in OUGD. In the learned grasp affordance model, a weighted mixture model of PDFs is

fit to the entire set of grasp samples. Each component PDF captures a single cluster (Box 3 of Fig. 3.1), and is described as a joint PDF in both the hand position and orientation space relative to the object. Given a set of grasp samples for an object, the parameters of the mixture model is found using expectation maximization (EM). Since EM is a gradient ascent method and local optima exist in our case, we perform many attempts of EM with random initial conditions. The best mixture model is selected by using the *Integrated Completed Likelihood* (ICL) metric. Assuming M_i is a mixture model that captures the distribution of grasp samples for a given object i, and that ${}^{o_i}_{g_j}T$ denotes a grasp sample j of object i, the *likelihood* of this grasp sample with respect to the grasp affordance model is described by:

$$p({}^{o_i}_{g_j}T|M_i).$$

Accordingly, assuming $\{{}^{o_i}_{g_j}T\}_{j=1..J_i}$ is the entire set of hand pose samples demonstrated for object i, the *log likelihood* of this set of grasp samples with respect to grasp affordance model M_i is described by:

$$L_i = \sum_{j=1}^{J_i} \log[p({}^{o_i}_{g_j}T|M_i)].$$

The above grasp affordance learning process provides us with grasp clusters for a single object (Box 3 of Fig. 3.1). We refer to each grasp cluster as a *proto-grasp*. This is because the affordance models learned by de Granville et al.'s approach are specific to individual objects. However, these models, alone, do not necessarily generalize to other objects of the same class because the alignment of these models is unknown and because of within-class variations in geometry (e.g., relative size of the objects). Nevertheless, these affordance models can serve as the basis for an object clustering process. First, we employ a stochastic gradient approach that identifies the ideal

alignment and scaling between object model pairs. The degree of match is measured in terms of the likelihood of the grasps for one object in terms of the model of the other object. Second, we identify groups of objects that mutually score well in terms of this metric. This process is detailed in the next section.

5.2 Object Clustering

Since we would like our grasp affordance models to be generalizable to objects that can be grasped in similar ways, we further group proto-grasps across different objects into a smaller set of grasp types. In particular, we want to identify which clusters in the hand pose space across different objects should be mapped to a single grasp type. One way to solve this problem is to use a *grasp-level matching* approach (Box 5 of Fig. 3.1). As indicated by the name, this approach clusters individual proto-grasps into grasp types. For example, this approach might find a match between the rims of a mug and a glass because a ball grasp is used for both. This approach requires some form of matching process between the proto-grasps. One possible way would be to compare the hand configuration (positions of the fingers) used for each cluster. However, the currently available sensing system precludes the accurate extraction of this information. The grasp-level matching between individual proto-grasps is made difficult by only knowing the hand poses relative to the object.

We instead choose to use an object-level matching approach that matches the entire set of proto-grasps associated with each object. The object-level matching approach clusters different objects by comparing their grasp affordance models rather than clustering individual proto-grasps. Because such an approach requires a complete match of objects, all partial matches are not considered. For example, a mug and a glass will not be grouped together, since they afford different sets of proto-grasps, despite the fact that these two sets may share some common proto-grasps.

We define two objects as *similar* if there is a match between the two sets of proto-grasps (grasp affordance models) associated with these two objects. We further group similar objects together into a single *object category*.

The question is: how to find a match between the two sets of grasp clusters associated with two objects. During the data collection process, even though the object coordinate frames are known and fixed for individual objects, there are no correspondences across objects. The coordinate frame of one mug is different from the other mugs since there is no guarantee that the pose sensor is placed at exactly the same location (relative to the object). In essence, the question is whether there exists a relative pose that brings the different grasp clusters into alignment with corresponding clusters from the other object. Specifically, if we describe a set of hand pose samples for the first object as $\{{}^{o_1}_{g_j}T\}_{j=1..J_1}$, the learned grasp affordance model for the first object is a mixture distribution that describes the likelihood of a given hand poses: $p({}^{o_1}_{g_j}T|M_1)$. For the second object, we have a set of hand pose samples $\{{}^{o_2}_{g_j}T\}_{j=1..J_2}$. The question is whether there exists a transformation ${}^{o_1}_{o_2}T$ and a scaling $R_{12}^T S_{12} R_{12}$ that results in a high log likelihood value for the following:

$$L_{12} = \sum_{j=1}^{J_2} \log[p(R_{12}^T S_{12} R_{12} \, {}^{o_1}_{o_2}T \, {}^{o_2}_{g_j}T|M_1)]. \tag{5.1}$$

In this equation, R_{12} is a rotation matrix to be determined that allows scaling about an arbitrary set of axes, and S_{12} is a diagonal scaling matrix. The detailed object matching algorithm is given in Algorithm 2 to 5. Our goal is to maximize Equation 5.1 with respect to the transformation parameters. Here, we have three unknowns: ${}^{o_1}_{o_2}T$, S_{12} and R_{12}. Specifically, in Algorithm 2, ${}^{o_1}_{o_2}T$ is decomposed into a 3D rotation, which is defined by a unit quaternion, q, and a 3D translation, which is defined by a vector $[p_x, p_y, p_z]$; the rotation matrix, R_{12}, is defined by a unit quaternion, r; and the scaling matrix, S_{12}, is defined by the three diagonal elements, $[s_x, s_y, s_z]$.

We use a gradient based search (Algorithm 3) to find the values of these variables $(q, p_x, p_y, p_z, s_x, s_y, s_z, r)$ that maximize L_{12}. In Algorithm 3, the search process starts with the initial values of these variables: $(\hat{q}, \hat{p_x}, \hat{p_y}, \hat{p_z}, \hat{s_x}, \hat{s_y}, \hat{s_z}, \hat{r})$. In addition, we constrain the amount of scaling, $[s_x, s_y, s_z]$, to be between 0.5 and 2, and the norms of q and r to be 1 (unit quaternions). However, since a gradient based search is subject to local extrema, we attempt multiple search starts with different initial conditions. This is reflected in Algorithm 2, where we call function *FMAXCON* multiple times by starting the search at different initial q's. Specifically, for each attempt, we regularly sample a set qs of unit quaternions from the entire space of rotations. The initial values of the other variables remain the same: the 3D translation, $[p_x, p_y, p_z]$, is initialized to $[0, 0, 0]$; the 3D scaling, $[s_x, s_y, s_z]$, is initialized to $[1, 1, 1]$; and the rotation, r, is initialized to $[1, 0, 0, 0]$. This generally works well in our case, since the translation of the sensor location from object to object is typically small, and the objects in the same category are typically of similar sizes. One can easily adapt this search process to more general cases by attempting different starting 3D translations, as well as a wider range of 3D scaling. In function *OBJMATCH*, we record the current best solution returned by function *FMAXCON*, and update it whenever a better solution is found by a new search attempt. After all search attempts, function *OBJMATCH* returns the matrices, ${}^{o_1}_{o_2}T$, S_{12}, and R_{12}, that give the highest value of L_{12}. The matrices, ${}^{o_1}_{o_2}T$, S_{12}, and R_{12}, are recovered from $(q, p_x, p_y, p_z, s_x, s_y, s_z, r)$ using function *ASSEMBLE* (Algorithm 5).

If the log likelihood, L_{12}, between a pair objects, 1 and 2, found by the above object matching algorithm is higher than some threshold (e.g., 0), we claim that an alignment (transformation, rotation and scaling) does exist between these two objects and the proto-grasps match in their structure. Then, the aligned proto-grasps are considered by our algorithm as being in the same grasp type. One advantage to such a process is that objects may be compared with themselves so as to identify

Algorithm 2 An object matching algorithm using grasp affordance models

function OBJMATCH($\{{}^{o_2}_{g_j}T\}_{j=1..J_2}, M_1$)
  $qs \leftarrow$ regularly sample a set of unit quaternions
  $\hat{p_x} \leftarrow 0, \hat{p_y} \leftarrow 0, \hat{p_z} \leftarrow 0$
  $\hat{s_x} \leftarrow 1, \hat{s_y} \leftarrow 1, \hat{s_z} \leftarrow 1$
  $\hat{r} \leftarrow [1\ 0\ 0\ 0]$
  $fval_{max} \leftarrow$ a very small number
  for all $\hat{q} \in qs$ **do**
    $[T, R, S, fval] \leftarrow$ FMAXCON($\{{}^{o_2}_{g_j}T\}_{j=1..J_2}, M_1, \hat{q}, \hat{p_x}, \hat{p_y}, \hat{p_z}, \hat{s_x}, \hat{s_y}, \hat{s_z}, \hat{r}$)
    if $fval > fval_{max}$ **then**
      $fval_{max} \leftarrow fval$
      $T^* \leftarrow T$
      $R^* \leftarrow R$
      $S^* \leftarrow S$
    end if
  end for
  return T^*, R^*, S^*
end function

Algorithm 3 A search algorithm that maximizes a target function with constraints

function FMAXCON($\{{}^{o_2}_{g_j}T\}_{j=1..J_2}, M_1, \hat{q}, \hat{p_x}, \hat{p_y}, \hat{p_z}, \hat{s_x}, \hat{s_y}, \hat{s_z}, \hat{r}$)
  given initial values: $\hat{q}, \hat{p_x}, \hat{p_y}, \hat{p_z}, \hat{s_x}, \hat{s_y}, \hat{s_z}, \hat{r}$,
  search for $q, p_x, p_y, p_z, s_x, s_y, s_z, r$ **that maximize**
    $fval =$ TARGET($\{{}^{o_2}_{g_j}T\}_{j=1..J_2}, M_1, q, p_x, p_y, p_z, s_x, s_y, s_z, r$)
  subject to:
    $0.5 \leq s_x, s_y, s_z \leq 2$
    $\|q\| = \|r\| = 1$
  $[T, R, S] \leftarrow$ ASSEMBLE($q, p_x, p_y, p_z, s_x, s_y, s_z, r$)
  return $T, R, S, fval$
end function

Algorithm 4 The target function that FMAXCON maximizes

function TARGET($\{{}^{o_2}_{g_j}T\}_{j=1..J_2}, M_1, q, p_x, p_y, p_z, s_x, s_y, s_z, r$)
  $[T, R, S] \leftarrow$ ASSEMBLE($q, p_x, p_y, p_z, s_x, s_y, s_z, r$)
  $L_{12} \leftarrow \sum_{j=1}^{J_2} \log[p(R^T S R T\ {}^{o_2}_{g_j}T | M_1)]$
  return L_{12}
end function

Algorithm 5 A function that assembles variables into matrices

function ASSEMBLE($q, p_x, p_y, p_z, s_x, s_y, s_z, r$)

$Q \leftarrow q2tr(q)$ ▷ convert unit quaternion to homogeneous transform

$R \leftarrow q2tr(r)$

$$T \leftarrow Q \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S \leftarrow \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

return T, R, S

end function

structural symmetries that may exist (e.g., a glass can be flipped end-to-end and have approximately the same set of proto-grasps as before). Another advantage of this approach is that it does not require an explicit matching process between grasp clusters, which could depend greatly based on how the examples are clustered for the different objects.

5.3 Refining Object Clusters

Equation 5.1 provides us a way to calculate a *similarity score* between a pair of objects. The similarity score between each pair of objects i and j is defined as:

$$s_{ij} = s_{ji} = \min(L_{ij}, L_{ji}), \tag{5.2}$$

where L_{ij} and L_{ji} are defined by Equation 5.1. Ideally, we can set a threshold (such as 0) for this similarity score, and group all objects with mutual similarity scores above the threshold together into a single object category. However, in reality, the similarity score between two objects belonging to different object categories can also achieve a high value. For example, the likelihood of samples captured on the mugs is usually

high given the affordance model learned on the set of glasses. This is because the grasps on the handle of a mug can be described pretty well by a girdle distribution, which is usually learned on the side of a glass.

The above observation suggests that we need some principled way to find clusters of objects that are mutually well-connected with high similarity scores. These clusters correspond to groups of objects that are mutually similar. To do this, we first introduce two visual representations of the relationships between objects. Given the similarity scores between each pair of objects, we can represent the relationships between objects using either a weighted adjacency matrix or a graph.

We define a weighted adjacency matrix as a symmetric matrix with its element, m_{ij}, calculated as:

$$m_{ij} = m_{ji} = \frac{\max(s_{ij}, 0)}{\max(s_{ii}, s_{jj})}.$$

In this equation, the similarity scores between all pairs of objects are pruned by a threshold of zero. Note that the diagonal elements corresponds to the similarity score of an object matching to itself, which is theoretically higher than that of the same object matching to other objects. This is because the learned grasp affordance model for an object is a maximum likelihood model given its own grasp samples. Given that the similarity score depends on the number of grasp clusters of an object, we use the diagonal element to normalize all the similarity scores when this object is matched to the other objects. By using $\max(s_{ii}, s_{jj})$ as the normalizer, we ensure that the normalized weighted adjacency matrix is symmetric. Therefore, an element m_{ij} in the adjacency matrix can be considered as a normalized similarity score.

An example weighted adjacency matrix is visualized in Fig. 5.1. In this figure, a weighted adjacency matrix is visualized by an image with different colors denoting the magnitudes of the matrix elements. The value range of an element (i.e., the normalized similarity score) is between 0 and 1, which is linearly mapped from blue

to red. The objects of the same true class are organized contiguously. Therefore, the non-zero similarity scores form square areas. However, we can also observe some entries in the adjacency matrix with positive similarity scores outside of the square regions, which correspond to matches between grasp affordance models of objects in different categories.

Similarly, we can visualize the relationships between objects using a weighted undirected graph. In such a graph, a vertex corresponds to an object, and an edge between two vertices is weighted by the normalized similarity score between the two corresponding grasp affordance models. An example graph is shown in Fig. 5.2, which corresponds to the weighted adjacency matrix visualized in Fig. 5.1. Again, we have pruned edges above a threshold of zero. Due to the complexity of this graph, edge weights are not shown for clarity. In this graph, nodes are color-coded based on true object categories. Although there are many extraneous edges between objects in different true categories, the object clusters are pretty clear. This is due to the fact that there are more "within-class" edges than "inter-class" edges.

The goal is to find the tightly-connected cliques in this graph, which correspond to groups of objects that are mutually similar. The expectation is that objects of the same class should have similar models and so should be able to explain each-other's data and achieve high mutual similarity scores. Furthermore, we expect that models should most often not generalize well across object classes. Hartuv and Shamir (2000) propose the Highly Connected Sub-graphs (HCS) algorithm that finds tightly-connected sub-graphs (clusters of nodes) in an undirected graph. A graph G is k-edge-connected if G remains connected after arbitrarily removing fewer than k vertices from G. Assuming that the number of vertices of a graph G is n (with $n > 1$), and G is k-edge-connected, Hartuv and Shamir define that G is called *highly connected* if $k > n/2$. A *cut* is an operation that separates a connect component of a graph by removing a set of edges. Sometimes, a cut also refers to the set of edges that

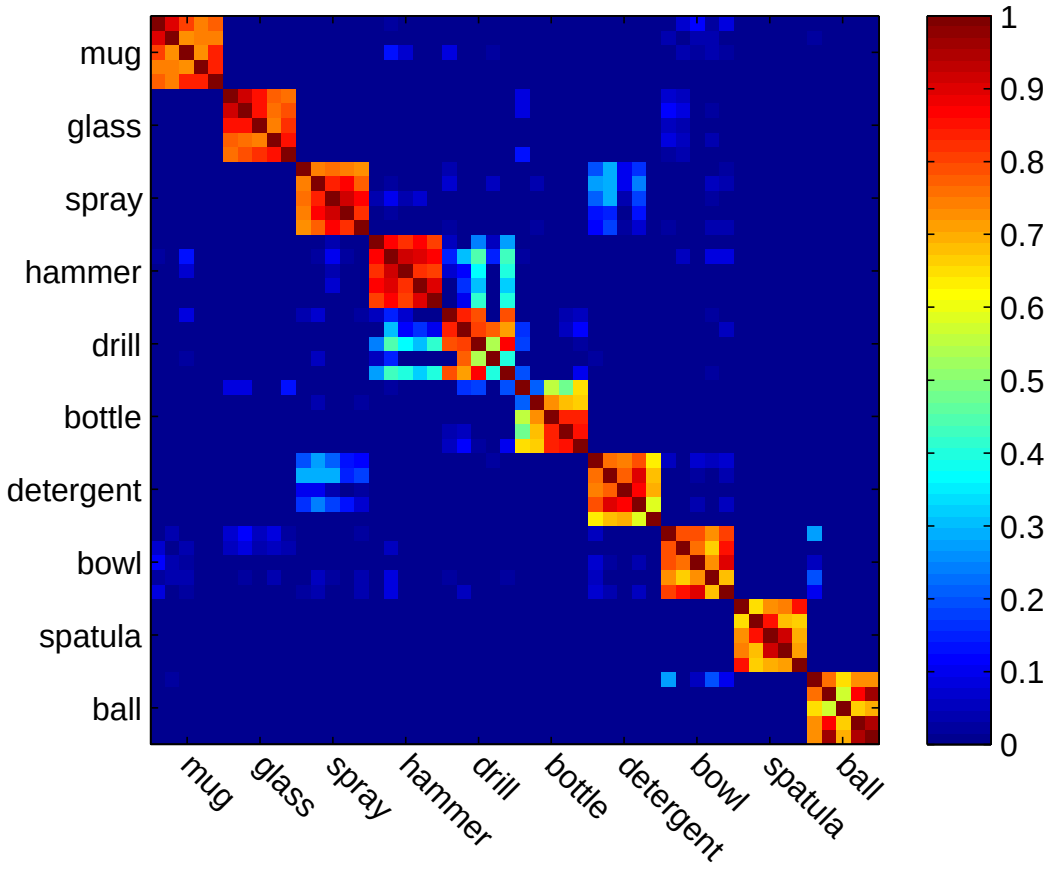


Figure 5.1: An example of a weighted adjacency matrix for all 50 objects in the OUGD before graph clustering. The axes are labeled by object categories (five individual objects in each object category). Each entry corresponds to the normalized similarity score between a pair of objects. The objects of the same true category are organized contiguously.

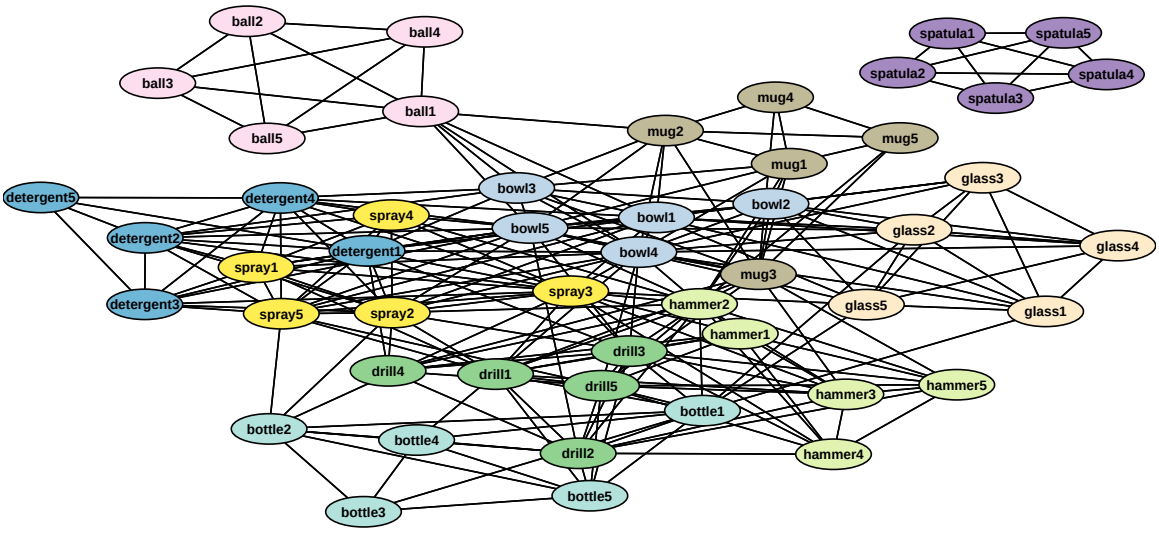


Figure 5.2: An example of the original similarity graph for all 50 objects in the OUGD before graph clustering. Nodes correspond to objects, which are color-coded based on true object categories. Edges connect similar objects (those whose similarity scores are above 0). Edges are weighted by normalized similarity scores (not shown for clarity).

have been removed. Particularly, a *minimum cut* is defined as a cut with a minimum number of edges. For a given a graph, the HCS algorithm recursively removes the minimum cut from connected components that are not highly connected, until all resultant sub-graphs are highly connected. However, they assume that the graph is unweighted, so it is not important which cut is removed as long as it is minimal. In our case, we have a weighted graph and prefer to remove the edges with lower similarity scores first. Therefore, we modify the HCS algorithm and adapt it to weighted graphs. The modified HCS algorithm is described in Algorithm 7. This algorithm recursively cuts connected components that are not highly connected, until all resultant sub-graphs are highly connected. Compared with Hartuv and Shamir's algorithm, we use function EDGECUT(G) to find a cut in a connected component G. EDGECUT(G) repeatedly removes the lowest weighted edge from G until G is disconnected. Another assumption that we make is that each cluster contains at least

one object. So, in function HCS, the search is pruned prior to the EDGECUT that would isolate the node, and returns the set of clusters before this EDGECUT.

Algorithm 6 The modified HCS graph clustering algorithm

```
function HCS(G)
    {G_i} ←FINDCC(G)                          ▷ Find connected components in G
    G' ← empty set
    for each g ∈ {G_i} do                      ▷ g: a connected component
        if g is NOT highly connected then
            g_old ← g                          ▷ Store the old g
            g ← EDGECUT(g)                     ▷ Cut g into a set of sub-graphs
            if g contains isolated node then
                g ← g_old                      ▷ Restore g
            else
                g ← HCS(g)                     ▷ Recursion on the set of sub-graphs
            end if
        end if
        G' ← G' ∪ g
    end for
    return G'
end function
```

Algorithm 7 An algorithm that finds a cut in a weighted graph

```
function EDGECUT(G)
    while G is connected do
        remove the lowest weighted edge in G
    end while
    return G
end function
```

5.4 Unified Grasp Affordance Models

The above process clusters objects into object categories in terms of how they are grasped. During the grasp affordance model matching process, Algorithm 2 not only measures the similarity score between a pair of objects, but also finds the alignment between them (specified by ${}^{o_1}_{o_2}T$, S_{12}, and R_{12} in Equation 5.1). For each object category, each object is aligned relative to the first object (arbitrarily chosen). Then,

for each object category, a unified grasp affordance model is learned by using grasp samples from all objects in this category (as opposed to a single object in the previous section). For each object category, we classify each grasp sample into one of the grasp types described by the unified grasp affordance model of this object category. Specifically, each grasp sample is assigned to its maximum likelihood class.

The object category induces a partition on the entire set of samples. Grasp type further partitions the set of samples. For instance, we may have a set of images containing different mugs and a corresponding set of grasp types: a ball grasp from the top and a handle grasp that involves both a palmar opposition and a side opposition (MacKenzie and Iberall, 1994). Likewise, we may have another set of images containing different hammers and a corresponding set of grasp types. In the latter case, there are three grasp types: a grasp from the top, a power grasp and a pad opposition about the length of the axis.

Chapter 6

Object Clustering Experiments

In order to test the performance of our object clustering algorithm, we use a 10-fold cross validation approach. For each object, we randomly split the grasp samples into 10 equal folds. Since different grasp types are demonstrated on an object in sequence, a random split ensures that each grasp type is represented in each fold. For each experiment, we use nine folds of data for grasp affordance model learning and matching (about 20,000 samples). The tuning parameters of the grasp affordance learning algorithm and their chosen values are shown in Table 6.1. These values were chosen based on exploratory experiments with reference to Palmer and Fagg (2009), and remain unchanged for all 10 experiments. Specifically, the maximum number of clusters, $B = 10$, was chosen to be larger than the largest number of grasp types demonstrated on any object (the spray bottle with seven grasp types), in order to prevent a ceiling effect. In the learning of the unified grasp affordance models, the total number of grasp types (B) was also chosen to be 10. The complexity punishment coefficient (ζ) was chosen to be 13, which is larger than the value (5) used for single-object grasp affordance model learning. This is due to the fact that more grasp samples are used for the unified grasp affordance model learning, and we have observed model over-fitting for some object categories with small ζ in exploratory experiments. Again, these parameter values remain unchanged for all unified grasp affordance model learning. More details to the sensitivity analyses on these tuning parameters can be found in de Granville (2008).

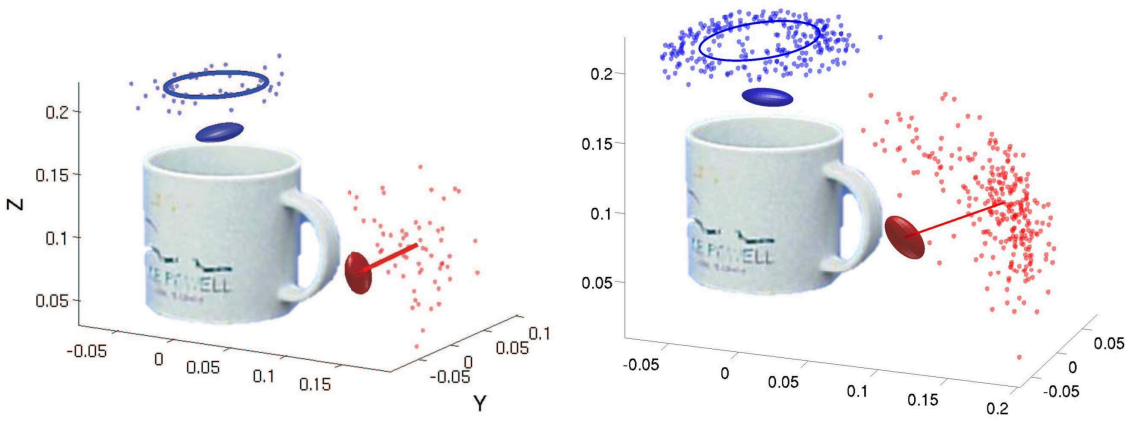
Table 6.1: The list of parameters and their chosen values for grasp affordance model learning.

Parameter	Single	Unified
Maximum number of clusters B	10	10
The number of EM attempts	40	40
The number of EM steps	20	20
ICL punishment factor ζ	5	13

6.1 Experimental Results

For each experiment, we first use de Granville et al. (2006, 2009) and Palmer and Fagg (2009)'s algorithm to learn grasp affordance models for each individual object in OUGD using 9 out of 10 training folds. We repeat this process for all 10 experiments. For comparison, the algorithm also learns a unified grasp affordance model for each object category. In order to do this, we use all 10 training folds to learn a grasp affordance model for each individual object and align the objects within each ground truth object category. Examples of both the single-object and unified grasp affordance models are shown in Fig. 6.1 to Fig. 6.10. In each of these figures, the panel on the left shows the grasp affordance model learned on nine training folds for a single object, and the panel on the right shows the corresponding unified grasp affordance model learned on all grasp samples from all objects in the same category after coordinate alignment.

In Fig. 6.1(a), a learned grasp affordance model is superimposed on the image of a mug. The 3D hand positions of grasp samples are represented as dots in the object coordinate frame. There are two clusters of grasp samples in this figure: the one near the top of the mug corresponds to top grasps, and the one near the handle of the mug corresponds to handle grasps. The ellipsoids represent grasp centroids described by 3D Gaussian distributions, which are translated by certain distances from the sensor positions (the sensor is located on the wrist). These translations are calculated on a grasp-by-grasp basis per Palmer and Fagg (2009)'s algorithm. The circle on top of

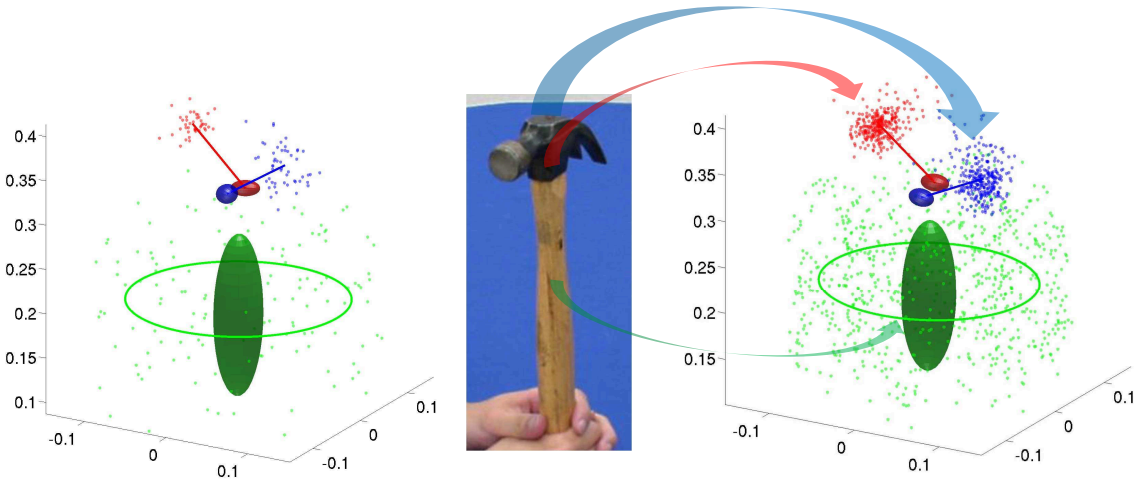


(a) Single (b) Unified

Figure 6.1: Grasp affordance models learned on the mugs using grasp samples from a single object (left), and grasp samples from all five objects (right). Blue: top ball grasp; red: handle grasp.

the mug represents a girdle distribution that captures the rotational symmetry of the top grasp. The straight line extending from the ellipsoid near the handle represents a DW distribution that captures the unidirectional property of the handle grasp. This grasp affordance model is consistent with our ground truth of demonstrated grasp types in Table 4.1. In Fig. 6.1, the grasp affordance models learned on a single mug and all 5 mugs are composed of the same number of clusters. We can also see that the population of grasp samples on the right panel is much larger than those on the left. Generally, more grasp samples produce more consistent grasp affordance models with the ground truth clustering of grasp types. However, the value of the complexity punishment coefficient, ζ, should be increased accordingly in order to prevent the grasp affordance learning algorithm from over-fitting.

Examples of grasp affordance models learned on the other object categories are shown in Fig. 6.2(a) to Fig. 6.10(a). In most of these grasp affordance models, the number of clusters found by the algorithm is consistent with the ground truth number of grasp types in Table 4.1. Exceptions are on the hammer, the drill and the spatula. For the hammer in Fig. 6.2(a), the grasp affordance learning algorithm learns

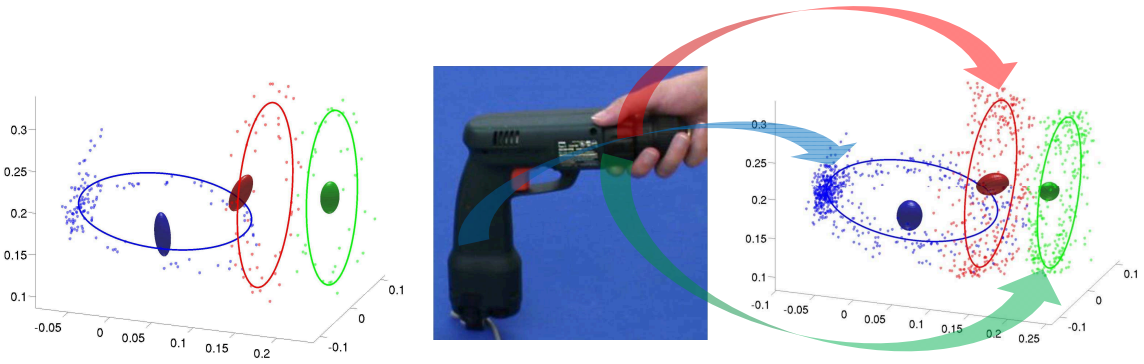


(a) Single (b) Unified

Figure 6.2: Grasp affordance models learned on the hammers using grasp samples from a single object (left), and grasp samples from all five objects (right). Red/blue: top grasps; green: handle precision/power grasps.

a single cluster (captured by a joint distribution of a 3D Gaussian and a girdle) that describes both the power and precision grasps on the handle. This is due to a relatively high complexity punishment coefficient ($\zeta = 5$). With a lower complexity punishment coefficient, the algorithm usually finds two clusters that describe the power and precision grasps separately. Also, we should note that the grasp affordance model learned on the hammers is symmetric about a 180 degrees rotation about the handle. When the algorithm aligns grasp affordance models of different hammers, there are two local maxima in the log likelihood space. Accordingly, the algorithm will select the higher local maximum in the log likelihood space. This happens if one of the hammers is rotated about 180 degrees from the others around the handle after coordinate alignment.

For the drill (Fig. 6.3a), the grasp affordance learning algorithm learns a single cluster (captured by a joint distribution of a 3D Gaussian and a girdle) that describes both the trigger (uni-directional) and handle grasps (rotationally symmetric). Again, this is due to a relatively high complexity punishment coefficient ($\zeta = 5$). With



(a) Single (b) Unified

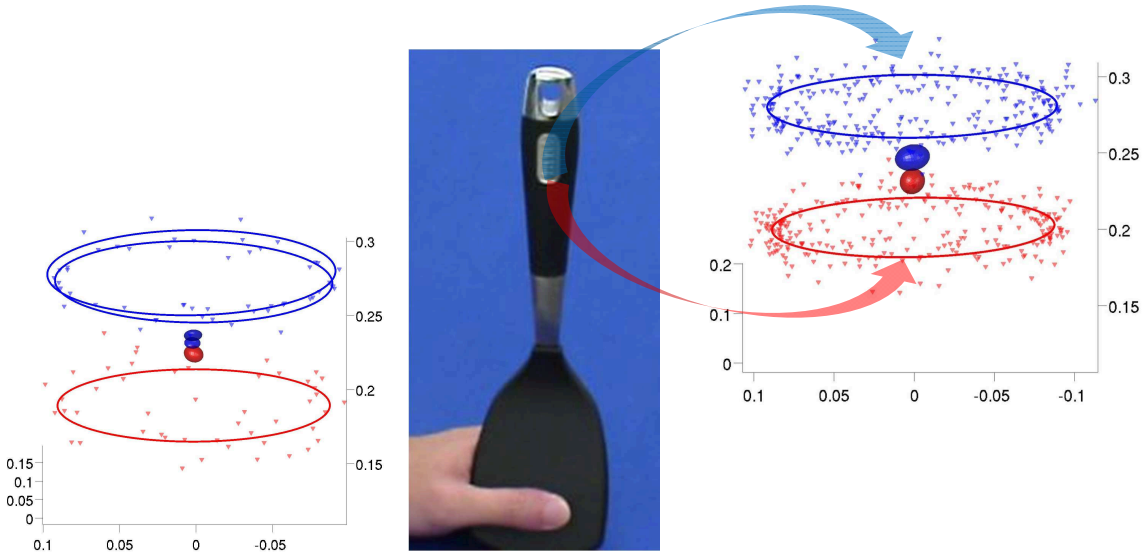
Figure 6.3: Grasp affordance models learned on the hand drills using grasp samples from a single object (left), and grasp samples from all five objects (right). Green/red: barrel grasps; blue: handle/trigger grasps.

a lower complexity punishment coefficient, the grasp affordance learning algorithm usually finds two clusters that describe these two grasp types separately.

For the spatula (Fig. 6.4a), ideally, we expect two clusters to be learned by the grasp affordance algorithm, which correspond to the overhand and underhand grasps on the handle of a spatula. However, the grasp affordance learning algorithm occasionally learns two clusters to describe one of these two grasp types. This may be due to that there is some bias away from uniform in the examples that are used in the training process.

The above results seem to recommend different complexity punishment coefficient for different object categories. However, we should note that the number of clusters learned by the grasp affordance learning algorithm has no direct influence on the performance of our grasp affordance matching algorithm. This is because the number of clusters is irrelevant when we calculate the similarity score between a pair of objects according to Equation 5.1.

For the glass (Fig. 6.5), both the single-object and unified grasp affordance models learn four clusters. This is consistent with the set of grasp types that are demonstrated by the human teacher. All four clusters are described by rotationally symmetric girdle



(a) Single (b) Unified

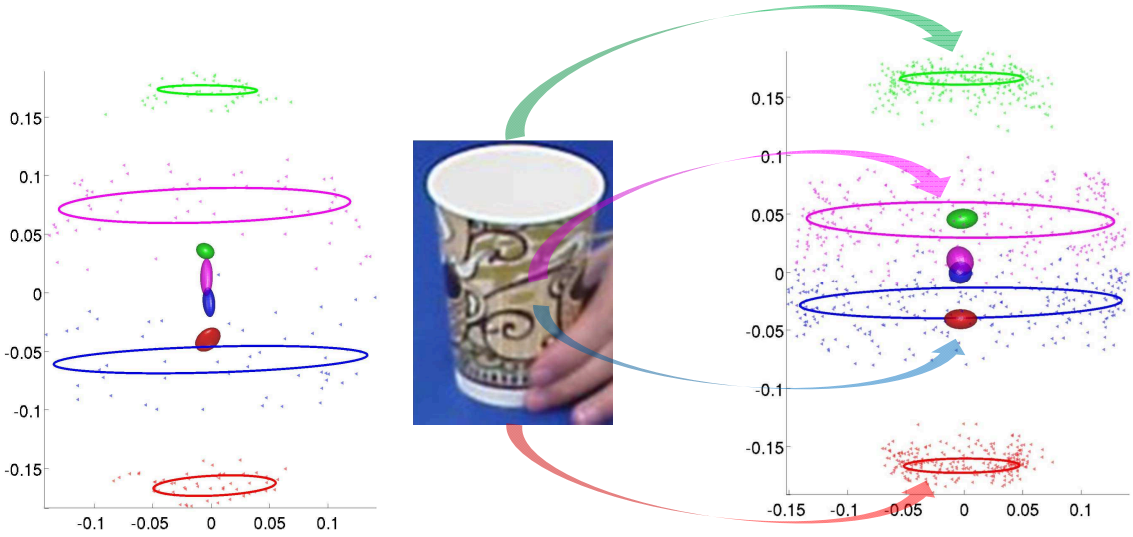
Figure 6.4: Grasp affordance models learned on the spatulas using grasp samples from a single object (left), and grasp samples from all five objects (right). Blue: overhand handle grasp; red: underhand handle grasp.

distributions.

For the spray bottle (Fig. 6.6), both the single-object and unified grasp affordance models learn seven clusters. This is consistent with the set of grasp types that are demonstrated by the human teacher. All seven clusters are described by unidirectional Dimroth-Watson distributions.

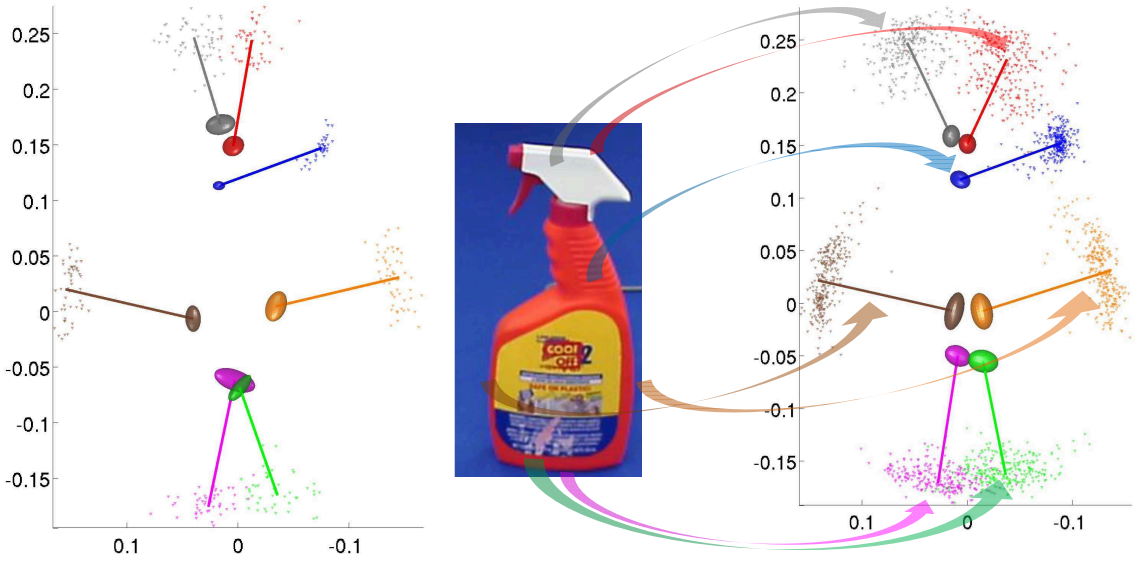
For the wine bottle (Fig. 6.7), both the single-object and unified grasp affordance models learn five clusters. This is consistent with the set of grasp types that are demonstrated by the human teacher. All five grasp types are rotationally symmetric, and therefore, are described by girdle distributions.

For the detergent bottle (Fig. 6.8), both the single-object and unified grasp affordance models learn five clusters. This is consistent with the set of grasp types that are demonstrated by the human teacher. The grasp type around the cap of the detergent bottle is rotationally symmetric, and therefore, is described by a girdle distribution. All the other four grasp types are unidirectional, and are described by



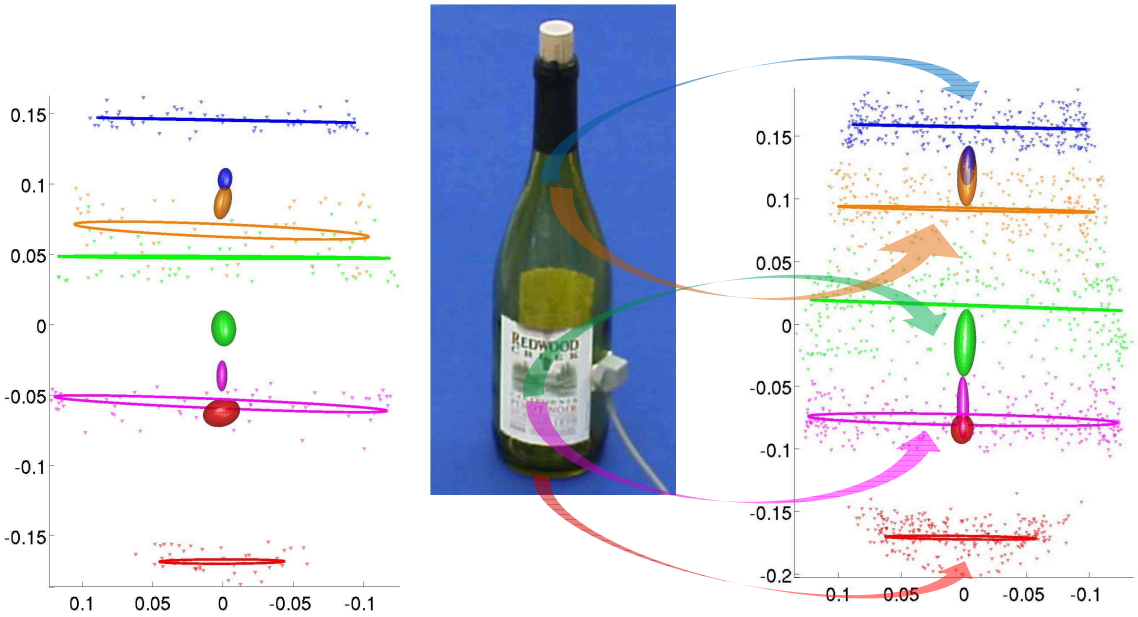
(a) Single (b) Unified

Figure 6.5: Grasp affordance models learned on the glasses using grasp samples from a single object (left), and grasp samples from all five objects (right). Green: top ball grasp; magenta: overhand side grasp; blue: underhand side grasp; red: bottom ball grasp.



(a) Single (b) Unified

Figure 6.6: Grasp affordance models learned on the spray bottles using grasp samples from a single object (left), and grasp samples from all five objects (right). Red/gray: nozzle grasps; blue: trigger grasp; brown/orange: side grasps; magenta/green: bottom grasps.



(a) Single (b) Unified

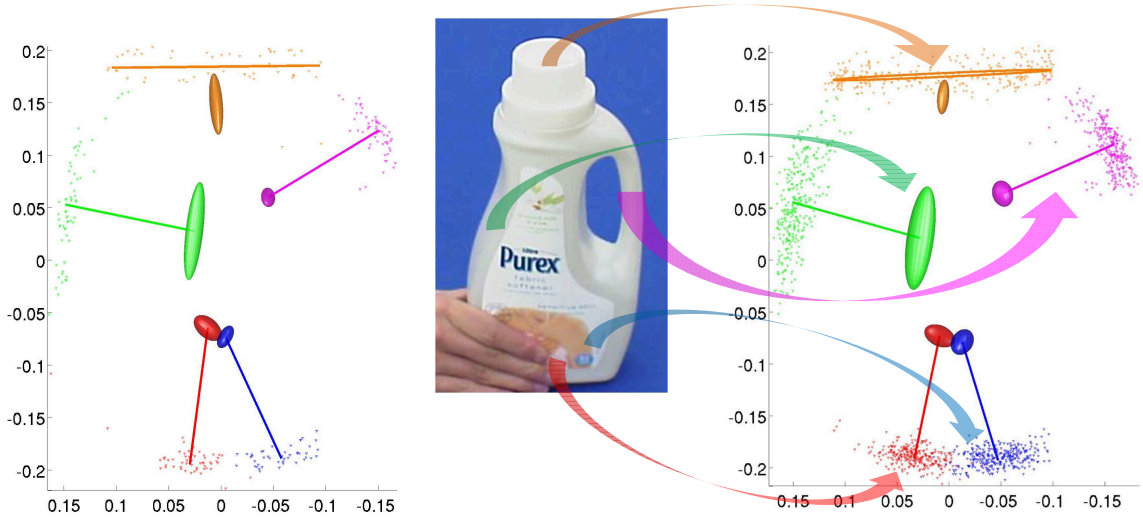
Figure 6.7: Grasp affordance models learned on the wine bottles using grasp samples from a single object (left), and grasp samples from all five objects (right). Blue/orange: neck grasps; green/magenta: side grasps; red: bottom ball grasp.

Dimroth-Watson distributions.

For the bowl (Fig. 6.9), both the single-object and unified grasp affordance models learn two clusters. This is consistent with the set of grasp types that are demonstrated by the human teacher. Both grasp types are rotationally symmetric, and are described by girdle distributions.

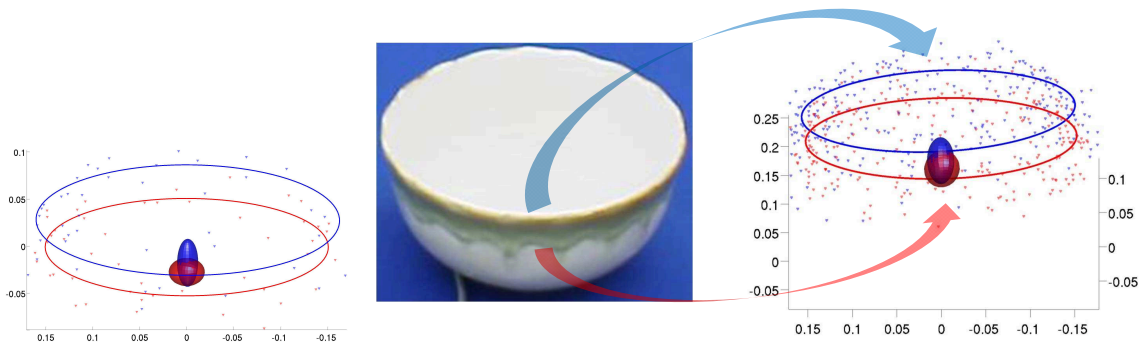
For the ball (Fig. 6.10), both the single-object and unified grasp affordance models learn a single cluster. This cluster is captured by a Dimroth-Watson distribution with a very low concentration about the mean hand orientation (the red bar). This is because the human teacher demonstrated a ball grasp with uniformly sampled hand orientations.

For the entire set of objects in OUGD, we compute their mutual similarity scores, s_{ij}, by using their grasp affordance models (Equations 5.1 and 5.2). For each experiment, we cluster objects by partitioning the similarity graph with the modified HCS



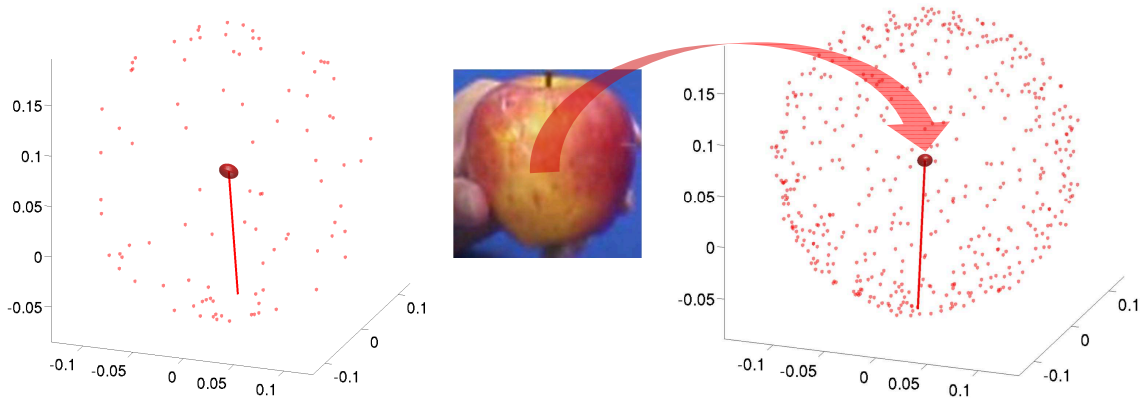
(a) Single (b) Unified

Figure 6.8: Grasp affordance models learned on the detergent bottles using grasp samples from a single object (left), and grasp samples from all five objects (right). Orange: cap grasp; green: side grasp; magenta: handle grasp; Blue/red: bottom grasps.



(a) Single (b) Unified

Figure 6.9: Grasp affordance models learned on the bowls using grasp samples from a single object (left), and grasp samples from all five objects (right). Blue: overhand rim grasp; red: underhand rim grasp.



(a) Single (b) Unified

Figure 6.10: Grasp affordance models learned on the balls using grasp samples from a single object (left), and grasp samples from all five objects (right). Red: ball grasp.

algorithm (Algorithm 7). Examples of different recursion steps for a single object clustering process is shown in Fig. 6.11 to 6.14. Fig. 6.11 shows the original graph with all edges bounded by a similarity score of zero. In this figure, we can see that the initial graph has many false positive matches between objects that belong to different categories. Two of the intermediate recursion steps are shown in Fig. 6.12 and 6.13. In these figures, the lower weighted edges are being removed gradually. In this example, nearly all of the removed edges are extraneous edges linking objects from different ground truth categories. The final solution is shown in Fig. 6.14. Note that the fifth cluster from the left (drills) is only 2-edge-connected, which is not highly connected according to the definition of Hartuv and Shamir. However, a further *EDGECUT* on this sub-graph will result in a single isolated node (drill 4). This is due to that several edges linking different drills have been removed in previous steps, since they have relatively lower similarity scores. Therefore, the object clustering algorithm terminates and returns a cluster with all five nodes.

We obtain consistent object clustering results for 9 of the 10 cross-validated experiments. In the tenth experiment, the original similarity graph contains many false positive edges linking hammers and drills. By comparing the grasp affordance mod-

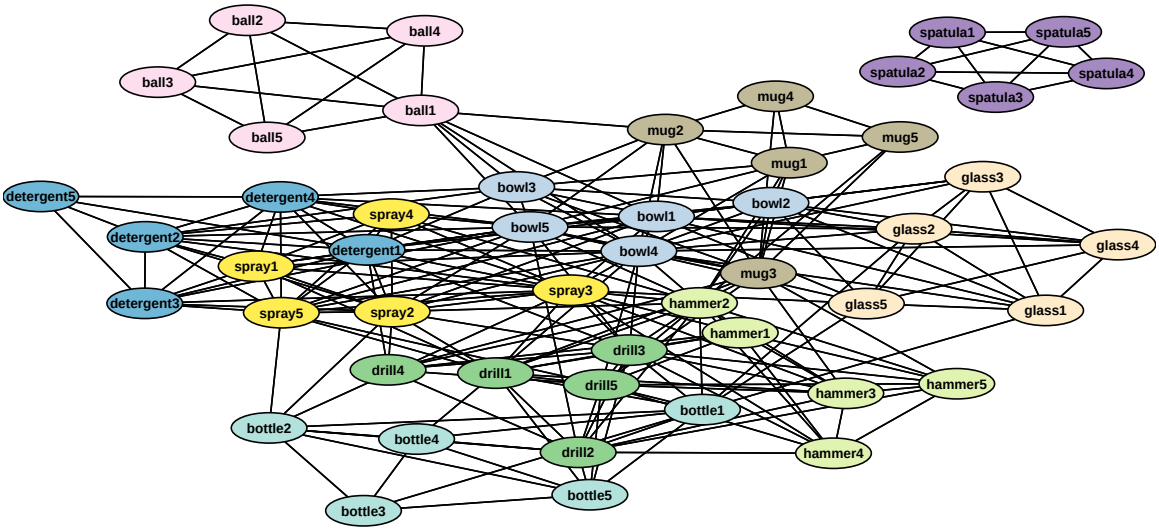


Figure 6.11: Initial graph

els learned for the hammers (Fig. 6.2a) and drills (Fig. 6.3a), intuitively we can see that the grasp affordance model of the drill can explain the samples of the hammer fairly well after a 90-degree rotation in the image plane. This will result relatively high similarity scores between hammers and drills, and some of them are even higher than those between different drills. In order to remove the extraneous edges between hammers and drills, the object clustering algorithm sacrifices some within-class edges of drills.

6.2 Sensitivity Analysis

One question that we are interested in is how much data our algorithm needs in order to cluster objects robustly. In order to answer this question, we vary the number of grasp samples used to learn grasp affordance models. In the previous sections, we have shown results with nine folds of training data. In this section, we repeat the above experiments and reduce the size of the training data set. Recall that the modified HCS algorithm will stop cutting a connected sub-graph and return the entire sub-graph

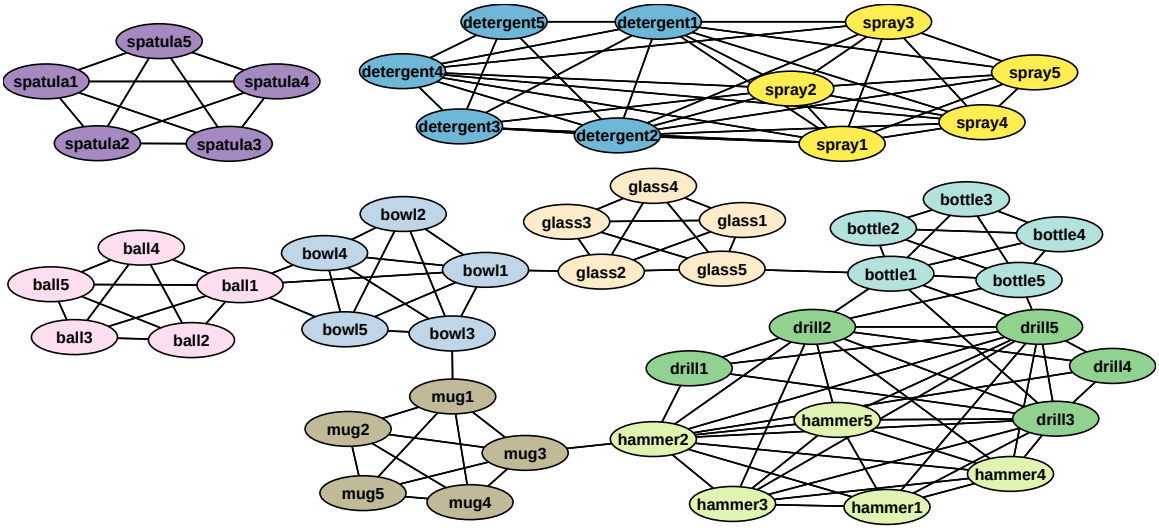


Figure 6.12: Graph after recursion 1

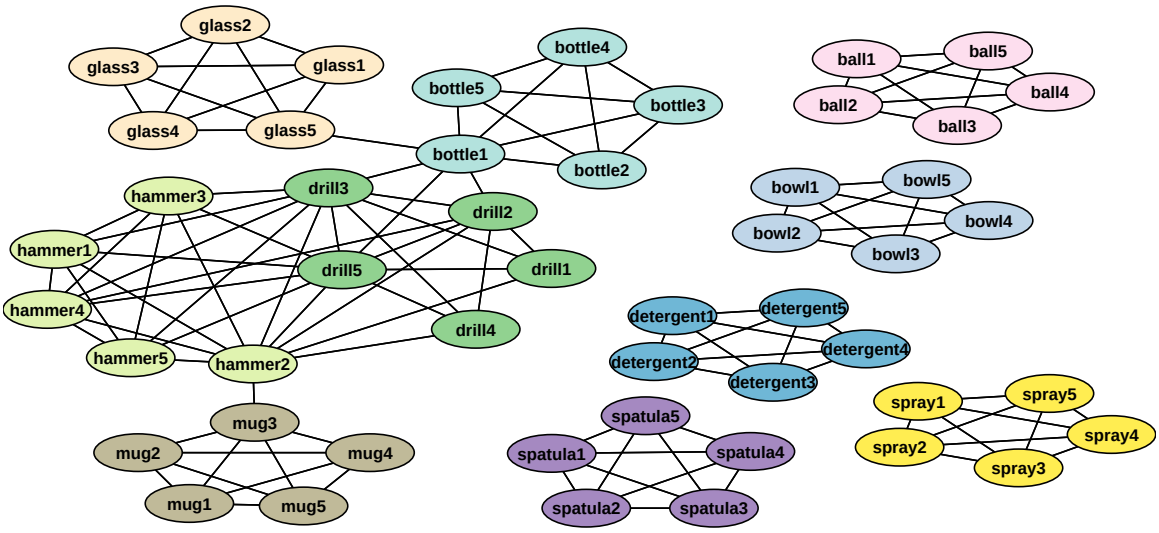


Figure 6.13: Graph after recursion 4

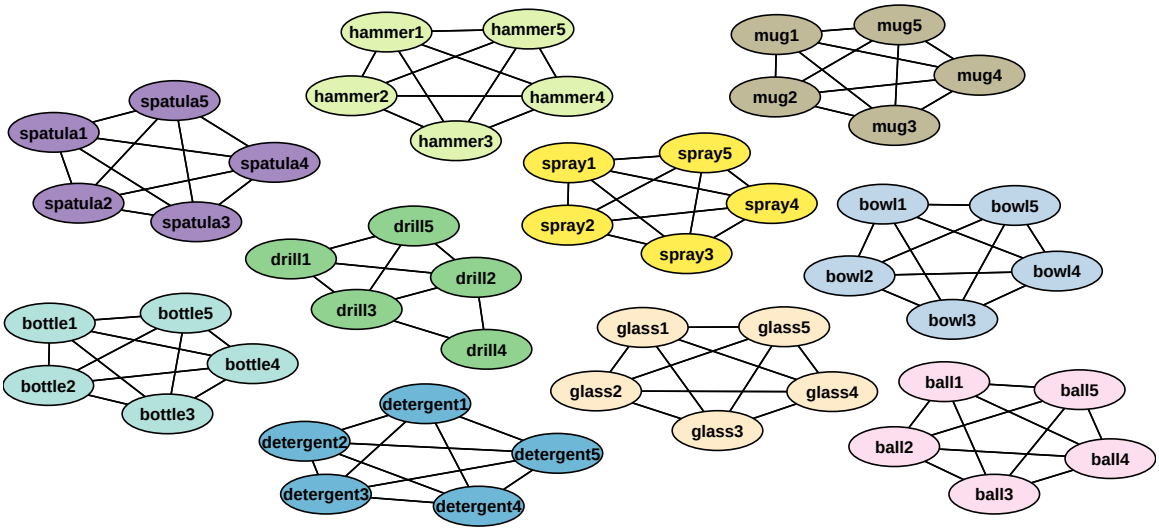


Figure 6.14: Final graph

as a single cluster in one of the two conditions: 1) there is a single isolated object that results from $EDGECUT$; 2) this sub-graph is a HCS. Also, we should note that our grasp affordance learning algorithm sub-divides the available grasp samples into one training set and three validation sets (with a 2:1:1 split). In the grasp affordance model matching process, only the training set of grasp samples (so about 1/2 of the total available grasp samples) is matched against a learned grasp affordance model.

The result of using six, seven, or eight folds of grasp samples (about 40 to 50 training grasp samples for each grasp type) is very similar to that of using 9 folds (about 60 training grasp samples for each grasp type). Using six folds of training data, in 9 out of 10 experiments, the algorithm clusters the objects in a manner that is consistent with the ground truth object categories. Specifically, the modified HCS algorithm stops cutting edges due to single isolated nodes in two experiments. One of the object partitioning results from the previous step is consistent with the ground truth, and the other is slightly different from the ground truth clustering. For the experiment that is inconsistent with the ground truth, the final set of object clusters is

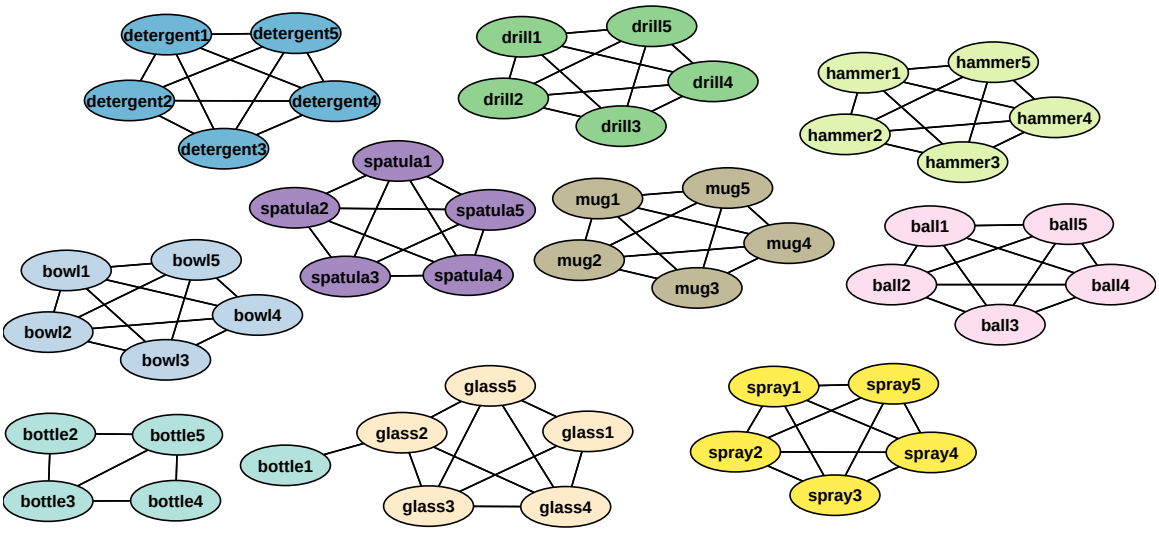


Figure 6.15: An example of object clustering using 6 folds of training data.

shown in Fig. 6.15. Our algorithm successfully clusters 8 object categories. However, one bottle is clustered together with the set of glasses. This is due to the fact that the set of grasp types demonstrated on the bottles and glasses are very similar.

Even with only 3 folds of grasp samples (about 20 training grasp samples for each grasp type), the graph clustering algorithm clusters most of the object categories correctly for some particular experiments (Fig. 6.16). However, the graph clustering algorithm fails to cluster all 10 object categories correctly for all 10 experiments. The grasp affordance models learned on such a small set of grasp samples are usually very generic. The learned grasp affordance model for an object can also explain a small set of grasp samples from an object in other object categories fairly well. This results in many inter-category edges weighted by high similarity scores that are inseparable from the within-class edges. In order to find a cut for a sub-graph that is not highly connected, the graph clustering algorithm removes some within-class edges as well as inter-category edges. The object clustering result in one of the experiments is shown in Fig. 6.16. In this example, 7 out of 10 object categories are clustered correctly

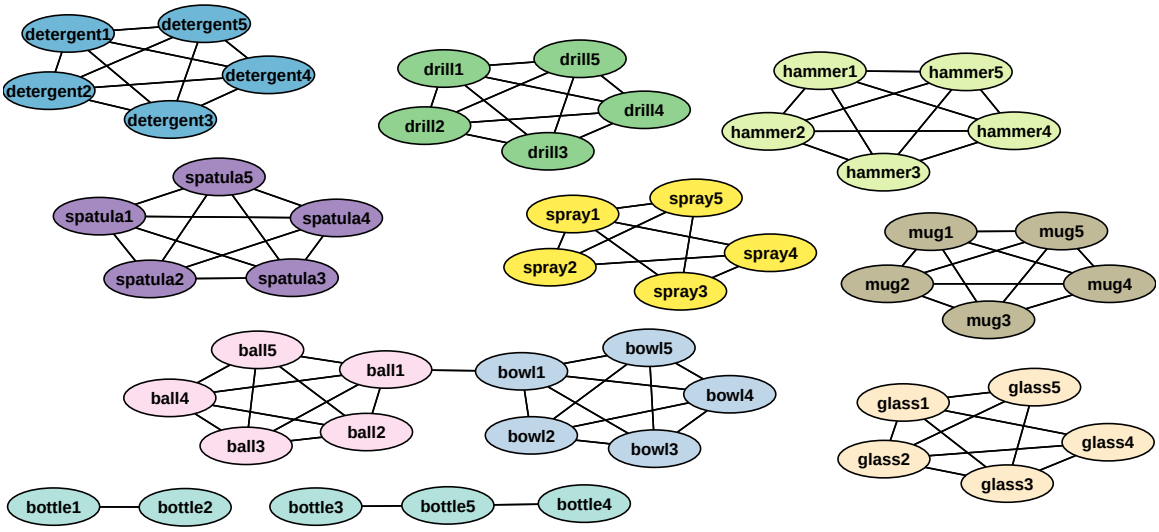


Figure 6.16: An example of object clustering using 3 folds of training data.

by the graph clustering algorithm. The cluster of balls and the cluster of bowls are grouped together by a single strong edge between ball 1 and bowl 1. Since this edge is very strong, in order to find a cut in this sub-graph, the graph clustering algorithm has to remove many other lower-weighted edges first. During this process, a single isolated node will be found and the graph clustering algorithm terminates and simply returns the sub-graph with all 10 nodes. The set of bottles is also separated into two clusters. This is because most of the edges between different bottles are not as strong as some of the inter-category edges, and have been removed during the process of finding a cut.

In order to measure the performance of our algorithm, we calculate the **Rand index** (Rand, 1971) between the clusters found by our algorithm and those by ground truth. Given a set of n elements in $S = \{O_1, ..., O_n\}$ and two partitions of S, $X = \{x_1, ..., x_r\}$ and $Y = \{y_1, ..., y_s\}$, the following is defined:

- a, the number of pairs of elements in S that are in the same set in X and in the same set in Y

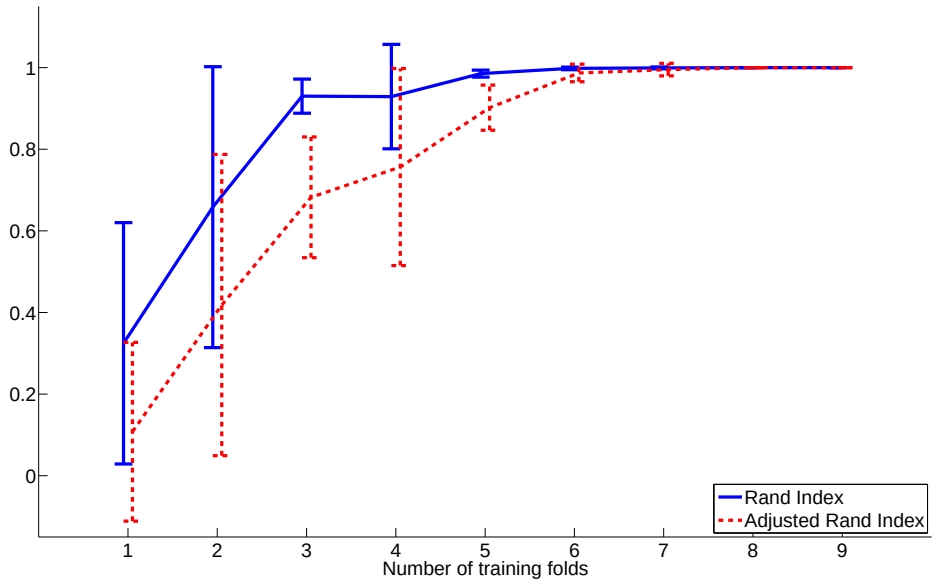


Figure 6.17: Object clustering performance with various number of training folds

- b, the number of pairs of elements in S that are in different sets in X and in different sets in Y
- c, the number of pairs of elements in S that are in the same set in X and in different sets in Y
- d, the number of pairs of elements in S that are in different sets in X and in the same set in Y

The Rand index, R, is defined as:

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}.$$

Intuitively, $a+b$ can be considered as the number of paired agreements between X and Y and $c+d$ as the number of paired disagreements between X and Y. A Rand index of 1 means perfect agreement between X and Y.

In Fig. 6.17, we show the mean and standard deviation of the Rand indices of 10 experiments as functions of training fold size. With 9 or 6 training folds, the clusters found by our algorithm are very close to the ground truth. The performance of our algorithm drops gracefully with fewer training folds. In the same figure, we also show a more strict performance measure: the adjusted Rand index (Hubert and Arabie, 1985). The adjusted Rand index adds on top of Rand index with corrections for chance agreement between X and Y:

$$\text{adjusted Rand index} = \frac{\text{Rand index} - \text{expected Rand index}}{1 - \text{expected Rand index}},$$

where the expected value of the Rand index is calculated according to Hubert and Arabie (1985). In Fig. 6.17, we can see that the adjusted Rand indices for 6 to 9 training folds are very close to the corresponding Rand indices, which are essentially close to 1. Even with this more strict performance measure, our algorithm achieves a mean adjusted Rand index close to 0.7 by using only 3 folds of training samples.

Chapter 7

Visual Learning Approach

The algorithm proposed in Chapter 5 1) clusters different objects into object categories, and for each object category, 2) clusters grasp examples into grasp types. This process induces a partition of all grasp examples by object and grasp type. The goal of the proposed visual learning algorithm is to learn a set of visual models for each partition. Particularly, these visual models can be used to recognize the key visual components that cue the grasp type.

7.1 Aspect Clustering

For each grasp type, there are multiple grasp examples across different objects. This set of grasp examples corresponds to different visual aspects of the component that has been grasped. Fig. 7.1 shows a set of image fragments of the mug handles corresponding to different examples of a handle grasp. Each image fragment corresponds to a grasp region generated by the contact location extraction process detailed in Section 4.2. We use multiple visual models to account for the appearance difference of these grasp regions. Assuming that different objects in an object category are already aligned by their coordinate frames, the proposed visual learning algorithm groups image fragments according to their aspects. Since these objects are already aligned, the image fragments corresponding to similar aspects have similar visual appearance. We discover the set of similarly-looking image fragments corresponding to neighboring aspects by clustering the camera orientations at which they are taken. Then, a single visual model is learned from the set of similarly-looking object components contained

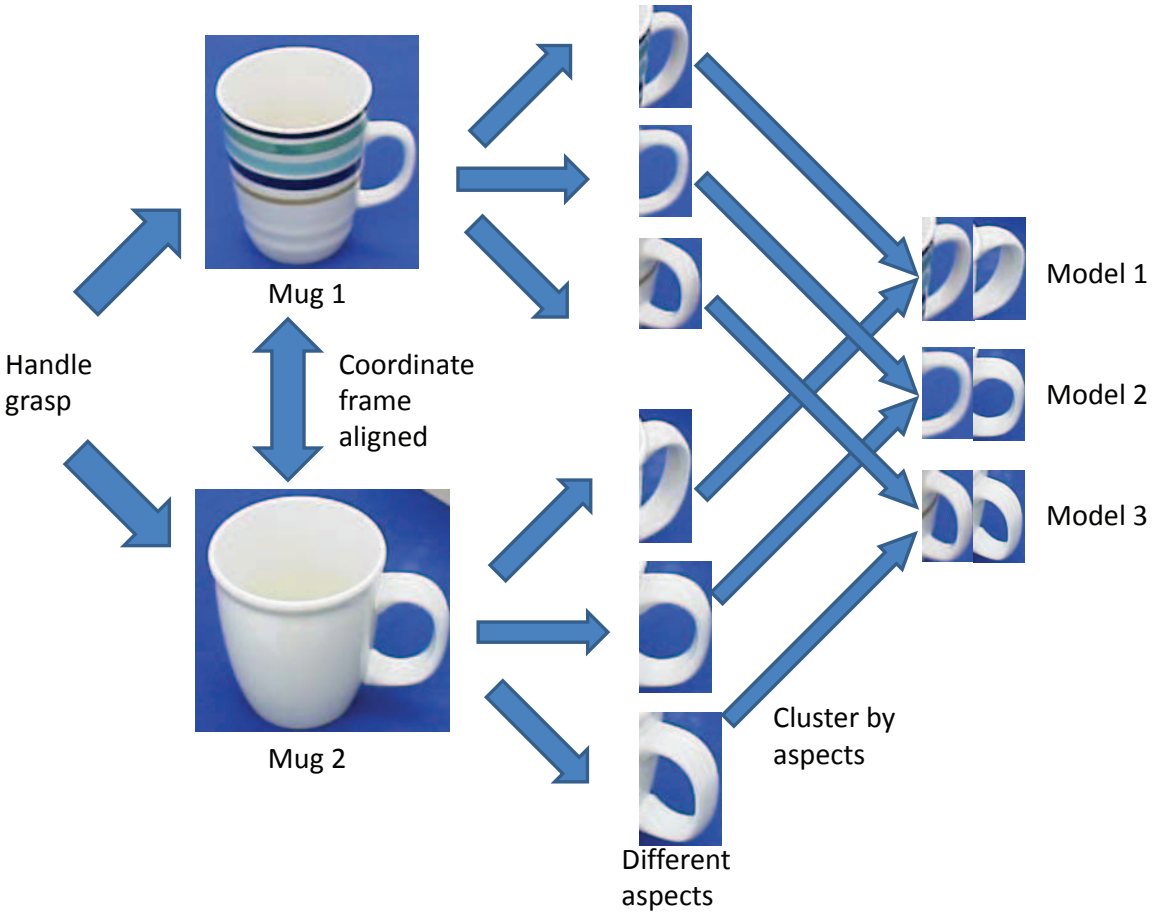


Figure 7.1: This figure illustrates how to cluster the grasp regions from multiple images for the handle grasp. First, the coordinate frames of the two mugs are aligned. For each mug, we assume that a set of sub-images that correspond to different aspects of the mug handles is extracted by identifying the grasp regions. Then, the similarly-looking sub-images are grouped together by aspect clustering across the mugs.

in the image fragments.

In Fig. 7.2, an *aspect sphere* is defined as a unit sphere in the object-centered coordinate frame. Each point on the aspect sphere encodes a camera position with the camera optical axis pointing to the object. For convenience, we attach coordinate frames to both the object and the camera as shown in this figure. As a means of organizing our data, the Z' axis of the object coordinate frame is defined by the symmetry axis if this object is rotationally symmetric. Otherwise, we directly use the coordinate frame defined by the Polhemus sensor that is attached to this object.

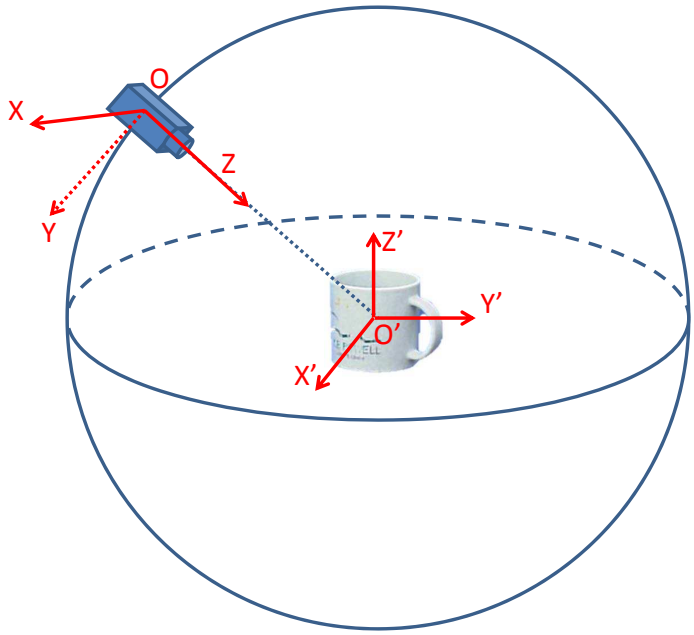


Figure 7.2: The aspect sphere.

The coordinate frame of the camera is defined in the standard way, with the Z axis aligning with the optical axis. We choose the orientation of the camera about Z, so that Y and Z' are aligned with one-another (though in opposite directions). Assuming that the camera is always pointing to the object center, O', the camera pose can be defined by a unit vector, $\overrightarrow{OO'} = [u_x, u_y, u_z]$, and the camera X direction. In our data collection process, the object is not intentionally rotated in the image plane. This choice implies that, while the sampling process covers the entirety of the aspect sphere, only one in-plan orientation is sampled at each point on the aspect sphere. This reduces the number of required samples during the data collection process and allows us to assume that no in-plane rotations must be accounted for in the shape model training process. However, there are exceptions at the north and south poles of the aspect sphere. For these aspects, in-image-plane rotations do exist due to the fact that we rotate the object about the Z' axis during data collection. As a result, the algorithm learns several different shape models for these aspects.

In order to learn shape models that are robust, it is important that they are responsive to small changes in viewing angle. The training set for a single shape model is selected from a cluster of image samples on the aspect sphere. The grasp affordance clusters are of two forms: unidirectional and rotationally symmetric. These each suggest object geometries that should also be reflected in the learned shape models. In particular, the visual models that are learned for a rotationally symmetric grasp should focus on visual features that are also invariant to rotations about the same axis. To this end, image samples are clustered together for the purposes of learning a specific shape model in one of two ways. For unidirectional grasp affordances, the image samples are drawn from a circular cap on the aspect sphere. These clusters are found based on all three components of the unit vector $\overrightarrow{OO'}$, together with a unit vector that defines the camera X direction. The camera X direction is added in order to differentiate different in-image-plane rotations, particularly for the aspects around the north and south poles of the aspect sphere. For rotationally symmetric grasp affordances, the image samples are drawn from a band around the aspect sphere that is perpendicular to the axis of rotation. These bands are found by clustering the u_z component of the unit vector $\overrightarrow{OO'}$ in the coordinate frame $X'O'Z'$.

In our implementation of the proposed algorithm, we use the mean shift algorithm (Comaniciu and Meer, 2002) to cluster aspects. One advantage of using the mean shift algorithm is that we do not need to commit to a certain number of clusters ahead of time. Per the above discussion, for unidirectional grasp types, the mean shift algorithm clusters a set of 6D vectors (3D for camera Z direction and 3D for camera X direction) on the aspect sphere. For rotationally symmetric grasp types, the clustering process takes place within a one DOF space, corresponding to the latitude on the aspect sphere, in which the polar axis is aligned with the axis of symmetry. Since the mean shift algorithm does not assume the shape of clusters *a priori*, we use a normal kernel for all our analysis. The only tuning parameter to the mean shift

algorithm is the kernel bandwidth, which determines the resolution of the clustering. This parameter is selected based on the stability and repeatability of the clustering results across different parameter values, as well as task-related domain knowledge. In our case, we choose the kernel bandwidth by observing how different are those images in a single cluster found by the mean shift algorithm. Generally, the set of images in a cluster should be similar enough in order for the visual model learning algorithm to learn a coherent shape model. In order to reduce the total number of visual models learned for a single grasp type, the set of images in a cluster should also cover a range of aspects on the aspect sphere. We chose kernel bandwidths separately for the unidirectional and rotationally symmetric grasp types based on exploratory experiments, and fix these values for all experiments described in Chapter 8.

Examples of aspect clusters that are found by the mean shift algorithm are shown in Fig. 7.3. In this figure, the dots are the camera aspect for the entire set of grasp samples of the five mugs. The coordinate of the mug is defined the same way as in Fig. 7.2. Each mug is aligned relative to the first mug (arbitrarily chosen) during the affordance model matching phase. Therefore, a point on the aspect sphere corresponds to consistent camera aspect on the different mugs. The entire set of grasp samples of mugs are first clustered by different grasp types described by the unified grasp affordance model (Section 5.4). In this figure, red dots correspond to examples of the handle grasp, and blue dots correspond to examples of the top grasp. For each grasp type, the mean shift algorithm further clusters the corresponding grasp examples. Two example clusters are highlighted and the corresponding image fragments contained in each cluster are shown. The cluster of image fragments on the lefthand side corresponds to handle grasps. The corresponding camera aspects on the aspect sphere aggregate around a single spot. The area of this spot reflects the resolution of the mean shift clustering algorithm, which is determined by the kernel bandwidth. In this cluster of image fragments, we can see that the object components (handles) are

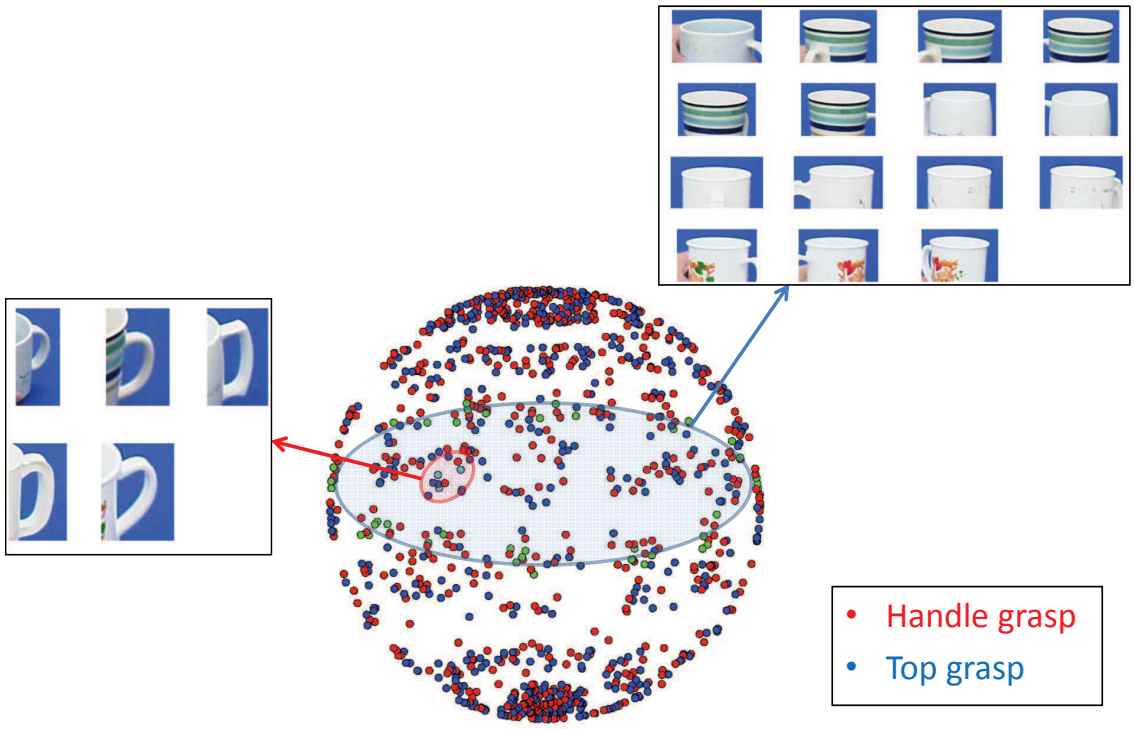


Figure 7.3: Example aspect clusters found by the mean shift algorithm. Each dot corresponds to the camera location of a grasp sample on the aspect sphere. Red dots: handle grasp. Blue dots: top grasp. Cyan dots: an example aspect cluster for the handle grasp. Green dots: an example aspect cluster for the top grasp. The corresponding object components are shown.

viewed from similar aspects, and therefore have similar visual appearance. Similarly, the cluster of image fragments on the righthand side corresponds to ball grasps from the top of mugs. The corresponding camera aspects on the aspect sphere aggregate around a band of the aspect sphere. The width of the band is determined by the kernel band width. Although these aspects spread out on the aspect sphere, the corresponding image fragments in this cluster still have similar visual appearance. This is due to the fact that the mugs are rotationally symmetric about the Z axis, and the camera orbits about the same axis. For each aspect cluster, we can see that the corresponding object components look visually similar to each other, which provides a set of related shapes that makes the visual model learning process feasible.

7.2 Visual Model Learning

For each aspect cluster, the visual learning algorithm learns a *shape model* by using PAS features (Ferrari et al., 2010) (detailed in the background section 2.4.2). For each aspect cluster, the algorithm uses both the left and right images in the stereo image pairs to learn a single shape model. This visual model learning process on average generates about 20 shape models for each grasp type that corresponds to different aspects. The algorithm further uses two steps of filtering to remove low-quality shape models and at the same time, determine the optimal model threshold t_i^* for each shape model i (where i is the model index).

High quality shape models are those that 1) match the training set well, and at the same time, 2) will not find many false positive matches on images that contain objects with different shapes. The first step of filtering is to match a learned shape model back to the set of training images in which this shape model is learned. The quality of the shape model is measured by the energy needed to match the set of model points to the corresponding points in a training image. This energy is measured by the amount of warping contained in the aligning TPS transform. Since a shape model is learned from a set of training images, we use the average energy of this set of images as a quality measure. The details of the shape model matching process and energy computation can be found in the background section 2.4.2. Intuitively, a low energy means that the learned shape model has a similar shape as the training image. However, this is not the case when the training images have very different shapes, which may be due to errors in the grasp region extraction process. For example, an aspect cluster for a handle grasp may be composed of a set of aspects where the mug handle is self-occluded and invisible. Due to this fact, the corresponding image fragments will contain random object components that are accidentally occluded by the grasping hand instead of mug handles (these are not occluded at all). Therefore,

a shape model learned from these image fragments will try to generalize to unrelated shapes and fail to match any particular training image very well. Due to the above reason, this shape model will have high average energy. Examples of models with their average energy are shown in Fig 7.4. In this figure, we can see that the shape model with lower average energy (left) is more discriminative and identifiable compare to the one with higher average energy (right).

For each grasp type in each object category, we keep the top k shape models based on their average energy. The basic idea is to select the value of k such that the different aspects corresponding to a grasp type are well covered. The number of shape models needed for each grasp type in order to cover the different aspects, not only depends on the robustness of the shape model, but also depends on the degree of symmetry of the object component. For example, a single shape model is sufficient to describe the top of a glass when it is rotated about its symmetry axis in front of a fixed camera. As a contrast, for object components without symmetries, more shape models are needed to account for shape variations due to aspect change. In the meanwhile, we would like to filter out a nontrivial number of shape models that have high average energy. Empirically, we choose to keep at least 80 percent of the learned shape models for each grasp type. If the number of learned shape models for a grasp type is too small (≤ 10 for a DW distribution or ≤ 5 for a girdle distribution), we simply keep all of them. Based on the above, we determine the value of k for each grasp type as follows:

$$k = \begin{cases} \min(m, \max(m \times 0.8, 10)) & \text{if DW} \\ \min(m, \max(m \times 0.8, 5)) & \text{if girdle} \end{cases},$$

where m is the total number of shape models learned for this grasp type before filtering.

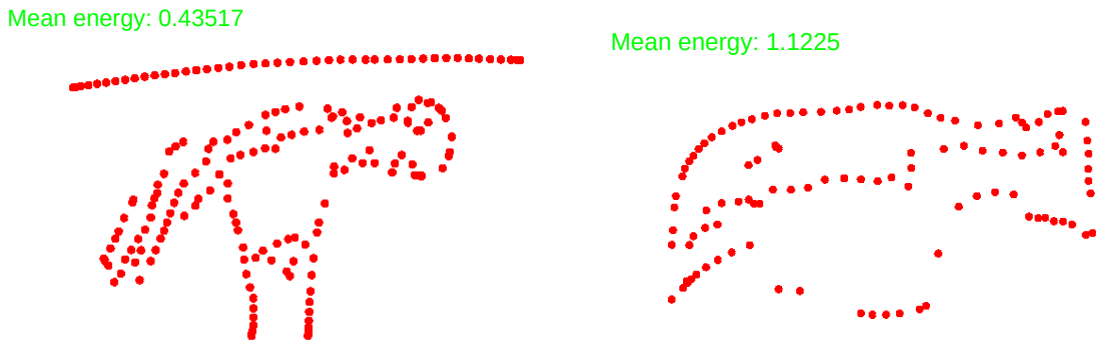
Besides measuring the quality of a shape model based on the training set of images,

we also would like to know how discriminative this shape model is when matching it to images that do not contain the target shape. Ideally, a shape model should be locally robust and globally discriminative. That is, a shape model should match very well with images that correspond to similar aspects as those of the training images, and at the same time, reject images that correspond to very different aspects. Therefore, in the second step of filtering, we measure the performance of a shape model on a separate validation set of images drawn from all different aspects. Given a set of *positive images* (where the target shape is observable), and a set of *negative images* (where the target shape is not observable), the performance is measured by the relative numbers of correct and incorrect matches. In this step of filtering, we only keep shape models with performance higher than a certain performance threshold. The individual model threshold t_i is selected as the one that maximize the chosen performance measure. In order to measure the relative numbers of correct and incorrect matches on a validation set composed of positive and negative images, we choose the κ statistic (Cohen, 1960) as the performance measure. Specifically, we calculate the κ statistic of a shape model when varying the threshold values. The κ statistic is defined as:

$$\kappa = \frac{\text{observed agreement} - \text{chance agreement}}{1 - \text{chance agreement}}, \tag{7.1}$$

where *observed agreement* is the fraction of correct image labellings by the algorithm ((TP+TN)/(TP+TN+FP+FN); TP = number of true positives; TN = true negatives; FP = false positives; FN = false negatives) and *chance agreement* is the fraction of correct image labellings by the best algorithm that has no information about the image. $\kappa \leq 0$ is interpreted as performance being no better than the fixed strategy; $\kappa = 1$ is interpreted as having perfect performance.

Unfortunately, for a given shape model, it is difficult to measure P/N exactly, due to the fact that the ground truth labeling of an image with regard to this shape



(a) A model with low average energy (b) A model with high average energy

Figure 7.4: Examples of shape models with average energy needed to match back to the training set of images.

model, does not exist (each image only has one demonstrated grasp). For a subset of validation images, whose aspect is close (down to symmetry) to the training images in which this shape model is learned, we may label them as positive for this particular shape model. However, by doing so, we may overestimate the positive set of images. Even though the aspect is consistent, the target object component may still be occluded by either the hand or the object itself. Even the images from which a shape model is learned may or may not actually contain this shape model. For instance, in some cases, a shape model of the mug handle is learned on the images in which the handle is self-occluded and invisible. This is in part why we need this filtering process to remove these "bad" models.

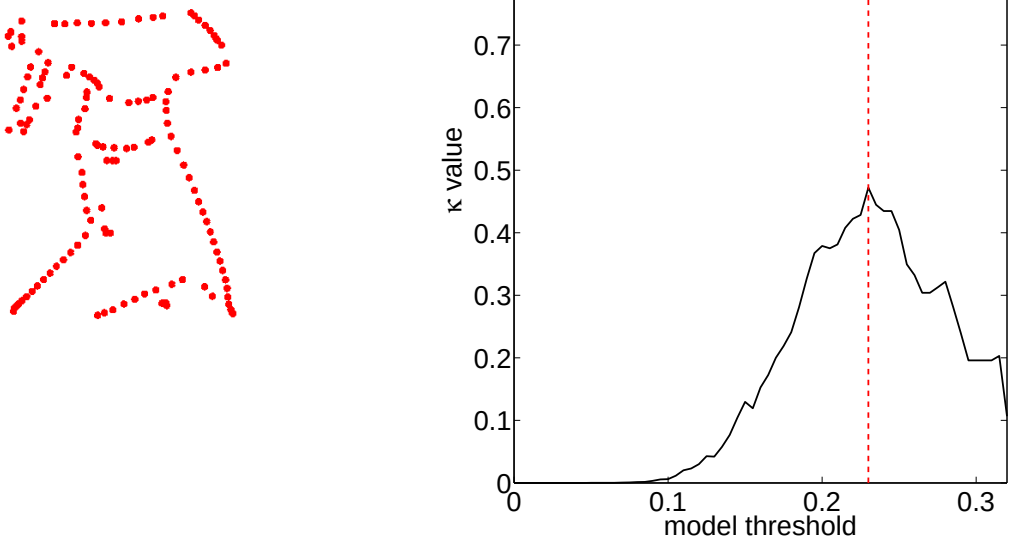
Instead, we use the following approach to find approximations to the true P/N. For each learned shape model, we match it to a validation set of images from the same object category. For each validation image, the shape model matching algorithm either finds a match of this shape model or not. If the algorithm finds a match, it also provides a matching score. Correspondingly, we use the algorithm proposed in Appendix B to estimate a ground-truth labeling for a given validation image and a match of a shape model. This labeling algorithm automatically labels a match

as either correct, wrong or unknown. Images that are labeled as unknown are not considered by the κ statistic. The set P is approximated by the set of images in which the matched shape model is labeled as *correct*. The set N is approximated by the set of images in which the matched shape model is labeled as *wrong*. By doing this, we assume that whenever the target object component shows up in a validation image, the shape model matching algorithm will detect it, and therefore the labeling algorithm will label this match as correct. Then, we define a model threshold, t_i, for model i that determines whether the corresponding shape model is detected or not in a given image. We select t_i that maximizes the κ statistic.

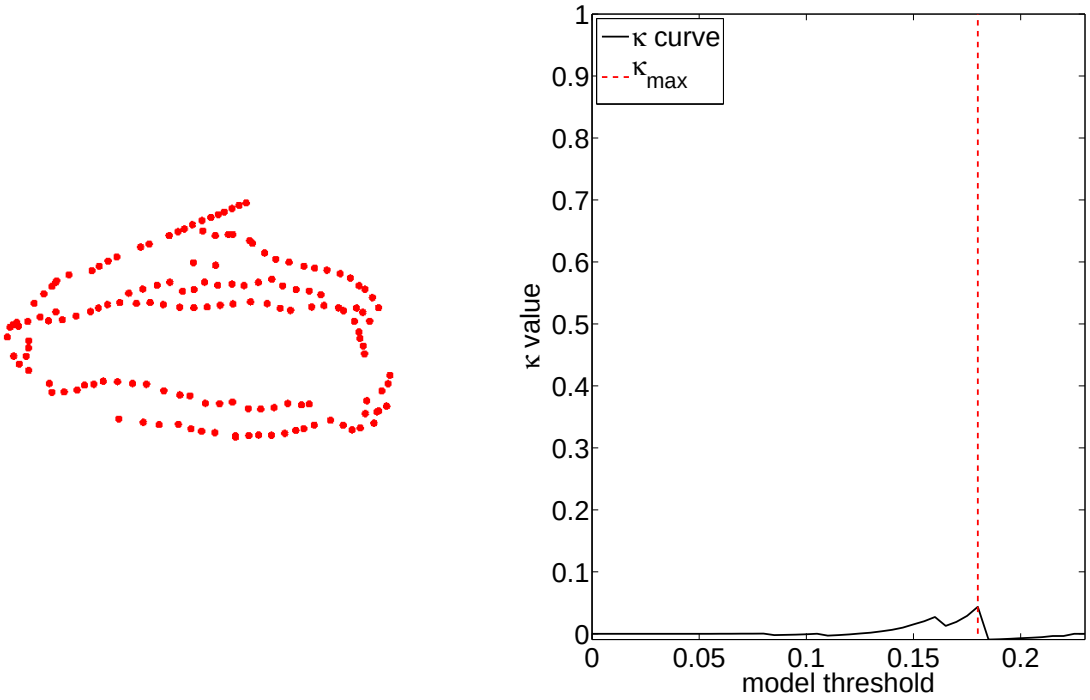
Examples of shape models with their corresponding κ curves are shown in Fig. 7.5. In this figure, both of the shape models are learned from a set of images when the spray bottle is grasped by the trigger. The shape model on the left corresponds to a frontal view of the spray bottle, while the shape model on the right corresponds to a top view. Intuitively, the shape model on the left is more discriminative and visually identifiable than the one on the right. The dotted vertical line indicates the selected model threshold t_i^*, which maximize the κ statistic. The corresponding maximum κ value, κ_{max}, is used as the quality measure of a shape model. In this filtering step, we only keep shape models with κ_{max} above some empirically determined threshold, κ_{th}. The shape model in Fig. 7.5(a) has a much higher κ_{max} than the one in Fig. 7.5(b), which is consistent with our intuition that the former is more discriminative than the latter.

7.3 Visual Model Matching

The visual learning process is done once a set of shape models are learned for some grasp types. In the testing phase, for a test image that contains a novel instance of an object from a previous known object category, we use the shape model matching



(a) A model with high κ value



(b) A model with low κ value

Figure 7.5: Examples of shape models with κ curves

(a) Single image matching

(b) Stereo matching

Figure 7.6: Each row corresponds to a stereo image pair taken by the left and right cameras. (a) A model of the mug top is matched to each image individually. (b) A model of the mug top is matched to both images simultaneously.

algorithm detailed in the background section 2.4.2 to match each learned shape model to this image.

Some example matches are shown in Fig. 7.6. In this figure, each row of images comes from a stereo image pair taken by the left and right cameras. In the first row of images, a shape model of the mug top is matched to each image individually. While this approach generally works well, the algorithm occasionally ends up with a mismatch on the background clutter. This motivates us to make use of the redundant information provided by both images in a stereo image pair.

7.3.1 Matching Shape Models with Stereo Constraints

By assuming that the target object is of similar distance to both cameras, the following holds: 1) the target shapes to be matched in a stereo image pair have similar scales, 2) the affine transforms and TPS warpings between the model and the target shapes across the stereo image pair are similar, and 3) the matched points in a stereo image pair correspond to each other (even though there may be small errors induced by the matching process). The corresponding points in a stereo image pair must also satisfy the *epipolar constraint* (Hartley and Zisserman, 2004). Specifically, the positions of a 3D point in two 2D images taken by two cameras from distinct locations are related by the epipolar constraint:

$$\mathbf{x}'^{T} F \mathbf{x} = 0,$$

where F is a 3×3 matrix known as the *fundamental matrix*, $\mathbf{x}'$ and $\mathbf{x}$ are the homogeneous coordinates of the 3D point in the first and second images respectively. In our experimental setup, since the two cameras are approximately related by a horizontal translation assuming the focus is sufficiently far, the epipolar constraint reduces to a simple scalar form:

$$y' = y, \tag{7.2}$$

that is, the y coordinates of the 3D point in a stereo image pair must be equal.

Based on the above idea, the process of matching a shape model to a stereo image pair consists of two steps. The first step is a Hough style voting process that combines evidence from both of the left and right images. First, the algorithm finds a set of candidate matches by applying Hough voting to each image separately. Given candidate i for the left image and candidate j for the right image, I define the following

function to measure the cost of combining these two candidates:

$$cost(i,j) = \exp\left(\frac{(y_i - y_j)^2}{2\sigma}\right) * \max\left(\frac{s_i}{s_j}, \frac{s_j}{s_i}\right). \tag{7.3}$$

The first term encodes the simplified epipolar constraint (Equation 7.2), whose value is minimized when candidates i and j have the same y coordinates. The second term measures the similarity of the two candidates in scale, whose value is minimized when candidates i and j have the same scale. The shape model matching algorithm uses Equation 7.3 to measure the cost of each pair of candidate matches in the left and right images. Then, the algorithm selects the top k pairs that give lowest costs. Each pair of candidate matches is composed of a candidate match in the left image and a candidate match in the right image.

The second step is to apply TPS-RPM for all k candidate pairs. A candidate pair in the first step specifies an initial center location and scale of the model in the left and right images respectively. In each image, we apply the single-image TPS-RPM algorithm as discussed previously. In our stereo matching algorithm, we assume that the distance between the two cameras is much smaller than the object distance. Therefore, the left and right images of the object should look similar to each other (i.e., the parallax is small). Based on this assumption, we match the shape model to both images by using two TPS mappings that are similar to each other.

After applying stereo constraints, the modified TPS-RPM procedure for a candidate pair is given in Algorithm 8. In this algorithm, there are actually two TPS-RPM processes running in parallel: one for the left image, and the other for the right image. The input variables to this algorithm are the two sets of model points in the left and right images, V_l and V_r, the two sets of image points in the left and right images, S_l and S_r, the deformation model, M_{deform}, which is learned from the training images, the initial and final values of Γ, Γ_{init} and Γ_{final}, the simulated annealing rate,

Algorithm 8 TPS-RPM

procedure TPS-RPM($V_l, V_r, S_l, S_r, M_{deform}, \Gamma_{init}, \Gamma_{final}, anneal\_rate, N$)
  $\Gamma = \Gamma_{init}$
  $V_l' = V_l$
  $V_r' = V_r$
  while $\Gamma > \Gamma_{final}$ **do**
    for $i = 1$ to N **do** ▷ repeat several times at each temperature
      ∇ given TPS warped model points, image points, update M
      M_l=CALC_M(V_l', S_l, Γ)
      M_r=CALC_M(V_r', S_r, Γ)
      ∇ given M, update the mapped image points
      $U_l \Leftarrow M_l S_l$
      $U_r \Leftarrow M_r S_r$
      ∇ constrain by deformation model
      U_l=PROJECT_INSIDE_VALID_REGION(U_l, M_{deform}, Γ)
      U_r=PROJECT_INSIDE_VALID_REGION(U_r, M_{deform}, Γ)
      ∇ given model points, mapped image points, update TPS transform
      $[w_r, d_r, w_r]$ =CALC_TRANSFORM(V_l, U_l)
      $[w_l, d_l, w_l]$ =CALC_TRANSFORM(V_r, U_r)
      ∇ apply stereo constraint
      $w_l \Leftarrow w_l + \Gamma(w_r - w_l)$
      $d_l \Leftarrow d_l + \Gamma(d_r - d_l)$
      $w_r \Leftarrow w_r + \Gamma(w_l - w_r)$
      $d_r \Leftarrow d_r + \Gamma(d_l - d_r)$
      ∇ transform model points by the current TPS
      V_l' =WARP_POINTS(V_l, w_l, d_l)
      V_r' =WARP_POINTS(V_r, w_r, d_r)
    end for
    $\Gamma = \Gamma * anneal\_rate$ ▷ note that $0 < anneal\_rate < 1$
  end while
  return $M_l, M_r, V_l', V_r', U_l, U_r, w_l, d_l, w_r, d_r$
end procedure

anneal_rate, which is a scalar in the range of 0 and 1, and the number of iterations per temperature, N. Here, Γ is the same temperature parameter as in Equation 2.1. At the beginning, Γ is initialized to Γ_{init}. The algorithm then iteratively reduces the temperature, Γ (line 28). For each temperature, the algorithm repeats a single step of TPS-RPM several times. A single step of TPS-RPM is detailed between line 8 and line 26. Specifically, for a single step of TPS-RPM, we apply stereo constraints on the left and right images as follows. Assuming that in a certain iteration, the current TPS mappings for the model points in the left and right images are $\{w_l, d_l\}$ and $\{w_r, d_r\}$, respectively, we define the update functions as shown in line 20 to line 23. Here, Γ determines the strictness of the stereo constraint. As discussed in Section 2.4.2, we decrease Γ from iteration to iteration in line 28. This is because the correspondence between the model points and the image points is coarser at the beginning of this iterative process, and we want the TPS mappings of the left and right images to be more constrained toward each other. As the iteration continues, the algorithm is more certain about the correspondence between model points and image points, and we want the TPS mappings to be less constrained so that the matched shape can fit better to local object contours in the left and right images (the contours are slightly different in these two images due to parallax). The rest of Algorithm 8 is the same as single image TPS-RPM, which is detailed in Section 2.4.2.

Since Algorithm 8 uses the same number of iterations for both the left and right images, the two TPS-RPM processes for a candidate pair converge at the same time. We define the overall matching score of the candidate pair to be the sum of the individual matching scores of the left and right images plus a weighted term that encodes the stereo constraint. This term measures how well the corresponding matched model points in the left and right images satisfy the epipolar constraint, which can

be measured by:

$$\sqrt{\sum_{a=1..K} (f(v_a, d_l, w_l)^T F f(v_a, d_r, w_r))^2},$$

where $f(v_a, d_l, w_l)$ is the mapping of model point v_a by the TPS $\{d_l, w_l\}$ in the coordinate frame of the left image, and $f(v_a, d_r, w_r)$ is the mapping of model point v_a by the TPS $\{d_r, w_r\}$ in the coordinate frame of the right image.

Fig. 7.6 compares the matches found by the single image and stereo matching algorithms. Fig. 7.6(a) shows the matches of a shape model learned on the top of mugs in the left and right images of a stereo pair using the original single image matching algorithm. Although the shape model is correctly matched on the top of the mug in the right image, the shape model is mismatched to background clutter in the left image. By applying the stereo constraints, a correct pair of corresponding matches of the same shape model are found in both the left and right images in Fig. 7.6(b).

7.3.2 Adding Negative Examples

The shape matching algorithm with stereo constraints helps with eliminating inconsistent false positive matches in a stereo pair. However, it is not uncommon to observe false positive matches that actually satisfy stereo constraints. This happens more often for simpler shape models, such as those that match the top or handle of a mug. This is because shape models learned from a small number of PAS features are less robust and image regions with many edges lead to many local maxima in the Hough voting space. This leads to false positive matches in the later TPS-RPM process.

For a set of putative PAS features generated in an image that contains a target shape, we want to be able to detect which features are discriminative. A better way to do this is to select those features that are not observed in a set of *negative examples*. Specifically, one can add negative training images to the shape model

learning process (Ferrari et al., 2008). The positive training set includes images of target objects while the negative training set may contain irrelevant objects or background clutter. The algorithm first learns a codebook C of PAS features from the training images. For a given region of an image, A feature vector is defined as a histogram whose element captures the frequency of each PAS type T_i in C observed inside this region. For a positive training image, a single feature vector is calculated for the image region that contains the target shape. For a negative training image, a set of feature vectors is calculated for some image regions that do not contain the target shape. In order to capture the relative positions of PAS features observed inside an image region, an image region is divided into sub-regions. Then, a histogram is calculated for each sub-region and the histograms for all sub-regions are appended together as a single feature vector. A SVM classifier is trained to distinguish the positive and negative sets of feature vectors.

Given a test image, the learned target shape is detected by a sliding window (SW) method. The algorithm slides rectangular windows with different offsets and scales across the image. For each window, the algorithm calculates the corresponding feature vector and classifies it by using the learned SVM classifier. As a result, the SW method detects a set of rectangular regions in which the target shape is likely to be observed.

The SW method is used in place of the Hough voting scheme to generate initial hypotheses that can be used to seed the TPS-RPM process. Similar to the Hough voting scheme, each hypothesis match of a shape model detected by the SW method specifies a x/y location, a scale and a matching score. The x/y location corresponds to the x/y position of the center of the detected window. The scale corresponds to the scale of the detected window relative to the average size of the bounding boxes of the target shapes in the training set. The matching score is the score calculated by the SVM classifier.

7.3.3 Performance Comparison between Shape Matching Algorithms

In this section, I compare the performance of several different shape model matching approaches, in order to decide which one to use in the visual learning algorithm proposed in this dissertation. In these exploratory experiments, I use 12 single-view images of different mugs as the positive training set. For each positive training image, I manually annotate bounding boxes that correspond to the mug, the handle and the top. I use two images that do not contain mugs as the negative training set. For testing, I take a set of 41 stereo image pairs of novel mugs by varying their orientations slightly in a cluttered scene as examples shown in Figures 7.6.

I use the code released by the authors[1] (Ferrari et al., 2010) with default parameters to learn three models on the training set of images that describe the mug, the handle and the top respectively. For each model, I compare four different algorithms that use sliding window (SW) or Hough transform for initial match and stereo image pair or single image for TPS-RPM. The experimental results are summarized in Table 7.1. The numbers shown in the table are the ratios of correct matches relative to the total number of images (82). The correctness of a match is manually determined by human annotation. A correctly identified grasp region is defined as the one whose error is small enough such that a hypothetical grasp of the identified region will be successful based on human judgment. Based on this criterion, some representative test images with the model shapes correctly detected by the SW-stereo method are given in Fig. 7.7, while some typical incorrect matches are given in Fig. 7.8.

This particular test set is difficult for shape matching because it contains images of novel objects and substantial aspect changes. By comparing the results, the stereo matching algorithm performs as well as the single image matching algorithm in most cases. In some other cases (Top-Hough, Top-SW and Handle-SW), the stereo

[1] http://www.vision.ee.ethz.ch/~calvin

Table 7.1: Performance Comparison between Shape Matching Algorithms. The numbers shown in the table are the ratios of correct matches relative to the total number of images (82). Each algorithm uses either Hough voting or sliding window (SW) to find initial matches; and uses either single image or stereo matching during TPS-RPM. Row: Hough voting or SW. Column: shape model type. The two ratios in each cell correspond to single/stereo matching.

	Mug	Handle	Top
Hough	0.7805/0.8049	0.2805/0.2195	0.1098/0.2439
SW	0.7317/0.7561	0.4024/0.6951	0.4756/0.7439

matching algorithm performs substantially better. One exception is that the stereo matching algorithm performs notably worse for the handle model. This is due to a consistent false positive match observed on the spray bottle in many stereo pairs. For the mug model, the SW algorithm performs comparably to the Hough voting algorithm. For the handle and top models, however, the SW algorithm performs substantially better. This indicates that the SW algorithm is especially helpful for the simpler models, which is consistent with our expectation. Based on the above results, I choose to use the SW-stereo approach in the proposed visual learning algorithm in the subsequent sections of this dissertation.

7.4 Grasp Identification

For a query stereo image pair, the above shape model matching algorithm finds a set of matches of some previously learned shape models. Each shape model is grasp type and aspect specific. If the matching score of a matched shape model is above the corresponding model threshold t_i^*, the proposed algorithm will assume that the corresponding grasp type is recognized and its location is specified by the boundary of this match. Given the estimated grasp locations in a stereo image pair and the pose of the calibrated camera (in the global coordinate frame), the 3D grasp location in the global coordinate frame can be calculated by stereo triangulation (e.g., Hartley and Zisserman, 2004; Saxena et al., 2008).

A shape model is learned from a set of training images, and each training image is associated with a grasp that is demonstrated by the human teacher. As part of the learning process, a shape model is associated with a set of example grasps. The hand orientations of these grasps provide a natural estimation of the hand orientation that the robot can use in order to grasp the object in the query image. Given the pose of the camera, we specify these hand orientations in the camera coordinate frame. Therefore, the robot can directly estimate the feasible hand orientations based on what it "sees." If a shape model matches a part of the object contained in the test image well (so, the aspect difference between the training and query images is small), the set of hand orientations (relative to the camera) can then be used to parameterize the grasp controller to reach the object at the recognized grasp location. So, which hand orientation in this set of orientations should the robot use? Recall that, for a unidirectional grasp type, a shape model is associated with a set of hand orientations that are concentrated around a single mean direction. Since the variation in this set of hand orientations is usually small, the robot can simply select the the mean orientation as the recommended hand orientation. However, for a rotationally symmetric grasp type, a shape model is associated with a manifold of hand orientations that are described by a girdle distribution. These hand orientations have a large variation about the symmetry axis of the girdle distribution. One possible way is to select a hand orientation in this manifold that has the shortest path length to the current arm configuration (see de Granville, 2008, for details).

In the scope of this dissertation, we do not explicitly deal with in-image-plane rotations. In OUGD, although the image samples corresponding to each grasp type cover a full range of aspects on the aspect sphere, they only cover a limited number of in-image-plane rotations for each aspect. For example, in OUGD, we only have images corresponding to two different in-image-plane rotations for a glass: upright and upside-down. Accordingly, our algorithm learns two distinct shape models that

capture both the upright and upside-down configurations. In order to estimate the grasp orientation for an object component in a test image that contains a large in-image-plane rotation compared to the training set of images, the estimated hand orientation should be adjusted by the rotation of the best matched shape model in the image plane. This requires that the learned shape models to be rotationally invariant in the image plane, and at the same time, be able to give an estimate of the orientation of a match in the image. A possible way to extend the original PAS features along these lines is given in Appendix A.

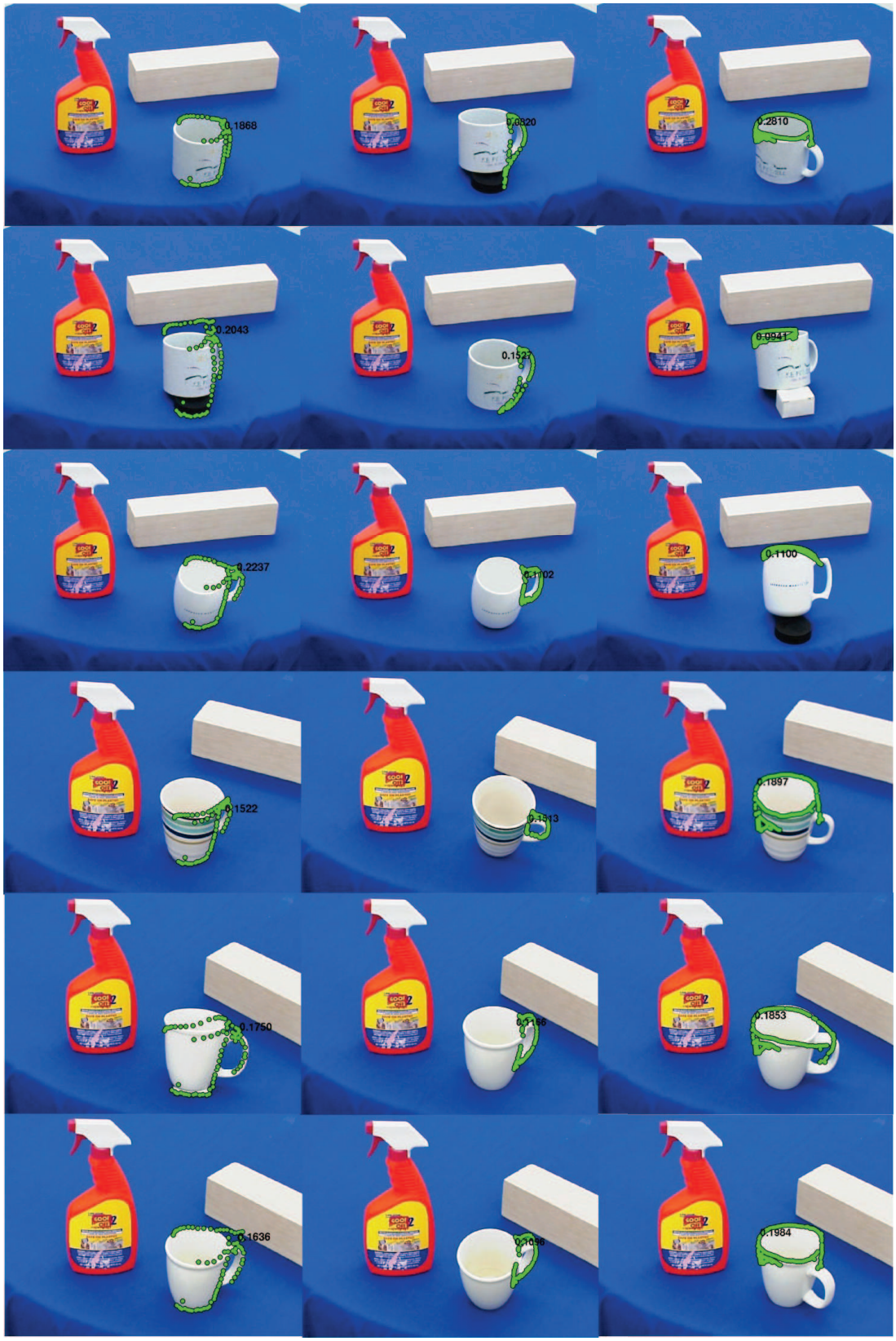


Figure 7.7: Some correct matches by using the SW-stereo method. The number in each image is the matching score.

Figure 7.8: Some incorrect matches by using the SW-stereo method. The number in each image is the matching score.

Chapter 8

Visual Learning Experiments

For the visual learning experiments, we have two major hypotheses: 1) the observed grasp types can be used to learn meaningful visual models that capture partial object shapes, and 2) a learned visual model can suggest an approximation to how the hand should be positioned relative to the object in novel scenarios containing novel objects. For the purposes of the visual learning experiments, we assume that the entire set of 50 objects in OUGD has been correctly partitioned into object classes, which is true in all 9-fold experiments according to the results in Section 6.1. In this chapter, we design an experiment to evaluate the performance of the proposed visual learning algorithm. We divide the entire data set into training, validation and testing sets. According to the visual learning algorithm proposed in the last chapter, we use the training, validation, and test data as follows. For each object category, we partition each grasp sample (and thus the corresponding image fragments) in the training set into one of the grasp types described by the unified grasp affordance model of this object category. For each grasp type, we further partition the training set of samples based their aspects. For each aspect cluster, the proposed algorithm learns a shape model from the training set of image fragments. On average, the learning algorithm gives us about 20 shape models for each grasp type that corresponds to different aspects. Given a validation set of images, we further use two steps of filtering to remove low-quality models and at the same time, determine the optimal model threshold t_i^* for each shape model i (where i is the model index). Finally, we evaluate the performance of the shape models that have survived the filtering process using a set of test images.

8.1 Experiment Setup

We evaluate the performance of our specific implementation of the proposed algorithm by using 5-fold leave-one-object-out cross-validation. First, we fix the order of the five objects in each object category. Then, for each object category, we choose the first object for testing, the second object for validation, and the remaining objects for training. For each experiment, we rotate the order of the five objects by one.

For all the results shown below, we assume that the foreground image has already been identified (the region of the image containing some object). It is this foreground image that is compared against the set of shape models. Each test image is matched against the set of shape models of the same object category to which this test image belongs[1]. If not otherwise noted, each shape model uses its own model threshold, t_i^*, that maximizes the κ statistic on the validation set.

8.2 Performance Measures

For each test image, the algorithm returns a list of matched shape models ordered by their matching score. The algorithm then puts forward the N top shape models. If **any** of these N top shape models is labeled as correct[2], we will claim this test image as correct. We call this approach Top N. By using the Top N approach, we label each test image as correct/wrong (if there is at least one match of some shape model) or no-match (if no shape model is matched above its model threshold). The test images labeled as correct form the true positive (TP) set, and those labeled as wrong form the false positive (FP) set. As we discussed before about the labeling of validation

[1]This version of the algorithm is called match-within-class. In Section 8.3.4, we compare the performance of match-within-class with that of match-against-all

[2]Given that we have thousands of test images for a single experiment, we propose an algorithm to automatically assess a matched shape model in a test image as correct or not as far as the predicted grasp location and orientation, without a human "in the loop." More details can be found in the Appendix B.

results during the second step of shape model filtering process, we do not have the true labeling of P/N for the test set of images for the same reason. This precludes us from using common statistics such as recall vs. precision to evaluate the performance of our algorithm. Instead, we choose to use rate of coverage vs. precision. Specifically, rate of coverage is defined as:

$$\text{rate of coverage} = (TP + FP)/\text{total number of images},$$

and precision is defined as:

$$\text{precision} = TP/(TP + FP).$$

We further define the probability of success by combining both rate of coverage and precision:

$$\text{probability of success} = \text{rate of coverage} \times \text{precision}. \tag{8.1}$$

The rate of coverage gives us some idea about the probability of finding a match (the prior) once the robot is confronted with a novel image. Once a match is found, the precision indicates the probability of that match being correct (the posterior). The probability of success combines both of these two measures: the probability of finding a correct match as far as hand location and orientation (the joint distribution).

8.3 Experimental Results

I begin by showing the algorithm's response to individual query images. An example of the top five matches on a test image of the mug is shown in Fig. 8.1. In this figure, the top row of images shows the same query image, but with the different

matching shape models. Each matched shape model (after TPS mapping) is shown as a set of green dots. The bounding box of the matched shape model is shown as a green rectangle, and the ground truth bounding box is shown as a red rectangle. As detailed in Appendix B, the match of a shape model is assessed based on the intersection over union ratio (IoU) between the predicted and ground truth bounding boxes and the error between the predicted and ground truth hand orientations (HOE). All the ground truth bounding boxes with which the matched bounding box has a IoU greater than 0.2 are shown. The matching score, IoU(s), and HOE(s) are shown below each of the query images. The bottom row of images are example hand positions/orientations recommended by each matched shape model. In the evaluation process (Algorithm 9), the algorithm compares the mean hand orientation of the set of training images from which the matched shape model is learned with the ground truth hand orientation associated with each test image. The responses are ordered by their matching scores, from high to low. Finally, below the bottom row of images, each response is labeled as either *correct* or *wrong*, which are determined by our automatic performance measuring process (Algorithm 9). A matched shape model is only labeled as correct if the IoU is greater than 0.2 and the HOE is smaller than 30 degrees. In this example, all of the top five matched shape models are learned on the top of mugs, due to the fact that the handle is occluded by the body of the mug. In this example, all top five responses are labeled as correct.

An extensive set of examples of the top five responses on different object categories is shown in Fig. 8.2 to Fig. 8.16. Fig. 8.2 shows another example of the top five matches on an image of the mug. In this case, there are only two matched shape models with matching scores above their respective model thresholds. Two ground truth bounding boxes are associated with this test image, which correspond to the handle and top grasps, respectively. For each matched shape model, we compare it with both ground truth bounding boxes. The separate IoUs and HOEs with respect

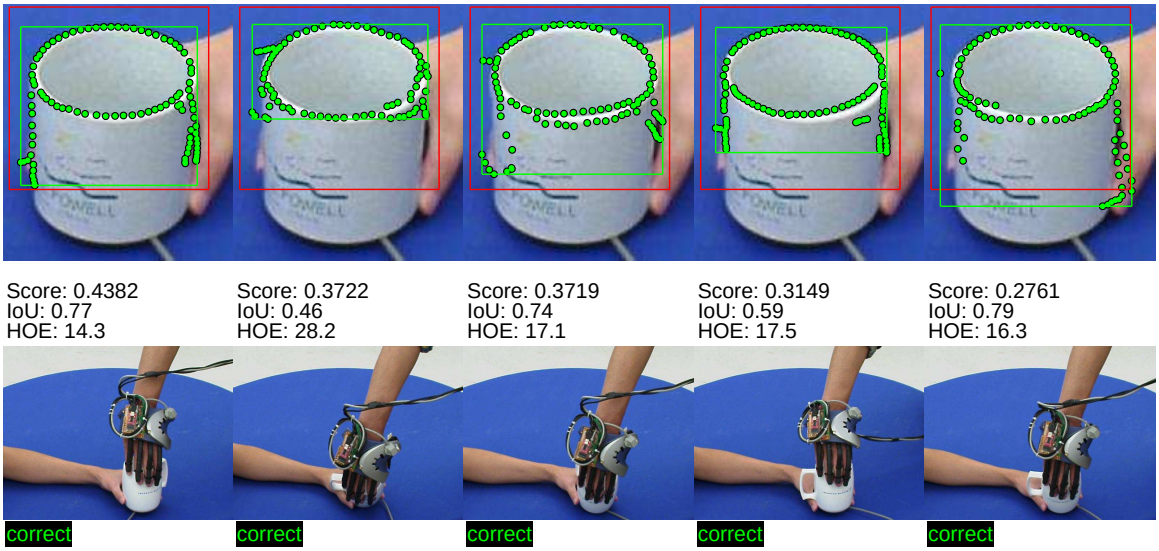


Figure 8.1: Top 5 matches on a mug. The top row of images shows the same query image, but with the different matching shape models. Each matched shape model (after TPS mapping) is shown as a set of green dots. The bounding box of the matched shape model is shown as a green rectangle, and the ground truth bounding box is shown as a red rectangle. The matching score, IoU, and HOE are shown below each of the query images. The bottom row of images are example hand positions/orientations provided by each matched shape model. The matches are ordered by their matching scores, from high to low. Below the bottom row of images, each match is labeled as either *correct* or *wrong*, which are determined by our automatic performance measuring process.

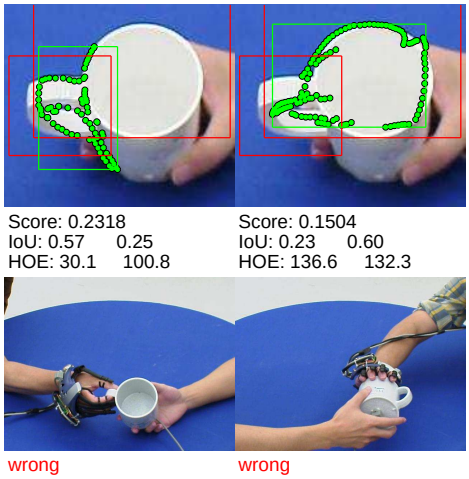


Figure 8.2: Top 5 matches on a mug. However, there are only 2 matched shape models with matching scores above their respective model thresholds.

to the two ground truth bounding boxes are shown below each query image. The first shape model is learned from the handles of mugs. Although this looks like a good match, the HOE is greater than the allowable threshold of 30 degrees. Therefore, it is labeled as wrong. The second matched shape model is learned from the tops of mugs, however, in an aspect that does not contain enough visual information. This response has IoUs greater than 0.2 with respect to both ground truth bounding boxes, but both of the HOEs are far greater than 30 degrees. Therefore, this response is also labeled as wrong.

Fig. 8.3 shows another example of the top five responses on an image of the mug. In this example, there is only one matched shape model with matching score above its threshold. This shape model is learned from the handles of mugs. Although this looks like a good match, the HOE is greater than the threshold of 30 degrees. The large HOE comes from an ambiguity in aspects, where the mug can be explained as either tilting toward the camera or away from the camera. As a result, this match is labeled as wrong.

Fig. 8.4 shows an example of the top five matches on an image of the glass. In this

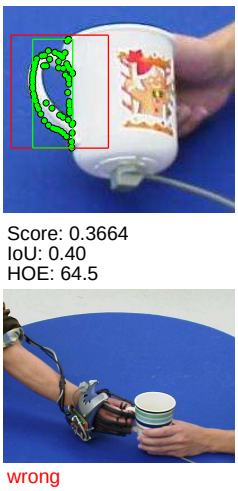


Figure 8.3: Top 5 matches on a mug. However, there is only 1 matched shape model with matching score above its model threshold.

case, there are only three matched shape models with matching scores above their respective thresholds. The first two shape models are learned from the tops of the glasses, and the third shape model is learned from the bottoms of the glasses. The first shape model matches the glass very well and has a matching score much higher than the other two. This match satisfies both the IoU and HOE criteria and is labeled as correct. The second shape model is learned from a top view of the top of the glass. The matched shape model has a large HOE due to very different aspects between the estimated hand orientation and the ground truth. The large shape warping (which can be captured by an affine transform) between the shape model and its response in the query image is reflected by a low matching score. For the third matched shape model, some of the model points are distracted by the texture of the glass. As a result, the matching score is low due to severe distortion. Nonetheless, it is labeled as correct since it satisfies both the hand location and orientation criteria.

Examples of the top five matched shape models on the spray bottle are shown in Fig. 8.5. The matched shape models correspond to top grasp, bottom grasp, top

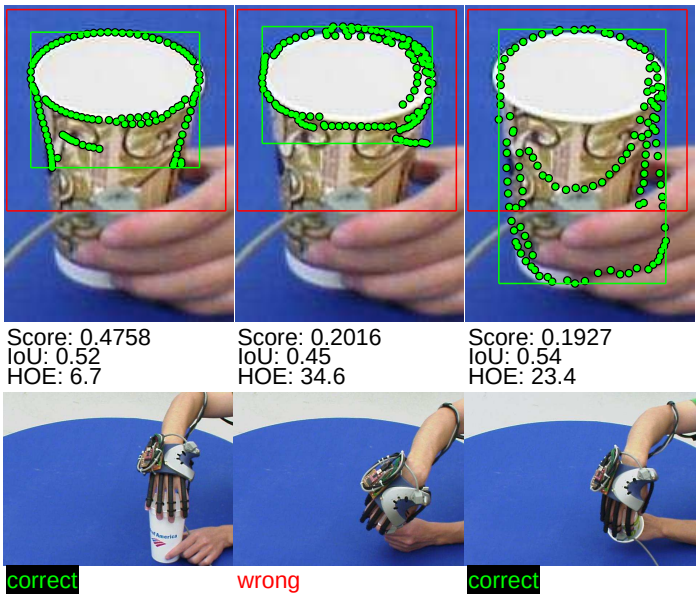


Figure 8.4: Top 5 matches on a glass. However, there are only 3 matched shape models with matching scores above their respective model thresholds.

grasp, side grasp, and side grasp respectively. We can see that only the top grasp is associated with shape models that can be readily identified by human. The first and fifth matched shape models match the object relatively well and are labeled as correct. The fourth matched shape model matches the object well but has a HOE greater than 30 degrees, and therefore is labeled as wrong. The second and third matched shape models are false positives, which are distracted by the inner lines on the spray bottle. The spray bottle is the most difficult object in the OUGD. We have demonstrated seven unidirectional grasp types on this object category. From the above example matches, we can see that most of these shape models do not contain informative visual cues for robust visual matching. A few shape models that are identifiable by human are learned for some very specific aspects, such as the first response in Fig. 8.5. The above observation is generally true in our experiments.

Fig. 8.6 and Fig. 8.7 show some examples of the top five matches on the test images of hammers. The first matched shape model in Fig. 8.6 is learned from the handles

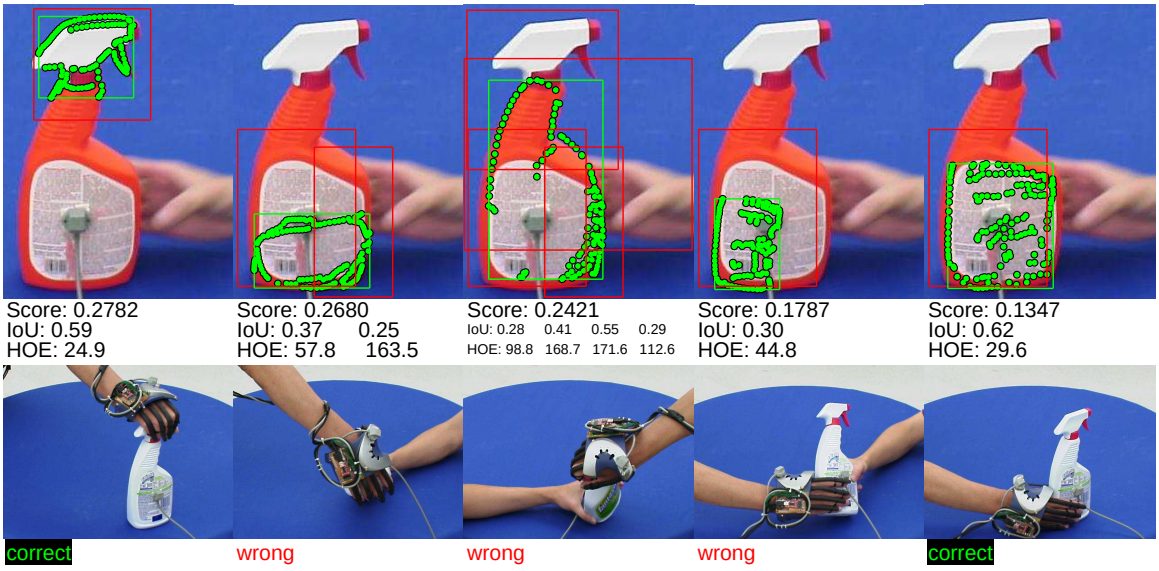


Figure 8.5: Top 5 matches on a spray bottle.

of hammers. The single bounding box corresponds to both the power and precision grasps on the handle, and this is why the two IoUs have equal values. This match is labeled as correct since it satisfies both the bounding box and hand orientation criteria. However, the shape models learned on the handles of hammers are usually insensitive to aspect change when the hammer is tilted towards or away from the image plane. These shape models can be matched to a wider range of aspects than is allowed by our error tolerance (30 degrees). The second matched shape model is learned on the head of hammers. This shape model tends to match equally well when the hammer is rotated by 180 degrees about the handle. This is because the grasp affordance models learned on hammers are symmetric about a 180 degrees rotation about the handle. As a result, in the grasp affordance model matching process, one hammer is aligned with the others when facing the opposite direction. Accordingly, in the shape model learning process, our algorithm learns a generic shape model that can be matched to hammers facing both directions. Due to the above reason, we acknowledge this symmetry in the performance evaluation process. Since we compare

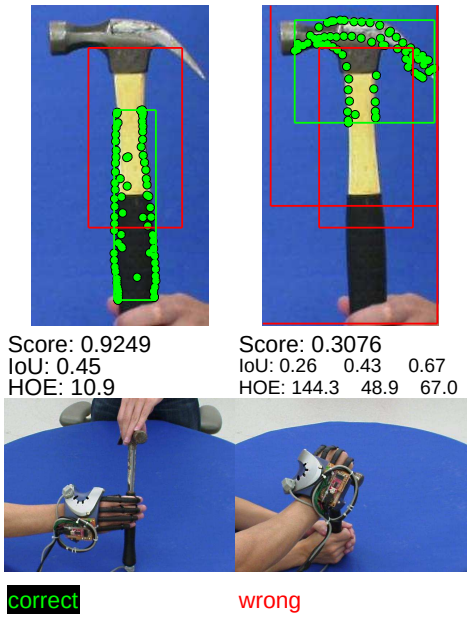


Figure 8.6: Top 5 matches on a hammer. However, there are only 2 matched shape models with matching scores above their respective model thresholds.

a matched bounding box to all ground truth bounding boxes, a matched shape model of hammer head will be labeled as correct if its prediction is consistent with the head grasp on a hammer facing either direction. In this case, however, the match is labeled as wrong due to a large hand orientation error (greater than 30 degrees).

Another example test image of a hammer is shown in Fig. 8.7. In this figure, both of the matched shape models are learned on the head of hammers. These shape models are learned from very similar aspects and both of them are labeled as correct.

Fig. 8.8 to Fig. 8.11 show some examples of the top five matched shape models on different hand drills. Generally, the shape models learned from the handles of drills generalize well to different hand drills and aspects where the handle is clearly visible. The unified grasp affordance model uses a single cluster to describe both the unidirectional (the trigger grasp) and rotationally symmetric grasp types on the handle. Therefore, the automatic performance measuring process does not discriminate between these two grasp types. The learned shape models can also generalize

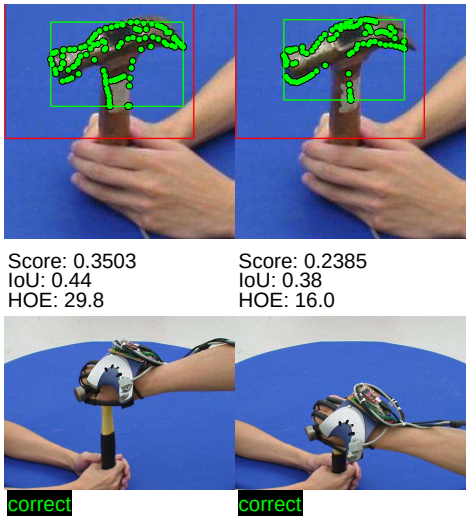


Figure 8.7: Top 5 matches on a hammer. However, there are only 2 matched shape models with matching scores above their respective model thresholds.

to handles with or without the extra battery compartment, and aspects where the handle is rotated about its symmetry axis.

Fig. 8.11 shows an example in which the top matching shape model is labeled as wrong. The hand drill in the test image has high contrast inner edges, which distracts the matched shape model. The second, third, and fourth responses are labeled as correct. The fifth response is labeled as wrong due to a large hand orientation error. Note that the two HOEs below each query image correspond to the two cases where the estimated hand orientation is compared to the trigger grasp (unidirectional) and the handle grasp (rotationally symmetric), respectively.

The wine bottle is one of the object categories in the OUGD that the proposed algorithm achieves high performance. This is due to the fact that wine bottles are rotationally symmetric. For grasp types that have rotational symmetry, the aspect clustering process will return clusters with adequate numbers of images, which allows the learning of meaningful shape models. Also, there is minimal ambiguity and self-occlusion in wine bottles. Fig. 8.12 shows some examples of the top five matches on

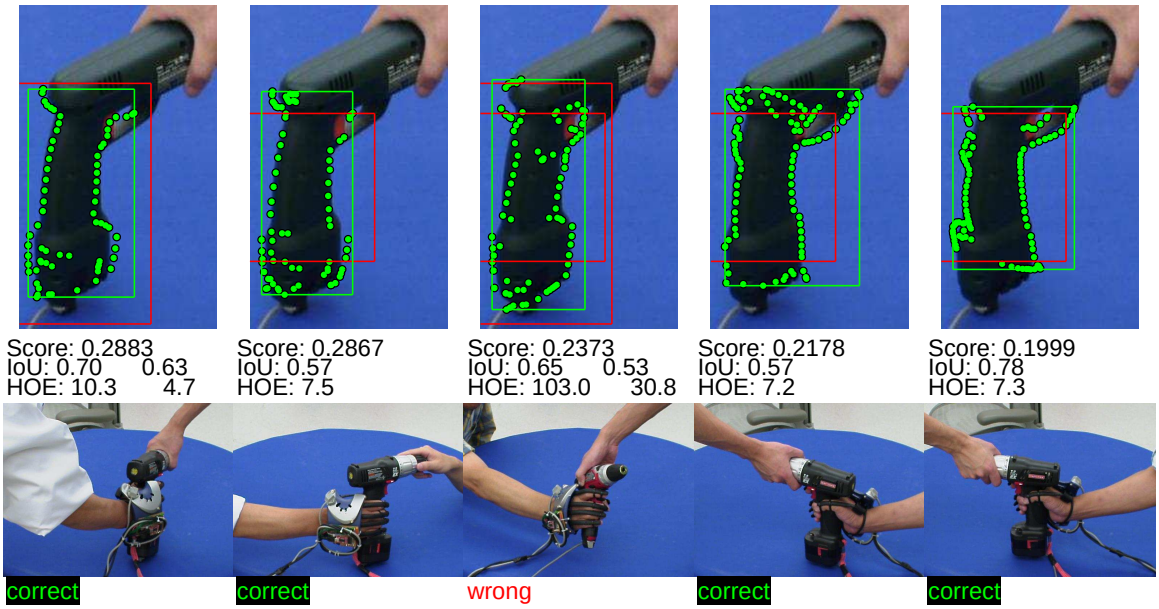


Figure 8.8: Top 5 matches on a hand drill.

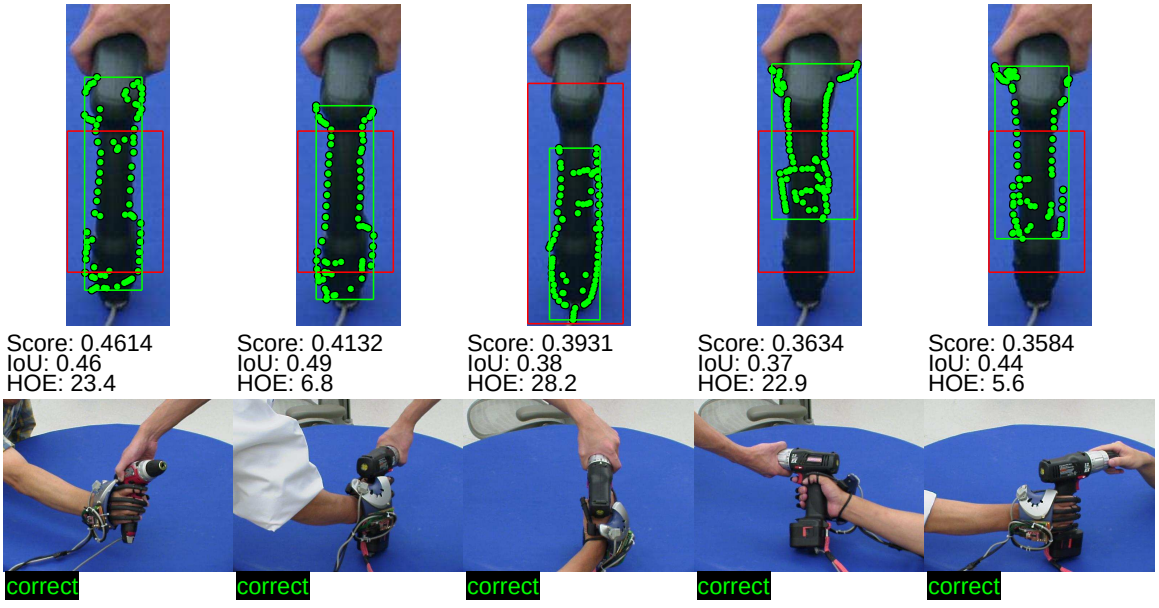


Figure 8.9: Top 5 matches on a hand drill.

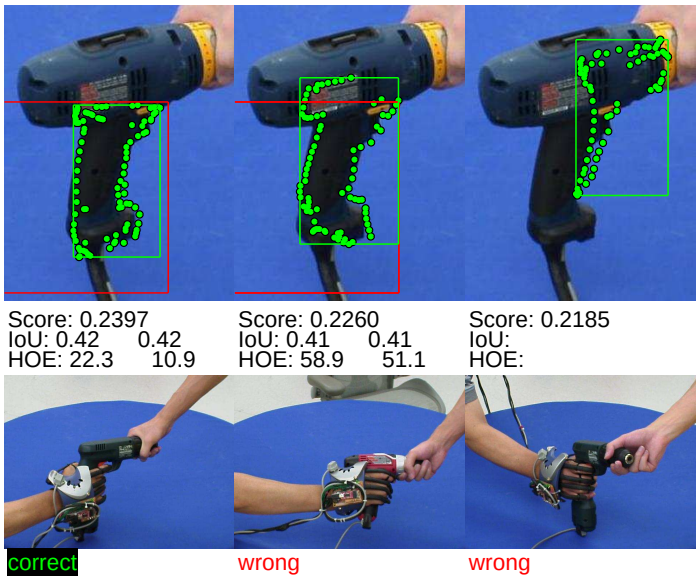


Figure 8.10: Top 5 matches on a hand drill. However, there are only 3 matched shape models with matching scores above their respective model thresholds.

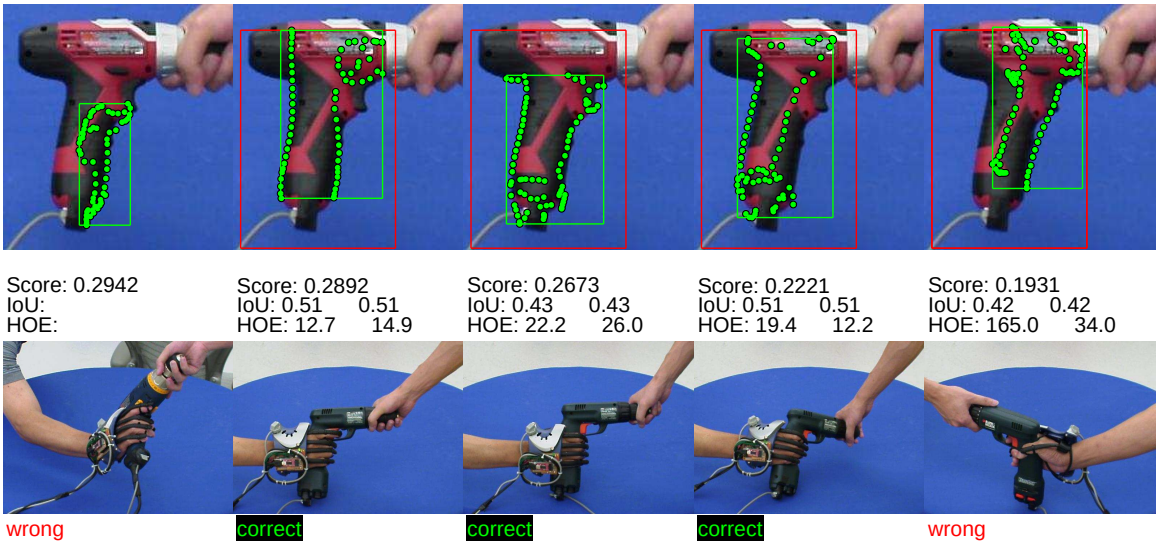


Figure 8.11: Top 5 matches on a hand drill.

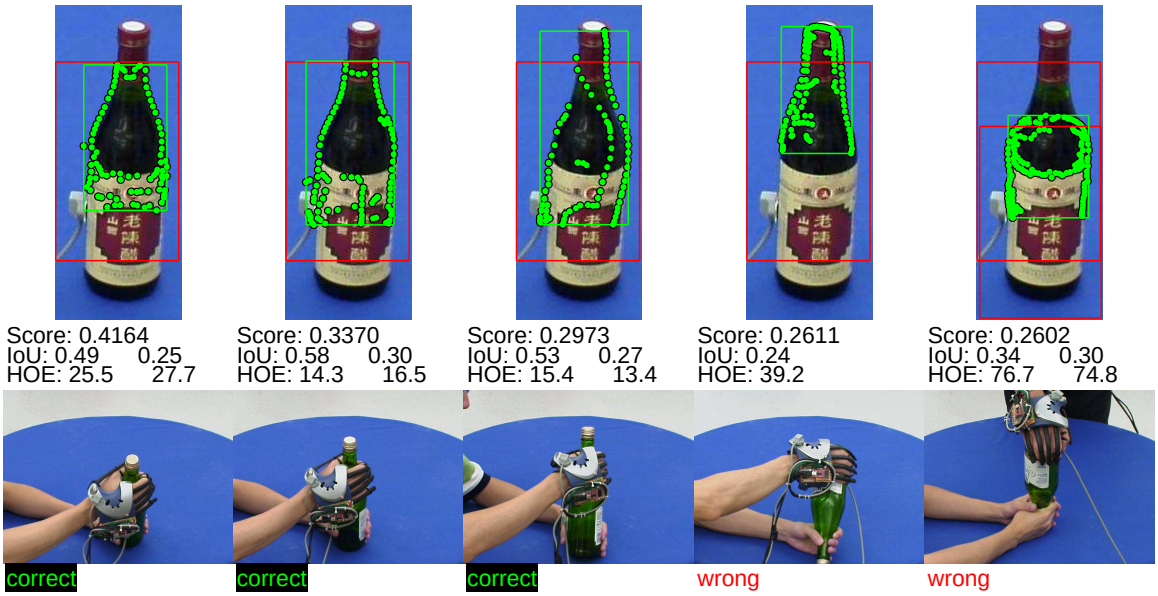


Figure 8.12: Top 5 matches on a wine bottle.

a wine bottle. The top 3 highest scored shape models match the neck of the wine bottle very well and are labeled as correct. The fourth shape model is learned from the bottom of wine bottles and is highly distorted when it is matched to the top of the bottle. This large distortion is captured by a non-rigid warp during TPS-RPM, and is reflected by the low matching score (though it is above threshold). The fifth match shows that a shape model learned from the bottom of wine bottles matches the body of the bottle. This example shows that the TPS-RPM can throw out outliers and only match to a subset of image points.

Fig. 8.13 and Fig. 8.14 show some examples of the top matched shape models on the images of detergent bottles. Similar to the spray bottles, the shape models corresponding to the side grasps and bottom grasps do not contain enough visual information for robust shape matching. However, the performance of the detergent bottle is much better than that of the spray bottle. This is because of the rotationally symmetric grasp that is demonstrated around the cap of the detergent bottle. In most

cases, the shape models associated with this top grasp can be robustly matched (the top three matches in Fig. 8.14). In Fig. 8.13, the first shape model is learned from the handles of detergent bottles. The handle grasp is not well matched. Due to low contrast caused by the shadow of the handle, the inner contours of the handle is usually not detected by the edge detector. This causes the shape models learned on the handles to fail to capture inner contours of the handles. The fourth shape model, which corresponds to a side grasp, does not contain differentiable visual information and gets distracted by the label. Note that two ground truth bounding boxes (red rectangles) in the fourth column of Fig. 8.14 are shown, which means that the matched bounding box has IoUs above 0.2 with both of them. The fifth match in Fig. 8.14 is labeled as correct coincidentally. This shape model, which corresponds to a handle grasp, is learned from images where the handles are not actually visible. Instead, it captures the shape of the cap. In the test image, this shape model detects the location of the cap correctly, and the hand orientation is also correct down to symmetry (since the matched object component is rotationally symmetric). Therefore, this match is labeled as correct.

Fig. 8.15 shows an example of the top five matches on an image of the bowl. In this example, only part of the edges are occluded by the grasping hand and thus included in the shape models, depending upon the size of the bowl. This gives rise to indiscriminative shape models that only contain a single curve as shown in the first three matches in Fig. 8.15. Since these shape models are insensitive to small affine distortions, the matched shape models can have large HOE. The fourth shape model is more robust compared to the other three shape models, since it captures both the upper and lower edges on the side of the bowl.

Fig. 8.16 shows an example of the top five matches on an image of the spatula. Similar as the handle of the hammer, the handle of the spatula has an ambiguity in aspect when the camera is tilted towards or away from the image plane. This causes

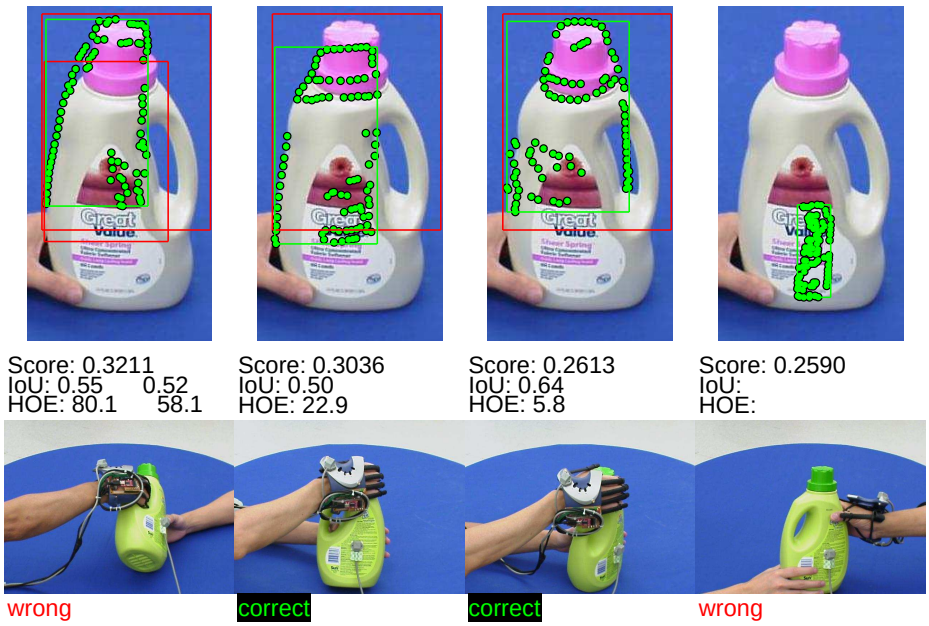


Figure 8.13: Top 5 matches on a detergent bottle. However, there are only 4 matched shape models with matching scores above their respective model thresholds.

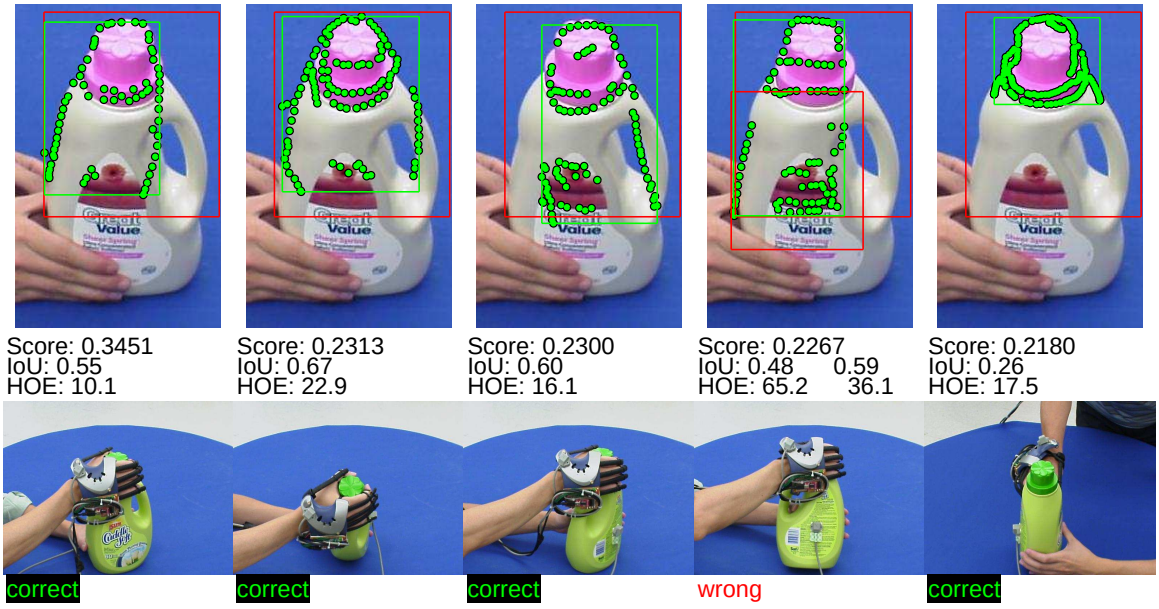


Figure 8.14: Top 5 matches on a detergent bottle.

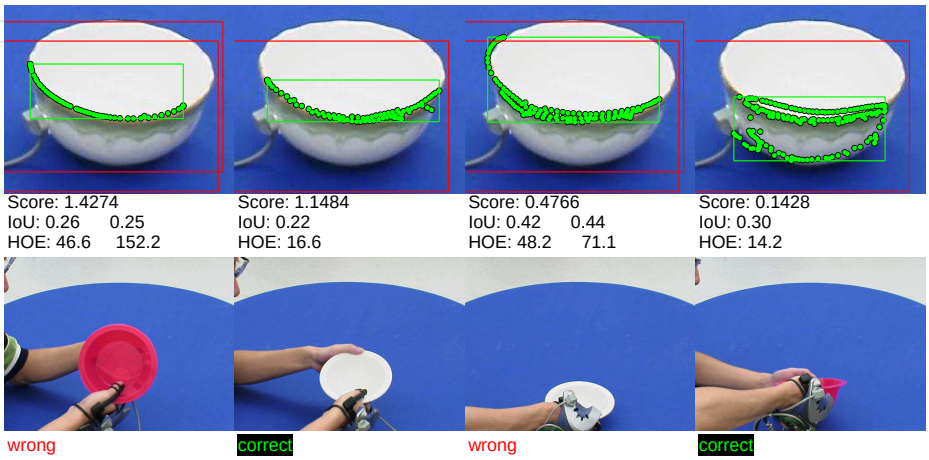


Figure 8.15: Top 5 matches on a bowl. However, there are only 4 matched shape models with matching scores above their respective model thresholds.

the large HOE of the first match, although the shape model matches the object contour very well. The HOE of the second match is smaller than 30 degrees and is labeled as correct.

8.3.1 Performance as a Function of Aspect

The precision for some object categories is low (for example, spray bottles). This is due to the fact that, for some objects, the visual properties that are relevant to a particular grasp are not robustly visible from all visual aspects. In part, this invisibility is due to self-occlusion. Fig. 8.17 shows algorithm performance as a function of visual aspect for the mugs. This figure has three panels, which show the rate of coverage, the probability of correct location and the probability of success respectively. Once a robot is confronted with an image of a novel object in a previous learned object category, we first would like to know whether the robot can recall a learned shape model from its memory and find a match in the image. Following that, we would like to verify whether the matched bounding box location is correct or not. Finally, we would like to verify the estimated hand orientation. In these figures, different colors

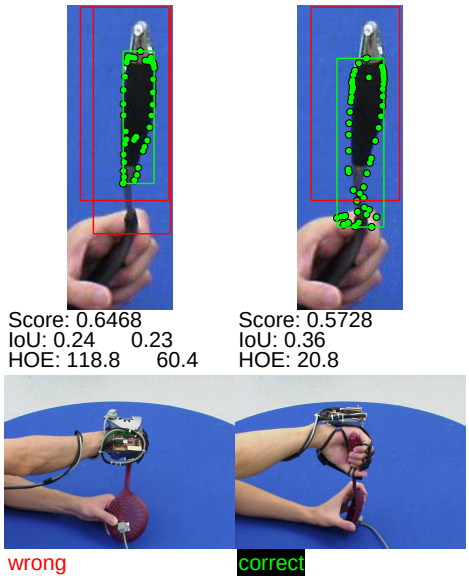


Figure 8.16: Top 5 matches on a spatula. However, there are only 2 matched shape models with matching scores above their respective model thresholds.

represent different values scaled in the range between the minimum and maximum probabilities. The dots represent test set samples from all five experiments, with green representing correct matches, red representing matches with wrong location and/or hand orientation and blue representing no matches. Several example test images corresponding to different aspects are shown in the third aspect sphere. Each example test image points to a single dot on the aspect sphere, whose location corresponds to the aspect at which the test image is taken. To interpret these different aspects, we can imagine a mug as being placed upright in the center of the sphere, with its handle facing left. Accordingly, the north pole of the aspect sphere corresponds to a top view of the mug, and the south pole corresponds to a bottom view of the mug. The arrow pointing from an example test image to a dot on the aspect sphere is color coded the same way as the dots.

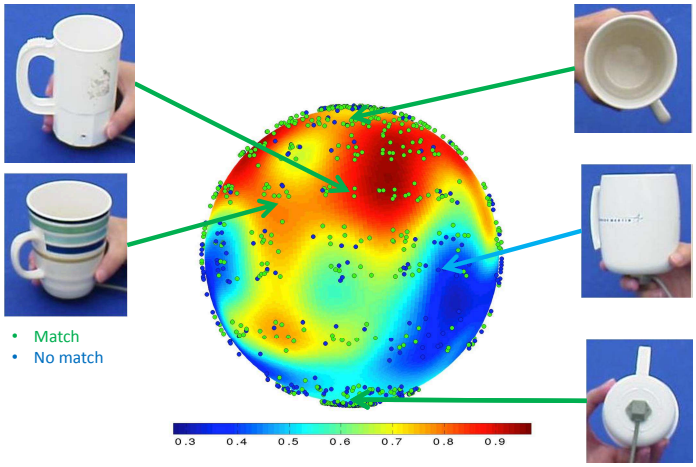
Fig. 8.17(a) shows the RoC as a function of aspect. On this aspect sphere, a dot is colored either green (match) or blue (no match). The range of the RoC is roughly

between 0.28 and 0.98. We can see that most of the regions with hotter colors aggregate on the upper hemisphere. This is because the shape models corresponding to the top grasp are more readily matched compared to the handle grasp, and these shape models are more robustly matched when the opening of the mug is visible. Also, there is a large hot region around the frontal view biased towards left. This region corresponds to some aspects at which the handle shape models are matched. Intuitively, the handle of a mug is best visible in some aspects close to the frontal view. Note that the right hand side of the aspect sphere corresponds to some aspects where the handle is totally occluded by the body of the mug. We expect that these aspects are not covered by handle shape models.

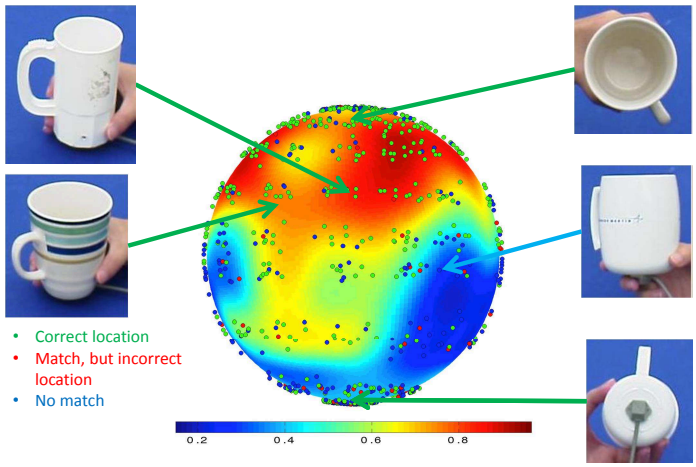
Fig. 8.17(b) shows the probability of correct grasp location (approximated by the bounding box of a matched shape model) as a function of aspect for the mug. The distribution of probabilities on the aspect sphere is very similar to that of the RoC. However, both the minimum and maximum probabilities are lower compared to those in Fig. 8.17(a). This is because we check the correctness of each matched bounding box. Because of this, some of the green dots in Fig. 8.17(a) become red in panel b. Since we calculate the probability of correct grasp location based only on the green dots, the overall probability is lower.

Similarly, in Fig. 8.17(c), we check both the bounding box and the hand orientation errors. Therefore, more green dots in Fig. 8.17(a) become red and the overall probability is even lower than that in Fig. 8.17(b). By comparing Fig. 8.17(b) and Fig. 8.17(c), we can see that the large hot region around the frontal view disappears. This indicates that most of the matched handle shape models have hand orientation errors greater than the error tolerance (30 degrees).

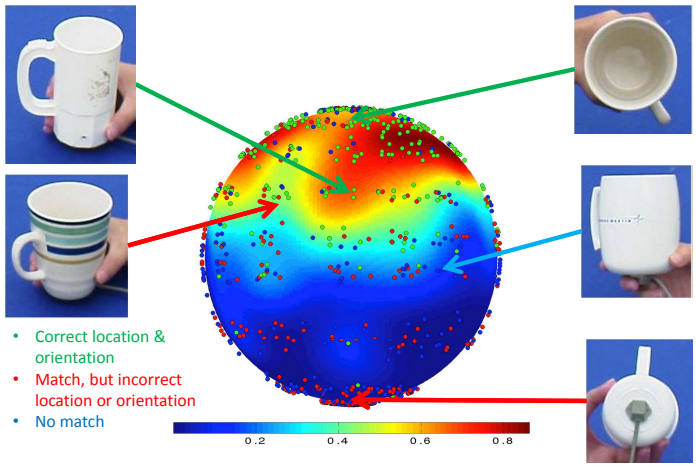
For the mug, we further show the performance as a function of aspect for different grasp types. Fig. 8.18 shows the RoC, probability of correct grasp location and probability of success as a function of aspect for the top grasp and Fig. 8.19 shows



(a) Rate of coverage as a function of aspect



(b) Probability of correct location as a function of aspect



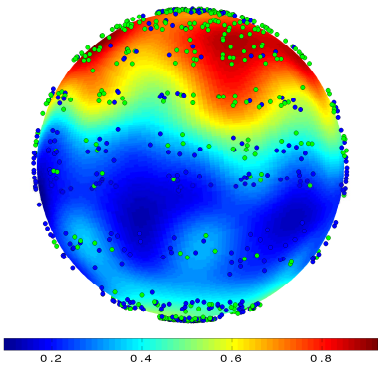
(c) Probability of success as a function of aspect

Figure 8.17: Performance as a function of aspect: mug

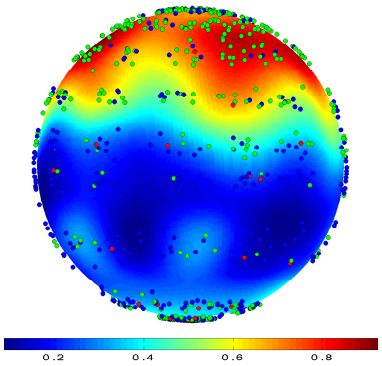
the above probabilities for the handle grasp. For each triplet of aspect spheres, we only match shape models corresponding to a single grasp type to the same set of test images as before. Again, if there are multiple shape models found in a test image with their matching scores above their respective model thresholds, we choose the one with highest matching score (Top 1). It is possible for a test image previously labeled as wrong (red dot) in Fig. 8.17(c) to be labeled as correct (green dot) if we consider two grasp types separately. This is because the highest scored shape model may be different if we only consider shape models from a single grasp type. In the following triplets of aspect spheres, we only show some example images corresponding to some camera locations on the aspect sphere in a single panel for clarity, since all three aspect spheres in a triplet are aligned consistently.

In Fig. 8.18,we can see that the triplet of aspect spheres of the top grasp is similar to that of the combined (Fig. 8.17). The difference is that the hot region around the frontal view no longer exists, since this region corresponds to matched shape models of the handle grasp. The minimum and maximum values of the probabilities on these aspect spheres are similar to their counterpart in Fig. 8.17. This suggests that most of the top 1 matched shape models in Fig. 8.17 are from top grasps. In Fig. 8.18, the distributions of the probabilities are similar for all three aspect spheres. This suggests that once a shape model of a top grasp is matched, the match is usually correct as far as the predicted grasp location and orientation.

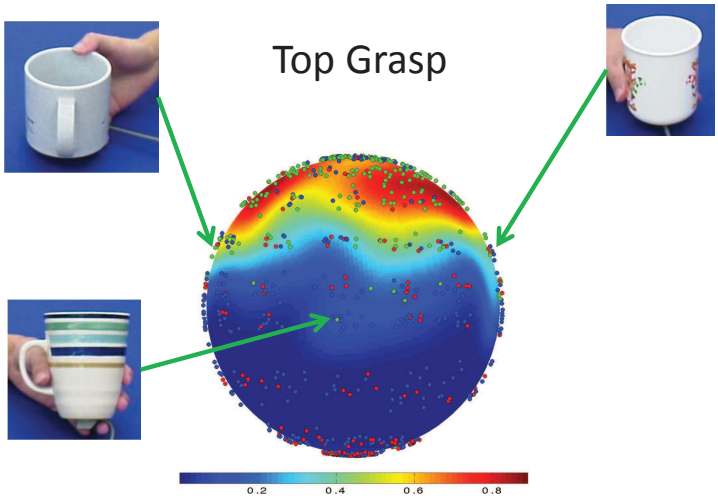
The handle grasp has a substantially lower maximum probability of success (Fig. 8.19c) than that of the combined (Fig. 8.17c). This is because the handle grasp is unidirectional and the handle of a mug can only be reliably matched in some frontal views as shown by the hot regions in Fig. 8.19(c). Since the probability of success is the product of the rate of coverage and precision, a substantially low rate of coverage will drag the probability of success down. Also we should note that some of the frontal views do not have any match (as indicated by blue dots in Fig. 8.19a). By inspection,



(a) Rate of coverage



(b) Correct location



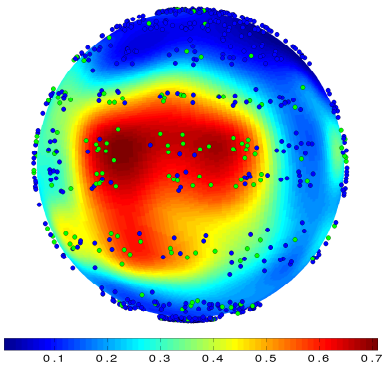
(c) Probability of success

Figure 8.18: Performance as a function of aspect: mug top

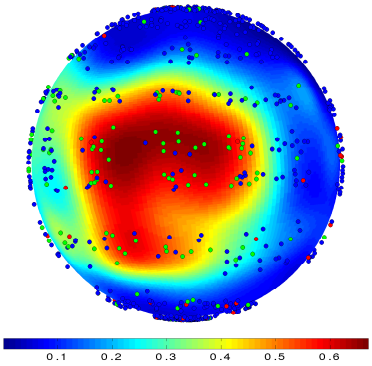
this is because the matched handle shape models have slightly lower matching scores than their respective model thresholds. This may suggest that the validation set has different statistics than the test set.

Another reason that the overall probability of success in Fig. 8.19(c) is low is due to large hand orientation errors (greater than 30 degrees). The learned shape models on the mug handles tend to be insensitive to a large range of aspects, especially when the mug is tilted towards or away from the image plane. The handle shape model deformation caused by tilting can be alternatively explained by mug handles with different heights, which is learned as within-class variation during the PAS shape model learning process. We can confirm this by comparing the three aspect spheres in Fig. 8.19. From Fig. 8.19(a) to Fig. 8.19(b), we can see that the distributions of the probabilities on the aspect spheres are very similar, and the maximum probability is slightly lower. This suggests that a matched handle shape model is usually correct as far as grasp location. Note that the right-hand side of the aspect sphere corresponds to aspects where the mug handle is totally invisible. Therefore, the probabilities in these regions are very low. From Fig. 8.19(b) to Fig. 8.19(c), we can see that once we check the hand orientation error, a substantial amount of the matched handle shape models with correct hand locations are labeled as wrong. This confirms our hypothesis that a common source of error comes from the hand orientation estimate.

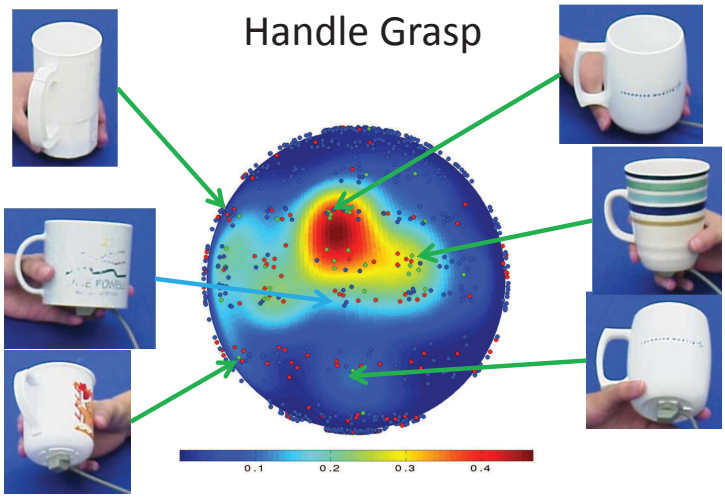
The corresponding backsides of the aspect spheres in Fig. 8.19 are shown in Fig. 8.20. It is somehow against our intuition that the backside of the aspect sphere is very different from the front side. This is partially due to the fact that not enough discriminative shape models are learned when the mug handle is facing the other side. We believe that this is due to a sampling issue in the data collection process. This problem can be alleviated by increasing the training set size. Another reason is that many matched shape models have matching scores that are just below their model thresholds.



(a) Rate of coverage

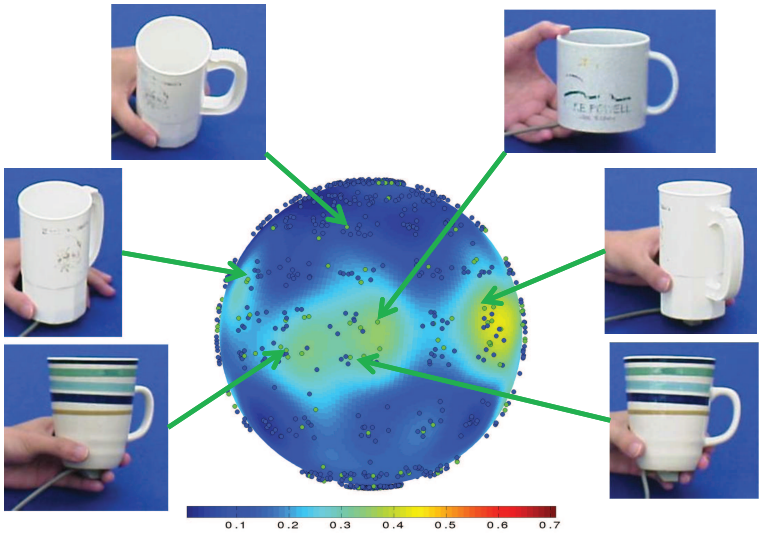


(b) Correct location

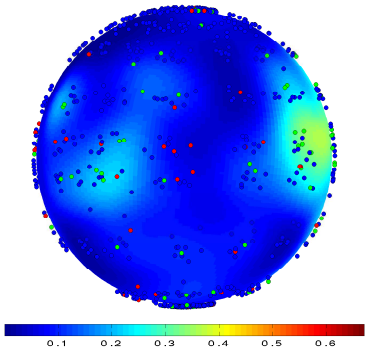


(c) Probability of success

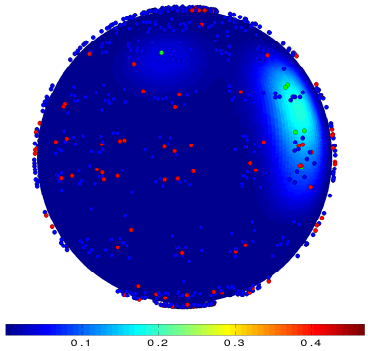
Figure 8.19: Performance as a function of aspect: mug handle front



(a) Rate of coverage

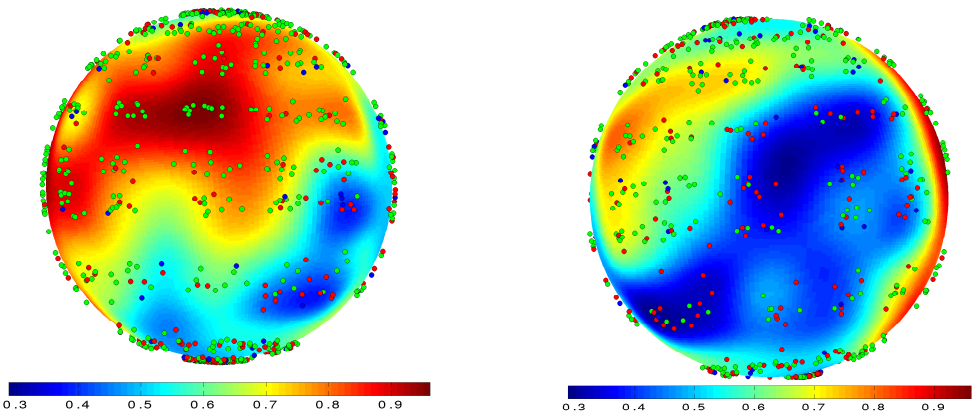


(b) Correct location



(c) Probability of success

Figure 8.20: Performance as a function of aspect: mug handle back



(a) Front (b) Back

Figure 8.21: Probability of correct grasp location as a function of aspect with uniform model threshold 0.1: mug handle front and back

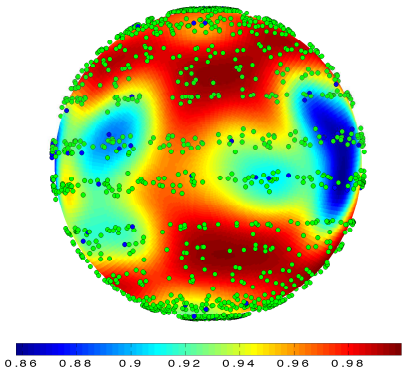
To confirm our hypothesis that the model thresholds corresponding to the mug handles are high such that some correctly matched shape models have matching scores below their model thresholds, we show the probability of correct grasp location as a function of aspect with a low uniform model threshold. The aspect sphere with uniform model threshold $t = 0.1$ is shown in Fig. 8.21. Compared to Fig. 8.19(b) and Fig. 8.20(b), we can see that the maximum probability is much higher and the high probability regions grow dramatically. However, the front and the back of the aspect sphere are still not symmetric. This further convinces us that an insufficient number of shape models have been learned when the handle is facing the the other side. In Fig. 8.21(b), most of the samples with wrong matches can be corrected by using the Top 3 version of the algorithm.

For the glass (Fig. 8.22), the probability of success is generally high across the entire aspect sphere. Exceptions are on the top and bottom views. These two views are ambiguous to each other, which causes large hand orientation errors. Also, the shape models learned on these aspects are too simple to be discriminative (usually only capturing circular shapes). There are also some strips on the aspect sphere with

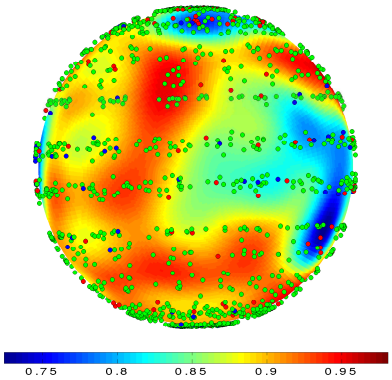
lower probabilities, which is due to the insensitivity of PAS-based shape models to small aspect changes. As a result, some shape models apply to a wider range of aspects than is allowed by our hand orientation criterion.

Note that a point on the aspect sphere only encodes the location of the camera relative to the object, so two points with the same camera location but different in-plane rotations will coincide. An image with a glass and an image with a upside-down glass will therefore correspond to the same aspect on the aspect sphere (if they can be explained by an in-plane rotation). This explains why the high probability regions in the triplet of aspect spheres in Fig. 8.22 are roughly symmetric about the equator. For the upper hemisphere, the opening of the top of glasses is visible, and the shape models learned on the top (and some learned on the bottom) of a set of glasses can be readily matched in these aspects. Likewise, for the lower hemisphere, the ellipse shape on the bottom of glasses is fully visible, and the shape models learned on the bottom (and some learned on the top) of glasses can be readily matched in these aspects.

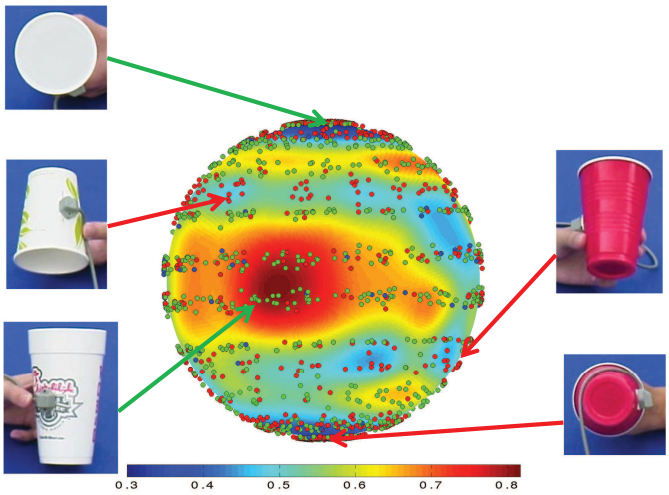
The triplet of aspect spheres for the spray bottles is shown in Fig. 8.23. The regions with higher probabilities of success include some frontal views, some other views in which the camera is slightly moved to the left of the frontal view, and some of the bottom views. According to Fig. 8.23(b), we can see that our algorithm actually matches target shape models with correct locations in a large amount of test images. The maximum probability of correct hand location is close to 0.9. However, when we add the hand orientation constraint, a large number of matched shape models are labeled as wrong due to hand orientations errors greater than 30 degrees. The spray bottles are the most difficult objects in OUGD for our algorithm. The reason in part is because all seven grasp types demonstrated on spray bottles are unidirectional. This means that the visual features will vary dramatically with visual aspect. Also, most of the grasp types, such as side grasps and bottom grasps, are associated with



(a) Rate of coverage



(b) Correct location



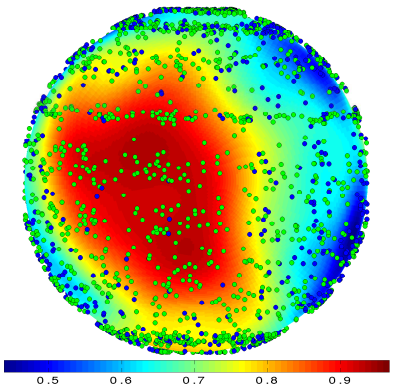
(c) Probability of success

Figure 8.22: Performance as a function of aspect: glass

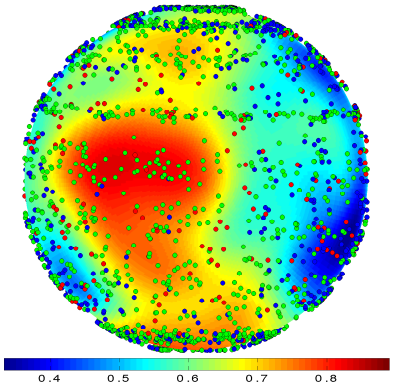
visually ambiguous object parts and difficult to recognize even for humans without larger context.

The hammer can be identified in most aspects, except the top and bottom views (Fig. 8.24a). The top and bottom views do not contain enough visual information for robust identification of the learned shape models. From Fig. 8.24(a) to Fig. 8.24(b), we can see that most of the matches are correct as far as grasp location. However, when comparing Fig. 8.24(b) and Fig. 8.24(c), we can see that the overall probabilities are lower with hand orientation constraint. One difficulty with the hammer is that the shape models learned on the handle are not very sensitive to aspect change when the hammer is tilted towards or away from the image plane, which causes large hand orientation errors. Note that two hammers in the example test images around the frontal view have inconsistent orientations. One hammer is rotated from the others about 180 degrees around its handle. This is because the grasp affordance models learned on hammers are symmetric about a 180 degrees rotation around the handle. As a result, in the grasp affordance model matching process, one hammer is aligned differently with the others. In the shape model learning process, our algorithm learns a generic shape model that can be matched to hammers facing both directions.

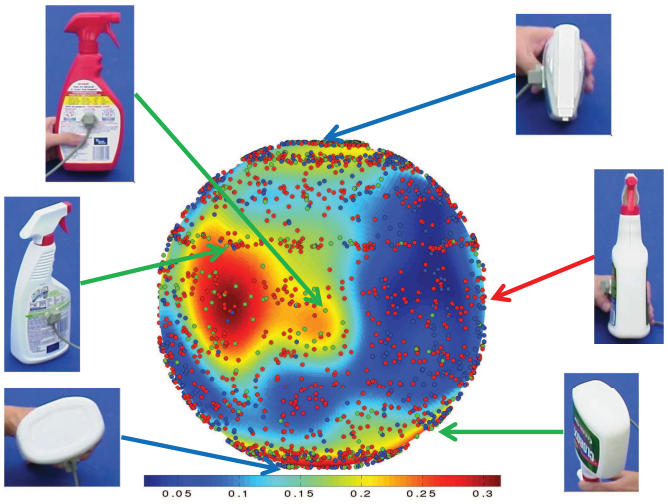
In Fig. 8.25 and Fig. 8.26, we separate the shape models corresponding to grasps on the head from those corresponding to grasps on the handle. By comparing Fig. 8.25(a) and Fig. 8.26(a), we can see that shape models corresponding to the head grasps cover more areas around the top views on the aspect sphere. This is because the head of a hammer is not occluded on the top view, which is the case for the handle. However, according to Fig. 8.25(c) and Fig. 8.26(c), neither the head nor the handle of the hammers is correctly identified for the top view when hand orientation is considered. The shape models corresponding to head grasps are best identified around the frontal views; while the shape models corresponding to handle grasps are correctly identified for most aspects surrounding the symmetry axis of the handle.



(a) Rate of coverage

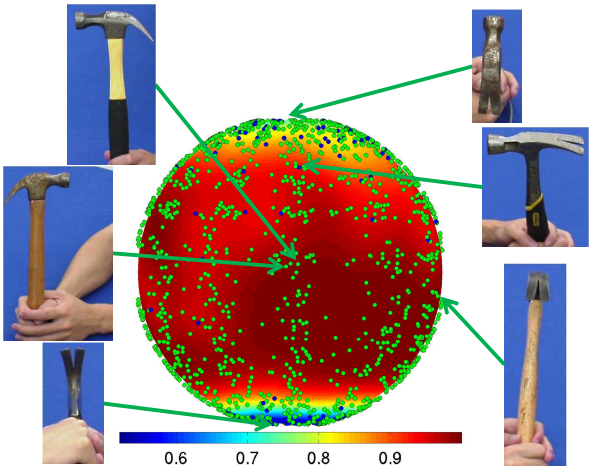


(b) Correct location

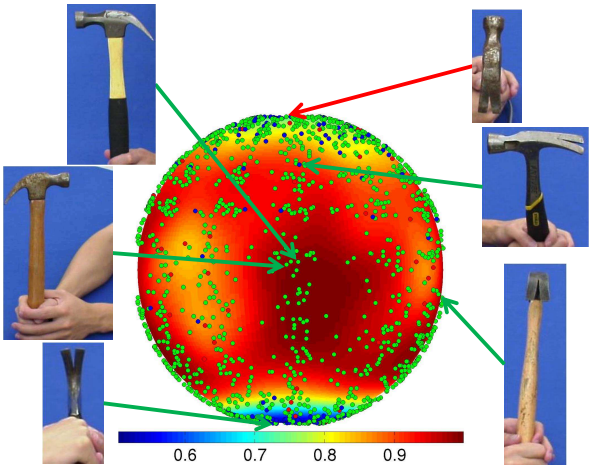


(c) Probability of success

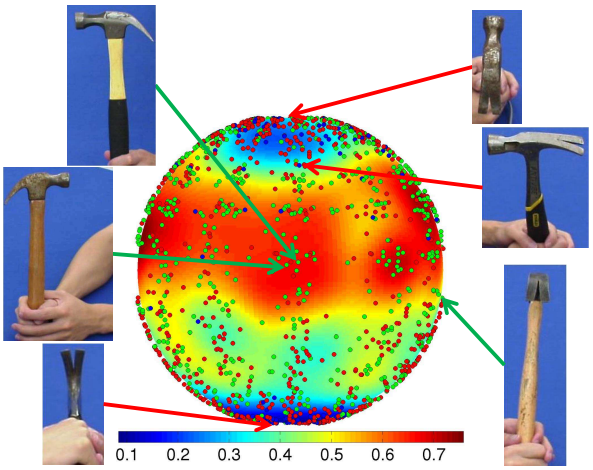
Figure 8.23: Performance as a function of aspect: spray bottle



(a) Rate of coverage

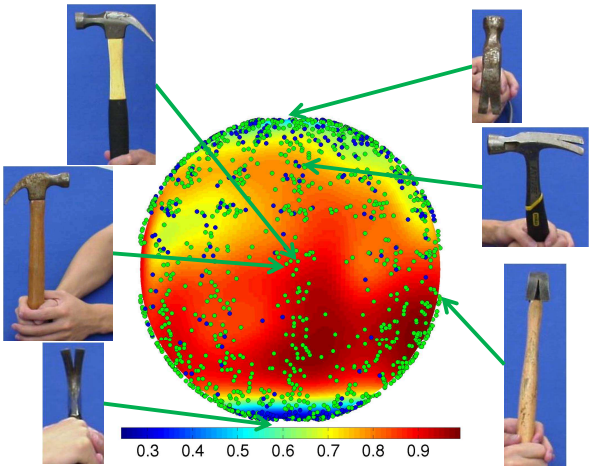


(b) Correct location

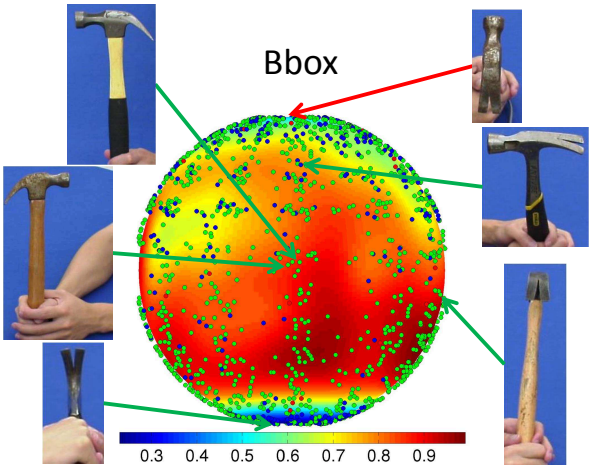


(c) Probability of success

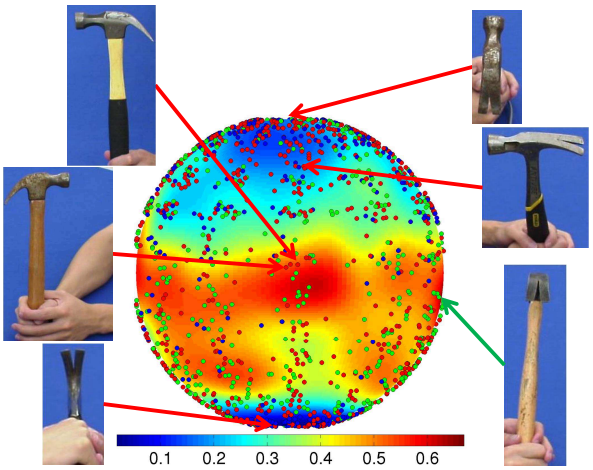
Figure 8.24: Performance as a function of aspect: hammer



(a) Rate of coverage

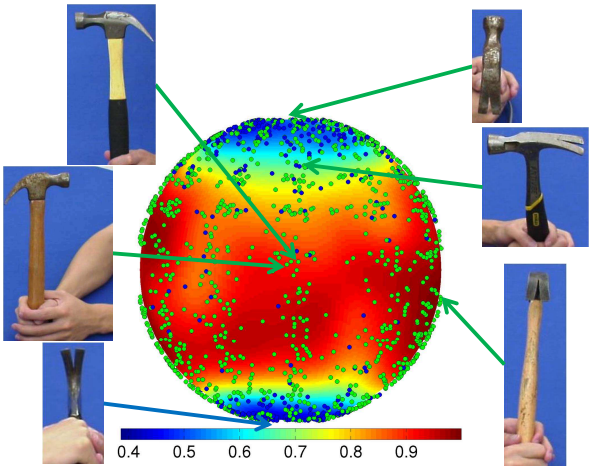


(b) Correct location

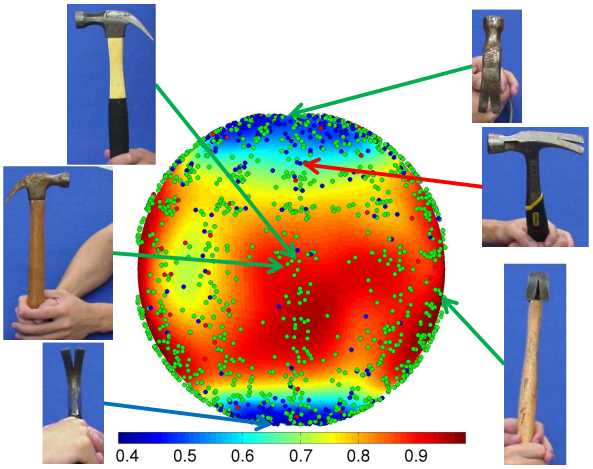


(c) Probability of success

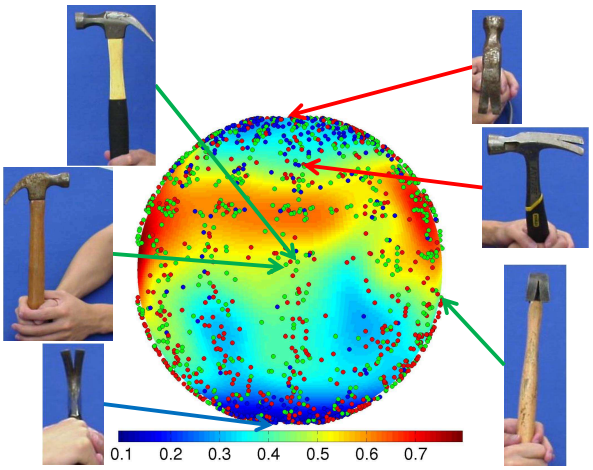
Figure 8.25: Performance as a function of aspect: hammer head



(a) Rate of coverage



(b) Correct location



(c) Probability of success

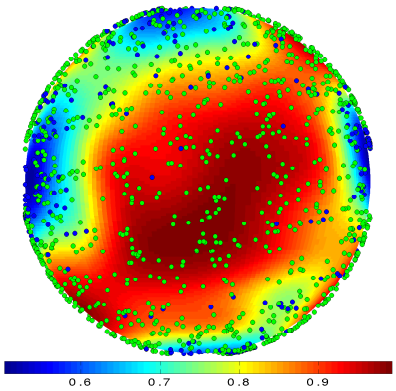
Figure 8.26: Performance of success as a function of aspect: hammer handle

The drill is usually robustly identified in frontal views (Fig. 8.27). In the top view, the handle is self-occluded. In the bottom view, the handle degenerates into a simple rectangular shape, which does not contain too much information for robust visual identification. Note that the drill is rotated by 90 degrees clockwise within the image plane when the barrel is grasped. So in these images, the ground truth bounding boxes for handle grasps do not exist. Similarly, for the test images where the drill handle is grasped by the human teacher, the ground truth bounding boxes for the barrel grasp do not exist.

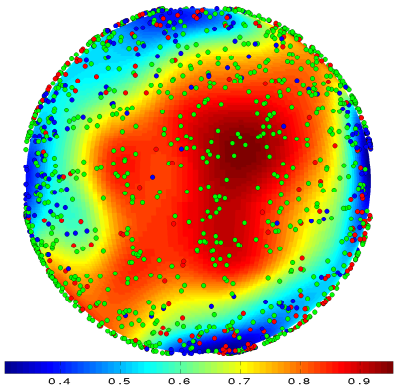
The triplet of aspect spheres for the wine bottles is shown in Fig. 8.28. The wine bottle is one of the object categories in the OUGD that the proposed algorithm achieves high performance, due to its rotational symmetry. In panel c, the probability of success is very similar to that of the glass (Fig. 8.22c). Exceptions are on the top view, where the ambiguity between the top and bottom views of the glasses causes large hand orientation errors.

The detergent bottles have similar shapes to those of the spray bottles (Fig. 8.29). The performance of the detergent bottle is much better than that of the spray bottle, due to a rotationally symmetric grasp around the cap. Similar to the spray bottle, the side and bottom grasps are not robustly identifiable without larger visual contexts. In some cases, the handle grasp is not reliably identifiable due to a low contrast caused by shadowing. A high probability region on the aspect sphere is where the camera is slightly above the equator of the aspect sphere. In these aspects, the top of a detergent bottle is readily matched, due to the fact that the cap of a detergent bottle is fully visible without self-occlusion.

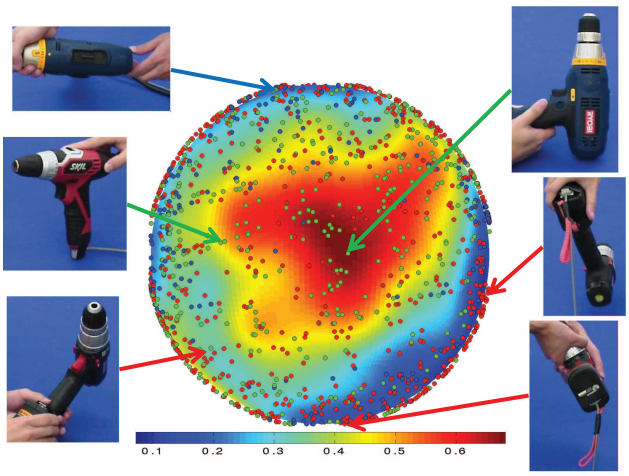
For the bowl (Fig. 8.30), the upper semi-sphere has a higher probability of success. The shape of the bowl is similar to that of a glass, with different aspect ratios. However, the bowl is more difficult to identify than the glass. The reason is that during the demonstrated grasp, the hand can only cover the bowl partially. This



(a) Rate of coverage

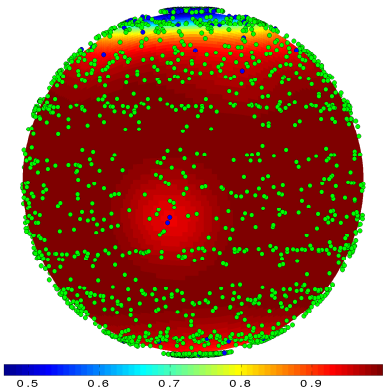


(b) Correct location

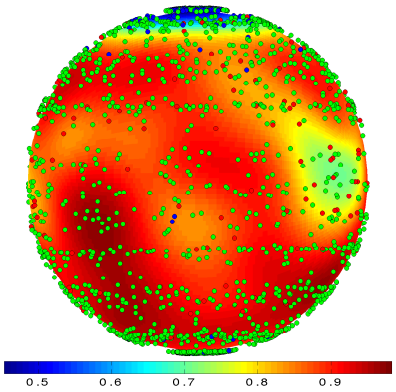


(c) Probability of success

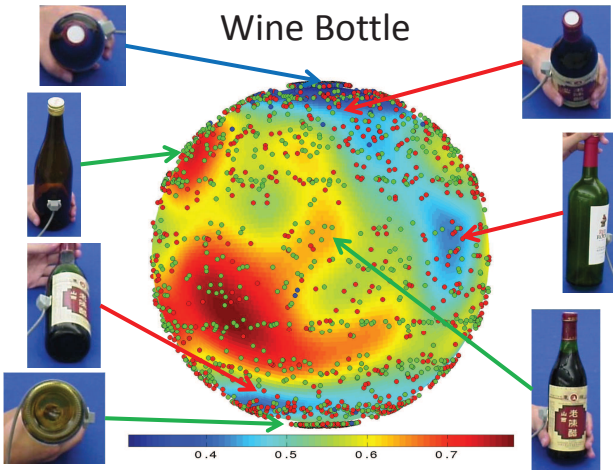
Figure 8.27: Performance as a function of aspect: drill



(a) Rate of coverage

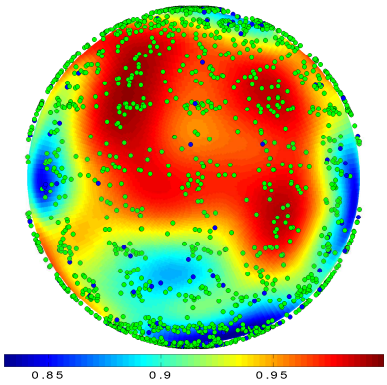


(b) Correct location

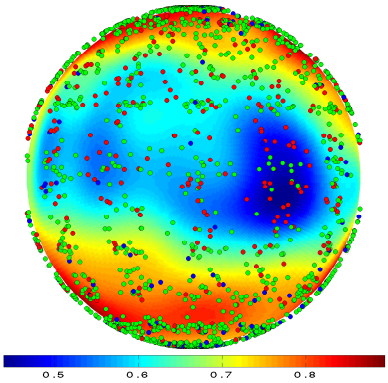


(c) Probability of success

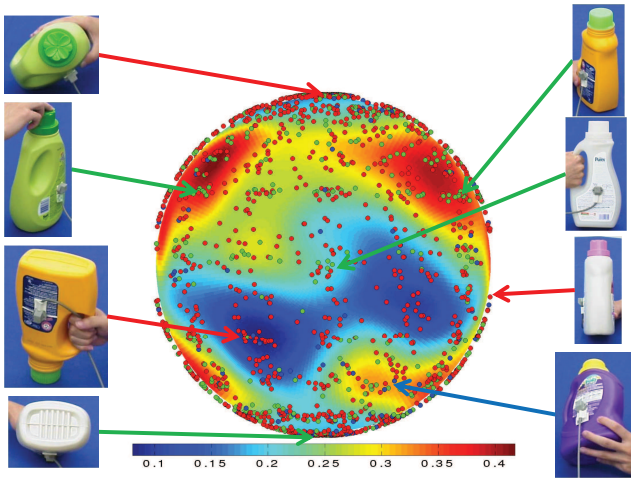
Figure 8.28: Performance as a function of aspect: wine bottle



(a) Rate of coverage



(b) Correct location



(c) Probability of success

Figure 8.29: Performance as a function of aspect: detergent bottle

means that the region of interest generally does not cover the entire rim. This usually yields a shape model with several contour lines that are not connected to each other. Sometimes, only a single curve is learned as a shape model. These shape models are not very discriminative and are ambiguous with aspect.

For the spatula (Fig. 8.31), the top and bottom views do not contain enough visual information. Also, the shape models learned on the handles are not very sensitive to camera tilt, which often results in large hand orientation errors.

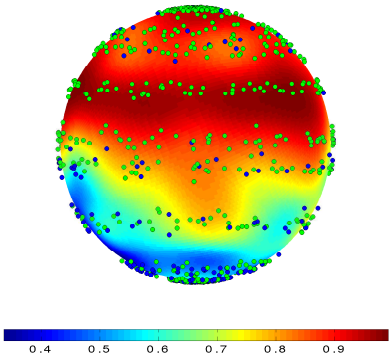
8.3.2 Aggregate Performance

From the example images in the previous section, we have seen that, from a number of viewing angles, the target object parts are either self-occluded or do not contain enough information for robust visual matching.

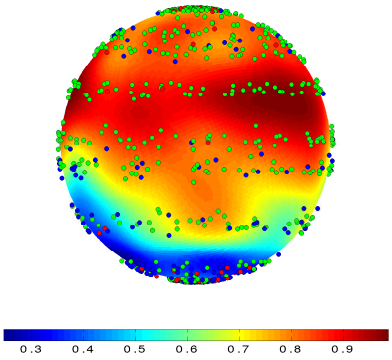
In this section, we show the aggregate performance of the algorithm for a large set of test images. In order to assess the performance of the proposed algorithm and to differentiate the contributions of the shape model matching and the hand orientation estimation processes, we compare the performance of our algorithm to several different baseline algorithms. All these random algorithms make use of the highest scored shape model in the test image (*top 1*). These different random algorithms are described as follows:

Random orientation. Given a correct bounding box matched by the highest scored shape model according to our algorithm, we randomly sample a hand orientation. Then, in the performance evaluation process, we measure the error between this random hand orientation and the ground truth hand orientation.

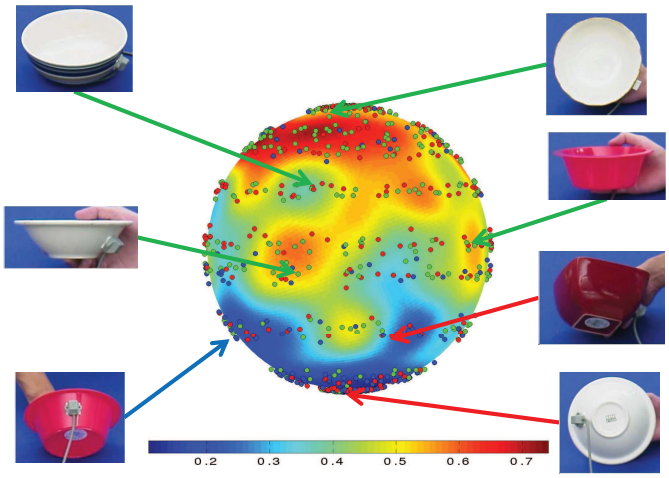
Random bounding boxes. Given a correctly estimated hand orientation by a the highest scored shape model according to our algorithm, we randomly generate a bounding box in the test image. The bounding boxes in the left and right images are generated separately without applying stereo constraints.



(a) Rate of coverage

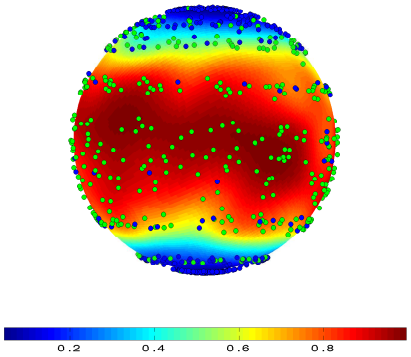


(b) Correct location

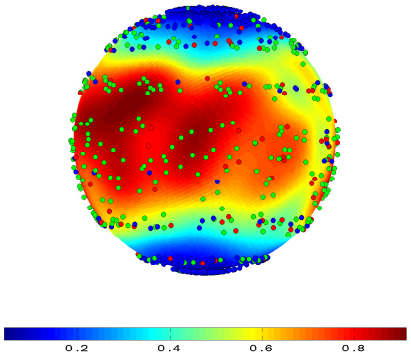


(c) Probability of success

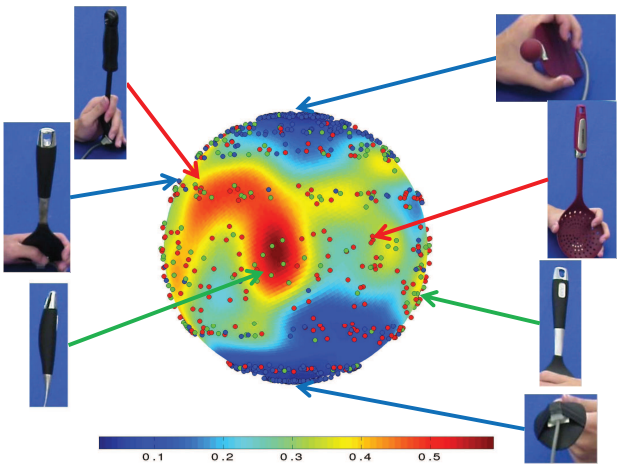
Figure 8.30: Performance as a function of aspect: bowl



(a) Rate of coverage



(b) Correct location



(c) Probability of success

Figure 8.31: Performance as a function of aspect: spatula

Random bounding boxes with stereo constraints. The same as above. However, the random bounding boxes generated in the left and right images are required to satisfy the stereo constraints.

In Fig. 8.32, we compare the mean precision of the above random algorithms with the proposed algorithm (for *top 1*, *top 2*, *top 3*, and *top all*). The bars represent mean performance over five experiments, and the whiskers on top of the bars represent standard deviation. In the horizontal direction, these bar plots are grouped according to the object categories. For all object categories, except for the balls, we match each test image against shape models from the same object categories. For the balls, since we have observed a ceiling effect on the performance, we match each test image of balls against shape models from all object categories in order to show some variance in performance. In each group of bars, the bars with red shading correspond to three random algorithms, while the bars with blue shading correspond to our proposed algorithm. For our proposed algorithm, besides the *top 1*, *top 2* and *top 3* approaches, we also show the performance of *top all*, which simply labels a test image as correct if any of the matched shape models is correct. Note that from *top 1* to *top all*, the performance monotonically increases, since more and more matched shape models are added. For the set of matched shape models considered for evaluation, we have *top 1* $\subseteq$ *top 2* $\subseteq$ *top 3* $\subseteq$ *top all*.

In seven out of ten object categories (all except the hammer, the spatula, and the ball), the algorithms from low to high performance are: *random orientation*, *random bounding boxes*, *random bounding boxes with stereo constraints*, and *top 1*. This is consistent with the original ordering in Fig. 8.32 from left to right. The reason that *random orientation* performs worse than *random bounding boxes* is because the hand orientation is in a higher dimensional space than 2D bounding boxes and thus has a higher number of degrees of freedom. In other words, it is easier to guess the bounding box than hand orientation. The *random bounding boxes with stereo*

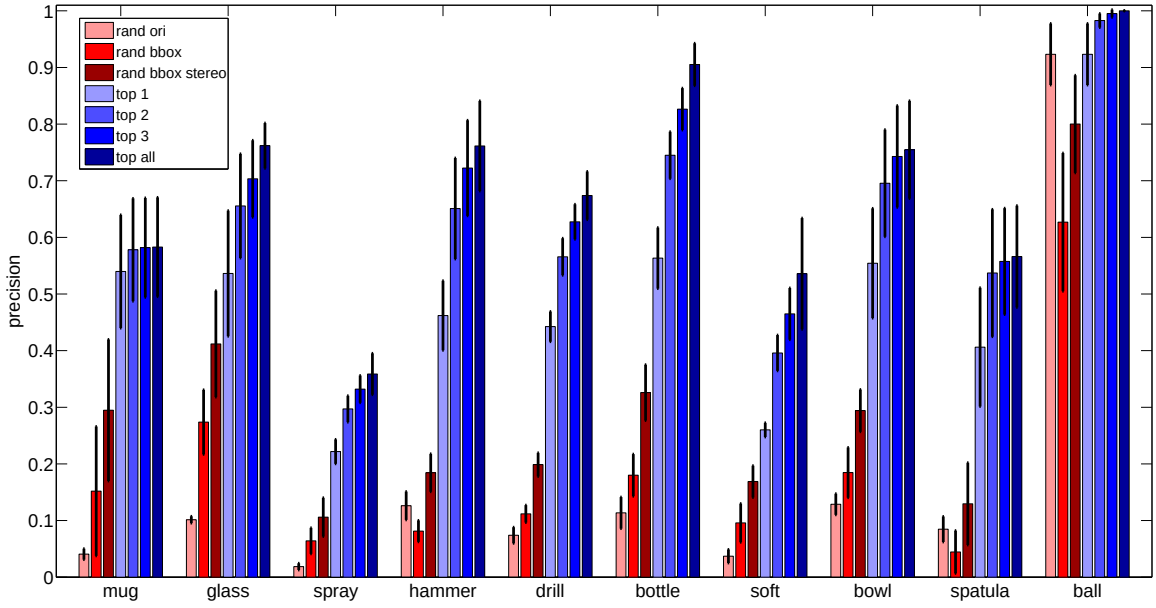


Figure 8.32: Performance comparison with random algorithms and oracle

constraints performs better than *random bounding boxes* since the former is required to satisfy the stereo constraints. This modification further reduces the number of degrees of freedom, and thus has a higher chance of guessing the bounding boxes correctly.

However, the above trend is not true for the hammer, spatula and ball. For the ball, the reason is that we consider arbitrary hand orientations as correct as long as the matched bounding boxes satisfy the IoU criteria. This is why the performance of *random orientation* is the same as that of *top 1*. For the hammer and spatula, it is relatively difficult to "guess" the correct bounding boxes for a test image, since both the hammer and spatula have narrow ground truth bounding boxes on the handles. A random bounding box tends to have small IoU with the ground truth bounding box. Also, it is relatively easy to guess the hand orientation since the grasps on the handles have rotational symmetry.

One of the oracle algorithms that we compare to is *top all* in Fig. 8.32. *Top all* uses all matched shape models (above their respective model thresholds t_i^*). If any of the matched shape models is correct, we label the image as being correctly identified. One possibility is that the robot incrementally tries each grasp until it finds one that works. Note that all algorithms (including the random algorithms) in Fig. 8.32 use the same set of model thresholds t_i^* to determine whether a shape model is matched or not. Because of this, the rates of coverage are the same for all algorithms for a particular object category.

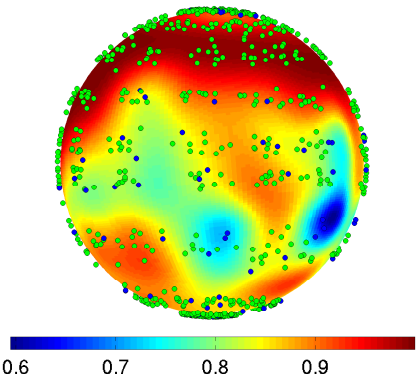
The precision of *top 1* is significantly higher than all random algorithms for all object categories except for the balls according to paired single-tail t-tests with Bonferroni adjustment ($\alpha = 0.017$) (Jensen and Cohen, 2000). For the balls, the *random orientation* algorithm has a precision exactly the same as the *top 1* approach; the precision of *top 1* is significantly better than that of the other two random algorithms ($p < 0.002$, in both cases). By aggregating results across all object categories, the *top 1* approach outperforms all random algorithms significantly (the same test as above). However, for all object categories except for the balls, the precision of *top 1* is significantly lower than any of *top 2*, *top 3* or *top all* (paired single-tail t-tests with Bonferroni adjustment, $\alpha = 0.017$). For the balls, the precision of *top 1* is not significantly lower than *top 2*, *top 3* or *top all* due to a ceiling effect ($p > 0.022$, 0.018 and 0.017, respectively). By aggregating results across all object categories, the precision of *top 1* is significantly lower than that of *top 2*, *top 3* or *top all* (the same test as above). This suggests that a statistically significant number of correct matches are not the highest scored among all matches, and therefore, it is generally beneficial to consider more matches besides the top one. However, for a majority of the 10 object categories (6 out of 10), there is no statistically significant difference between *top 3* and *top all*, which means that most of the correct matches are ranked within the first 3 highest scored matches. For hammers, drills, wine bottles, and

detergent/softener bottles, there is a performance gain by considering more than the top 3 matches ($p < 0.010$, 0.025, 0.002, and 0.047, respectively, paired t-test).

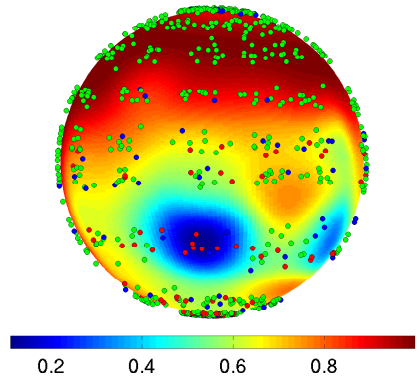
8.3.3 Generalization Across Object Categories

In the previous section, the experimental results are based on within-class shape models. One question that we would like to ask is how well do the shape models generalize between object categories? To answer this question, we first match the shape models learned on the glasses to the test images of mugs. The corresponding triplet of aspect spheres is shown in Fig. 8.33. We can see that these figures are essentially similar to those when matching against within-class shape models (Fig. 8.17). The shape models learned on the top or bottom of the set of glasses are more readily matched than the other shape models. These shape models are essentially very similar to those learned on the top of mugs. In Fig. 8.33(a), the rate of coverage is generally very high across all aspects, especially the upper hemisphere. Again, this can be explained by the fact that the top shape models can be more robustly identified when the opening of the mug is visible. Another reason that the rate of coverage is very high is because there are more shape models learned on both the top and bottom of glasses (roughly twice as many as those learned on the top of mugs). When considering grasp location errors, the maximum probability is lower (Fig. 8.33a), but still is as high as about 0.9. Fig. 8.33(c) looks very similar to Fig. 8.17(c), which means that the shape models learned on the top and bottom of glasses actually achieve similar performance to those learned on the top of mugs.

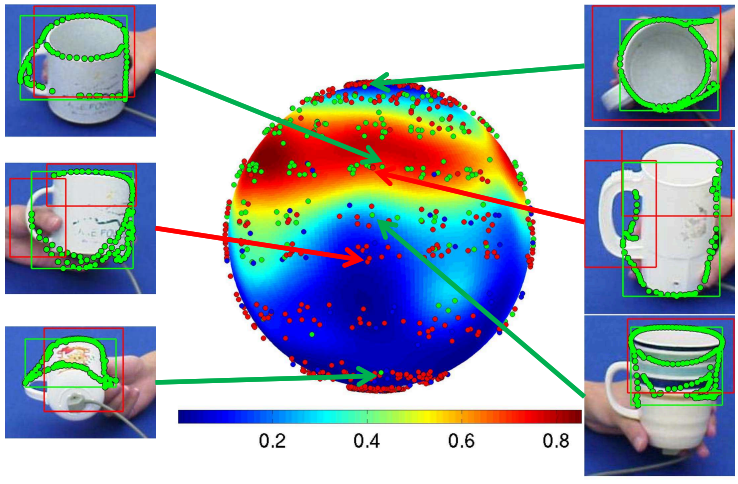
To further confirm the above observation, we also show the performance of the shape models learned on the mugs as a function of aspect when matched against test images of glasses (Fig. 8.34). Compared with Fig. 8.33, the maximum probabilities are generally lower in these figures. This is because the number of shape models learned on the top of mugs is fewer (roughly half) than those learned on the top and



(a) Rate of coverage



(b) Correct location

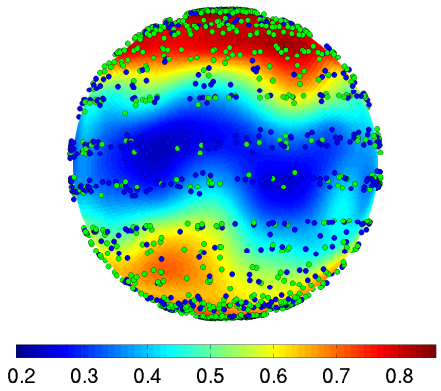


(c) Probability of success

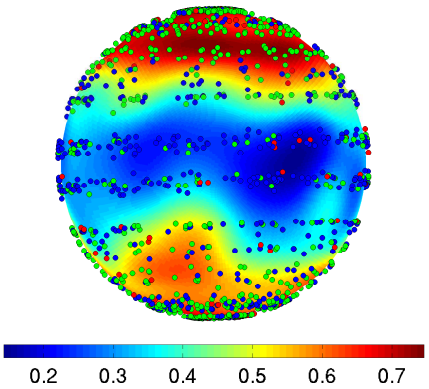
Figure 8.33: Performance as a function of aspect: mugs match against the shape models learned on the glasses

bottom of glasses. Another difference is that besides the upper hemisphere, the lower hemisphere also has higher probability than the other regions. The upper and lower hemispheres are roughly symmetric about the equator. This is because of the fact that we rotate the glasses 180 degrees in the image plane when grasping the bottom. For example, when the top of the glass is grasped during data collection, a point on the upper hemisphere of the aspect sphere corresponds to an aspect where the opening of the glass is visible. When the bottom of the glass is grasped, imagine that the glass is rotated 180 degrees in the image plane. This gives us an image of an upside-down glass with the opening visible, but the bottom surface is invisible. Since aspects with different in-plane rotations correspond to the same point on the aspect sphere, the image with the upside-down glass corresponds to the same point as the original image on the aspect sphere. Symmetrically, a point on the lower hemisphere corresponds to an aspect where the bottom surface of an upside-down glass is fully visible. Similar to the opening of the glasses, the shape models learned on the top of mugs are more robustly matched when the bottom surface of the glasses are fully visible, which corresponds to aspects on the lower hemisphere.

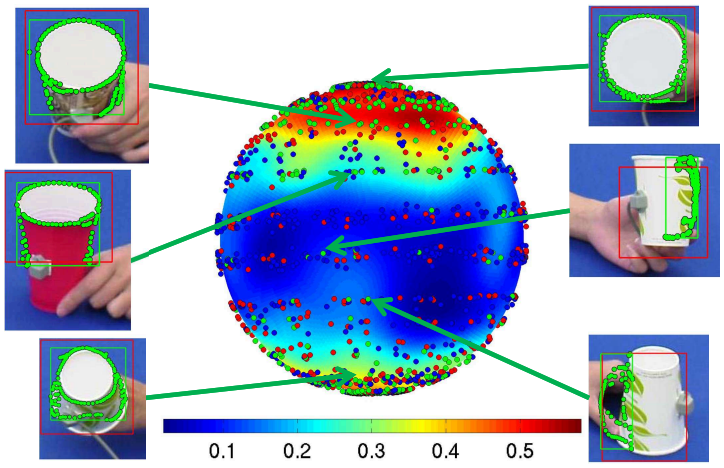
So what happens when we match test images of mugs against shape models that do not look similar to mug components? To answer this question, in Fig. 8.35, we show the performance of the shape models learned on spatulas as a function of aspect when matched against test images of mugs. Ideally, the shape models of spatulas should not "pop out" from images of mugs (with matching scores lower than model thresholds). Therefore, in this triplet of figures, we are interested in whether the rate of coverage is low. In Fig. 8.35(a), we can see that the maximum rate of coverage (about 0.4) is much lower than that in Fig. 8.17(a) (close to 1). The locations of the high probability regions tend to be random. Some example matches are shown with arrows pointing to the corresponding aspects (green dots) on the aspect sphere. In the top two test images, the shape models are learned on the top views of spatulas,



(a) Rate of coverage



(b) Correct location

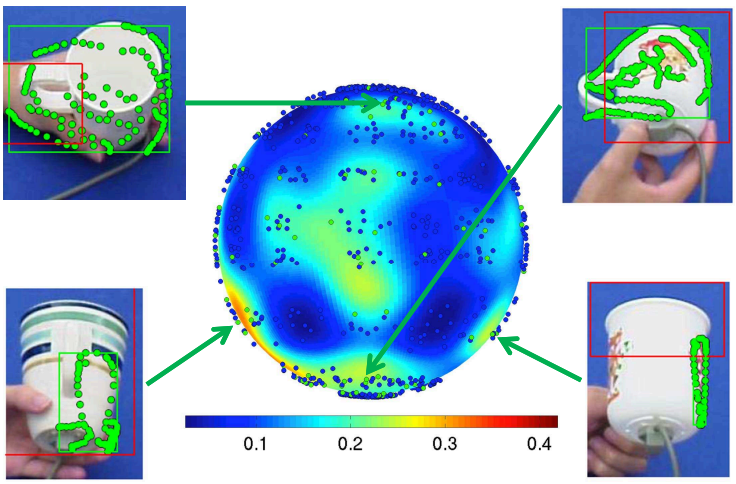


(c) Probability of success

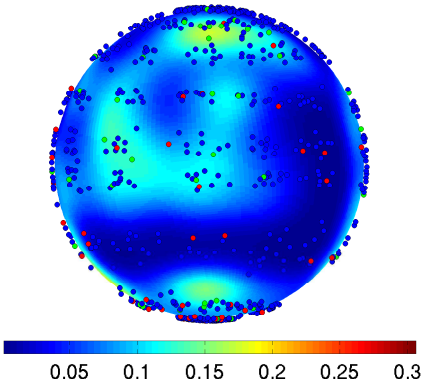
Figure 8.34: Performance as a function of aspect: glasses match against the shape models learned on the mugs

which accidentally capture part of the holding hand and data glove. In the bottom two test images, the shape models are learned on the handle of spatulas. These shape models accidentally match to the textures and patterns of mugs. Some of these false positive matches accidentally overlap with the ground truth bounding boxes (IoUs greater than 0.2, Fig. 8.35b). However, none of these cases actually suggest correct hand orientations (Fig. 8.35c). This is consistent with our expectation and serves as a sanity check for our experimental result evaluation process.

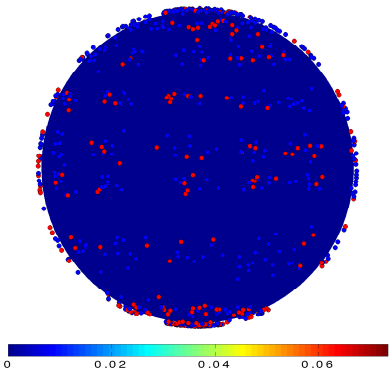
In Fig. 8.36(a), we show the generalization of learned shape models to different object categories for a single experiment. In this figure, each entry in the matrix is the rate of coverage when matching shape models of a particular grasp type (row) to images from all object categories (column). In each entry, the range of the value is between 0 and 1, with hotter colors representing higher values. For some particular grasp types, such as those learned from balls, drill handles, and bottle bodies, the corresponding shape models match to a wide range of object categories and achieve high rates of coverage. These shape models are usually simple and not very discriminative when matched to test images from other object categories. Therefore, some of the high value entries in Fig. 8.36(a) are true generalizations beyond object categories and the others are false positive matches. In order to know which entries are true generalizations, we use the proposed labeling algorithm to evaluate each match and show the corresponding probability of success matrix in Fig. 8.36(b). By comparing the two matrices in Fig. 8.36, the values of most entries decrease from Fig. 8.36(a) to Fig. 8.36(b), since we check the correctness of matches found in each test image and only consider the test images that are labeled as correct in latter. Recall that the PoS is the product of RoC and precision (Equation 8.1). Both the RoC and precision need to be high in order to achieve high PoS. In Fig. 8.36(b), the peak value is located at the right-bottom corner, which corresponds to the grasp type learned on balls matching to test images of balls. In this particular case, the PoS is close



(a) Rate of coverage



(b) Correct location



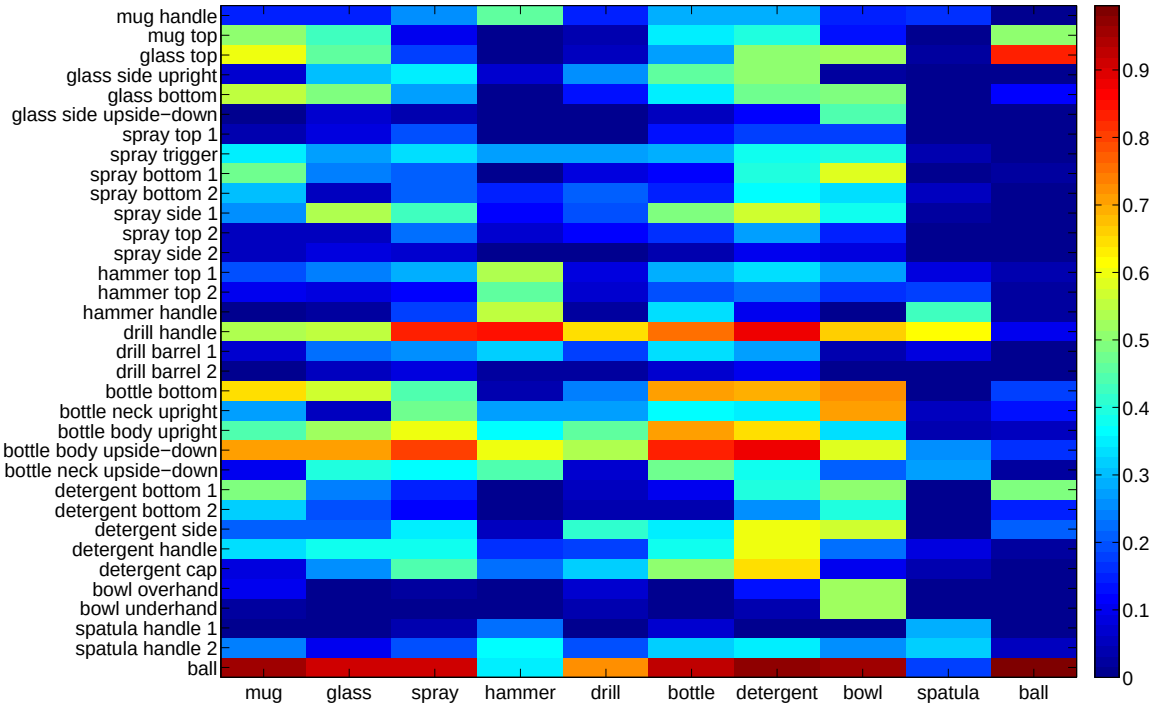
(c) Probability of success

Figure 8.35: Performance as a function of aspect: mugs match against the shape models learned on the spatulas

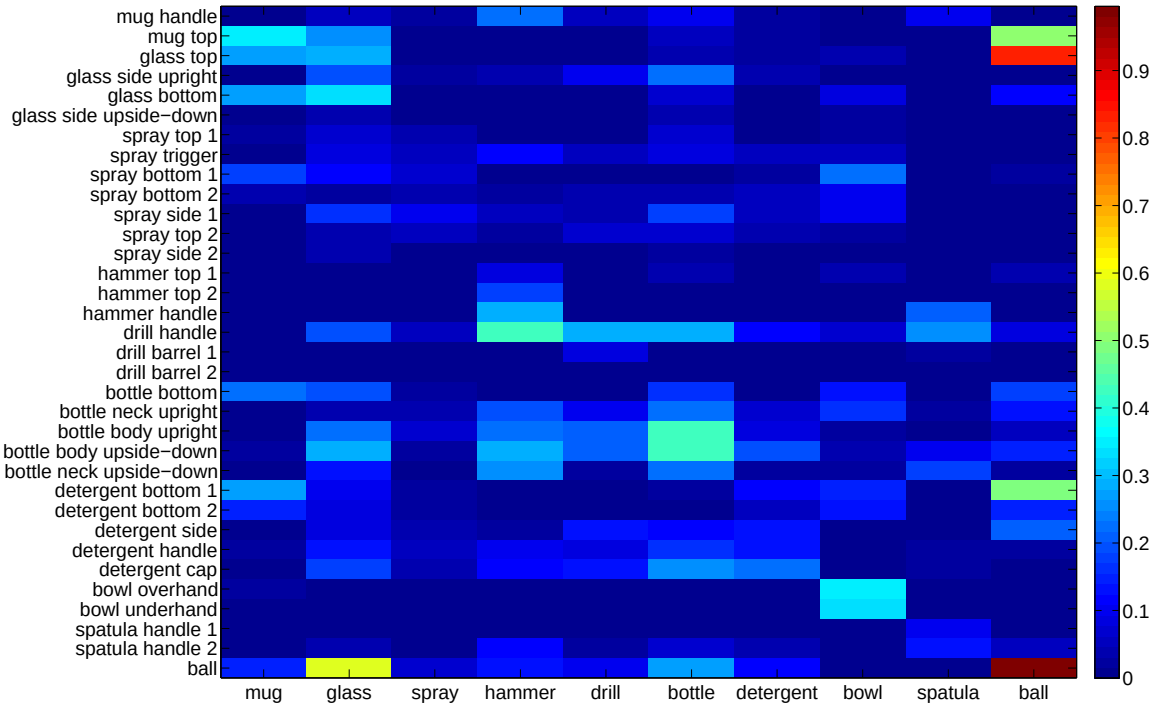
to 1, since both the RoC and precision are close to one. The second highest entry corresponds to the grasp type learned on the top of glasses matching to test images of balls. Symmetrically, the entry corresponding to the shape models learned on the balls also achieve high probability of success on the test images of glasses. This is because the top of a glass has a similar shape as a ball. Likewise, the shape models learned on the top of mugs also generalize to balls, and vice versa. The shape models learned on the top of mugs and the top/bottom of glasses achieve similar performance on the test images of mugs and glasses (the entries around the left-top corner), which is consistent with our observation in Fig. 8.33 and Fig. 8.34. Note that some of the entries in Fig. 8.36(b) may still contain matches that are accidentally labeled as correct by the labeling algorithm.

In order to show the generalizations between different object categories more clearly, we show the corresponding graphs in Fig. 8.37. In these figures, green nodes correspond to grasp types, and yellow nodes correspond to the test images of some object categories. A directed edge pointing from a grasp type to an object category indicates a generalization between them. For visual clearance, we only show edges with $RoC > 0.5$ in Fig. 8.37(a), and edges with $PoS > 0.2$ in Fig. 8.37(b). In Fig. 8.37(a), we can see that shape models from many different grasp types achieve $RoC > 0.5$ on the images of bowls and detergent bottles. Shape models corresponding to some grasp types, such as *bottle body upside-down*, achieve $RoC > 0.5$ on the test images of many different object categories. However, some of these edges correspond to RoCs that are exaggerated by false positive matches. In Fig. 8.37(b), most of the false positive edges are pruned out by the labeling algorithm. An edge in Fig. 8.37(a) that is removed in Fig. 8.37(b) usually corresponds to a false generalization caused by false positive matches; while an edge that exists in both Fig. 8.37(a) and Fig. 8.37(b) usually corresponds to a true generalization.

The observed generalizations of shape models across object categories in Fig. 8.37(b)

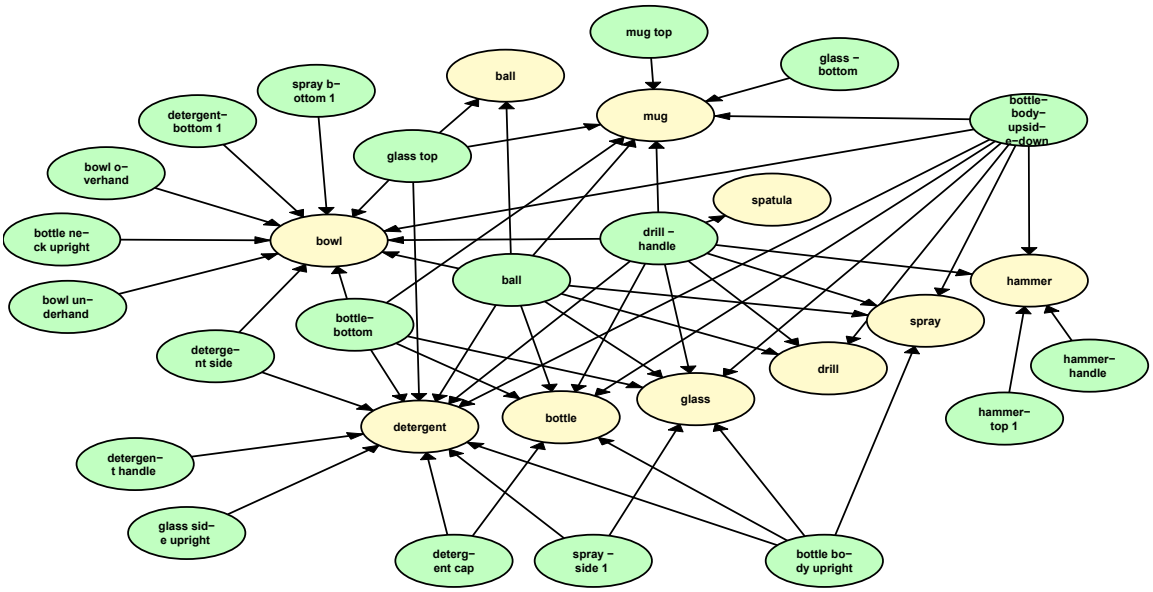


(a) Rate of coverage

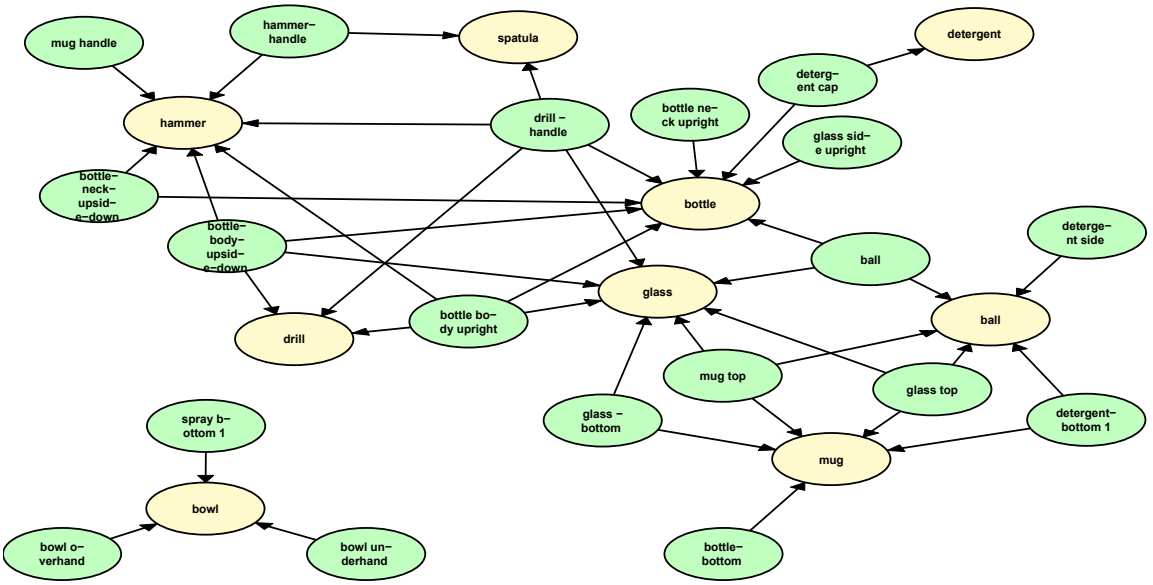


(b) Probability of success

Figure 8.36: The performance of matching shape models of different grasp types to images from all object categories.



(a) All generalizations with $RoC > 0.5$

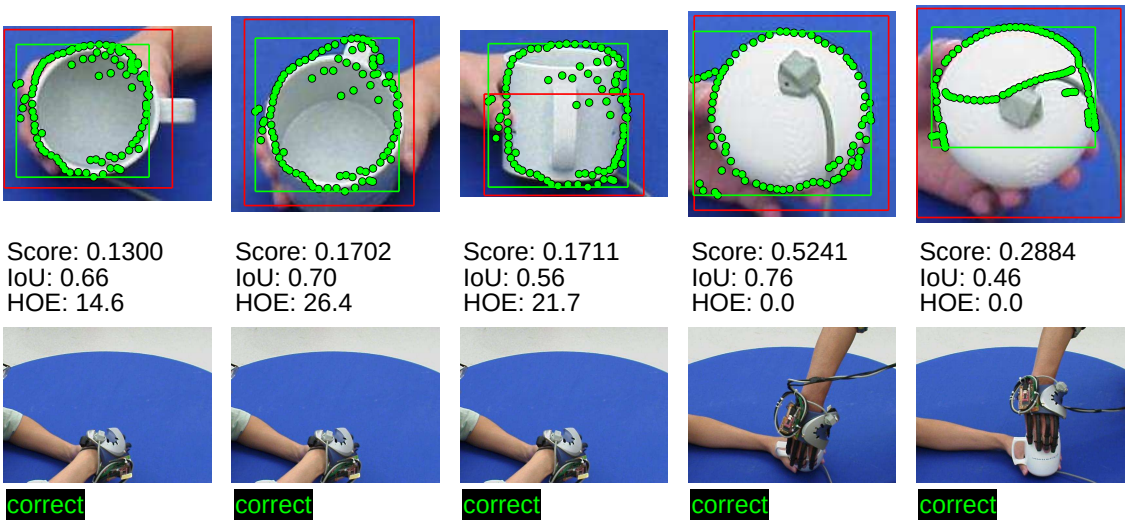


(b) All generalizations with $PoS > 0.2$

Figure 8.37: Graph representation of typical generalizations between object categories

are summarized in the table below. In this table, $A \Rightarrow B$ denotes a generalization from a shape model learned on object A to test images of object B; $A \Leftrightarrow B$ denotes a mutual generalization between objects A and B. Each generalization case in this table is indexed into some example test images with matched shape models (Fig. 8.38 to 8.57). These figures are composed of representative examples of generalization across object categories. Both true positive matches (those that are labeled as correct by the automatic labeling algorithm) and false positive matches (those that are labeled as wrong by the automatic labeling algorithm) are given. However, we should note that some of the true positives are counter-intuitive matches that are coincidentally labeled as correct by the labeling algorithm due to consistent hand location/orientation with the ground truth.

- glass top/bottom $\Leftrightarrow$ mug top $\Leftrightarrow$ ball — Fig. 8.38
- glass side $\Leftrightarrow$ bottle body — Fig. 8.39, 8.40
- spray bottom $\Rightarrow$ mug top — Fig. 8.41(a), (b)
  $\Rightarrow$ bowl — Fig. 8.41(c), (d), (e)
- spray side $\Rightarrow$ glass side — Fig. 8.42
  $\Rightarrow$ wine bottle body — Fig. 8.43
- hammer handle $\Leftrightarrow$ spatula handle — Fig. 8.44, 8.45
- drill handle $\Rightarrow$ hammer handle — Fig. 8.46
  $\Leftrightarrow$ wine bottle — Fig. 8.47, 8.48
  $\Rightarrow$ spatula handle — Fig. 8.49
- bottle bottom $\Rightarrow$ mug top — Fig. 8.50
  $\Rightarrow$ glass top/bottom — Fig. 8.51, 8.52
  $\Rightarrow$ bowl — Fig. 8.53



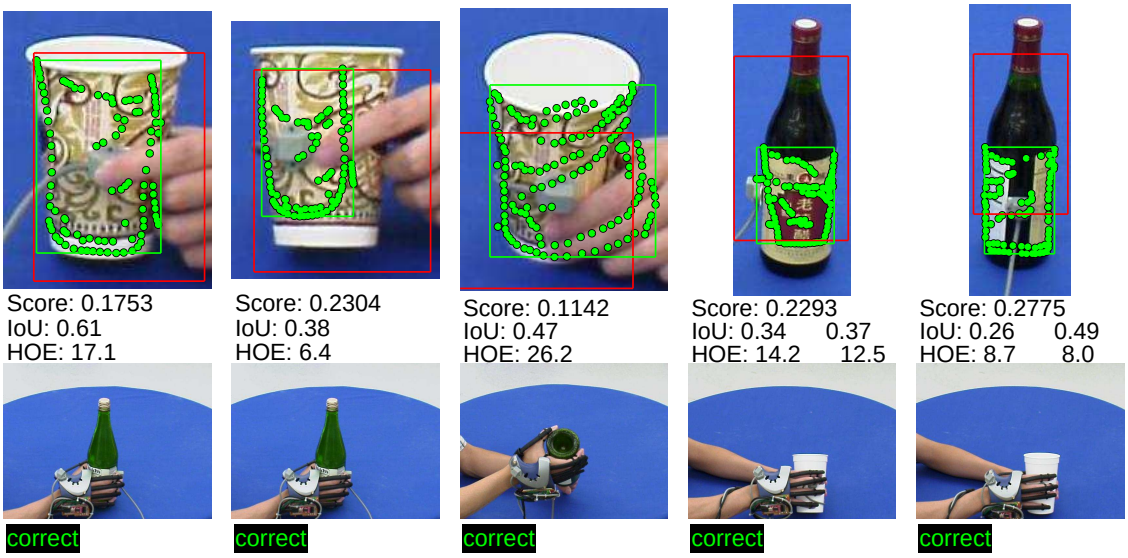
(a) (b) (c) (d) (e)

Figure 8.38: Shape model generalization between mugs and balls

- bottle body upside-down ⇒ glass side Fig. 8.39(c)
  ⇒ hammer handle Fig. 8.54
  ⇔ detergent bottle side/cap Fig. 8.55, 8.56
- detergent bottle bottom ⇔ ball Fig. 8.57

In Fig. 8.38, we show some generalization between mugs and balls. Particularly, the shape models learned on the top of mugs and balls capture similar circular shapes. In Fig. 8.38(c), the match of a ball shape model is coincidentally labeled as correct, due to the fact that the estimated hand location and orientation is consistent with those of a handle grasp. In Fig. 8.38(d) and 8.38(e), the matches of shape models learned from the top of mugs are labeled as correct since they satisfy the IoU criterion. The HOEs are always 0 for test images of balls, since we consider arbitrary hand orientation as correct.

In Fig. 8.39, we show some generalization between the glass and the wine bottle. Particularly, the shape models learned from the side grasps of the glass and the wine bottle capture similar shapes. In Fig. 8.39(b), the match of the shape model

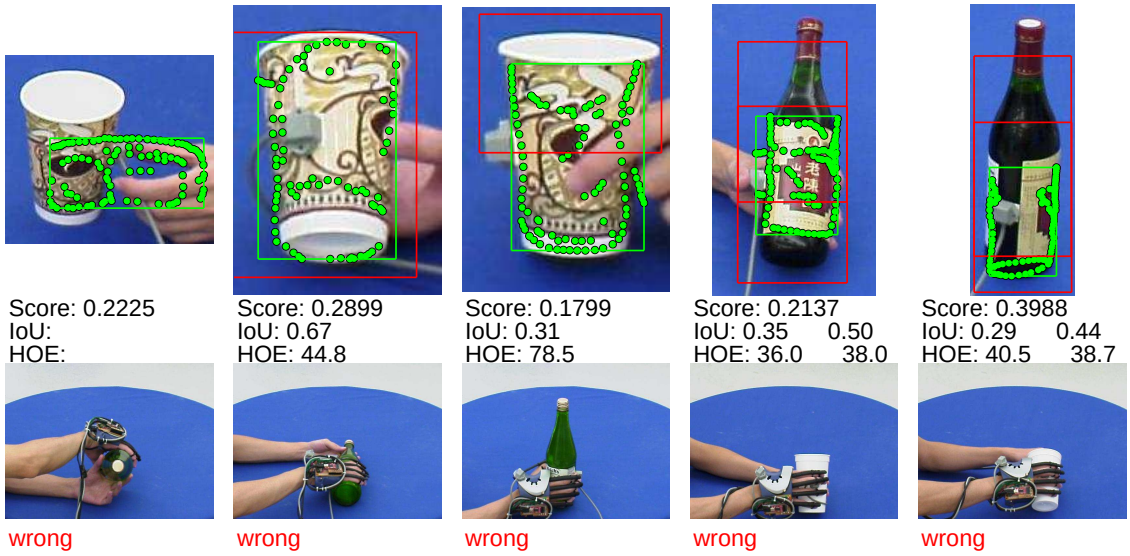


(a) (b) (c) (d) (e)

Figure 8.39: Shape models generalize between glass and wine bottle.

is distracted by inner textures of the glass and the sensor. However, this match is still labeled as correct by the assessment algorithm, since it satisfies both the hand location and orientation criteria.

In Fig. 8.40, we show some false positive matches between the glass and the wine bottle. In Fig. 8.40(a), the shape model learned from the top view of wine bottles are matched to the inner texture of the glass and part of the hand that holds the glass. This match does not overlap with any of the ground truth bounding boxes with IoU greater than 0.2, and therefore, is labeled as wrong by the assessment algorithm. In Fig. 8.40(b), 8.40(d),and 8.40(e), the matches of shape models are labeled as wrong due to hand orientation errors that are larger than 30 degrees, although these matches match the gross shape of the query objects fairly well. In Fig. 8.40(c), the match of the shape model actually estimates both the hand location and orientation accurately. However, this query image of the glass does not have a ground truth that corresponds to a side grasp. As a result, the matched shape model is only compared to the ground truth hand location and orientation corresponding to a top grasp, and therefore, is



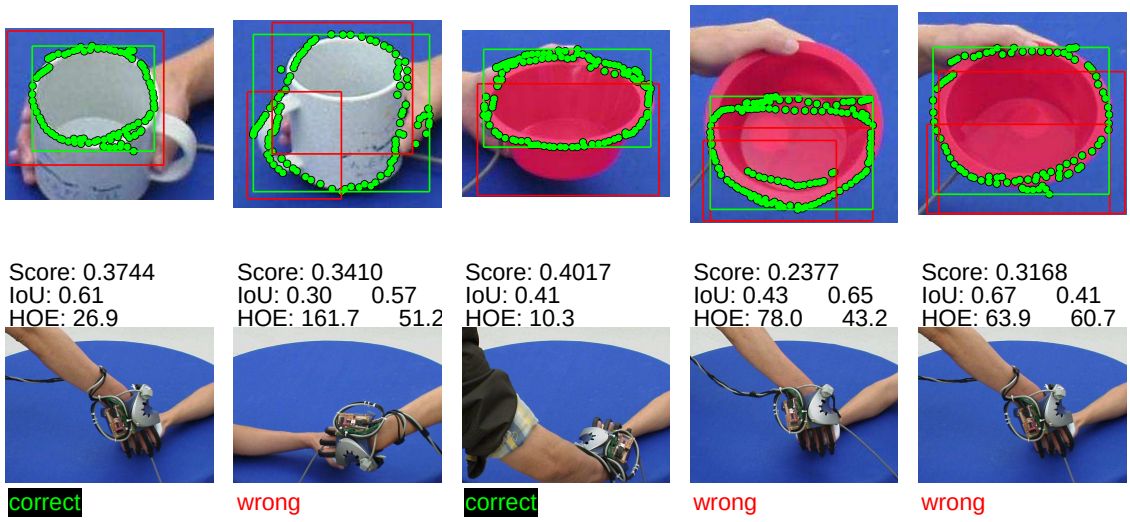
(a) (b) (c) (d) (e)

Figure 8.40: Shape models generalize between glass and wine bottle.

labeled as wrong due to a large hand orientation error.

In Fig. 8.41, shape models learned from bottom of spray bottles are matched to the images of mugs and bowls. These shape models capture simple ellipse shapes, and can match very well to the top of mugs and bowls for certain aspects. Although some of these matches satisfy both the hand location and orientation criteria (Fig. 8.41a, c), some of them have large hand orientation errors (Fig. 8.41d, e). In Fig. 8.41(b), the shape model is matched to the outer contour of a mug. This match is compared to both ground truths that are associated with this query image. This match is labeled as wrong, since it has large hand orientation errors with both ground truth hand orientations. In Fig. 8.41(d), the match of the shape model is distracted by the inner edge of the bowl. This match causes large hand orientation errors, and therefore, is labeled as wrong by the assessment algorithm.

In Fig. 8.42, shape models corresponding to a side grasp of the spray bottles are matched to the query images of a glass. During the grasp region extraction, the approximated grasp locations on the objects tend to be inaccurate, due to the fact that

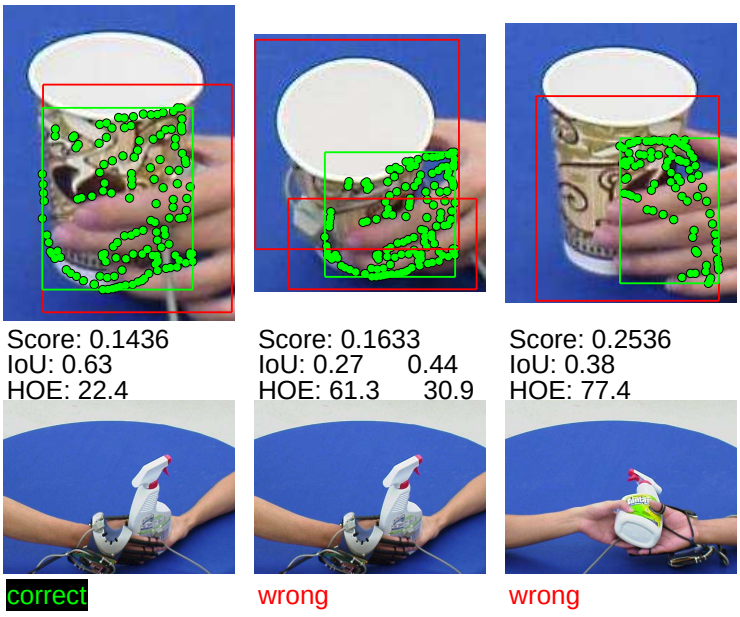


(a) (b) (c) (d) (e)

Figure 8.41: Shape models learned on the bottom of spray bottles generalize to mugs and bowls.

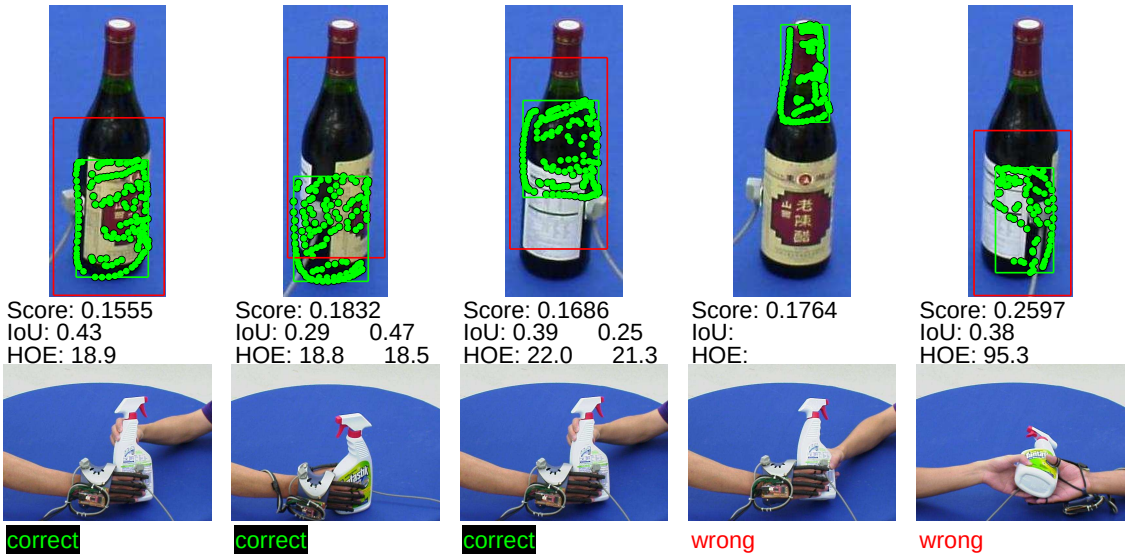
only a small portion of the spray bottle is occluded by the grasping hand. As a result, the shape models learned from these grasp locations are noisy and discontinuous. In this figure, most of these matches are distracted by the inner texture of the glass and part of the hand that holds the glass.

However, the shape models corresponding to the side grasp of spray bottles tend to match well on the body of wine bottles (Fig. 8.43). This is in part due to the fact that the wine bottle also has a label, and the shape models match well to both the outer and inner edges of the label. In addition, the hand orientations associated with the grasps on the body of the wine bottles are consistent with those from the side of the spray bottles. Therefore, the matches in Fig. 8.43(a), 8.43(b), and 8.43(c) are labeled as correct by the assessment algorithm. In Fig. 8.43(d), the match does not have IoU greater than 0.2 with any ground truth bounding box, and therefore, is labeled as wrong. In Fig. 8.43(e), the shape model is labeled as wrong due to a large hand orientation error.



(a) (b) (c)

Figure 8.42: Shape models learned on the side of spray bottles generalize to a glass.



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

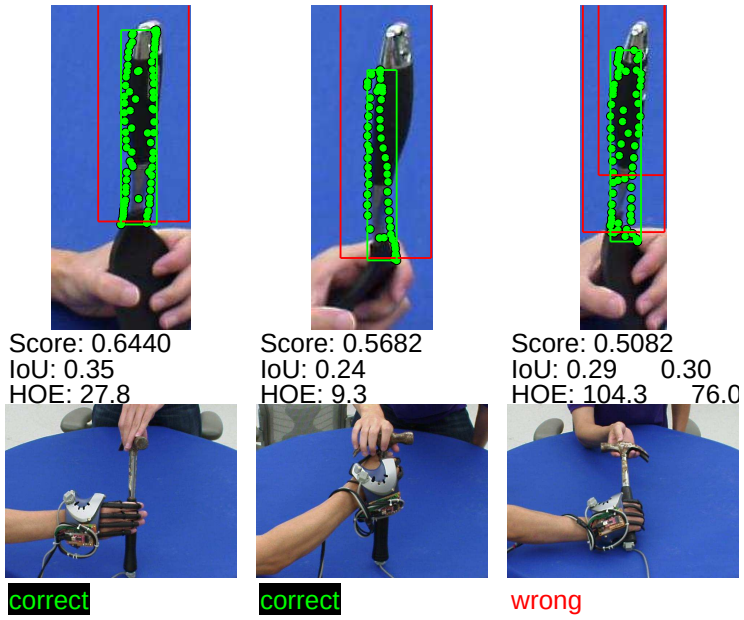
Figure 8.43: Shape models learned on the side of spray bottles generalize to glasses and wine bottles.

A major part of generalization between shape models comes from the handles of different objects, such as the hammer, the spatula, and the drill. From Fig. 8.44 to 8.49, we show an extensive set of example matches of the shape models that are learned from the handles of different objects.

In Fig. 8.44 and 8.45, we show the generalization between the handles of spatulas and those of the hammers. In most cases, the shape models learned from the handles of spatulas and hammers capture similar elongated cylindrical shapes. These shape models usually generalize well between spatulas and hammers (Fig. 8.44a, b, 8.45a). However, there is an ambiguity in aspect when the handle is tilted towards or away from the camera. Due to this reason, some of the matches cause large hand orientation errors (Fig. 8.44c, 8.45d). In Fig. 8.45(b), the shape model is learned from a top view of the spatula. This shape model captures a small part of the spatula and a large part of the grasping hand. In the query image, this shape model matches well to a top view of the hammer, and is coincidentally labeled as correct. In Fig. 8.45(c), the same shape model is matched to a frontal view of the hammer head. However, this match is labeled as wrong due to a large hand orientation error.

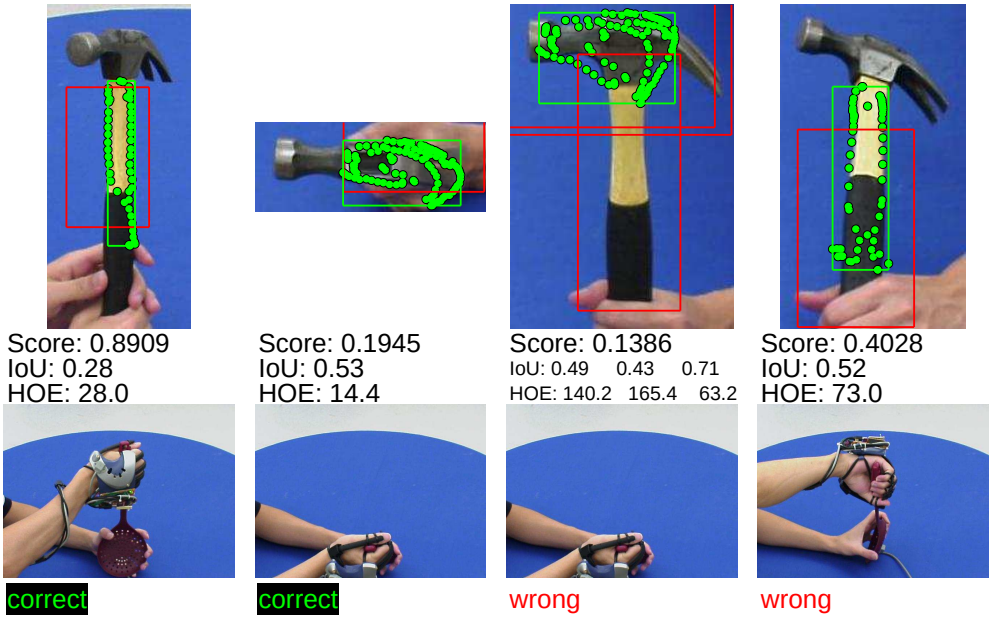
In Fig. 8.46, shape models learned from the handle of drills are matched to the query images of hammers. Again, in most cases, the matched shape models predict hand locations and orientations accurately and are labeled as correct by the assessment algorithm (Fig. 8.46a, b, c). However, in some other cases, the matches cause large hand orientation errors (Fig. 8.46d, e).

In Fig. 8.47 and 8.48, shape models learned from the handle of drills are matched to the images of wine bottles. In Fig. 8.47(a), 8.47(b), and 8.47(c), the shape models match well to the neck or the body of the wine bottle, and are labeled as correct. In Fig. 8.47(d) and 8.47(e), the shape models are distracted by the edges of the label and the sensor, and are coincidentally labeled as correct by the assessment algorithm. Some false positive matches are shown in Fig. 8.48. These matches are labeled as



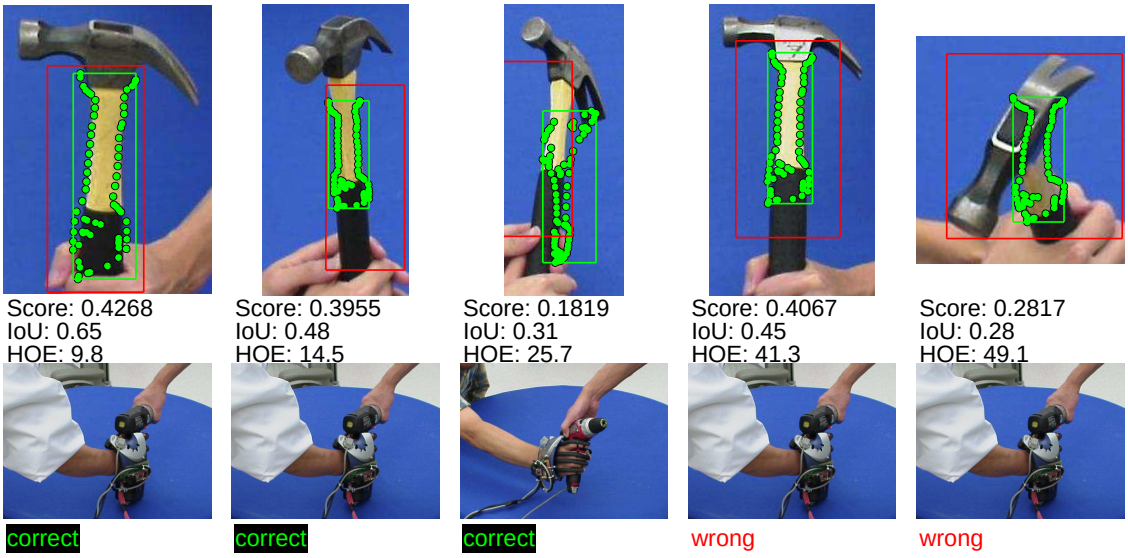
(a) (b) (c)

Figure 8.44: Shape models generalize between hammer and spatula handles.



(a) (b) (c) (d)

Figure 8.45: Some generalizations between hammer and spatula handles that are labeled as false positives by the labeling algorithm.



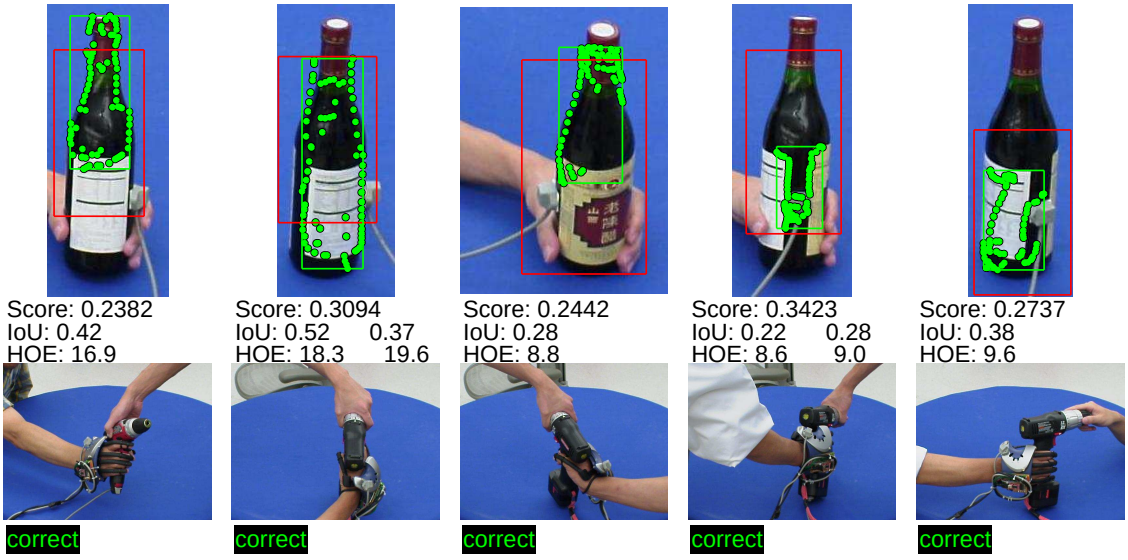
(a) (b) (c) (d) (e)

Figure 8.46: Shape models generalize between drill and hammer handles.

wrong by the assessment algorithm due to large hand orientation errors.

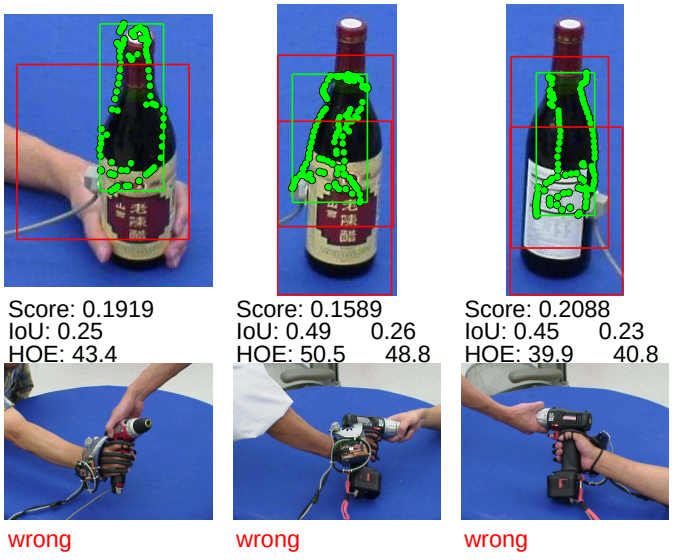
In Fig. 8.49, shape models learned from the handle of drills are matched to the images of spatulas. Most of these shape models match very well to the handles of spatulas. However, there is an ambiguity in aspect when the spatula is tilted towards or away from the camera. Due to this reason, some of the matches cause large hand orientation errors (Fig. 8.49d, 8.49e).

Another widely observed shape generalization between object categories corresponds to the shape models learned from the bottom of wine bottles matching to the top of the mug, the glass, and the bowl. An extensive set of such examples is shown in Fig. 8.50, 8.51, 8.52, and 8.53. From viewing angles where the opening of the mug, the glass, or the bowl are clearly visible (slightly above the frontal view), the shape models learned from the bottom of wine bottles can usually match robustly to the query images. In some cases, the corresponding hand orientation errors are slightly above 30 degrees (Fig. 8.50b, 8.52a, and 8.53c). In some other cases, the learned shape models are too simple to be discriminative, and are matched to a small set of



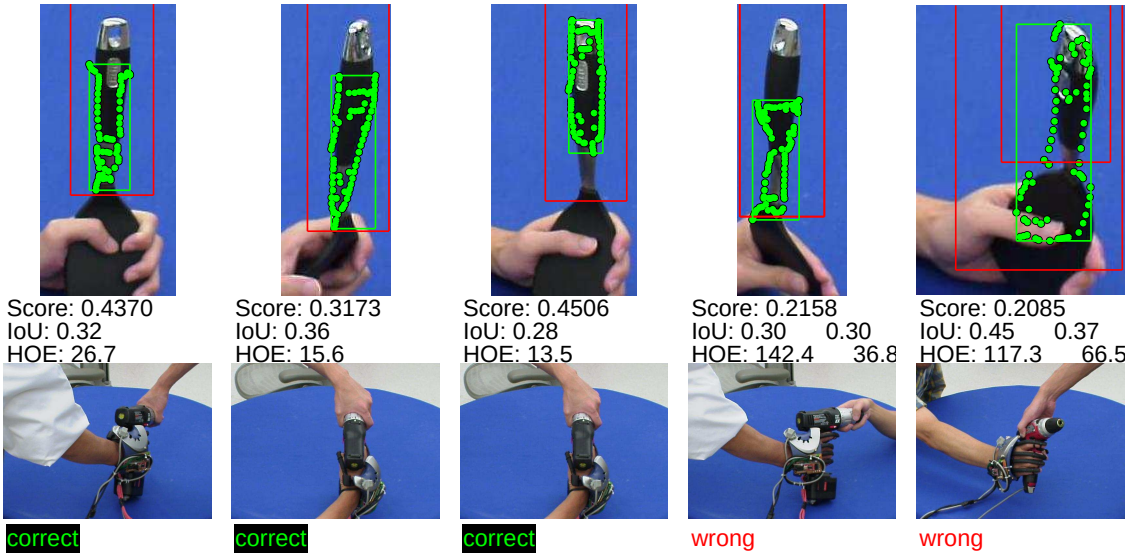
(a) (b) (c) (d) (e)

Figure 8.47: Shape models generalize between wine bottles and drill handles.



(a) (b) (c)

Figure 8.48: Shape models generalize between wine bottles and drill handles.



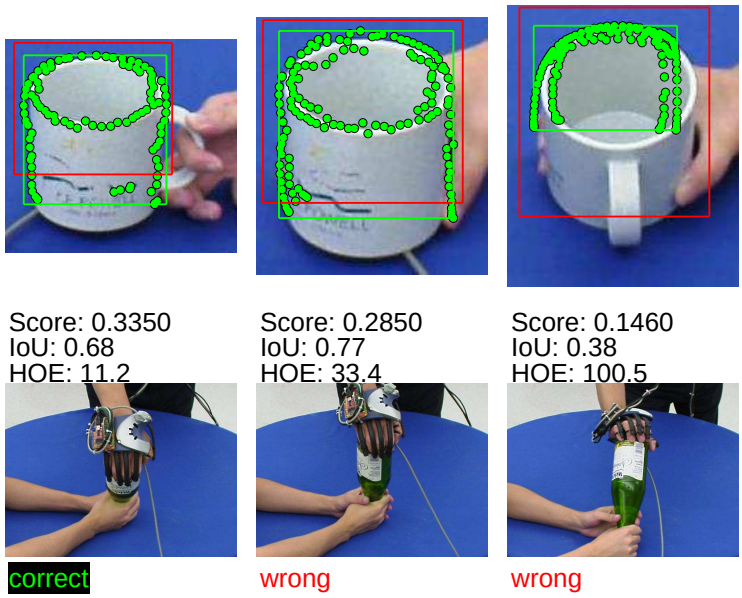
(a) (b) (c) (d) (e)

Figure 8.49: Shape models generalize between drill and spatula handles.

edges in a query image (Fig. 8.50c, 8.52b).

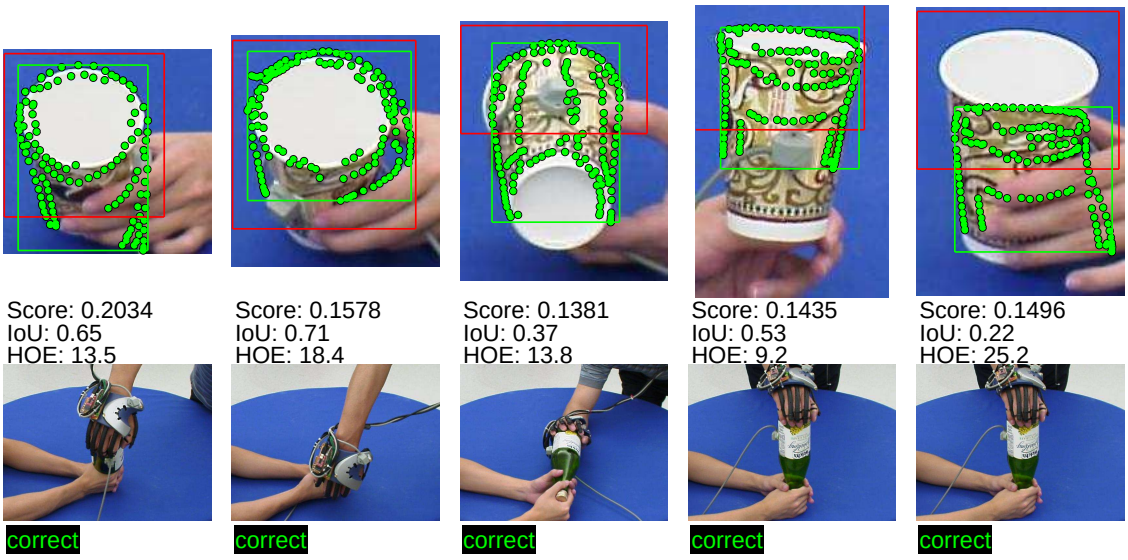
In Fig. 8.54, a shape model corresponding to a side grasp of wine bottles is matched to the images hammers. In Fig. 8.54(a) and 8.54(b), the matches are labeled as correct due to consistent hand location and orientation with the ground truth. However, this shape model can match to a wide range of aspects when the hammer is tilting towards or away from the camera. This causes large hand orientation errors in some cases (Fig. 8.54c).

In Fig. 8.55, the shape models learned from the bottom of wine bottles match well to the the caps of detergent bottles. Due to the fact that the cap of a detergent bottle is rotationally symmetric, these shape models can usually match well to a wide range of aspects. Some false positive matches are shown in Fig. 8.56. In Fig. 8.56(a), 8.56(b), and 8.56(c), the shape model is learned from a top view of the wine bottle, when the human teacher is grasping from the side. Theoretically, the grasp regions extracted from these images should be empty, since the grasping hand does not occlude any part of the bottle. However, due to noise (e.g., the bottle is slightly moved during the



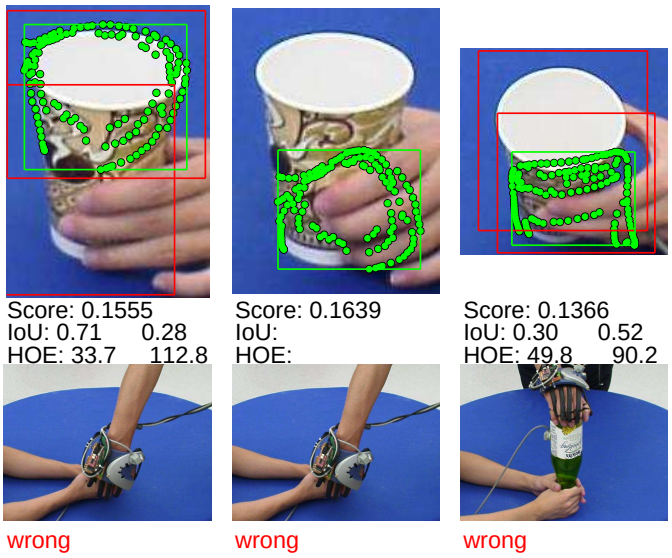
(a) (b) (c)

Figure 8.50: Shape models generalize between mugs and wine bottles.



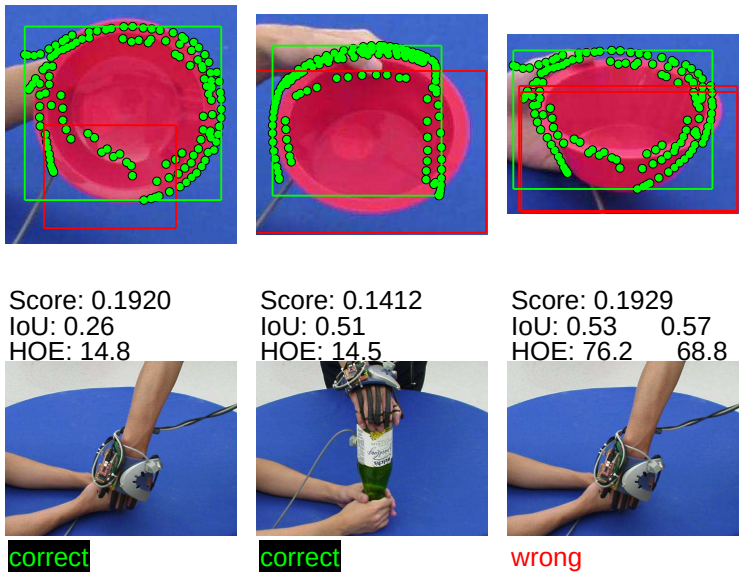
(a) (b) (c) (d) (e)

Figure 8.51: Shape models generalize between glasses and wine bottles.



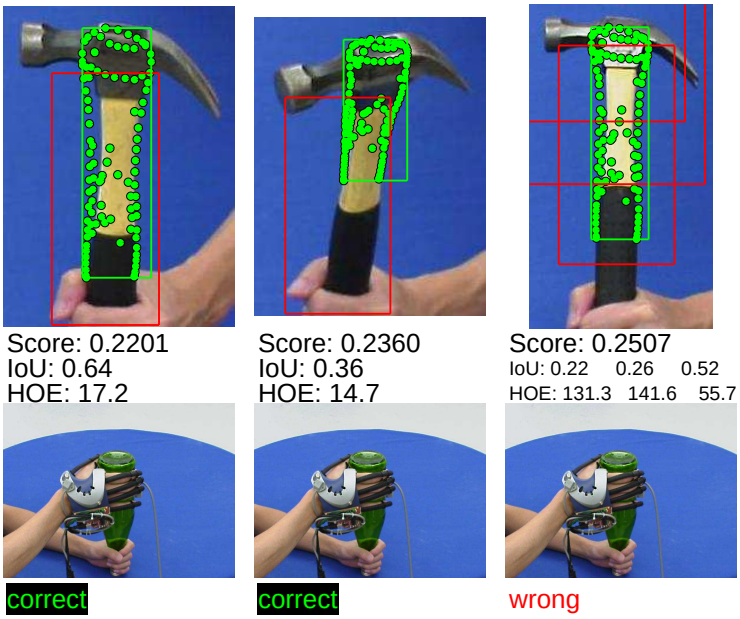
(a) (b) (c)

Figure 8.52: Shape models generalize between glasses and wine bottles.



(a) (b) (c)

Figure 8.53: Shape models generalize from wine bottles to bowls.

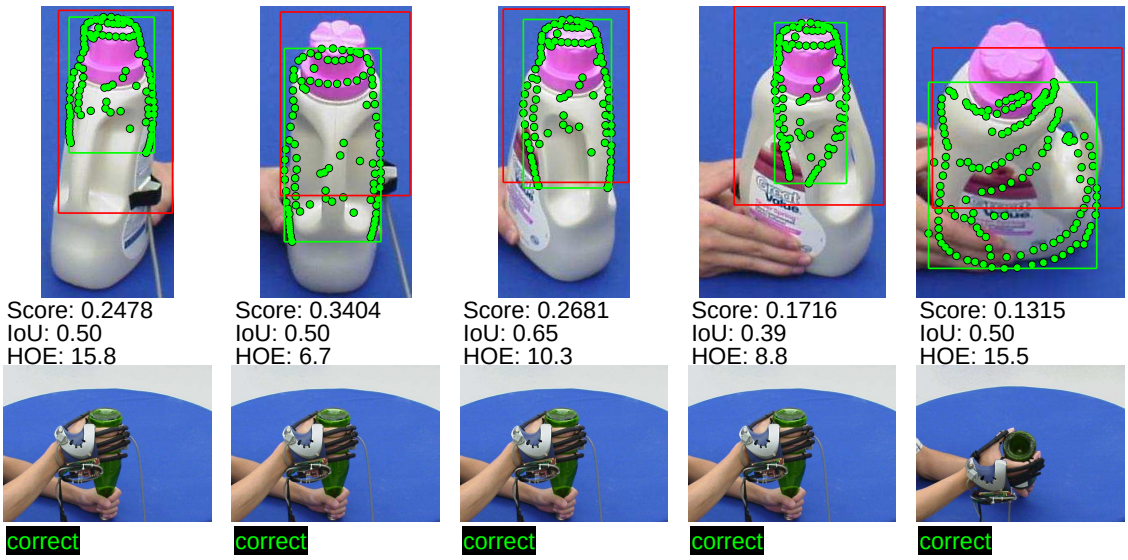


(a) (b) (c)

Figure 8.54: Shape models generalize from wine bottles to hammer handles.

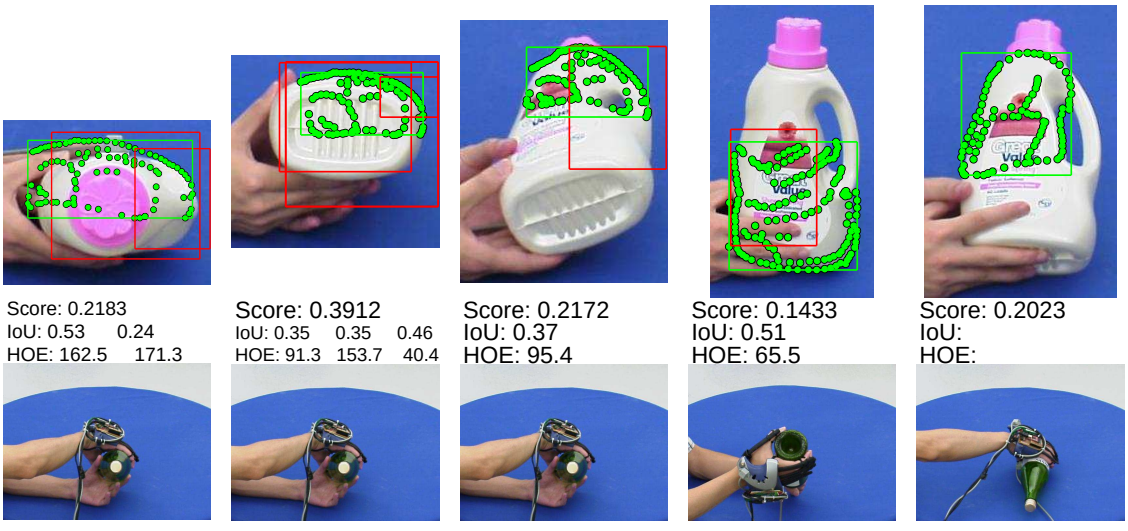
grasp), our algorithm still finds some plausible image regions as the grasp regions. These image regions usually do not contain related shapes, and therefore, the shape models learned from these regions tend to be general and indiscriminative (although a large portion of these shape models have already been filtered out during the model filtering process). As a result, these shape models usually cause false positive matches in query images. In Fig. 8.56(d) and 8.56(e), the learned shape models capture partial textures on the labels, and are distracted by the inner edges on the labels of the detergent bottle.

In Fig. 8.57, the shape models corresponding to bottom grasps on the detergent bottles match well to the query images of balls. This is in part due to the fact that the query images of balls are usually texture-free. As a result, the shape model can usually find the correct location of the ball. Since we consider arbitrary hand orientation as correct, the corresponding match is labeled as correct as far as the predicted hand location is correct.



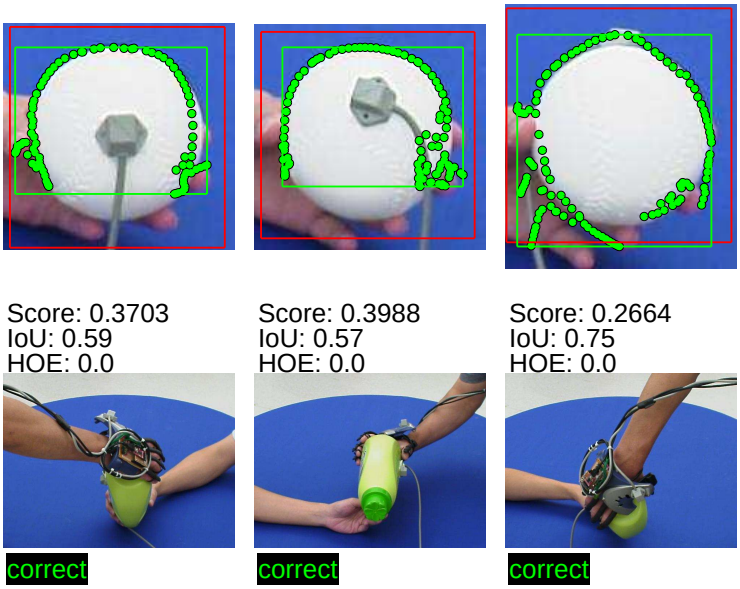
(a) (b) (c) (d) (e)

Figure 8.55: Shape models generalize from wine bottles to detergent bottles.



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

Figure 8.56: Some generalizations between wine and detergent bottles that are labeled as false positives by the labeling algorithm.



(a) (b) (c)

Figure 8.57: Shape models learned on the bottom of detergent bottles generalize to balls.

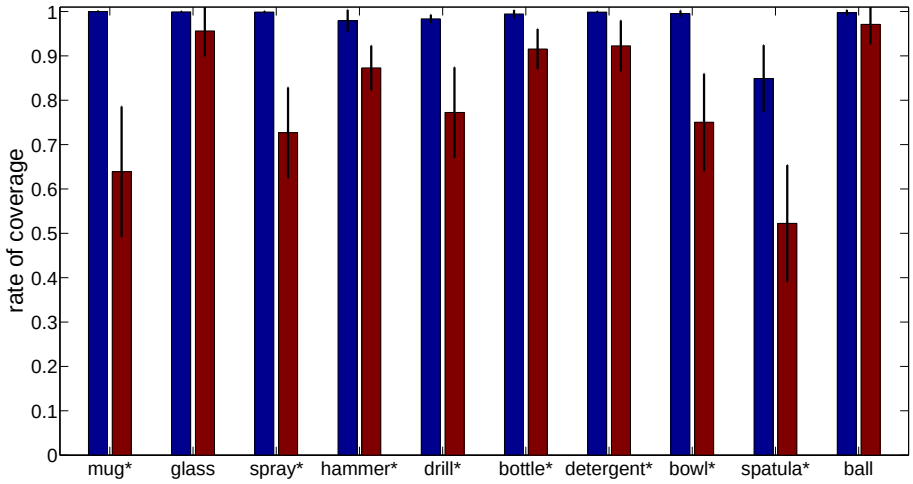
8.3.4 Match-against-all vs. Match-within-class

One question that we would like to ask is how does the performance of matching a test image against all shape models (match-against-all) look like when compared to matching against the shape models learned within the same object category (match-within-class)? In order to compare the performance of match-against-all and match-within-class, we use *Top 1* here. In the case of match-within-class, we assume that we know *a priori* what the object class label is.

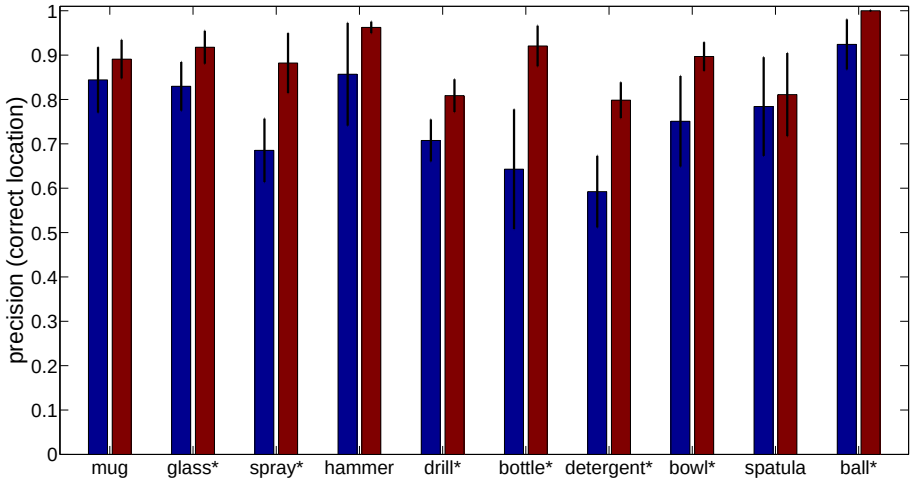
The rate of coverage, precision (correct location), and precision (correct location and orientation) of match-against-all and match-within-class are compared in Fig. 8.58. The difference between precision (correct location) and precision (correct location and orientation) is that we only check the IoU between the ground truth and predicted grasp regions for the former. For all objects, the rate of coverage of match-within-class is lower than that of match-against-all. This is because

that match-against-all considers a much larger set of shape models, which covers a much larger set of aspects on the aspect sphere. However, for all object categories, the precision (correct location) and precision (correct location and orientation) of match-within-class are higher than those of match-against-all. This is because that the proposed algorithm has a smaller chance of finding false positive matches when a given test image is matched against a set of shape models from the same object category. Given the definition of precision (TP/(TP+FP)), the proposed algorithm finds more true positive matches relative to the total number of matches when a given test image is matched against within-class shape models. By comparing Fig. 8.58(b) and 8.58(c), we can see that the precision is much higher without considering hand orientation errors for both match-against-all and match-within-class. This is consistent with our previous observation that the proposed algorithm achieves higher performance on predicting grasp regions than hand orientations. For most object categories, the superiority of match-within-class is more obvious when hand orientation error is considered. This is reflected by the larger difference between the two bars in each pair in Fig. 8.58(c), compared with those in Fig. 8.58(b).

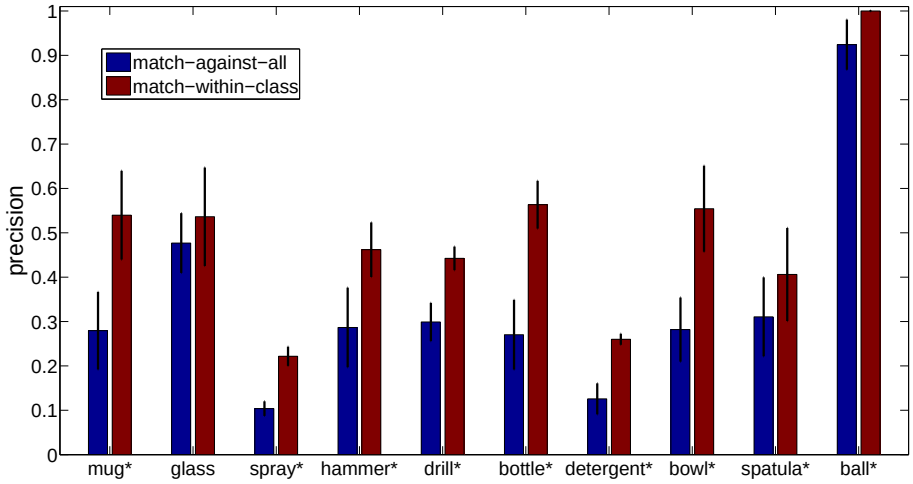
Most of the above comparisons are statistically significant. The statistically significant comparisons are indicated by asterisks next to the object category names on the x axis. In Fig. 8.58(a), the comparisons for object categories glass and ball are not statistically significant (single-tail paired t-test, $p < 0.09$ and 0.11, respectively). Due to ceiling effects on these two object categories, the match-against-all and match-within-class methods perform very similar to each other. The objects in these two categories are rotationally symmetric. The shape models learned on these objects cover most of the aspects very well and achieve high rate of coverage. For the mugs and spatulas, match-against-all and match-within-class have very different rates of coverage. The spatula shape models that survive after the two steps of the filtering process only cover the aspects where the handle is clearly visible (Fig. 8.31a).



(a) Rate of coverage

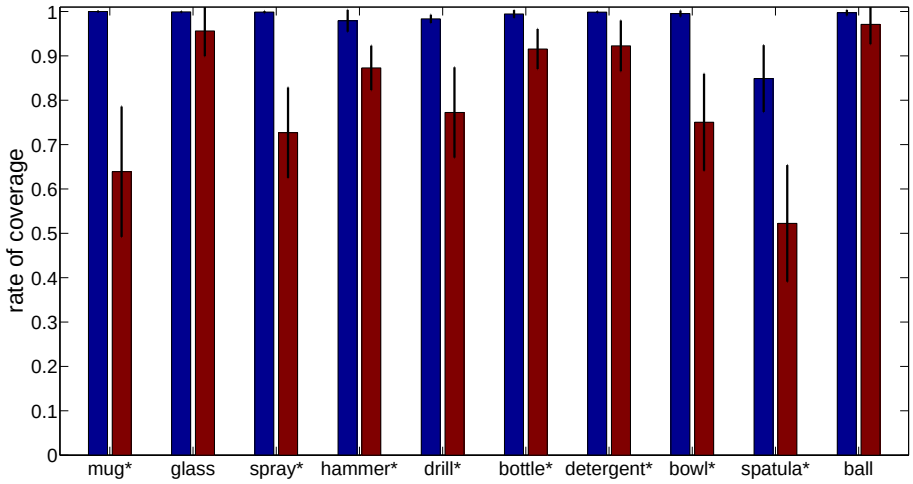


(b) Precision (correct location)

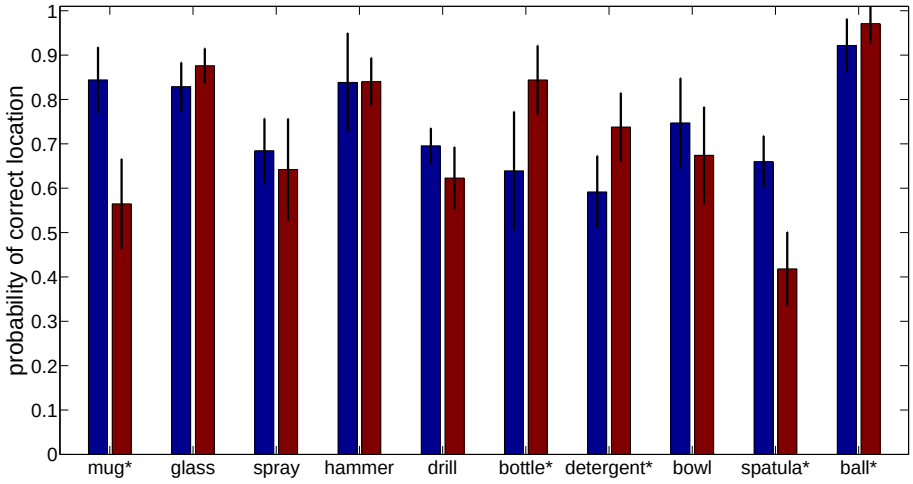


(c) Precision

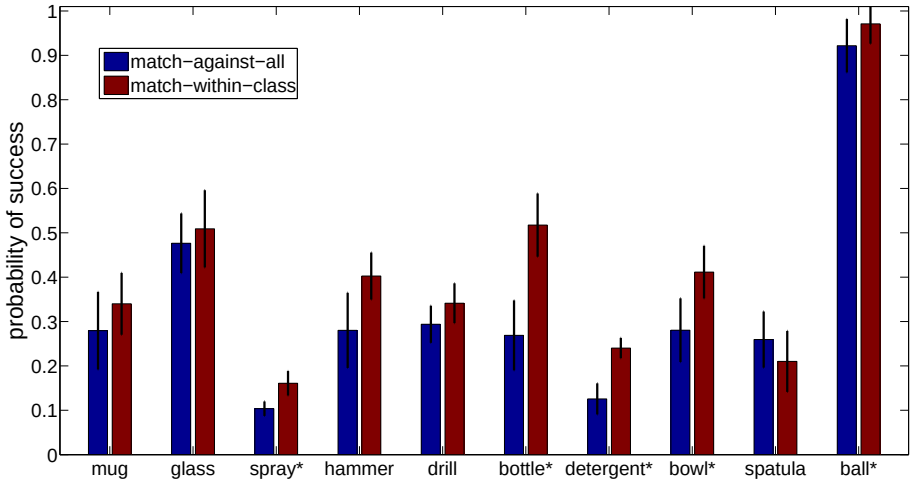
Figure 8.58: Performance comparison: match-against-all vs. match-within-class



(a) Rate of coverage



(b) Probability of correct location



(c) Probability of success

Figure 8.59: Performance comparison: match-against-all vs. match-within-class

Because of this, the rate of coverage for the spatula is low when only matching to within-class shape models. For the mugs, it is beneficial to match test images against shape models that are learned from other object categories, such as those learned from the top/bottom of a set of glasses, the bottom of wine bottles, and the balls. These shape models are learned from object components that resemble the top of mugs, which generalize very well across object categories. As a result, match-against-all achieves significantly higher rate of coverage on the mugs compared to match-within-class. However, due to the fact that these shape models usually match to a wider range of aspects than that is allowed by our HOE criterion, a large number of these generalizations are labeled as wrong. As a result, match-against-all achieves significantly lower precision on the mugs compared to match-within-class. In Fig. 8.58(b), the comparisons for object categories mug, hammer, and spatula are not statistically significant (single-tail paired t-test, $p < 0.09$, 0.07, and 0.28, respectively). For these object categories, the proposed algorithm performs well for both match-against-all and match-within-class as far as finding the correct grasp regions. In Fig. 8.58(c), the comparison for object category glass is not statistically significant (single-tail paired t-test, $p < 0.06$). The glass object category is fully rotationally symmetric. The objects in this category are relatively simple and free of complex texture, and it is not very likely for shape models from other categories to find a good match on these objects. As a result, our algorithm achieves similar precision for match-against-all and match-within-class on the glass object category.

In Fig. 8.59, we compare the rate of coverage, probability of correct location, and probability of success of match-against-all and match-within-class. This triplet of bar plots is arranged consistently with the triplets of aspect spheres we have seen in the previous section. Fig. 8.59(a) is the same as Fig. 8.58(a), we include it here for

comparison purpose. Between Fig. 8.58 and 8.59, we have the following relationship:

$$\begin{aligned} \text{probability of correct location} &= \text{precision (correct location)} \times \text{rate of coverage,} \\ \text{probability of success} &= \text{precision} \times \text{rate of coverage.} \end{aligned}$$

Therefore, in order to achieve high probability of success, the algorithm has to achieve both high rate of coverage and high precision. In Fig. 8.59(b), match-against-all achieves higher performance than match-within-class for the mug, the spray bottle, the hand drill, the bowl, and the spatula. Particularly, for the mug and the spatula, match-against-all performs significantly better than match-within-class (paired t-test, $p < 0.01$ in both cases). This is because the proposed algorithm achieves high rates of coverage on these two object categories. For the glass, the wine bottle, the detergent bottle, and the ball, match-within-class performs better than match-against-all as far as probability of correct location. Particularly, for the wine bottle, the detergent bottle, and the ball, these comparisons are significant (paired t-test, $p < 0.02$, 0.01, and 0.05, respectively). For the hammer, match-against-all and match-within-class achieve similar performance. The above comparisons indicate that match-against-all performs comparably well as match-within-class as far as predicting grasp regions. However, in Fig. 8.59(c), match-within-class outperforms match-against-all for most object categories (all except the spatula) when hand orientation error is considered. Particularly, for the spray bottle, the wine bottle, the detergent bottle, the bowl, and the ball, match-within-class performs significantly better than match-against-all (paired t-test, $p < 0.01$ in all cases). For the spatula, match-within-class achieves slightly lower probability of success than match-against-all, due to the fact that the latter achieves much higher rate of coverage than the former. However, this comparison is insignificant (paired t-test, $p < 0.09$).

8.3.5 The Quality of Synthesized Bounding Boxes

For a given image, there typically exists multiple suggested grasps. However, the human teacher will have demonstrated but one of these grasps. When the proposed algorithm matches this demonstrated grasp, it can serve as a ground truth against which to measure the algorithm's choice. When this demonstrated grasp is not the one suggested by the algorithm, we need an alternative method of measuring the quality of this choice. Our approach is to synthesize an answer given other samples of the same object and from similar viewing angles (Section 4.2). A key question, however, is how well do these synthesized answers compare to the true ground truth. In order to answer this question, for a single object category, we compare the performance of a set of shape models for a certain grasp type on two different sets of test images: the one in which such a grasp type has been demonstrated (so the ground truth bounding boxes for this grasp type come directly from the demonstration) and the one in which such a grasp type has not been demonstrated (so the ground truth bounding boxes are synthesized by using the method in Section 4.2).

In the following results, we select the mug object category. For mugs, two grasp types are demonstrated during training: a ball grasp from the top and a handle grasp. For each in the five mugs, we compare the performance of the following: 1) top shape models matched to images in which top grasps are demonstrated (top-GT), 2) top shape models matched to images in which top grasps are synthesized (top-syn-GT), 3) handle shape models matched to images in which handle grasps are demonstrated (handle-GT), and 4) handle shape models matched to images in which handle grasps are synthesized (handle-syn-GT). Here, we consider a test image as correctly identified if the matched shape model with highest matching score is labeled as correct (so, the Top 1 approach).

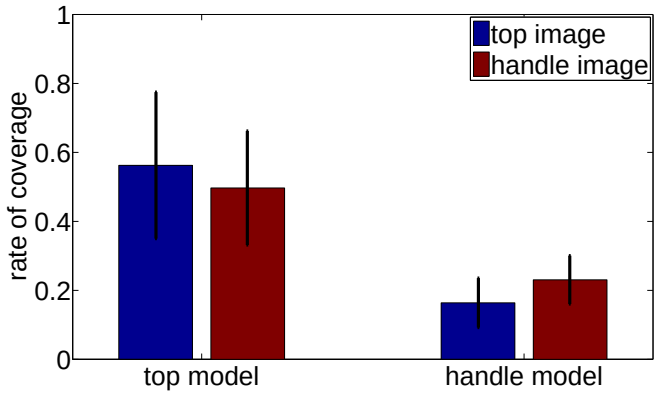
The comparisons of rate of coverage and precision between the ground truth

bounding box and synthesized bounding box are shown in Fig. 8.60(a) and Fig. 8.60(b). Each bar is a mean of five experiments, corresponding to five mugs, and the whiskers are standard deviations. In each figure, the bars are grouped by the grasp types of shape models, either top grasps (the first group) or handle grasps (the second group). In each group of two bars, the first one corresponds to a set of test images in which top grasps are demonstrated (top image), and the second one corresponds to a set of test images in which handle grasps are demonstrated (handle image). Recall that for each test image, only a single grasp type is demonstrated by the human teacher. Therefore, these two sets of test images are mutually exclusive and bisect the entire set of test images. An unpaired t-test fails to show a statistical difference in the rate of coverage between the top model pair, or the handle model pair ($p < 0.17$). Furthermore, an unpaired t-test fails to show a statistical difference in the precision between the top and handle model pairs, respectively ($p < 0.64$). This indicates that the synthesized ground truth bounding boxes are as accurate as the ground truth bounding boxes generated by image differencing.

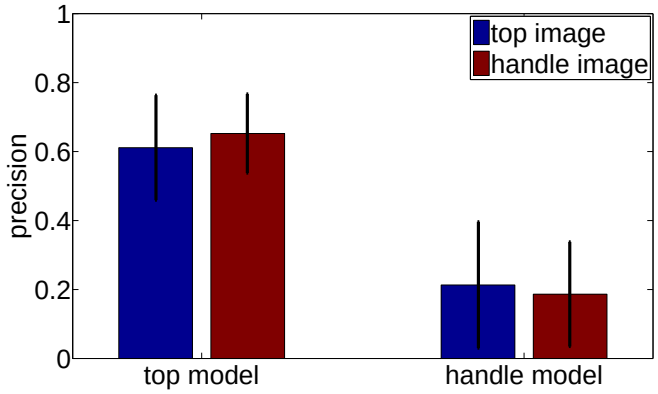
8.4 Sensitivity Analyses

So far, we have shown results for a specific implementation of the proposed algorithm. Within the algorithm, there are a variety of parameters that must be chosen (Table 8.1). We have chosen many of these parameters empirically, based on validation set performance. In this section, we focus on several parameters, whose values have critical impacts on the experimental performance. For the results that we have reported in the previous sections, we determine the values of key parameters by using a separate validation set. For the purpose of sensitivity analyses in this section, we change their values and observe their influence on the performance.

One of the key parameters is the κ threshold, κ_{th}, that we choose to filter out less



(a) Rate of coverage



(b) Precision

Figure 8.60: Performance comparison: ground truth bounding box vs. synthesized bounding box

Table 8.1: The list of parameters and their chosen values.

sub-algorithm	Parameter	Value
aspect clustering mean shift	bandwidth on z (girdle)	0.01
	min cluster size (girdle)	10
	bandwidth on camera position (DW)	0.06
	bandwidth on camera x direction (DW)	0.2
	min cluster size (DW)	3
model learning	PAS difference threshold	4.0‡
	max number of images to learn codebook	10‡
	number of PAS types	30‡
	number of tiles	30†
SVM classifier	number of negative images	100
	SVM type	C-SVC†
	kernel type	polynomial†
	polynomial degree	10†
model filtering I	percentage of models kept	80%
	number of models kept	10(DW)/5(girdle)
model filtering II	κ threshold: κ_{th}	0.1
model matching	number of initial matches	10
	enlarge model bounding box ratio	1.3‡
	sliding window stepsize	5 pixels†
	sliding window scaling factor	$2^{1/4}$†
	max sliding window size	min(model size $*$ 2.5, image size)
	min sliding window size	model size $*$ 0.8
score weighting	W_{err}: point matching error	10
	W_{nomp}: number of matched points	0.1
	W_{tps}: warping energy	1
	W_{aff}: amount of affine change	1
	W_{epi}: epipolar constraint	0.01
evaluation	IoU error bound	0.2‡
	HOE bound	30°

†These values are from Ferrari et al. (2008)
‡These values are from Ferrari et al. (2010)

discriminative shape models during the second step of filtering process. A κ_{th} value of 0.1 usually gives us about 300 to 350 shape models for all object categories in a single experiment.

We also would like to know whether we have made a reasonable choice for the model thresholds. Given that the entire parameter space is huge (one model threshold for each shape model), we will reduce our search to a single dimension and use a uniform model threshold t for all shape models. Specifically, we compare the performance of individual model thresholds t_i^*'s and that of the uniform model threshold, t. The hypothesis is that using separate model threshold, t_i^*, for each individual shape model should outperform using uniform model threshold, t. This is because the former approach can increase the model thresholds for shape models that are less discriminative based on their validation set performance. This helps to eliminate some of the false positive matches at an early stage.

The rate of coverage (RoC) vs. precision curves when varying the uniform model threshold t are shown in Fig. 8.61 to Fig. 8.80. The curves are means and the shadings are standard deviations of five experiments. *Top 1*, *top 2* and *top 3* correspond to three specific choices of our algorithm. The performance of model-specific threshold t_i^* is superimposed in these figures as single points.

Fig. 8.61 shows the rate of coverage vs. precision curves for the mugs. In this figure, the three marked points, which correspond to *top 1*, *top 2*, and *top 3* with individual model thresholds, are all above the three curves with uniform model thresholds. This means that given the same rate of coverage, the algorithms with individual model thresholds achieve higher precision. Note that once the set of model thresholds t_i^* is determined, the rate of coverage is fixed. This explains why the three markers in this figure have the same rates of coverage. The three curves corresponding to algorithms with uniform model thresholds generally increase with higher t (lower RoC). With higher t, the RoC monotonically decreases, since fewer shape models have matching

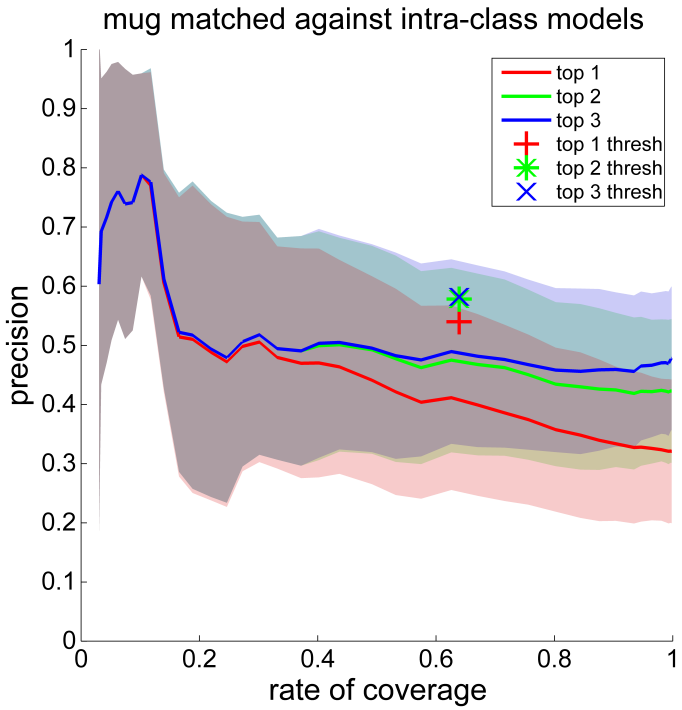


Figure 8.61: Rate of coverage vs. precision curves for the mugs by varying the uniform model threshold, t (match-within-class). Red: *Top 1*. Green: *Top 2*. Blue: *Top 3*. The curves are means and the shadings are standard deviations of five experiments. Each test image is matched against shape models within the same object category. The three marked points correspond to using individual model thresholds, t_i^*, which are automatically chosen based on the validation set performance.

scores above t. The three curves terminate when any of the experiments achieve a RoC of zero, since the precision is meaningless in this case. Because only a small number of test images have matches when t is high, the precision may jump erratically. Therefore, we do not focus on the part of the curve with very low RoC (such as ≤ 0.1).

In general, these three curves are correlated with each other, since *top 1* $\subseteq$ *top 2* $\subseteq$ *top 3*. When increasing t to some value, *top 3* will join *top 2* if there are ≤ 2 shape models with matching scores above t. Similarly, when increasing t further to some value, *top 2* will join *top 1* if there are ≤ 1 shape models with matching scores above t. For the *top 1* version of our algorithm, with higher t, the RoC will be equal or lower, since the number of images with matches above t (TP+FP) is equal or fewer than that of lower t. Similarly, with higher t, the number of test images that are labeled as correct (TP) can either decrease or remain the same, depending upon whether a correct or a wrong match is removed. For example, with higher t, if a wrong match in a test image is removed, the rate of coverage will decrease, while the precision, TP/(TP+FP), will increase. This is because the number of correct images (TP) remains the same, while the number of matches (TP+FP) decreases. However, if a correct match in a test image is removed, both the rate of coverage and precision will decrease. This is because the new precision, (TP-1)/(TP+FP-1), is smaller than the old precision, TP/(TP+FP). In the general case, if k matches are removed, which include m correct matches and $k - m$ wrong matches, the rate of coverage decreases, while the change of precision is uncertain (depending upon the ratio, m/k). In the cases where more than one matched shape models are involved (*top 2* and *top 3*), the relative change between TP and (TP+FP) is more complicated. Since these curves are means of five experiments, they reflect the overall trends of the change of precision with respect to the rate of coverage. The behaviors of the three curves are summarized as follows.

Top 1: for all test images, if the false positives have relatively higher matching

scores than the true positives, the precision will decrease as RoC decreases.

Top 2/3: for a single test image, if the false positives have relatively higher matching scores than the true positives, the precision will decrease as the RoC decreases. This is because the lower scored shape models are always first to be eliminated from the top N matched shape models when increasing t.

We should note that, when t is sufficiently large, the RoC will decrease to zero. In this case, the precision is undefined, and we will terminate the curve whenever this happens. Also, the maximum value of RoC may be smaller than 1. This is because the PAS shape model matching algorithm sometimes fails to find any shape model in a test image.

The trends of these curves can be summarized as follows. For all objects, except the balls, the precision of the *top 1* curve generally increases with higher t and lower rate of coverage. This means the true positives usually have higher matching scores than the false positives across all test images.

For mugs, hammers, drills and spatulas, the curves corresponding to *top 2/3* generally increase with higher t. With higher t, the rate of coverage will monotonically decrease, since fewer shape models will have matching scores above t. For these objects, in the top 2/3 matches, the true positive matches usually have higher matching scores than the false positive matches. As t increases, more false positive matches will be eliminated, thus the precision increases.

For glasses, with higher t, the *top 2/3* curves first decrease and then increase. This suggests that for some images with lower matching scores, the false positives have relatively higher scores than the true positives out of the top 2/3 best scored matches. However, for images with higher matching scores, the true positives have relatively higher scores than the false positives out of the top 2/3 best scored matches.

For spray bottles, wine bottles, detergent bottles and bowls, the *top 2/3* curves keep decreasing until a very low rate of coverage. This suggests that in most of the

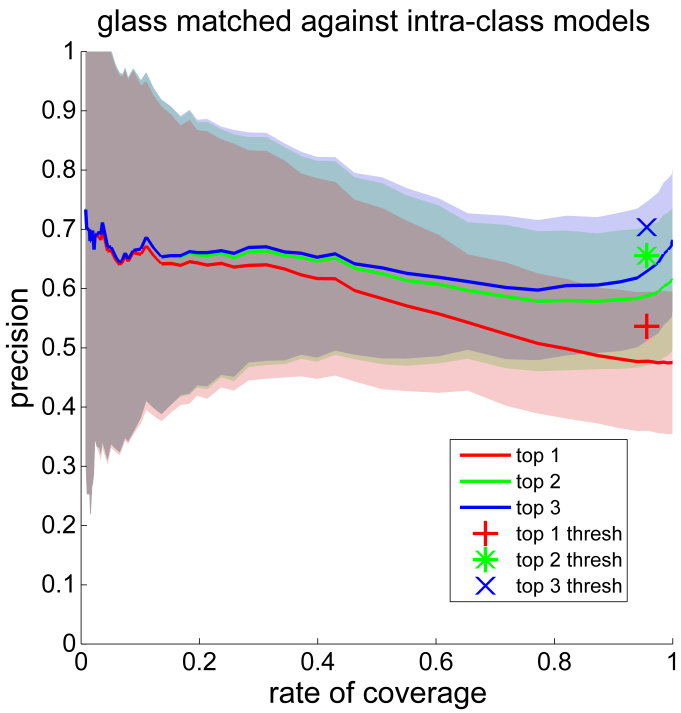


Figure 8.62: RoC vs. precision for the glasses by varying t (match-within-class)

test images, the false positives have relatively higher scores than the true positives out of the top 2/3 best scored matches.

For the balls, we have observed that the precision is always 1, and the rate of coverage is very close to 1. In order to show some variation in the rate of coverage vs. precision curve, we match the balls to all shape models, rather than only the shape models learned from balls. Because of this, the performance of our algorithm on the balls is not directly comparable to that of the other objects. All three curves decrease as the rate of coverage decreases. This suggests that in most of the test images, the false positives have relatively higher scores than the TPs. This may be partially due to the fact that we match the balls against all shape models.

For all object categories, the performance of the automatically chosen optimal model threshold t_i^* is substantially better than that of the uniform threshold t. In some cases, *top 1* with individual model thresholds is even better than *top 3* with

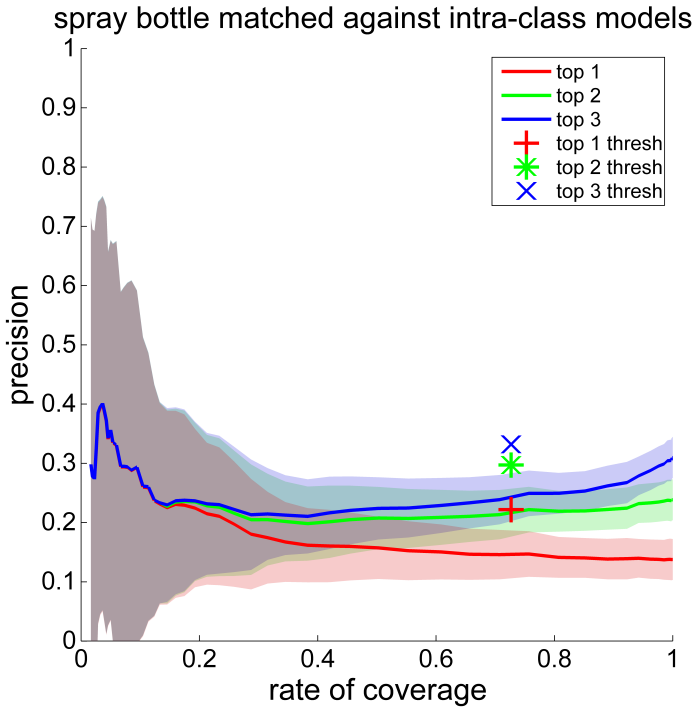


Figure 8.63: RoC vs. precision for the spray bottles by varying t (match-within-class)

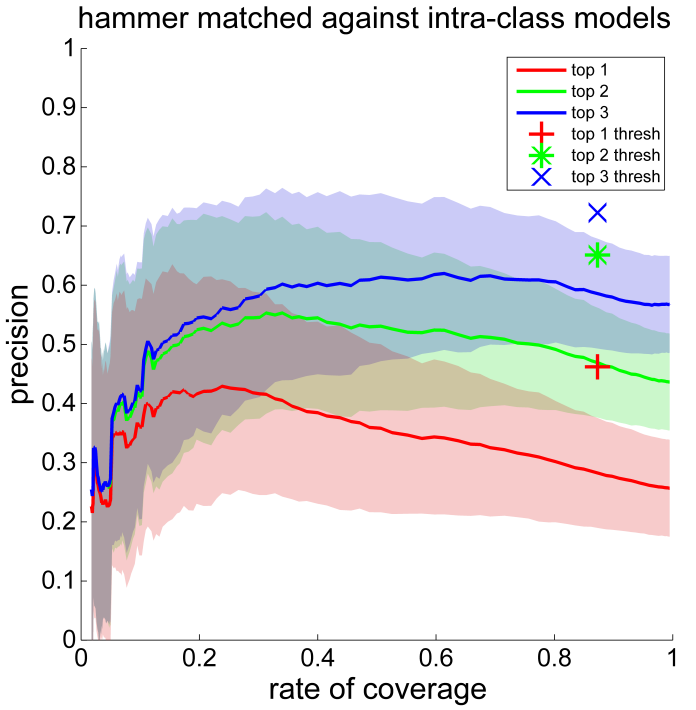


Figure 8.64: RoC vs. precision for the hammers by varying t (match-within-class)

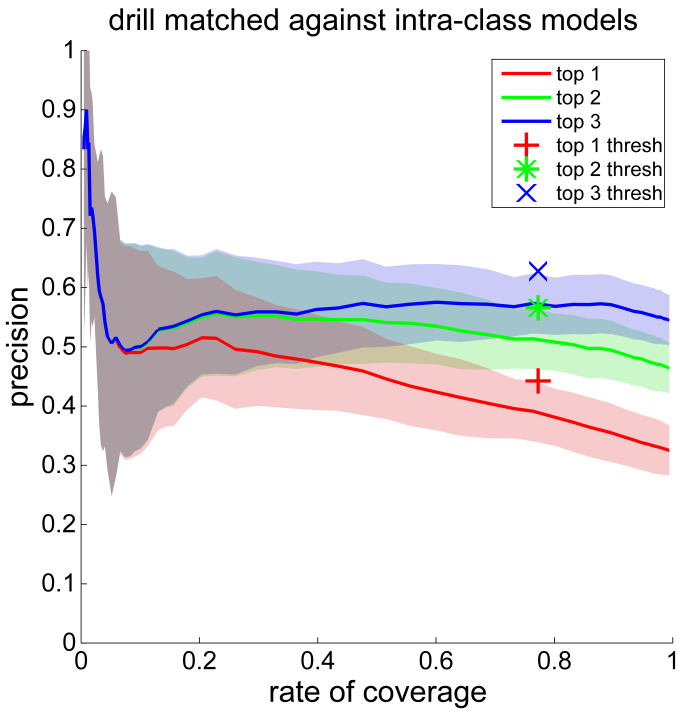


Figure 8.65: RoC vs. precision for the hand drills by varying t (match-within-class)

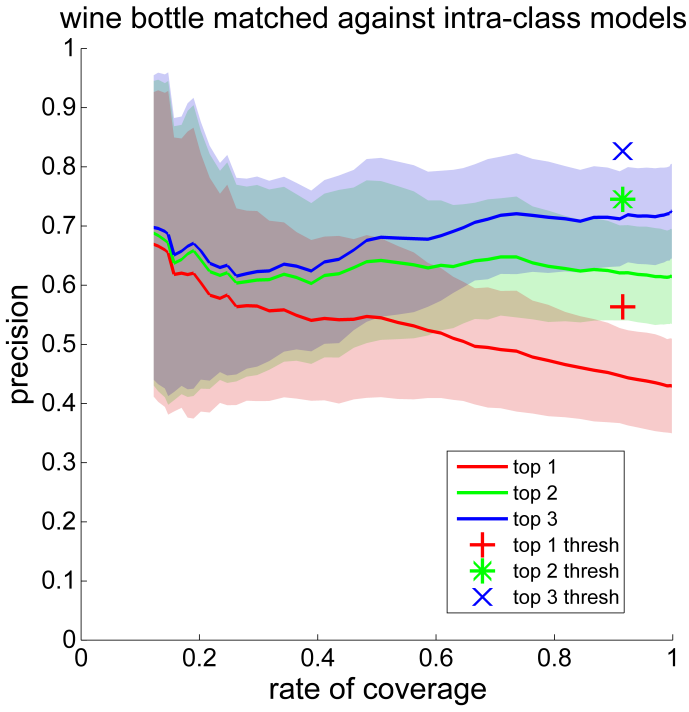


Figure 8.66: RoC vs. precision for the wine bottles by varying t (match-within-class)

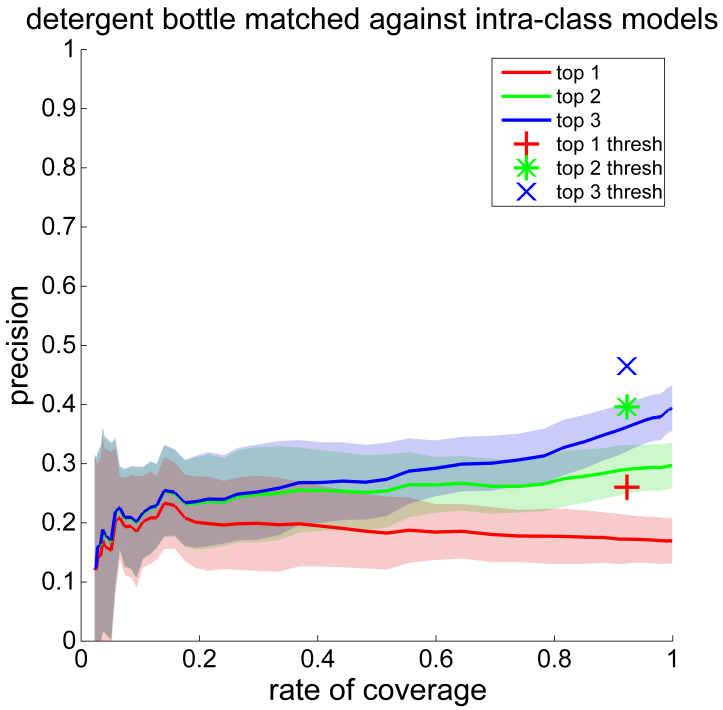


Figure 8.67: RoC vs. precision for the detergent bottles by varying t (match-within-class)

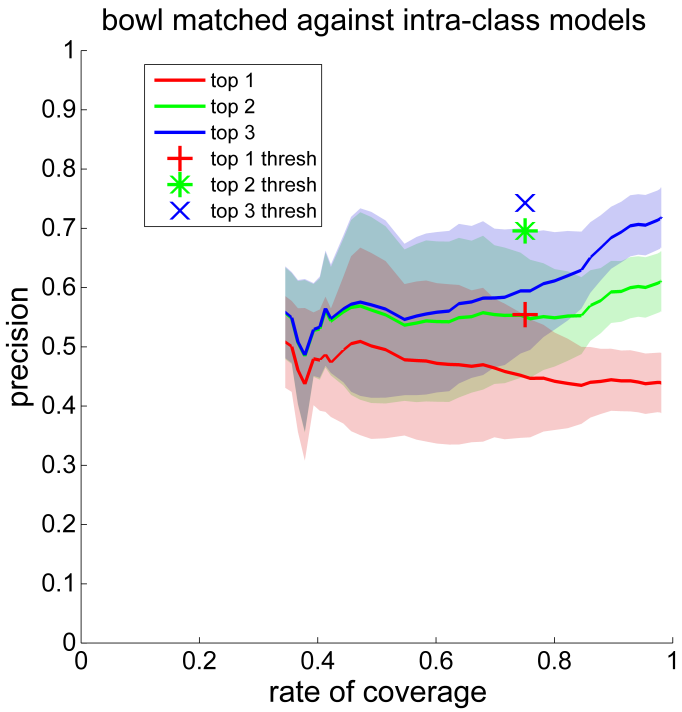


Figure 8.68: RoC vs. precision for the bowls by varying t (match-within-class)

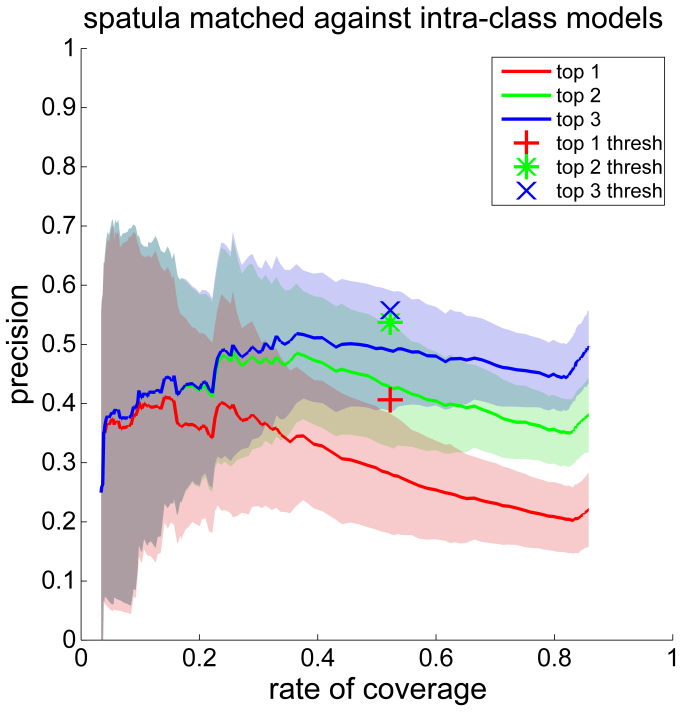


Figure 8.69: RoC vs. precision for the spatulas by varying t (match-within-class)

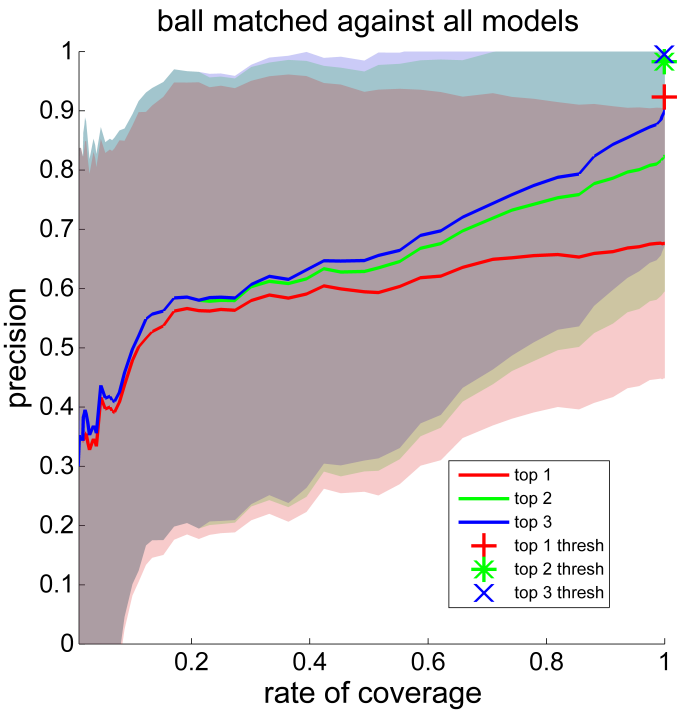


Figure 8.70: RoC vs. precision for the balls by varying t (match-against-all)

uniform model threshold (indicated by a red cross above the blue curve). This shows the advantage of choosing model specific thresholds. For shape models that are not very discriminative on the validation set, we would like to increase their model thresholds, in order to decrease chance of false positive matches on the test set. Also, some shape models tend to have higher matching scores than the other shape models, so we would like to increase their model thresholds accordingly.

8.4.1 Varying the κ Threshold

In all the above analyses, we have chosen the κ threshold (κ_{th}) to be 0.1 in the shape model filtering step II. In the following, we are interested in the performance change when varying κ_{th}. For these experiments, we use the model thresholds, t_i^*, which are automatically determined by using the validation set.

Fig. 8.71 shows the RoC vs. precision curves for the mugs when varying κ_{th}. We

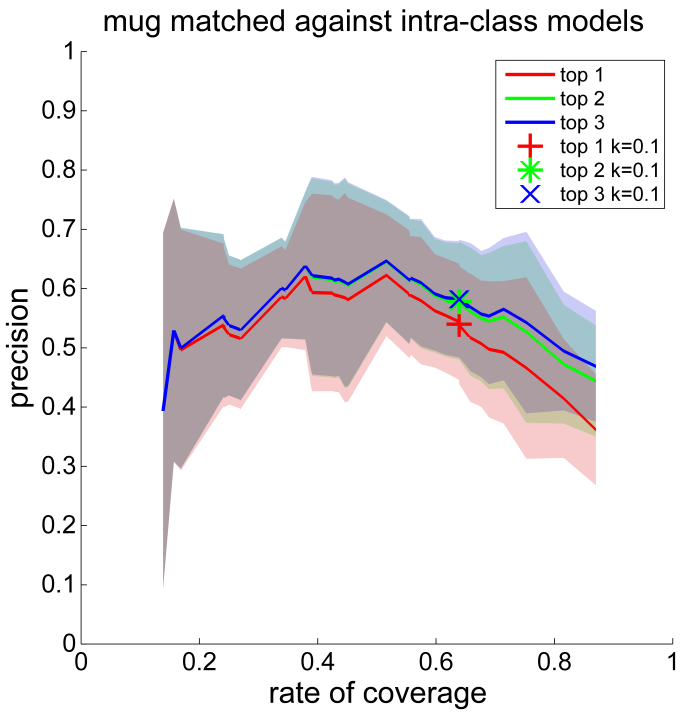


Figure 8.71: Rate of coverage vs. precision curves for the mugs by varying the κ threshold, κ_{th} (match-within-class). Red: *Top 1*. Green: *Top 2*. Blue: *Top 3*. The curves are means and the shadings are standard deviations of five experiments. Each test image is matched against shape models within the same object category. The three marked points correspond to the performance of using $\kappa_{th} = 0.1$.

vary κ_{th} in the range of $[0, 1]$. In this figure, the three curves go from right to left as κ_{th} increases. The three marked points, which correspond to *top 1*, *top 2*, and *top 3* with $\kappa_{th} = 0.1$, are on the three curves with various κ_{th}'s, since they correspond to a particular choice of the κ_{th}. By choosing different κ_{th} values, the three marked points slide along the corresponding curves. Ideally, we would like to choose a κ_{th} value that yields a high RoC value, as well as a reasonably high precision (not much lower than the peak). Note that the three marked points always have the same RoC. This is because they choose the same $\kappa_{th} = 0.1$ and individual model thresholds t_i^*. In other words, they match a test image against the same set of shape models during the testing phase, as well as measuring the RoC.

The effect of changing κ_{th} is similar to that of changing the uniform model thresh-

old, t. The κ statistic is used to measure the performance of a learned shape model. In this sense, shape models with higher κ values should be more discriminative and have relatively higher precision. When increasing κ_{th}, more shape models will be filtered and, thus, the RoC will monotonically decrease. If the shape model filtering algorithm works well, the shape models that are less discriminative will be filtered first and, thus, the precision should generally increase. When κ_{th} is sufficiently high, more of the shape models that contribute to the true positive matches will be filtered out than those that contribute to the false positive matches, and the precision will drop.

Again, these three curves are correlated with each other, since *top 1* $\subseteq$ *top 2* $\subseteq$ *top 3*. When κ_{th} is sufficiently high, *top 3* will join *top 2* if there are ≤ 2 shape models with $\kappa \geq \kappa_{th}$. Similarly, when increasing κ_{th} further to some value, *top 2* will join *top 1* if there are ≤ 1 shape models with $\kappa \geq \kappa_{th}$.

RoC vs. precision curves for the other object categories are shown in Fig. 8.72 to Fig. 8.80. For all object categories other than the wine bottles, these curves generally increase as κ_{th} increases, which indicates the effectiveness of our model filtering process. For the wine bottles, these curves generally decrease as κ_{th} increases. This suggests that most of the shape models (even those with low κ values) are discriminative enough and actually contribute to the TP matches. When κ_{th} increases, some of the "good" shape models will be filtered out and the overall precision drops. From these curves, we can see that our choice of $\kappa = 0.1$ is good for all object categories. This is reflected by the fact that the marked points on these curves balance well between RoC and precision.

8.4.2 Revisiting Model Filtering

In Section 8.3.1, we have discussed that the rate of coverage of the handle grasp on the mugs is substantially lower than that of the top grasp (by comparing Fig. 8.18 and

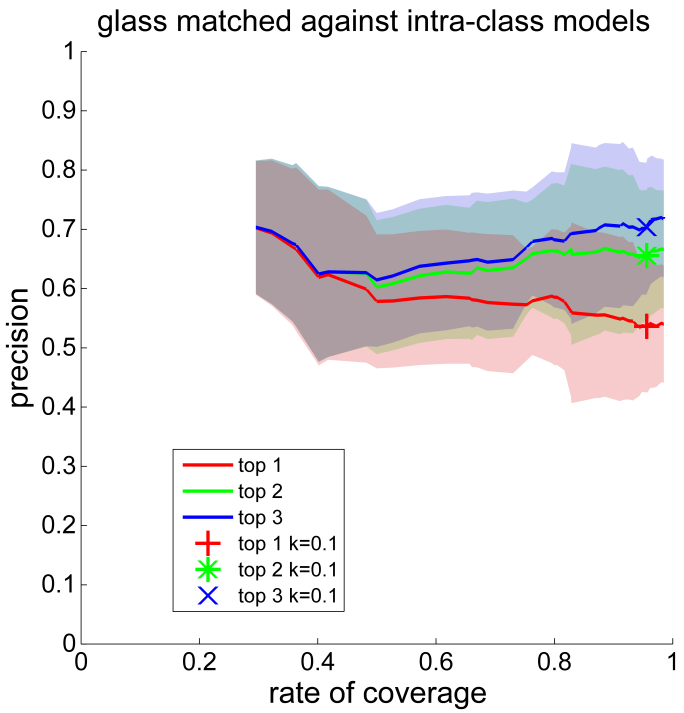


Figure 8.72: RoC vs. precision for the glasses by varying κ_{th} (match-within-class)

Fig. 8.19). This in part is because insufficient number of handle shape models survive the model filtering process. One possible way to relax the model filtering process so that more shape models can survive is to use a different performance measure. In the second step of the filtering process, we measure the performance of a shape model based on its κ statistic. An alternative approach is to measure the performance of a shape model based on the Kolmogorov-Smirnoff distance (KSD, Utgoff and Clouse, 1996). Similar to the model filtering process with κ statistic, we select t_i^* that gives the KSD between the true positive rate (TPR = TP/(TP+FN); TP = number of true positives; FN = false negatives) and false positive rate (FPR = FP/(FP+TN); FP = false positives; TN = true negatives) of a validation set. In addition, we filter out shape models that have KSD lower than some threshold, δ_{th}.

The κ and KSD curves of a handle shape model is compared in Fig. 8.81. The same validation results are used to generate these two curves. For the same shape model,

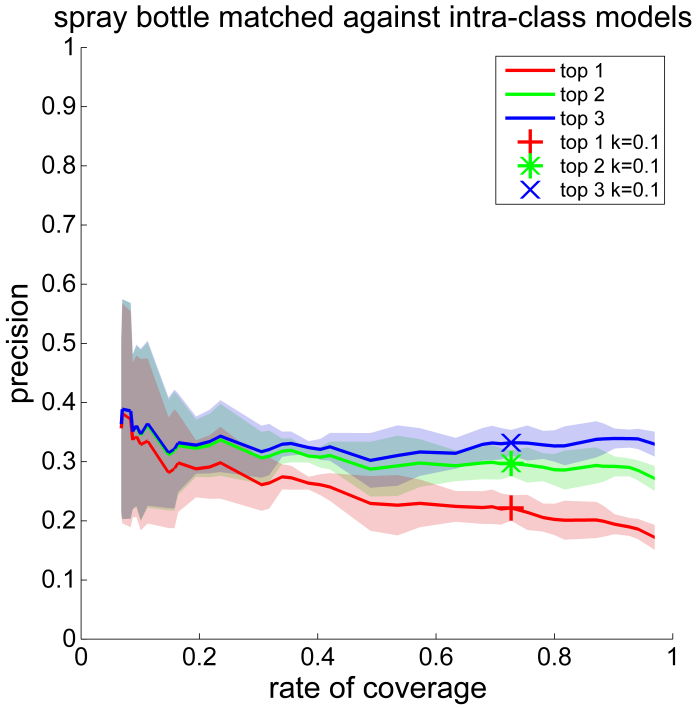


Figure 8.73: RoC vs. precision for the spray bottles by varying κ_{th} (match-within-class)

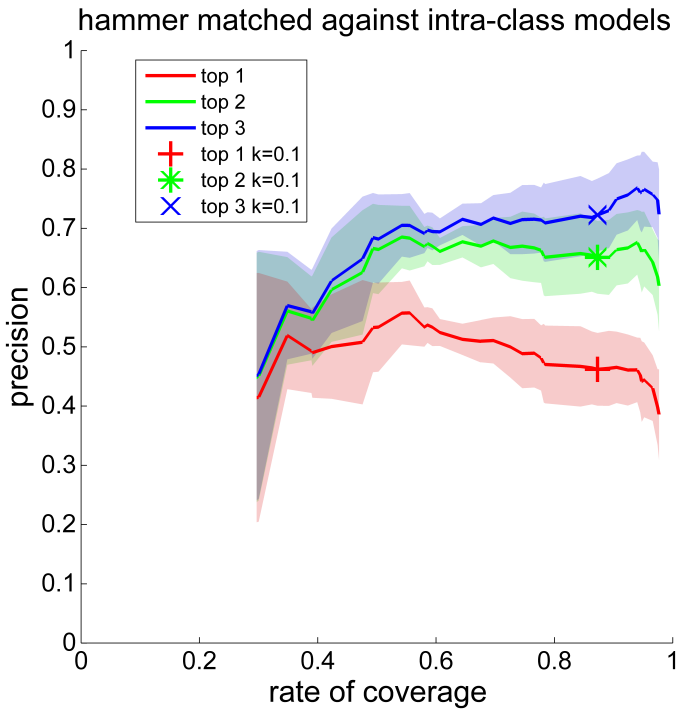


Figure 8.74: RoC vs. precision for the hammers by varying κ_{th} (match-within-class)

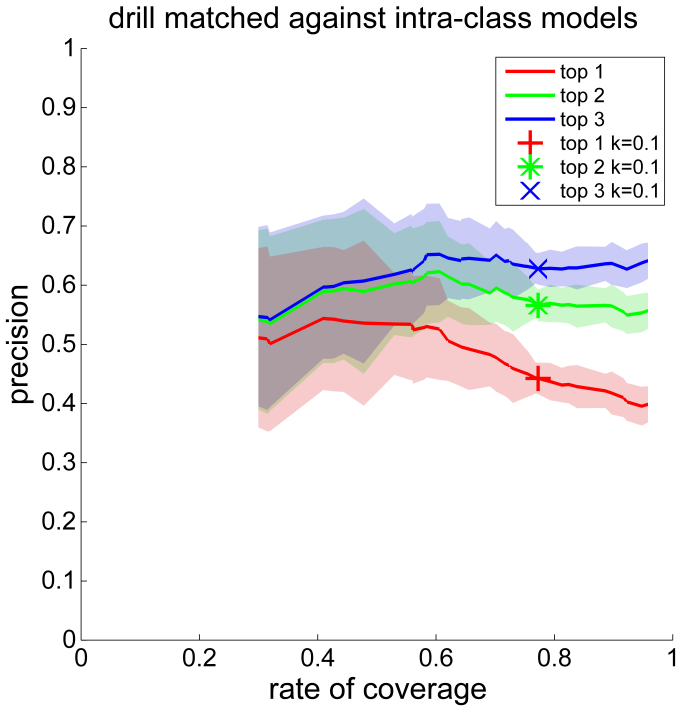


Figure 8.75: RoC vs. precision for the hand drills by varying κ_{th} (match-within-class)

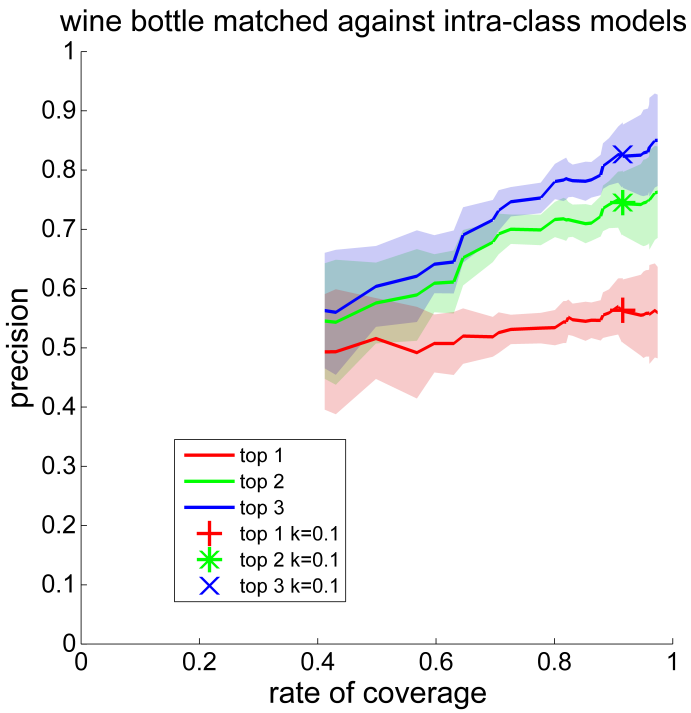


Figure 8.76: RoC vs. precision for the wine bottles by varying κ_{th} (match-within-class)

the κ statistic is stricter than KSD, since the former takes chance agreement into consideration (Equation 7.1). Therefore, more shape model will survive the filtering process when KSD is used. Also, the model threshold chosen by KSD tends to be lower than that of κ. In order to achieve the same rate of coverage (so, the same number of shape models survive the filtering process), the κ threshold, κ_{th}, has to be lower than the KSD threshold, δ_{th}.

In Fig. 8.82, we compare the performance of κ and KSD for mugs by varying κ_{th} and δ_{th}, respectively. As in the last section, we show the rate of coverage vs. precision curves. The marked points on these two curves correspond to $\kappa_{th} = 0.1$ and $\delta_{th} = 0.1$, respectively. Here, we only show the results of the *Top 1* version of our algorithm. The starting points on the right of these two curves correspond to $\kappa_{th} = 0$ and $\delta_{th} = 0$. When $\kappa_{th} = \delta_{th}$, the KSD method keeps more shape models than the κ statistic. In addition, the model thresholds selected by KSD tend to be lower than those selected

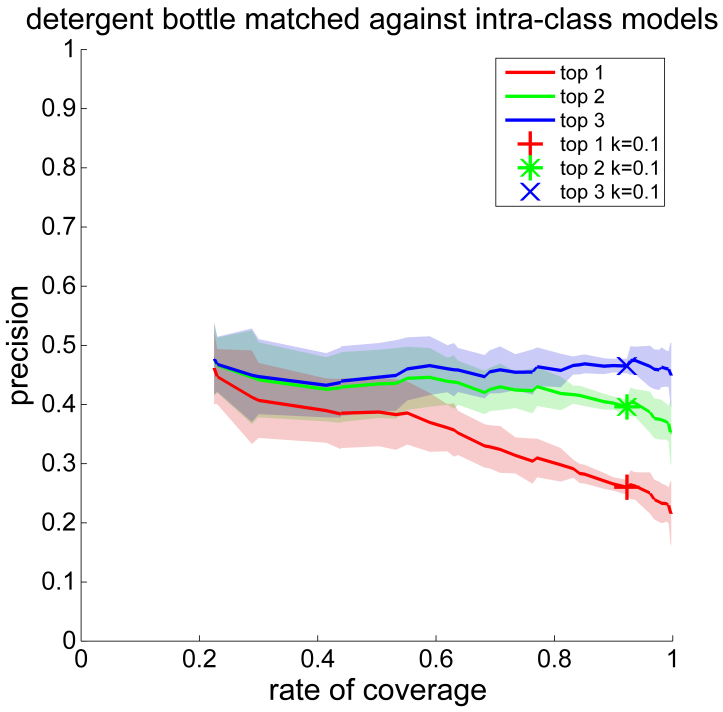


Figure 8.77: RoC vs. precision for the detergent bottles by varying κ_{th} (match-within-class)

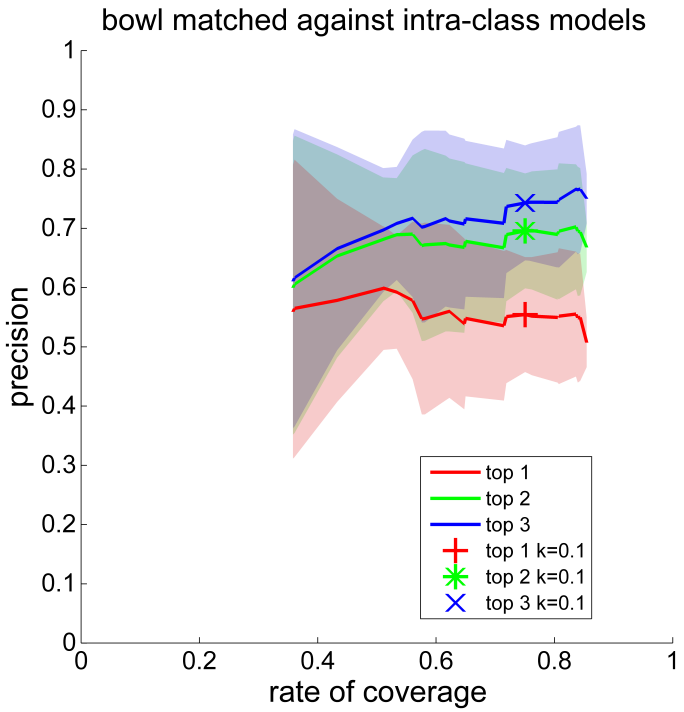


Figure 8.78: RoC vs. precision for the bowls by varying κ_{th} (match-within-class)

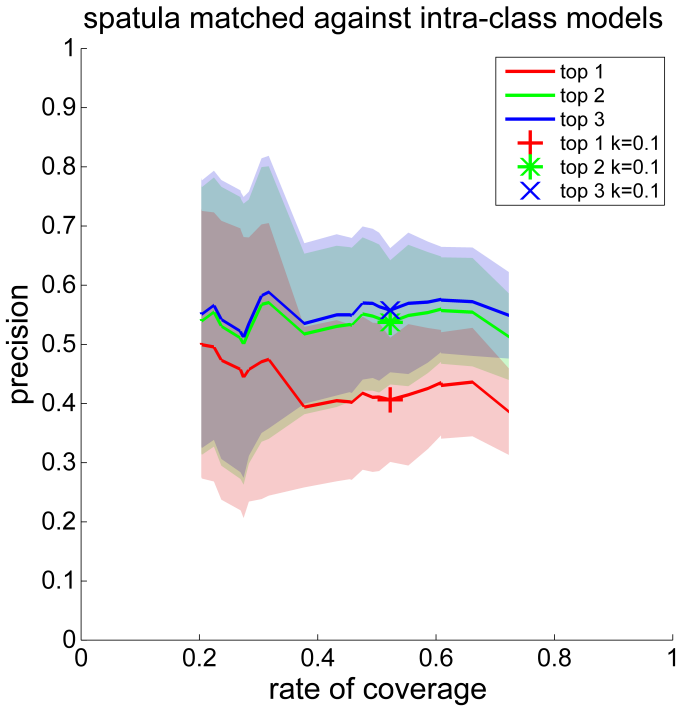


Figure 8.79: RoC vs. precision for the spatulas by varying κ_{th} (match-within-class)

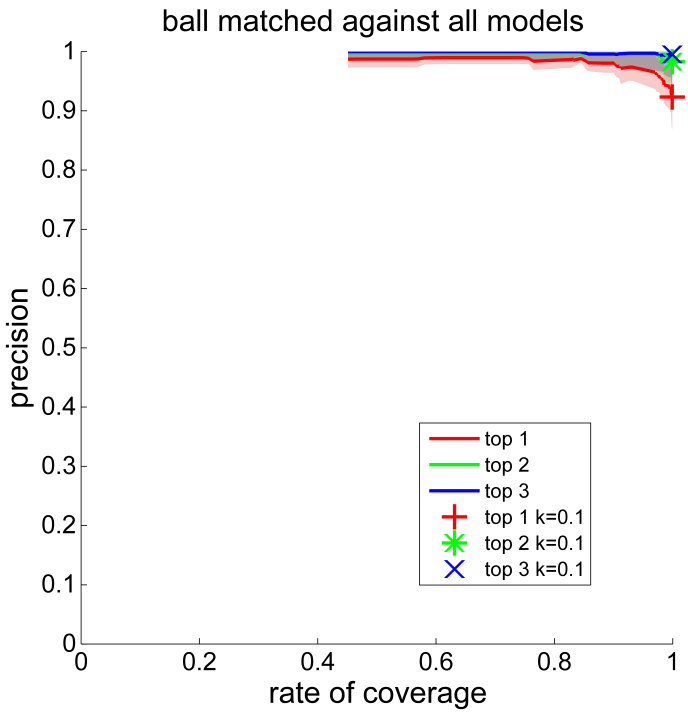
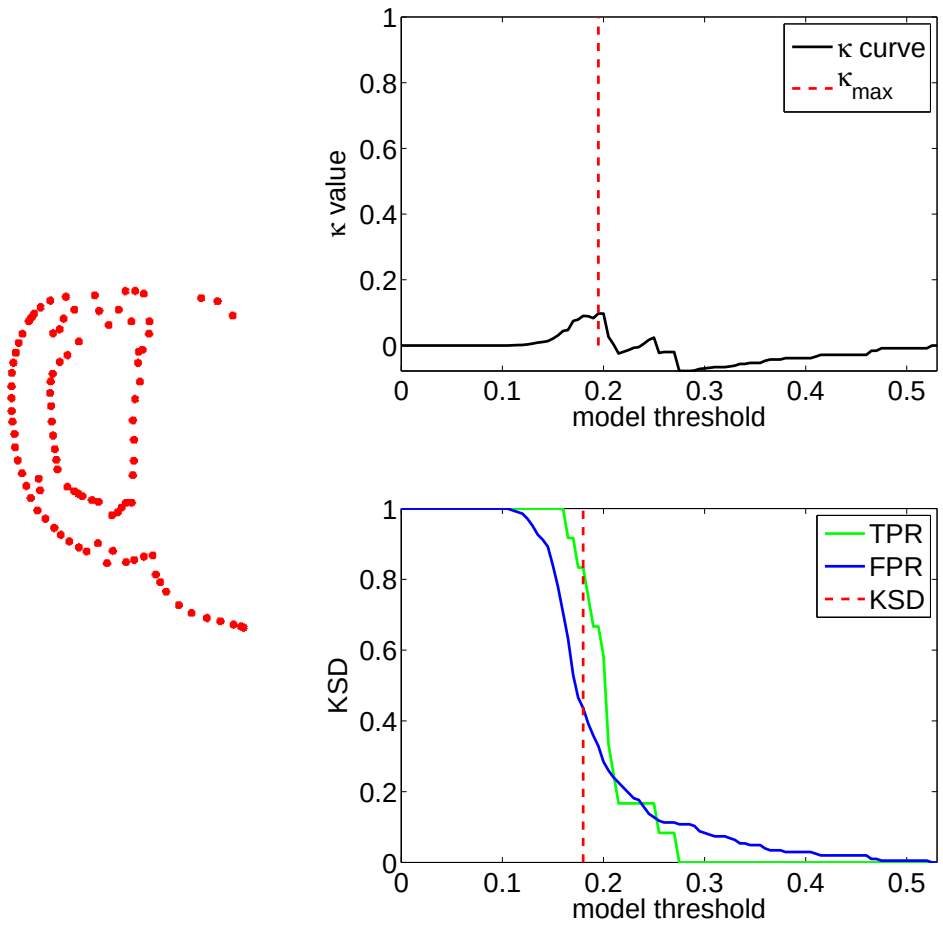


Figure 8.80: RoC vs. precision for the balls by varying κ_{th} (match-against-all)

by κ, which results in more matches for a given set of test images. Because of the above reasons, the performance curve of the KSD method starts with higher rate of coverage than that of κ. Although the KSD method achieves higher rate of coverage than κ when $\kappa_{th} = \delta_{th}$, the corresponding precision is substantially lower than that of κ. This is reflected by the fact that the κ curve is much higher above the KSD curve for the whole range of rates of coverage. Except for a small region around the rate of coverage of 0.9, the κ statistic substantially outperforms the KSD as far as precision. This indicates that the shape models selected by the κ statistic have higher quality than those selected by KSD, and thus, achieve higher precision when matching to test images. This is consistent with our expectation, since the κ is a stricter statistic than KSD by considering chance agreement. The above observations are generally true for all object categories.



(a) A handle shape model (b) The corresponding κ and KSD curves

Figure 8.81: Example κ and KSD curves for a handle shape model

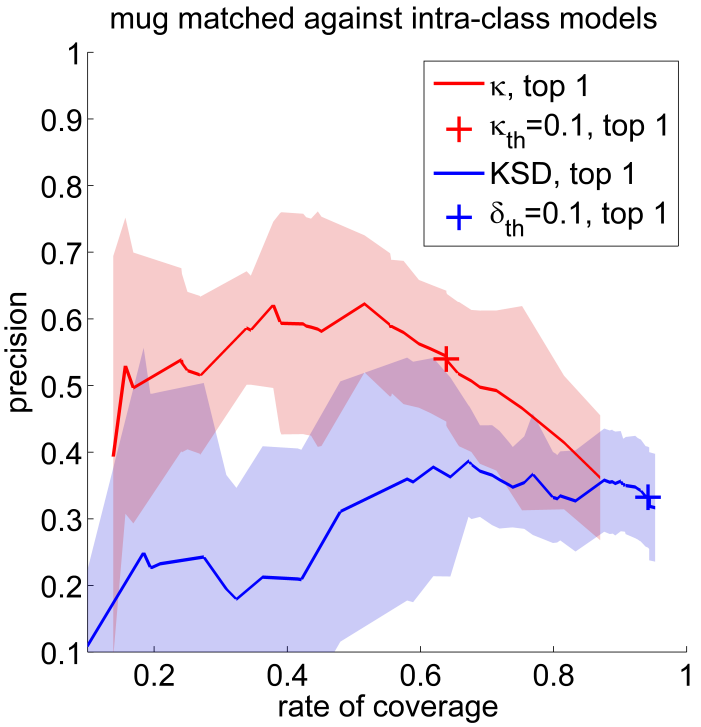


Figure 8.82: Rate of coverage vs. precision for different κ_{th} and δ_{th}. Red: κ. Blue: KSD. The curves are means and the shadings are standard deviations of five experiments for the mugs. Each test image is matched against shape models within the same object category. The marked points correspond to $\kappa_{th} = \delta_{th} = 0.1$.

8.5 Discussion

The experimental results discussed in this chapter have confirmed our experimental hypotheses. First, the different grasp types in an object category successfully drive the learning of meaningful visual models. As shown by the extensive set of example matches, most of these visual models are visually identifiable to humans. Second, in many cases, the learned visual models successfully estimate the grasp types and hand poses in novel images that contain novel objects in a known object category. The experimental results have also shown that, these shape models, which capture partial object shapes, generalize well to novel objects with similar components to those in the training set. Sometimes, these shape models can even accommodate large shape variations, such as the generalization between drill and hammer handles.

The OUGD is a challenging data set from both computer vision and robotics grasping perspective. In our experiments, one challenge is to recognize object components rather than the object itself. A substantial number of object components do not contain enough information for robust visual matching, and some of them look ambiguous with respect to different viewing angles. These object components are difficult to differentiate even for humans without larger contexts. Also, our algorithm is required to differentiate a full range of viewing angles, and, at the same time, to be able to generalize to novel instances in a previously learned object category. These two points actually conflict with each other. Differentiating different aspects requires the learned shape models to be sensitive to shape variations due to aspect change, while generalizing to novel objects requires the learned shape models to be insensitive to small shape variations within an object category. However, the shape variation of a shape model may due to both aspect and object identity changes, which usually can not be decoupled. The PAS based shape model learned on the handle of a mug allows shape deformations along certain dimensions that are exhibited on the train-

ing set of images (for example, mug handles with different heights). This is key to the generalization to novel objects in a learned object category, and also part of the reason that we chose PAS features in the first place. By doing so, we sacrifice some of the abilities for the learned shape models to describe visual aspects accurately. The shape model that captures different heights of mug handles allows a wide range of visual aspects when the mug is tilted away or towards the image plane. Due to this choice, we observe large hand orientation errors for a substantial number of test images.

Given the above difficulties, the experimental results have shown promising results. One should note that the proposed algorithm is independent of particular visual operators. One can easily use other visual operators in place of PAS-based shape models in our framework. Some of the candidate visual operators may focus instead on depth images (Ohbuchi et al., 2008; Goldfeder and Allen, 2011).

In our experiments, we choose to match the learned shape models to foreground images (which usually contain the entire target object and part of the hand that is holding the object). Matching to the entire image will slightly increase the computation time (only during the sliding window matching phase) and moderately decrease the performance. We choose to only match to the foreground because the foreground images can usually be extracted robustly with object-background segmentation on 2D or range images.

In our experimental results, the shape models associated with rotationally symmetric object components generally perform better than those associated with unidirectional object components. Object components with rotational symmetries are usually simpler and easier to match, and free of self occlusions. In addition, for object components with rotational symmetry, we usually have adequate training samples since we acknowledge this symmetry during the aspect clustering process. In the PAS shape model learning process, more training data usually yield more robust

shape models. This ensures that a learned shape model generalizes well across novel images in the same object category. In contrast, the appearance of object components without rotational symmetries usually change dramatically with small aspect change. This indicates that we need more shape models to cover different aspects for object components associated with unidirectional grasp types. Given a certain number of training grasp examples, more shape models means less training data for the learning of each shape model. For unidirectional grasp types, we typically have only 5 to 10 images in a single aspect cluster from which to learn a shape model. Based on our experience, the shape models learned from such a small number of training images tend to capture partial texture information of individual objects, and tend not to generalize well to object components with similar shapes. All together, the above suggests that we need more samples for object components without rotational symmetries.

For a query image, the proposed algorithm is not intended to precisely determine hand position and orientation. Instead, the proposed algorithm provides a set of initial guesses that could then be refined by other information (such as haptic feedback, Coelho, Jr. and Grupen, 1997; Platt, Jr., 2006; Wang et al., 2007). In our experimental results, one version of our algorithm returns a set of *Top N* matched shape models. Finding N candidate grasps, instead of only the top one, allows the robot to further refine and sub-sample these grasps based on "real" experience (Detry et al., 2009; Goldfeder and Allen, 2011).

8.5.1 Common Source of Errors

The common source of errors made by the proposed algorithm is summarized as follows:

1. A shape model is too generic (indiscriminative). For example, the algorithm learns a shape model that contains a single curve on edge of the bowl (Fig. 8.15), or a

shape model for a drill handle that is learned from a degenerated view. Although we use two steps of filtering shape models to alleviate this problem, some indiscriminative shape models can still survive. A possible solution may be to integrate visual features of different modalities (such as those calculated on depth images) into our PAS based shape models.

2. A shape model is too specific. We have shown generalization of shape models learned by our algorithm to novel objects in known object categories. However, some of the learned shape models do not generalize well to novel objects with similar shapes. Some shape models still capture object-specific textures such as those on the labels, and sometimes, even on the hand that holds the object, or the data glove. This problem can be alleviated by increasing the training set size.

3. Ambiguity in aspects. Multiple explanations can be made to a 2D shape without depth cues. An ellipse in a 2D image may correspond to a circle in 3D space. In our experimental results, the shape models learned on the hammer handles are insensitive to camera tilt. Another example is the ambiguity between the top view and bottom view of a glass. These problems are inherent in 2D visual models and can only be solved by incorporating some form of 3D depth information.

4. A shape model is insensitive to small aspect changes. We choose to use PAS-based shape models, since they can generalize to novel objects with small variations in shape compared to the training set of objects. However, shape deformations can come from both aspect change and within-class object shape variation. In other words, we trade off some of the sensitivity to aspect change by allowing generalization to novel objects. A possible solution is to incorporate 3D depth information.

8.5.2 Running Time

The current running speed of our algorithm is slow, since for each test image, our algorithm has to match to more than 200 shape models. One way to speed up this

process is to only match to a subset of shape models given the current task. This requires some type of semantic representation of an object category. For example, if we want to grasp a mug in order to drink from it, we only need to search for a subset of grasps that will not block the top of the mug.

In the above performance evaluation, the *Top 2* and *Top 3* versions of our algorithm suggest more than one grasp hypotheses. A robot can further eliminate some grasp hypotheses that do not perform reliably in practice (Detry et al., 2009), and only keep those with high success rate.

For a grasp type with rotational symmetry, a matched shape model may suggest a manifold of hand orientations surrounding the symmetry axis. We can select one hand orientation from this manifold of orientations that is closest to the current hand pose. For details of this approach, refer to de Granville (2008).

Chapter 9

Conclusions and Future Work

J. J. Gibson (1977) argued that objects are represented in the brain (in part) in terms of how the agent interacts with the objects. In particular, he argued that the categories of objects are determined by the categories of this physical interaction. In this dissertation, I propose a visual learning algorithm that is driven by the categories of grasps that are used to interact with a set of objects. Based on Gibson's idea, the proposed algorithm categorizes objects based on how they are grasped by a human teacher. For each object category, the proposed algorithm maps partial visual models directly to hand positions and orientations that can be used for grasp control. When a robot is confronted with a novel object, the proposed algorithm allows the robot to visually index into the appropriate set of grasps that can be applied to the object. The proposed algorithm maps from partial visual models directly to hand positions and orientations without explicit object recognition. This allows generalization to novel objects with similar partial shapes.

In this work, the grasp experience of a robot comes from human demonstration. The OUGD is composed of 50 objects in 10 different categories. In the OUGD, each object is associated with multiple examples of each of several grasp types, and each grasp sample is associated with a triplet of stereo image pairs: background, object and object with hand. By using this triplet of stereo image pairs, the proposed algorithm automatically approximates the visual locations of the object-hand contacts.

The two key contributions of this dissertation are as follows:

1. An algorithm learns object categories based on how the objects are

grasped. The different ways (grasp types) that an object can be grasped are captured by its grasp affordance model. The proposed algorithm then measures the similarity between each pair of objects by matching their grasp affordance models in structure. In reality, the affordance model matching result is noisy in that objects from different true categories can also partially match to each other's affordance model in structure. The similarity relationship between objects is represented by a graph, and a modified Highly Connected Sub-graphs algorithm is used to find the highly connected sub-graphs in the entire graph. These highly connected sub-graphs usually correspond to objects that belong to the same true category.

In the affordance matching experiments, by using sufficient training data, the modified Highly Connected Sub-graphs algorithm clusters the entire set of objects in the OUGD consistently with the ground truth. The performance of the proposed algorithm decreases gracefully when the amount of training data decreases. Even with a small amount of training data (about 20 training samples for each grasp type), the proposed algorithm clusters most of the object categories correctly for some particular experiments.

2. An algorithm learns visual operators that predict grasp actions for novel objects. For each object category found above, the proposed algorithm learns a unified grasp affordance model by aggregating grasp samples from all objects in this category. The unified grasp affordance model then partitions the entire set of grasp samples based on which grasp type they belong to. Each partition of grasp samples is associated with a set of corresponding image fragments in which the object-hand contacts happen. For each grasp type, these image fragments are further partitioned by aspect, Pair of Adjacent Segments shape model is then learned for each partition of image fragments. Each shape model is then associated with a hand orientation that is appropriate for grasping the object.

When the hypothetical robot is confronted with a novel image pair, the proposed

algorithm identifies which of the learned shape models occur within the image. I extend the original PAS shape matching algorithm to include stereo image constraints. This results in a more robust shape model matching process, compared with single image shape matching. A shape model is detected if it matches to a sub-region of the image with a matching score above its model threshold. When there are multiple shape models matched in the image, the proposed algorithm ranks these matches based on their matching scores. Since a shape model is associated with a grasp type and a relative hand pose, a matched shape model can suggest a possible grasp plan that can be used by the robot to reach and grasp the target object.

In the visual learning experiments, the proposed algorithm learns shape models that are usually visually identifiable by humans. With automatically extracted grasp regions, the proposed algorithm successfully learns visual models that capture the gross shapes of target object components. These shape models capture within-class shape variations, and are robust to small aspect changes. In many cases, these shape models match well to novel objects, and predict grasp type and hand poses reliably. However, some visually indiscriminative shape models still survive the shape model filtering process, and do not perform well for some particular aspects. The learned shape models usually estimate the location of the hand reliably, while some shape models match to a wider range of aspects than is allowed by the chosen error bound. The proposed visual learning algorithm performs significantly better than any baseline algorithm. For a given query image, it is usually beneficial to consider more candidate matches of shape models other than the one that gives the highest matching score. The experimental results show extensive generalizations of shape models across different object categories. Some of these generalizations even involve objects with very different gross shapes (such as a hammer and a hand drill). However, the learned shape models successfully generalize between object components with similar shapes in these objects.

9.1 Conclusions

This dissertation is concluded as follows:

1) Objects can be categorized by the way in which they are grasped and manipulated. The affordance based object categorization scales well with the amount of training data. The different grasp types in an object category can be used to drive the learning of meaningful visual models that capture partial object shapes.

2) Because the PAS-based visual models capture common shapes and within-class variations in a set of training image fragments, they generalize well to novel objects with similar shapes. However, the PAS-based shape models are insensitive to small variations in visual aspect. In addition, since the PAS-based shape matching operates on 2D images (although a stereo image pair is used), some visual ambiguities are inevitable.

3) Visual models describing object components generalize well to novel objects with similar partial shapes. This generalization is very important in robot grasp learning, since it greatly reduces the total number of objects that a robot needs to learn to grasp.

4) A matched visual model can suggest an approximation to how the hand should be positioned and oriented relative to the object in novel scenarios containing novel objects. In many situations, the grasp types and corresponding hand poses are estimated accurately by matching 2D shape models. However, for certain visual aspects, the hand orientations are not estimated reliably. This is in part due to the fact that the PAS-based visual model is insensitive to small aspect change. This is also due to some visual ambiguities in 2D shape matching.

9.2 Future Work

In the experiments, some shape models apply to a wider range of aspects than is allowed by the hand orientation error criterion. For example, handles can appear in 2D to be the same over a wide range of out-of-plane orientations. As a result, the proposed algorithm achieves high performance on finding the relevant part of the object, but performs poorly when estimating the appropriate hand orientation. In a future implementation of the algorithm, we plan to integrate depth information to make the final decisions about hand orientation. Specifically, depth information can be derived from range sensors or dense disparity maps. For all object categories, since we have recorded both the left and right images in stereo pairs, the corresponding dense disparity maps could be calculated accordingly.

Another extension to the proposed algorithm is to integrate hand configuration besides hand location and orientation into the grasp affordance clustering process (de Granville, 2008). This will allow matching between proto-grasps directly without object categorization. Also, this will allow for the pruning of redundant grasp types associated with different object categories. An example is that the top grasp learned from the mugs is very similar to the top grasp learned from the set of glasses. The hand configuration can be described as clusters in eigengrasp space (Ciocarlie et al., 2007).

The current running time of the proposed algorithm is expensive. This is due to the fact that, for each test image, our algorithm needs to match to a large set of shape models. One way to alleviate this problem is to integrate semantic context. Given the current task, a robot only needs to consider a subset of relevant shape models. For example, if the current task is to grasp an object in the environment that allows pouring water into it, a robot may only need to consider shape models corresponding to a handle grasp of the mug, a side grasp of the glass, or a side grasp of the bowl.

In the scope of this dissertation, the hypothetical robot observes a human teacher demonstrating a set of grasp examples. We instrument both the object and the grasping hand. This is intended to model an agent's ability to track the relative pose of an object as it manipulates that object. In the future work, we would like the robot to learn from its own experience in grasping and manipulating objects. This would require that the object pose be estimated during the manipulation process. The robot could use a combination of depth, visual and proprioceptive cues to estimate pose.

Bibliography

Aleotti, J. and S. Caselli, 2011: Part-based robot grasp planning from human demonstration. *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2011*, IEEE, 4554–4560.

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

Argall, B. D., S. Chernova, M. Veloso, and B. Browning, 2009: A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, **57**, 469–483.

Asada, M., K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, 2009: Cognitive developmental robotics: A survey. *IEEE Transactions on Autonomous Mental Development*, **1**, 12–34.

Asada, M., K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi, 2001: Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, **37**, 185–193.

Augustine, E., L. B. Smith, and S. S. Jones, 2011: Parts and relations in young children's shape-based object recognition. *Journal of Cognition and Development*, **12**, 556–572.

Balaguer, B. and S. Carpin, 2010a: Efficient grasping of novel objects through dimensionality reduction. *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2010*, 1279–1285.

—, 2010b: Learning end-effector orientations for novel object grasping tasks. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Humanoids 2010*, 302 –307.

Bar-Aviv, E. and E. Rivlin, 2006: Functional 3D object classification using simulation of embodied agent. *Proceedings of the British Machine Vision Conference, BMVC 2006*, 307–316.

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., 1995: On the closure properties of robotic grasping. *The International Journal of Robotics Research*, **14**, 319–334.

Biederman, I., 1987: Recognition-by-components: a theory of human image understanding. *Psychological Review*, **94**, 115–147.

Blake, A., M. Taylor, and A. Cox, 1993: Grasping visual symmetry. *Proceedings of the IEEE International Conference on Computer Vision, ICCV 1993*, 724–733.

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.

Brooks, R. A., C. B. Ferrell, R. Irie, C. C. Kemp, M. Marjanovi, B. Scassellati, and M. M. Williamson, 1998: Alternative essences of intelligence. *Psychiatry Interpersonal and Biological Processes*, **1**, 961–968.

Castellini, C., T. Tommasi, N. Noceti, F. Odone, and B. Caputo, 2011: Using object affordances to improve object recognition. *IEEE Transaction on Autonomous Mental Development*, **3**, 207–215.

Chui, H. and A. Rangarajan, 2003: A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, **89**, 114–141.

Ciocarlie, M., C. Goldfeder, and P. K. Allen, 2007: Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. *Proceedings of the Robotics: Science & Systems 2007 Workshop - Sensing and Adapting to the Real World*, Electronically published.

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

Cohen, J., 1960: A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**, 37–46.

Comaniciu, D. and P. Meer, 2002: Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 603–619.

Cutkosky, M. R., 1989: On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, **5**, 269–279.

de Granville, C., 2008: *Learning Grasp Affordances*. Master's thesis, School of Computer Science, University of Oklahoma, Norman, OK.

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, ICDL 2006*, electronically published.

de Granville, C., D. Wang, J. Southerland, and A. H. Fagg, 2009: Grasping affordances: Learning to connect vision to hand action. *The Path to Autonomous Robots; Essays in Honor of George A. Bekey*, G. Sukhatme, ed., Springer, 59–80.

Detry, R., E. Başeski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater, 2009: Learning object-specific grasp affordance densities. *Proceedings of the IEEE International Conference on Development and Learning, ICDL 2009*, 1–7.

Dufournaud, Y., R. Horaud, and L. Quan, 1998: *Robot Stereo-hand Coordination for Grasping Curved Parts*, volume 2. British Machine Vision Association, 760–769 pp.

Fergus, R., P. Perona, and A. Zisserman, 2003: Object class recognition by unsupervised scale-invariant learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Los Alamitos, CA, USA, volume 2, 264–271.

Ferrari, C. and J. Canny, 1992: *Planning optimal grasps*, volume 3. IEEE, 2290–2295 pp.

Ferrari, V., L. Fevrier, F. Jurie, and C. Schmid, 2008: Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**, 36–51.

Ferrari, V., F. Jurie, and C. Schmid, 2010: From images to shape models for object detection. *International Journal of Computer Vision*, **87**, 284–303.

Forssén, P.-E. and D. G. Lowe, 2007: Shape descriptors for maximally stable extremal regions. *Proceedings of the IEEE International Conference on Computer Vision, ICCV 2007*, IEEE Computer Society, Rio de Janeiro, Brazil, 1–8.

Fukunaga, K. and L. Hostetler, 1975: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, **21**, 32–40.

Gallese, V., L. Fadiga, L. Fogassi, and G. Rizzolatti, 1996: Action recognition in the premotor cortex. *Brain*, **119**, 593–609.

Gibson, J. J., 1966: *The Senses Considered as Perceptual Systems*. Allen and Unwin.

—, 1977: The theory of affordances. *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, R. E. Shaw and J. Bransford, eds., Lawrence Erlbaum, Hillsdale, NJ, 67–82.

Goldfeder, C. and P. K. Allen, 2011: Data-driven grasping. *Autonomous Robots*, 1–20.

Goldfeder, C., M. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen, 2009: Data-driven grasping with partial sensor data. *Proceedings of the IEEE/RSJ International Conference on Robots and Intelligent Systems, IROS 2009*, 1278–1283.

Gordon, I. and D. Lowe, 2006: What and where: 3D object recognition with accurate pose. *Toward Category-Level Object Recognition*, 67–82.

Graf, M., 2006: Coordinate transformations in object recognition. *Psychological Bulletin*, **132**, 920–945.

Griffin, G. and P. Perona, 2008: Learning and using taxonomies for fast visual categorization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008*, 1–8.

Hartley, R. and A. Zisserman, 2004: *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Hartuv, E. and R. Shamir, 2000: A clustering algorithm based on graph connectivity. *Information Processing Letters*, **76**, 175–181.

Hubert, L. and P. Arabie, 1985: Comparing partitions. *Journal of Classification*, **2**, 193–218.

Hummel, J. E. and I. Biederman, 1992: Dynamic binding in a neural network for shape recognition. *Psychological Review*, **99**, 480–517.

Ikeuchi, K., H. K. Nishihara, B. K. Horn, P. Sobalvarro, and S. Nagata, 1986: Determining grasp configurations using photometric stereo and the prism binocular stereo system. *The International Journal of Robotics Research*, **5**, 46–65.

Jeannerod, M., 1997: The cognitive neuroscience of action. *Trends in Cognitive Sciences*, **1**, 238.

Jensen, D. and P. Cohen, 2000: Multiple comparisons in induction algorithms. *Machine Learning*, **38**, 309–338.

Johnson, M., D. Mareschal, G. Csibra, C. Nelson, and M. Luciana, 2001: The development and integration of the dorsal and ventral visual pathways: A neurocomputational approach. *Handbook of developmental cognitive neuroscience*, 139–151.

Jonquières, S., M. Devy, F. Huynh, and M. Khatib, 1999: Object recognition for a grasping task by a mobile manipulator. *Proceedings of the 1st International Conference on Computer Vision Systems, ICVS 1999*, Springer-Verlag, London, UK, ICVS 1999, 538–551.

Kilner, J. M., A. Neal, N. Weiskopf, K. J. Friston, and C. D. Frith, 2009: Evidence of mirror neurons in human inferior frontal gyrus. *Journal of Neuroscience*, **29**, 10153–10159.

Kjellström, H., J. Romero, and D. Kragić, 2011: Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, **115**, 81–90.

Kotovsky, L. and D. Gentner, 1996: Comparison and categorization in the development of relational similarity. *Child Development*, **67**, 2797.

Kragic, D. and H. I. Christensen, 2002: *Vision techniques for robotics manipulation and grasping*. Stockholm, SE.

Leibe, B., A. Leonardis, and B. Schiele, 2008: Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, **77**, 259–289.

Lowe, D. G., 2004: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**, 91–110.

Lungarella, M. and G. Metta, 2003: Beyond gazing, pointing, and reaching: A survey of developmental robotics. *EPIROB*, 81–89.

Lungarella, M., G. Metta, R. Pfeifer, and G. Sandini, 2003: Developmental robotics: a survey. *Connection Science*, **15**, 159–190.

MacKenzie, C. L. and T. Iberall, 1994: *The Grasping Hand*. Elsevier-North Holland.

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

Mareschal, D. and M. H. Johnson, 2003: The "what" and "where" of object representations in infancy. *Cognition*, **88**, 259–276.

Marszalek, M., 2008: *Past the limits of bag-of-features*. Ph.D. thesis, Institut Polytechnique de Grenoble.

Mason, M. T. and J. K. Salisbury Jr., 1985: *Robot Hands and the Mechanics of Manipulation*. The MIT Press, xxvi + 298 pp.

Metta, G. and P. Fitzpatrick, 2003: Early integration of vision and manipulation. *Proceedings of the International Joint Conference on Neural Networks 2003*, **11**, 2703–2703.

Mikolajczyk, K. and C. Schmid, 2005: A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**, 1615–1630.

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

Milner, A. D. and M. A. Goodale, 1995: *The Visual Brain in Action*. Number 27 in Oxford Psychology Series, Oxford University Press.

—, 2008: Two visual systems re-viewed. *Neuropsychologia*, **46**, 774–785.

Napier, J., 1993: *Hands*. Princeton University Press.

Ohbuchi, R., K. Osada, T. Furuya, and T. Banno, 2008: Salient local visual featuers for shape-based 3D model retrieval. *Proceedings of the IEEE International Conference on Shape Modeling and Applications, SMI 2008*, 93–102.

Opelt, A., A. Pinz, and A. Zisserman, 2006: A boundary-fragment-model for object detection. *Proceedings of the European conference on Computer Vision, ECCV 2006*, Springer-Verlag, Berlin, Heidelberg, II: 575–588.

Palmer, T. J. and A. H. Fagg, 2009: Learning grasp affordances with variable centroid offsets. *Proceedings of the IEEE/RSJ International Conference on Robots and Intelligent Systems, IROS 2009*, 1265–1271.

Pereira, A. F., K. H. James, S. S. Jones, and L. B. Smith, 2010: Early biases and developmental changes in self-generated object views. *Journal of Vision*, **10**, 1–13.

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

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.

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

Rancourt, D., L.-P. Rivest, and J. Asselin, 2000: Using orientation statistics to investigate variations in human kinematics. *Applied Statistics*, **49**, 81–94.

Rand, W. M., 1971: Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, **66**, 846–850.

Rao, K., G. Medioni, H. Liu, and G. A. Bekey, 1988: Robot hand-eye coordination: shape description and grasping. *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 1988*, IEEE Computer Society Press, 407–411.

Rivest, L.-P., 2001: A directional model for the statistical analysis of movement in three dimensions. *Biometrika*, **88**, 779–791.

Rizzolatti, G. and L. Craighero, 2004: The mirror-neuron system. *Annual Review of Neuroscience*, **27**, 169–192.

Romea, A. C., D. Berenson, S. Srinivasa, and D. Ferguson, 2009: Object recognition and full pose registration from a single image for robotic manipulation. *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2009*, electronically published.

Rothganger, F., S. Lazebnik, C. Schmid, and J. Ponce, 2006: 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, **66**, 231–259.

Sahbani, A., S. El-Khoury, and P. Bidaud, 2012: An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, **60**, 326–336.

Saxena, A., J. Driemeyer, and A. Y. Ng, 2009: Learning 3-D object orientation from images. *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2009*, electronically published.

Saxena, A., J. Driemeyer, and A. Y. Ng, 2008: Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, **27**, 157–171.

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

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.

Sporns, O., 2003: Embodied cognition. *MIT Handbook of Brain Theory and Neural Networks*, M. Arbib, ed., MIT Press, Cambridge, MA, 395–398.

Stanley, K., Q. M. J. Wu, and W. A. Gruver, 2000: Implementation of vision-based planar grasp planning. *IEEE Transactions on Systems, Man, and Cybernetics*, **30**, 517–524.

Stark, M., P. Lies, M. Zillich, J. Wyatt, and B. Schiele, 2008: Functional object class detection based on learned affordance cues. *Proceedings of the 6th International Conference on Computer Vision Systems, ICVS 2008*, Santorini, Greece, oral presentation.

Stoytchev, A., 2005: Toward learning the binding affordances of objects: A behavior-grounded approach. *Proceedings of the AAAI Spring Symposium on Developmental Robotics*, electronically published.

Street, S. Y., K. H. James, S. S. Jones, and L. B. Smith, 2011: Vision for action in toddlers: the posting task. *Child Development*, **82**, 2083–2094.

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*.

Umiltà, M., E. Kohler, V. Gallese, L. Fogassi, L. Fadiga, C. Keysers, and G. Rizzolatti, 2001: I know what you are doing: A neurophysiological study. *Neuron*, **31**, 155–165.

Utgoff, P. E. and J. A. Clouse, 1996: A Kolmogorov-Smirnoff metric for decision tree induction. Technical Report Computer Science Technical Report 96-3, University of Massachusetts, Amherst.

Varadarajan, K. and M. Vincze, 2011: Affordance based Part Recognition for Grasping and Manipulation. *Workshop at the IEEE International Conference on Robotics and Automation, ICRA 2011*, electronically published.

Wang, D., 2007: *A 3D Feature-Based Object Recognition System for Grasping*. Master's thesis, School of Computer Science, University of Oklahoma, Norman, OK.

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

Weng, J., J. Mcclelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, 2001: ARTIFICIAL INTELLIGENCE: Autonomous mental development by robots and animals. *Science*, **291**, 599–600.

Woods, K., D. Cook, L. Stark, and K. Bowyer, 1995: Learning membership functions in a function-based object recognition system. *Journal of Artificial Intelligence Research*, **3**, 177–222.

Ying, L. Y. L., J. L. Fu, and N. S. Pollard, 2007: Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions on Visualization and Computer Graphics*, **13**, 732–747.

Appendix A

Rotationally-invariant PAS features

Given the way that the PAS descriptor is constructed, it is not rotationally invariant since the ordering of the two edges and the edge orientations are defined in the image coordinate frame. When the camera is rotated within the image plane, the set of edges in the image will also rotate accordingly. However, the image coordinate frame is always the same. Thus, the PAS descriptors calculated from the original scene will no longer match to those calculated from the rotated scene. However, the affine transformation in a TPS mapping allows small rotations to be encoded, which makes the learned shape model be able to match to a shape rotated slightly (up to about 25 degrees) in the image plane (the first pair of images in Figure A.1).

To make the PAS descriptor rotationally invariant, the key is to use a coordinate frame that is relative to each PAS feature. Since the PAS descriptor is calculated based on the order of the two segments, we need a unique way to determine the order of the two segments, so that the same PAS always gives the same ordering after rotation. The original approach may give an inconsistent ordering, since the relative left-right and/or top-bottom locations between two segments may change after the image is rotated. For this reason, I reorder the two segments based on their lengths. Without loss of generality, I order the shorter segment as the first segment. The first two elements of the PAS descriptor denote a unit vector pointing from the midpoint of the first segment to the midpoint of the second segment. It is natural to use this vector as the x axis of the PAS coordinate frame. The y axis can be determined accordingly by rotating the x axis 90 degrees clockwise (the right hand rule). The

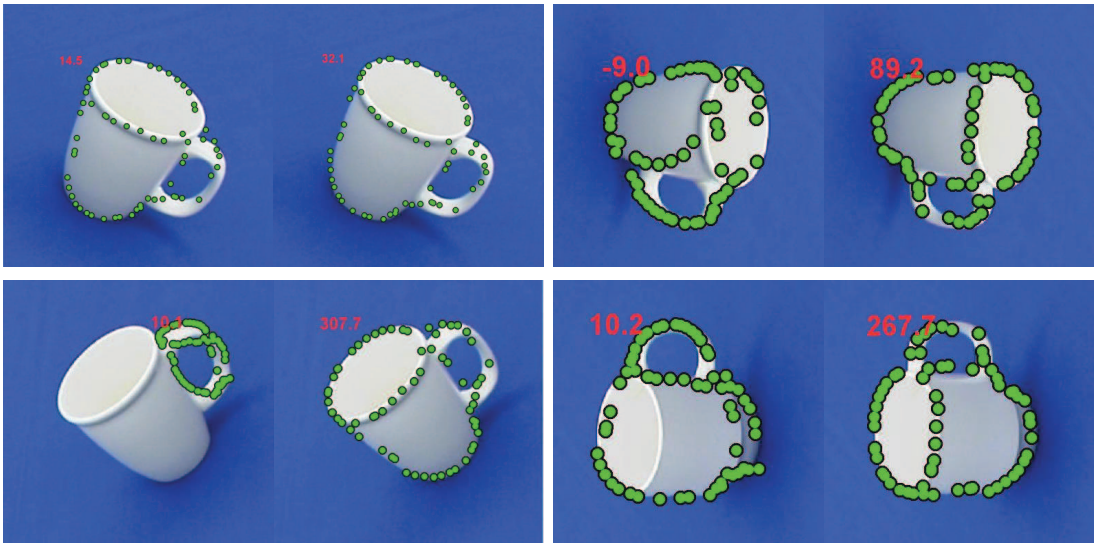


Figure A.1: Adding rotational invariance to the PAS features. There are four groups of images and two images next to each other in each group. The image on the left in each group is recognized by the original PAS features. The image on the right in each group is recognized by the rotational-invariant PAS features. The number in each image is the estimated in-plane rotation in degrees.

direction of the x axis is also used as the canonical orientation of the PAS. The segment orientations θ_1 and θ_2 are calculated based on the PAS coordinate frame. The segment lengths $\frac{l_1}{N_d}$ and $\frac{l_2}{N_d}$ remain the same in the new descriptor, since they are rotationally invariant.

During the matching process, the algorithm matches the model PAS features to those observed in a test image. Since the modified PAS features are rotationally invariant, they can match to arbitrarily rotated objects in the test image. Recall that in the original PAS matching, each match votes for a center location (x/y) and a scale in a 3D Hough space. As a contrast, in my extension, each match votes for a center location, a scale and a rotation of the model in a 4D Hough space. Each local maximum in this case specifies a center location, a scale and an initial rotation of the model. Again, the local maxima are used as initial conditions for the TPS-RPM step. Given each initial condition, the algorithm centers, scales and rotates the shape model in the test image accordingly based on the Hough voting result. Given a initial

condition, a subset of edgels in the test image is cropped by using the translated, scaled and rotated bounding box of the shape model. Given the model points and the subset of image points, the algorithm uses TPS-RPM to find a mapping between them, as before. The estimated rotation of the recognized shape is the sum of the initial rotation found by the Hough voting and the rotation specified by the affine component d of the TPS mapping.

In each pair of images in Figure A.1, the left image shows a match of the mug model found by using the original PAS features. The right image shows a match found by using the rotationally invariant PAS features. The original PAS features can only tolerate in-plane rotations up to about 25 degrees, while the rotationally invariant PAS features work well on all rotations that we examined. However, the discriminative power of the PAS descriptor is reduced because of this extra degree of freedom in rotation. The rotationally invariant PAS features are more likely to incorrectly match an image, as compared with the original PAS features. As a result, the 4D Hough voting process returns many false positive local maxima. An alternative approach is to regularly sample a series of rotations, and for each rotation, use the 3D Hough voting approach to find the local maxima in x/y position and scale. For each candidate in each orientation, we use TPS-RPM to then refine the match. Finally, the best match across all sampled orientations is selected as the one with the highest score. Currently, this approach yields the most promising results. However, rotationally-invariant PAS features were not used in the results reported in this dissertation.

Appendix B

Automatically Evaluating Shape Model Matches

One of the difficulties in the performance evaluation process is how to properly assess whether a matched shape model in a test image is correct or not. Since we have thousands of test images for a single experiment, we would like to label them automatically, without a human "in the loop." One possible solution is to measure 1) whether a matched shape model overlaps with the ground truth bounding box of a test image above some threshold value, together with 2) whether the difference between the estimated and ground truth aspects is smaller than some threshold value. If a matched shape model in a test image satisfies both 1 and 2, we label this test image as correct. However, this approach will not allow the generalization of a shape model to other objects that share similar parts with those objects on which this shape model is originally learned. For example, the shape model learned from the top of a glass may match well with the top of a mug in a test image, and we would like to label this match as correct. The above approach fails in this case, because it is meaningless to compare aspects of objects from different object categories, whose coordinate alignment is unknown. Also, there are multiple ground truth bounding boxes associated with a single test image, and for a given shape model, we do not know which ground truth bounding box we should compare with. In the forementioned example, our algorithm simply has no idea about the fact that a shape model of the mug top is expected to be observed on the top of a glass. In order to tackle this problem, one way is to manually build connections between these similar object parts. However, this turns out to be difficult since some object parts only look similar to each other

for some particular viewing angles. For example, a ball looks exactly like a glass when observed from the top (without 3D depth information).

Given the above difficulties, we propose an algorithm that automatically labels a match based on the estimated grasp, i.e., the hand location and orientation. In order for a match in a test image to be labeled as *correct*, it needs to satisfy the following: 1) the matched shape model has to overlap very well with one of the ground truth bounding boxes corresponding to an object part; and 2) the estimated hand orientation (relative to the camera) should align very well with the ground truth hand orientation associated with this object part. We assume that a matched shape model is correct if it satisfies the above two criteria with any of the ground truth hand locations and orientations associated with a test image. Accordingly, we label this test image as correct. Since we do not compare the match to a single ground truth grasp type corresponding to a certain object component, there is no need to know *a priori* which grasp type this match corresponds to across different object categories. Also, we do not need to explicitly calculate the alignment of two objects in order to compare hand orientations, since these hand orientations are relative to the camera coordinate frame. When a mug is observed from a point of view where its top looks like that of a glass, grasps with the same orientations relative to the observer can be applied to both them. In the scope of this dissertation, we assume that there is no in-plane rotation between the shape model and its match in the test image.

Specifically, the algorithm that we use to assess a set of matched shape models in a test image is detailed by Algorithm 9. For a given test image with the matching results of a set of shape models, we use Algorithm 9 to label a match corresponding to shape model i, where i is a unique model index. Recall that each test image is associated with a set of grasp types that are automatically found by the grasp affordance learning algorithm: $\mathcal{G} = \{G_m\}_{m=1..M}$, with M denoting the total number of grasp types. Each of these grasp types is associated with a ground truth bounding box

and a hand orientation, which are found by either image differencing or synthesizing (Appendix). Given the above, the *for* loop iterates through each ground truth grasp type, G_m, associated with a test image, I, and compares the hand location and orientation estimated by shape model i with the ground truth bounding box, B_m, and hand orientation associated with G_m. The degree to which a matched bounding box overlaps with a ground truth bounding box is measured by the intersection over union ratio (IoU). The hand orientation error, HOE, is calculated as the mean of errors between the estimated hand orientation and all hand orientations (down to symmetry, as explained below) associated with the set of training images in which this shape model is learned.

Algorithm 9 An algorithm that automatically labels a match of a shape model in a test image. Input: a test image, I, and a match of shape model i, $match_i$

```
procedure ASSESSMATCH(I, match_i)
    if match_i.score > t_i^* then
        valid ← 0
        for all G_m in 𝒢 do                    ▷ 𝒢 comes from the grasp affordance model
            [B_m, O_m, PDF] =GET_GT(I, G_m)
            if B_m is NOT empty then
                valid = 1
                if IoU(match_i.boundingbox, B_m) > 0.2 &
                   HOE(match_i.orientation, O_m, PDF) < 30° then
                    return correct
                end if
            end if
        end for
        if valid then
            return wrong
        else
            return invalid
        end if
    else
        return not found
    end if
end procedure
```

Before entering the *for* loop, we first check whether the the matching score of

Algorithm 10 A function that calculates the IoU between two bounding boxes

function IOU($bbox_1, bbox_2$) ▷ $bbox$: bounding box
  return $bbox_1 \cap bbox_2 / bbox_1 \cup bbox_2$
end function

Algorithm 11 A function that calculates the HOE between the estimated and ground truth hand orientations

function HOE(O_{est}, O_{gt}, PDF)
  ▽ O_{est} comes from a set of samples from which a shape model is learned
  for all O_i in O_{est} **do**
    if PDF is girdle **then**
      ▽ project O_i onto the plane spanned by $PDF.u_1$ and $PDF.u_2$
      O_{min} =PROJECT($O_i, PDF.u_1, PDF.u_2$)
    end if
    hoe_i =ROTATION_ANGLE(O_{min}, O_{gt})
  end for
  return MEAN(hoe)
end function

the matched shape model exceeds its model threshold t_i^*. If so, the algorithm enters the *for* loop and evaluates the quality of this match, $match_i$, against all grasp types, $\mathcal{G} = \{G_m\}_{m=1..M}$, associated with the test image. We first check the IoU between the detected and ground truth bounding boxes. Note that the ground truth bounding boxes associated with some of the grasp types may be empty. This is due to the fact that we only demonstrate a single grasp for this image and a close neighbor may not exist during our bounding box synthesis process. The test image is labeled as *valid* if at least one ground truth bounding box exists. If the detected and ground truth bounding boxes have an IoU greater than 0.2, we further calculate the mean hand orientation error, HOE. If a matched shape model satisfies both the IoU and HOE criteria, we label it as *correct*.[1] The *for* loop terminates after it iterates through all grasp types. A valid test image will be labeled as *correct* if the matched shape model is labeled as correct with respect to any of the grasp types in $\mathcal{G}$. Otherwise, the test image is labeled as *wrong*. So, given a valid test image, a shape model in the

[1] In the case of *correct location*, we only check the IoU criterion.

repository can either be *found* or *not found*. If a shape model is found, it can either be *correct* or *wrong*. If this test image is matched against a set of shape models, we further repeat the above assessment process for the other shape models in this set. The entire assessment process is done after we have checked the matches of all shape models for this test image.

The key component to the above assessment algorithm is the calculation of hand orientation error (Algorithm 11). This error measure respects any symmetry property of a ground truth grasp type, G_m, associated with the test image. For a uni-directional grasp type, whose ground truth hand orientations are described by a Dimroth-Watson distribution, we directly calculate the hand orientation error between the ground truth and each predicted hand orientation.[2] Since a shape model is learned from a set of grasp samples, each grasp sample in this set recommends a separate hand orientation. Since we do not have any pre-knowledge to determine which hand orientation to use out of this set of orientations that are suggested by the shape model, we simply calculate the mean of hand orientation errors between each of the hand orientation in this set and the ground truth hand orientation. The hand orientation error between the ground truth and a single predicted hand orientation is measured by the magnitude of the minimal rotation angle that aligns the two.

In the case where the ground truth hand orientation is captured by a girdle distribution, we take into consideration the rotational symmetry in order to calculate the hand orientation errors. A girdle distribution describes a manifold of orientations that are spanned by its two axes (two orthogonal unit quaternions), u_1 and u_2. Our goal is to find the ground truth hand orientation, O_{min}, in this manifold of hand orientations that has minimal rotation to the predicted hand orientation, O_i (represented by a unit quaternion). Specifically, O_{min} is found by projecting the predicted hand

[2]Note that the grasp type associated with the matched shape model can be either uni-directional or rotationally symmetric, which does not make a difference in the HOE calculation

orientation, O_i, to the plane spanned by u_1 and u_2. More details of this process can be found in de Granville (2008). Again, since there are multiple hand orientations associated with the shape model, the mean hand orientation error is calculated. If the mean hand orientation error is smaller than 30 degrees, the assessment algorithm will consider the estimated hand orientation as correct.

By using the above assessment algorithm, we acknowledge the generalization of shape models beyond object categories. An example is given in Fig. B.1. In this example, the shape model is originally learned on the bottom of a set of glasses. However, our assessment algorithm still assesses a match of this shape model on the top of a mug as correct, since a ball grasp on the bottom of a glass will very likely succeed on the top of a mug, as far as the hand location and orientation are concerned.

Caveats. Attention should be paid when assessing matched shape models learned from spatulas, bowls, and the barrels of hand drills. For these objects, multiple grasp types (overhand/underhand) share visually identical grasp regions. So, a matched shape model of a grasp type will be mis-labeled if it is compared to a ground truth hand orientation corresponding to the other grasp type that shares the same grasp region. However, this is not a problem for our assessment algorithm, since it compares a matched shape model to ground truth hand orientations of all grasp types associated with a test image. For balls, we assess a matched shape model only based on the IoU criterion, since the ground truth hand orientation is arbitrary.

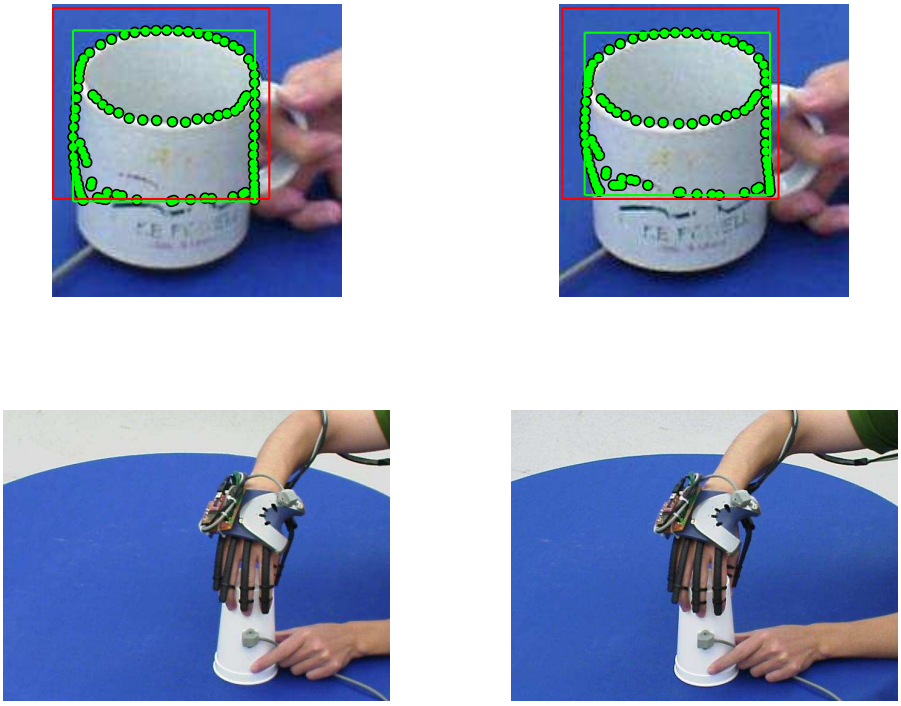


Figure B.1: A shape model learned on the bottom of glasses matches the top of a mug. The first row: a test stereo image pair. The second row: a training stereo image pair from which the matched shape model is learned. Green dots: matches of the shape model. Green boxes: the bounding boxes that enclose the matches of the shape model. Red boxes: the ground truth bounding boxes that enclose the rim of the mug. The proposed algorithm labels this match as correct based on the hand location and orientation criteria.