

# Robotic Grasping of Novel Objects using Vision

Ashutosh Saxena, Justin Driemeyer, Andrew Y. Ng

Computer Science Department

Stanford University, Stanford, CA 94305

{asaxena, jdriemeyer, ang}@cs.stanford.edu

## Abstract

We consider the problem of grasping novel objects, specifically ones that are being seen for the first time through vision. Grasping a previously unknown object, one for which a 3-d model is not available, is a challenging problem. Further, even if given a model, one still has to decide where to grasp the object. We present a learning algorithm that neither requires, nor tries to build, a 3-d model of the object. Given two (or more) images of an object, our algorithm attempts to identify a few points in each image corresponding to good locations at which to grasp the object. This sparse set of points is then triangulated to obtain a 3-d location at which to attempt a grasp. This is in contrast to standard dense stereo, which tries to triangulate every single point in an image (and often fails to return a good 3-d model). Our algorithm for identifying grasp locations from an image is trained via supervised learning, using synthetic images for the training set. We demonstrate this approach on two robotic manipulation platforms. Our algorithm successfully grasps a wide variety of objects, such as plates, tape-rolls, jugs, cellphones, keys, screwdrivers, staplers, a thick coil of wire, a strangely shaped power horn, and others, none of which were seen in the training set. We also apply our method to the task of unloading items from dishwashers.<sup>1</sup>

## 1 Introduction

In this paper, we address the problem of grasping novel objects that a robot is perceiving for the first time through vision.

Modern-day robots can be carefully hand-programmed or “scripted” to carry out many complex manipulation tasks, ranging from using tools to assemble complex machinery, to balancing a spinning top on the edge of a sword [Shin-ichi and Satoshi, 2000]. However, autonomously grasping a previously unknown object still remains a challenging problem. If we are trying to grasp a previously known object, or if we

<sup>1</sup>A preliminary version of this work was described in [Saxena *et al.*, 2006b; 2006a].

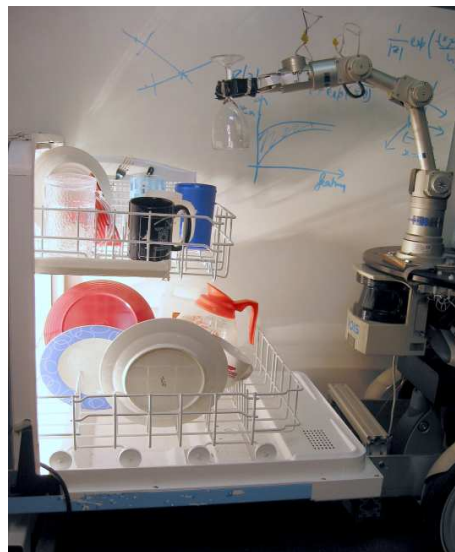


Figure 1: Our robot unloading items from a dishwasher.

are able to obtain a full 3-d model of the object, then various approaches such as ones based on friction cones [Mason and Salisbury, 1985], form- and force-closure [Bicchi and Kumar, 2000], pre-stored primitives [Miller *et al.*, 2003], or other methods can be applied. However, in practical scenarios it is often very difficult to obtain a full and accurate 3-d reconstruction of an object seen for the first time through vision. For stereo systems, 3-d reconstruction is difficult for objects without texture, and even when stereopsis works well, it would typically reconstruct only the visible portions of the object. Even if more specialized sensors such as laser scanners (or active stereo) are used to estimate the object’s shape, we would still only have a 3-d reconstruction of the front face of the object.

In contrast to these approaches, we propose a learning algorithm that neither requires, nor tries to build, a 3-d model of the object. Instead it predicts, directly as a function of the images, a point at which to grasp the object. Informally, the algorithm takes two or more pictures of the object, and then tries to identify a point within each 2-d image that corresponds to a good point at which to grasp the object. (For example, if trying to grasp a coffee mug, it might try to iden-



Figure 2: Some examples of objects on which the grasping algorithm was tested.

tify the mid-point of the handle.) Given these 2-d points in each image, we use **triangulation to obtain a 3-d position** at which to actually attempt the grasp. Thus, rather than trying to triangulate every single point within each image in order to **estimate depths**—as in dense stereo—we only attempt to triangulate one (or at most a small number of) points corresponding to the 3-d point where we will grasp the object. This allows us to grasp an object without ever needing to obtain its full 3-d shape, and applies even to textureless, translucent or reflective objects on which standard stereo 3-d reconstruction fares poorly (see Figure 6).

To the best of our knowledge, our work represents the first algorithm capable of grasping novel objects (ones where a 3-d model is not available), including ones from novel object classes, that we are perceiving for the first time using vision.

This paper focuses on the task of **grasp identification**, and thus we will consider only objects that can be picked up without performing complex manipulation.<sup>2</sup> We will attempt to grasp a number of common office and household objects such as toothbrushes, pens, books, cellphones, mugs, martini glasses, jugs, keys, knife-cutters, duct-tape rolls, screwdrivers, staplers and markers (see Figure 2). We will also address the problem of unloading items from dishwashers.

The remainder of this paper is structured as follows. In Section 2, we describe related work. In Section 3, we describe our learning approach, as well as our **probabilistic model for inferring the grasping point**. In Section 4, we describe our robotic manipulation platforms. In Section 5, we describe the **motion planning/trajectory planning** for moving the manipulator to the grasping point. In Section 6, we report the results of extensive experiments performed to evaluate our algorithm, and Section 7 concludes.

<sup>2</sup>For example, picking up a heavy book lying flat on a table might require a sequence of complex manipulations, such as to first slide the book slightly past the edge of the table so that the manipulator can place its fingers around the book.

## 2 Related Work

Most work in robot manipulation assumes availability of a complete 2-d or 3-d model of the object, and focuses on designing control and planning methods to achieve a successful and stable grasp. Here, we will discuss in detail prior work that uses learning or vision for robotic manipulation, and refer the reader to [Bicchi and Kumar, 2000; Mason and Salisbury, 1985; Shimoga, 1996] for a more general survey of past work in robotic manipulation.

In **simulation environments** (without real world experiments), learning has been applied to robotic manipulation for several different purposes. For example, [Pelossof *et al.*, 2004] used Support Vector Machines (**SVM**) to estimate the quality of a grasp given a number of features describing the grasp and the object. [Hsiao *et al.*, 2007; Hsiao and Lozano-Perez, 2006] used partially observable Markov decision processes (**POMDP**) to choose optimal control policies for two-fingered hands. They also used imitation learning to teach a robot whole-body grasps. [Miller *et al.*, 2003] used heuristic rules to generate and evaluate grasps for three-fingered hands by assuming that the objects are made of basic shapes such as spheres, boxes, cones and cylinders each with **pre-computed grasp primitives**. All of these methods **assumed full knowledge of the 3-d model** of the object. Further, these methods were not tested through real-world experiments, but were instead modeled and evaluated in a simulator.

Some work has been done on using vision for real world grasping experiments; however most were limited to grasping 2-d planar objects. For uniformly colored planar objects lying on a uniformly colored table top, one **can find the 2-d contour of the object quite reliably**. Using local visual features (based on the 2-d contour) and other properties such as form- and force-closure, the methods discussed below decide the 2-d locations at which to place (two or three) fingertips to grasp the object. [Piater, 2002; Coelho *et al.*, 2001] **estimated 2-d hand orientation using K-means clustering** for simple objects (specifically, square, tri-

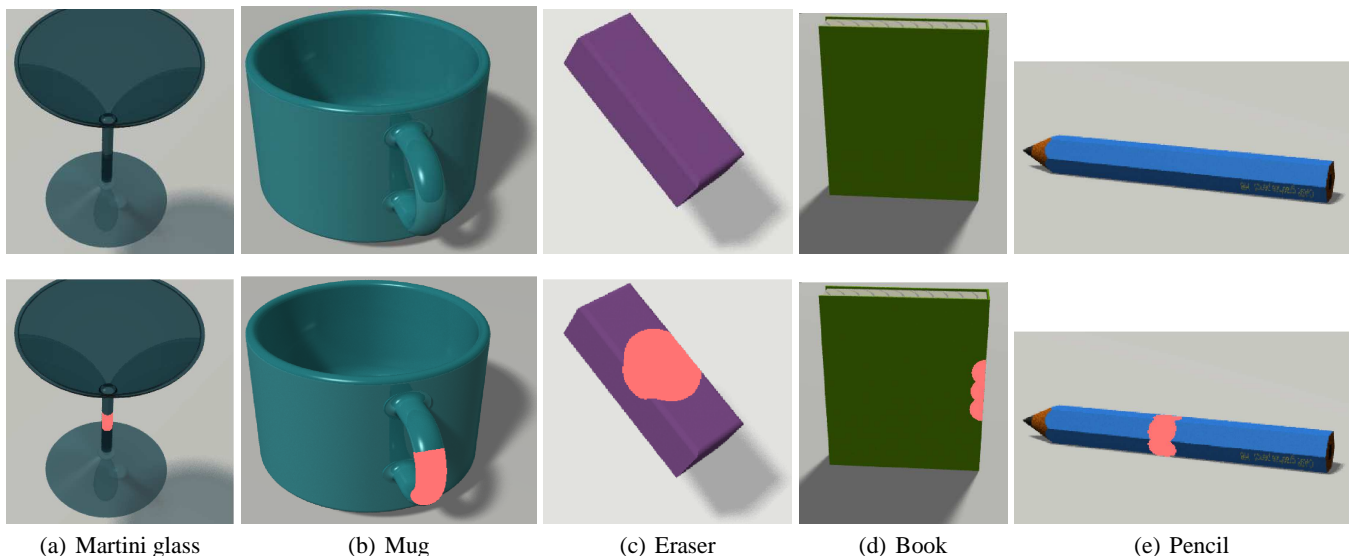


Figure 3: The images (top row) with the corresponding labels (shown in red in the bottom row) of the five object classes used for training. The classes of objects used for training were martini glasses, mugs, whiteboard erasers, books and pencils.

angle and round “blocks”). [Morales *et al.*, 2002a; 2002b] calculated 2-d positions of three-fingered grasps from 2-d object contours based on feasibility and force closure criteria. [Bowers and Lumia, 2003] also considered the grasping of planar objects and chose the location of the three fingers of a hand by first classifying the object as circle, triangle, square or rectangle from a few visual features, and then using pre-scripted rules based on fuzzy logic. [Kamon *et al.*, 1996] used Q-learning to control the arm to reach towards a spherical object to grasp it using a parallel plate gripper.

If the desired location of the grasp has been identified, techniques such as visual servoing that align the gripper to the desired location [Kragic and Christensen, 2003] or haptic feedback [Petrovskaya *et al.*, 2006] can be used to pick up the object. [Platt *et al.*, 2005] learned to sequence together manipulation gaits for four specific, known 3-d objects. However, they considered fairly simple scenes, and used online learning to associate a controller with the height and width of the bounding ellipsoid containing the object. For grasping known objects, one can also use Learning-by-Demonstration [Hueser *et al.*, 2006], in which a human operator demonstrates how to grasp an object, and the robot learns to grasp that object by observing the human hand through vision.

The task of identifying where to grasp an object (of the sort typically found in the home or office) involves solving a difficult perception problem. This is because the objects vary widely in appearance, and because background clutter (e.g., dishwasher prongs or a table top with a pattern) makes it even more difficult to understand the shape of a scene. There are numerous robust learning algorithms that can infer useful information about objects, even from a cluttered image. For example, there is a large amount of work on recognition of known object classes (such as cups, mugs, etc.), e.g., [Schneiderman and Kanade, 1998]. The performance of these object recognition algorithms could proba-

bly be improved if a 3-d model of the object were available, but they typically do not require such models. For example, that an object is cup-shaped can often be inferred directly from a 2-d image. Our approach takes a similar direction, and will attempt to infer grasps directly from 2-d images, even ones containing clutter. [Saxena *et al.*, 2005; 2007d] also showed that given just a single image, it is often possible to obtain the 3-d structure of a scene. While knowing the 3-d structure by no means implies knowing good grasps, this nonetheless suggests that most of the information in the 3-d structure may already be contained in the 2-d images, and suggests that an approach that learns directly from 2-d images holds promise. Indeed, [Marotta *et al.*, 2004] showed that humans can grasp an object using only one eye.

Our work also takes inspiration from [Castiello, 2005], which showed that cognitive cues and previously learned knowledge both play major roles in visually guided grasping in humans and in monkeys. This indicates that learning from previous knowledge is an important component of grasping novel objects.

Further, [Goodale *et al.*, 1991] showed that there is a dissociation between recognizing objects and grasping them, i.e., there are separate neural pathways that recognize objects and that direct spatial control to reach and grasp the object. Thus, given only a quick glance at almost any rigid object, most primates can quickly choose a grasp to pick it up, even without knowledge of the object type. Our work represents perhaps a first step towards designing a vision grasping algorithm which can do the same.

### 3 Learning the Grasping Point

We consider the general case of grasping objects—even ones not seen before—in 3-d cluttered environments such as in a home or office. To address this task, we will use an image of the object to identify a location at which to grasp it.



Because even very different objects can have similar sub-parts, there are certain visual features that indicate good grasps, and that remain consistent across many different objects. For example, jugs, cups, and coffee mugs all have handles; and pens, white-board markers, toothbrushes, screwdrivers, etc. are all long objects that can be grasped roughly at their mid-point (Figure 3). We propose a learning approach that uses visual features to predict good grasping points across a large range of objects.

In our approach, we will first predict the 2-d location of the grasp in each image; more formally, we will try to identify the projection of a good grasping point onto the image plane. Then, given two (or more) images of an object taken from different camera positions, we will predict the 3-d position of a grasping point. If each of these points can be perfectly identified in each image, then we can easily “triangulate” from these images to obtain the 3-d grasping point. (See Figure 8a.) In practice it is difficult to identify the projection of a grasping point into the image plane (and, if there are multiple grasping points, then the correspondence problem—i.e., deciding which grasping point in one image corresponds to which point in another image—must also be solved). This problem is further exacerbated by imperfect calibration between the camera and the robot arm, and by uncertainty in the camera position if the camera was mounted on the arm itself. To address all of these issues, we develop a probabilistic model over possible grasping points, and apply it to infer a good position at which to grasp an object.

### 3.1 Grasping Point

For most objects, there is typically a small region that a human (using a two-fingered pinch grasp) would choose to grasp it; with some abuse of terminology, we will informally refer to this region as the “grasping point,” and our training set will contain labeled examples of this region. Examples of grasping points include the center region of the neck for a martini glass, the center region of the handle for a coffee mug, etc. (See Figure 3.)

For testing purposes, we would like to evaluate whether the robot successfully picks up an object. For each object in our test set, we define the successful grasp region to be the region where a human/robot using a two-fingered pinch grasp would (reasonably) be expected to successfully grasp the object. The error in predicting the grasping point (reported in Table 1) is then defined as the distance of the predicted point from the closest point lying in this region. (See Figure 4; this region is usually somewhat larger than that used in the training set, defined in the previous paragraph.) Since our gripper (Figure 1) has some passive compliance because of attached foam/rubber, and can thus tolerate about 0.5cm error in positioning, the successful grasping region may extend slightly past the surface of the object. (E.g., the radius of the cylinder in Figure 4 is about 0.5cm greater than the actual neck of the martini glass.)

### 3.2 Synthetic Data for Training

We apply supervised learning to identify patches that contain grasping points. To do so, we require a labeled training set, i.e., a set of images of objects labeled with the 2-d location of

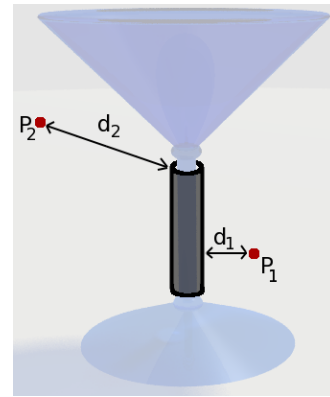


Figure 4: An illustration showing the grasp labels. The labeled grasp for a martini glass is on its neck (shown by a black cylinder). For two predicted grasping points  $P_1$  and  $P_2$ , the error would be the 3-d distance from the grasping region, i.e.,  $d_1$  and  $d_2$  respectively.

the grasping point in each image. Collecting real-world data of this sort is cumbersome, and manual labeling is prone to errors. Thus, we instead chose to generate, and learn from, synthetic data that is automatically labeled with the correct grasps.

In detail, we generate synthetic images along with correct grasps (Figure 3) using a computer graphics ray tracer.<sup>3</sup> There is a relation between the quality of the synthetically generated images and the accuracy of the algorithm. The better the quality of the synthetically generated images and graphical realism, the better the accuracy of the algorithm. Therefore, we used a ray tracer instead of faster, but cruder, OpenGL style graphics. [Michels *et al.*, 2005] used synthetic OpenGL images to learn distances in natural scenes. However, because OpenGL style graphics have less realism, their learning performance sometimes *decreased* with added graphical details in the rendered images.

The advantages of using synthetic images are multi-fold. First, once a synthetic model of an object has been created, a large number of training examples can be automatically generated by rendering the object under different (randomly chosen) lighting conditions, camera positions and orientations, etc. In addition, to increase the diversity of the training data generated, we randomized different properties of the objects such as color, scale, and text (e.g., on the face of a book). The time-consuming part of synthetic data generation was the creation of the mesh models of the objects. However, there are many objects for which models are available on the internet that can be used with only minor modifications. We generated 2500 examples from synthetic data, comprising objects from five object classes (see Figure 3). Using synthetic data also allows us to generate perfect labels for the training set with the exact location of a good grasp for each object. In contrast,

<sup>3</sup>Ray tracing [Glassner, 1989] is a standard image rendering method from computer graphics. It handles many real-world optical phenomenon such as multiple specular reflections, textures, soft shadows, smooth curves, and caustics. We used PovRay, an open source ray tracer.



Figure 5: Examples of different edge and texture filters (9 Laws’ masks and 6 oriented edge filters) used to calculate the features.

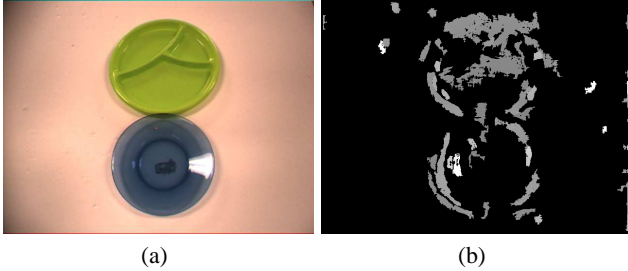


Figure 6: (a) An image of textureless/transparent/reflective objects. (b) Depths estimated by our stereo system. The grayscale value indicates the depth (darker being closer to the camera). Black represents areas where stereo vision failed to return a depth estimate.

collecting and manually labeling a comparably sized set of real images would have been extremely time-consuming.

We have made the data available online at:

<http://ai.stanford.edu/~asaxena/learninggrasp/data.html>

### 3.3 Features

In our approach, we begin by dividing the image into small rectangular patches, and for each patch predict if it contains a projection of a grasping point onto the image plane.

Instead of relying on a few visual cues such as presence of edges, we will compute a battery of features for each rectangular patch. By using a large number of different visual features and training on a huge training set (Section 3.2), we hope to obtain a method for predicting grasping points that is robust to changes in the appearance of the objects and is also able to generalize well to new objects.

We start by computing features for three types of local cues: edges, textures, and color. [Saxena *et al.*, 2007c; 2007a] We transform the image into YCbCr color space, where Y is the intensity channel, and Cb and Cr are color channels. We compute features representing edges by convolving the intensity channel with 6 oriented edge filters (Figure 5). Texture information is mostly contained within the image intensity channel, so we apply 9 Laws’ masks to this channel to compute the texture energy. For the color channels, low frequency information is most useful for identifying grasps; our color features are computed by applying a local averaging filter (the first Laws mask) to the 2 color channels. We then compute the sum-squared energy of each of these filter outputs. This gives us an initial feature vector of dimension 17.

To predict if a patch contains a grasping point, local image features centered on the patch are insufficient, and one has to use more global properties of the object. We attempt to capture this information by using image features extracted at multiple spatial scales (3 in our experiments) for the patch.

Objects exhibit different behaviors across different scales, and using multi-scale features allows us to capture these variations. In detail, we compute the 17 features described above from that patch as well as the 24 neighboring patches (in a 5x5 window centered around the patch of interest). This gives us a feature vector  $x$  of dimension  $1 * 17 * 3 + 24 * 17 = 459$ .

Although we rely mostly on image-based features for predicting the grasping point, some robots may be equipped with range sensors such as a laser scanner or a stereo camera. In these cases (Section 6.3), we also compute depth-based features to improve performance. More formally, we apply our texture based filters to the depth image obtained from a stereo camera, append them to the feature vector used in classification, and thus obtain a feature vector  $x_s \in \mathbb{R}^{918}$ . Applying these texture based filters this way has the effect of computing relative depths, and thus provides information about 3-d properties such as curvature. However, the depths given by a stereo system are sparse and noisy (Figure 6) because many objects we consider are textureless or reflective. Even after normalizing for the missing depth readings, these features improved performance only marginally.

### 3.4 Probabilistic Model

Using our image-based features, we will first predict whether each region in the image contains the projection of a grasping point. Then in order to grasp an object, we will statistically “triangulate” our 2-d predictions to obtain a 3-d grasping point.

In detail, on our manipulation platforms (Section 4), we have cameras mounted either on the wrist of the robotic arm (Figure 11) or on a frame behind the robotic arm (Figure 9). When the camera is mounted on the wrist, we command the arm to move the camera to two or more positions, so as to acquire images of the object from different viewpoints. However, there are inaccuracies in the physical positioning of the arm, and hence there is some slight uncertainty in the position of the camera when the images are acquired. We will now describe how we model these position errors.

Formally, let  $C$  be the image that would have been taken if the actual pose of the camera was exactly equal to the measured pose (e.g., if the robot had moved exactly to the commanded position and orientation, in the case of the camera being mounted on the robotic arm). However, due to positioning error, instead an image  $\hat{C}$  is taken from a slightly different location. Let  $(u, v)$  be a 2-d position in image  $C$ , and let  $(\hat{u}, \hat{v})$  be the corresponding image position in  $\hat{C}$ . Thus  $C(u, v) = \hat{C}(\hat{u}, \hat{v})$ , where  $C(u, v)$  is the pixel value at  $(u, v)$  in image  $C$ . The errors in camera position/pose should usually be small,<sup>4</sup> and we model the difference between  $(u, v)$  and  $(\hat{u}, \hat{v})$  using an additive Gaussian model:  $\hat{u} = u + \epsilon_u$ ,  $\hat{v} = v + \epsilon_v$ , where  $\epsilon_u, \epsilon_v \sim N(0, \sigma^2)$ .

<sup>4</sup>The robot position/orientation error is typically small (position is usually accurate to within 1mm), but it is still important to model this error. From our experiments (see Section 6), if we set  $\sigma^2 = 0$ , the triangulation is highly inaccurate, with average error in predicting the grasping point being 15.4 cm, as compared to 1.8 cm when appropriate  $\sigma^2$  is chosen.

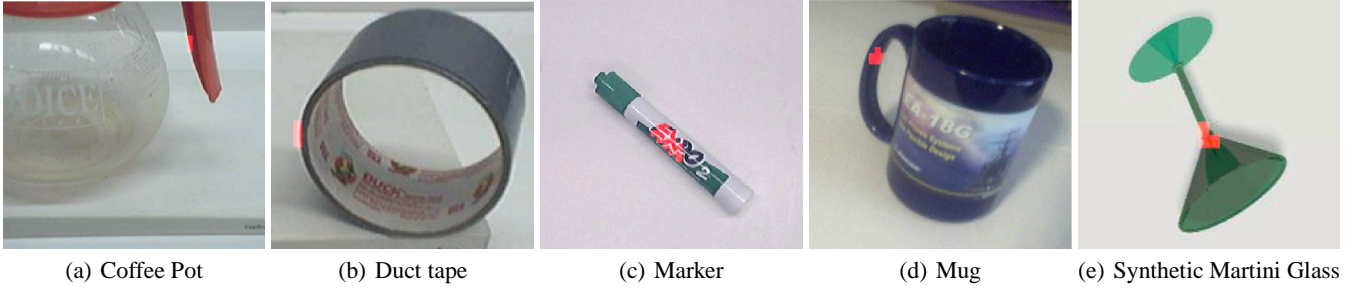


Figure 7: Grasping point classification. The red points in each image show the locations most likely to be a grasping point, as predicted by our logistic regression model. (Best viewed in color.)

Now, to predict which locations in the 2-d image are grasping points (Figure 7), we define the class label  $z(u, v)$  as follows. For each location  $(u, v)$  in an image  $C$ ,  $z(u, v) = 1$  if  $(u, v)$  is the projection of a grasping point onto the image plane, and  $z(u, v) = 0$  otherwise. For a corresponding location  $(\hat{u}, \hat{v})$  in image  $\hat{C}$ , we similarly define  $\hat{z}(\hat{u}, \hat{v})$  to indicate whether position  $(\hat{u}, \hat{v})$  represents a grasping point in the image  $\hat{C}$ . Since  $(u, v)$  and  $(\hat{u}, \hat{v})$  are corresponding pixels in  $C$  and  $\hat{C}$ , we assume  $\hat{z}(\hat{u}, \hat{v}) = z(u, v)$ . Thus:

$$P(z(u, v) = 1|C) = P(\hat{z}(\hat{u}, \hat{v}) = 1|\hat{C})$$

$$= \int_{\epsilon_u} \int_{\epsilon_v} P(\epsilon_u, \epsilon_v) P(\hat{z}(u + \epsilon_u, v + \epsilon_v) = 1|\hat{C}) d\epsilon_u d\epsilon_v \quad (1)$$

Here,  $P(\epsilon_u, \epsilon_v)$  is the (Gaussian) density over  $\epsilon_u$  and  $\epsilon_v$ . We then use logistic regression to model the probability of a 2-d position  $(u + \epsilon_u, v + \epsilon_v)$  in  $\hat{C}$  being a good grasping point:

$$P(\hat{z}(u + \epsilon_u, v + \epsilon_v) = 1|\hat{C}) = P(\hat{z}(u + \epsilon_u, v + \epsilon_v) = 1|x; \theta)$$

$$= 1/(1 + e^{-x^T \theta}) \quad (2)$$

where  $x \in \mathbb{R}^{459}$  are the features for the rectangular patch centered at  $(u + \epsilon_u, v + \epsilon_v)$  in image  $\hat{C}$  (described in Section 3.3). The parameter of this model  $\theta \in \mathbb{R}^{459}$  is learned using standard maximum likelihood for logistic regression:  $\theta^* = \arg \max_{\theta} \prod_i P(z_i|x_i; \theta)$ , where  $(x_i, z_i)$  are the synthetic training examples (image patches and labels), as described in Section 3.2. Figure 7a-d shows the result of applying the learned logistic regression model to some real (non-synthetic) images.

**3-d grasp model:** Given two or more images of a new object from different camera positions, we want to infer the 3-d position of the grasping point. (See Figure 8.) Because logistic regression may have predicted multiple grasping points per image, there is usually ambiguity in the correspondence problem (i.e., which grasping point in one image corresponds to which grasping point in another). To address this while also taking into account the uncertainty in camera position, we propose a probabilistic model over possible grasping points in 3-d space. In detail, we discretize the 3-d work-space of the robotic arm into a regular 3-d grid  $G \subset \mathbb{R}^3$ , and associate with each grid element  $j$  a random variable  $y_j$ , so that  $y_j = 1$  if grid cell  $j$  contains a grasping point, and  $y_j = 0$  otherwise.

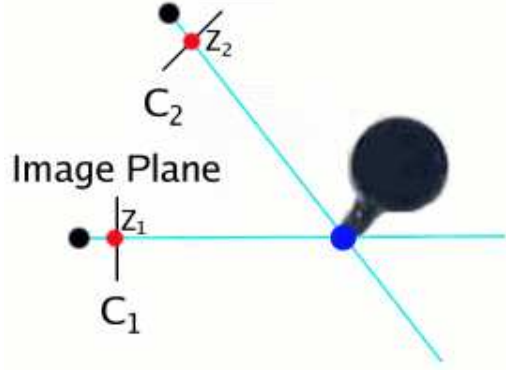


Figure 8: (a) Diagram illustrating rays from two images  $C_1$  and  $C_2$  intersecting at a grasping point (shown in dark blue). (Best viewed in color.)

From each camera location  $i = 1, \dots, N$ , one image is taken. In image  $C_i$ , let the ray passing through  $(u, v)$  be denoted  $R_i(u, v)$ . Let  $G_i(u, v) \subset G$  be the set of grid-cells through which the ray  $R_i(u, v)$  passes. Let  $r_1, \dots, r_K \in G_i(u, v)$  be the indices of the grid-cells lying on the ray  $R_i(u, v)$ .

We know that if any of the grid-cells  $r_j$  along the ray represent a grasping point, then its projection is a grasp point. More formally,  $z_i(u, v) = 1$  if and only if  $y_{r_1} = 1$  or  $y_{r_2} = 1$  or  $\dots$  or  $y_{r_K} = 1$ . For simplicity, we use a (arguably unrealistic) naive Bayes-like assumption of independence, and model the relation between  $P(z_i(u, v) = 1|C_i)$  and  $P(y_{r_1} = 1 \text{ or } \dots \text{ or } y_{r_K} = 1|C_i)$  as

$$P(z_i(u, v) = 0|C_i) = P(y_{r_1} = 0, \dots, y_{r_K} = 0|C_i)$$

$$= \prod_{j=1}^K P(y_{r_j} = 0|C_i) \quad (3)$$

Assuming that any grid-cell along a ray is equally likely to be a grasping point, this therefore gives

$$P(y_{r_j} = 1|C_i) = 1 - (1 - P(z_i(u, v) = 1|C_i))^{1/K} \quad (4)$$

Next, using another naive Bayes-like independence assumption, we estimate the probability of a particular grid-cell

$y_j \in G$  being a grasping point as:

$$\begin{aligned}
P(y_j = 1|C_1, \dots, C_N) &= \frac{P(y_j = 1)P(C_1, \dots, C_N|y_j = 1)}{P(C_1, \dots, C_N)} \\
&= \frac{P(y_j = 1)}{P(C_1, \dots, C_N)} \prod_{i=1}^N P(C_i|y_j = 1) \\
&= \frac{P(y_j = 1)}{P(C_1, \dots, C_N)} \prod_{i=1}^N \frac{P(y_j = 1|C_i)P(C_i)}{P(y_j = 1)} \\
&\propto \prod_{i=1}^N P(y_j = 1|C_i)
\end{aligned} \tag{5}$$

where  $P(y_j = 1)$  is the prior probability of a grid-cell being a grasping point (set to a constant value in our experiments). One can envision using this term to incorporate other available information, such as known height of the table when the robot is asked to pick up an object from a table. Using Equations 1, 2, 4 and 5, we can now compute (up to a constant of proportionality that does not depend on the grid-cell) the probability of any grid-cell  $y_j$  being a valid grasping point, given the images.<sup>5</sup>

**Stereo cameras:** Some robotic platforms have stereo cameras (e.g., the robot in Figure 10); therefore we also discuss how our probabilistic model can incorporate stereo images. From a stereo camera, since we also get a depth value  $w(u, v)$  for each location  $(u, v)$  in the image,<sup>6</sup> we now obtain a 3-d image  $C(u, v, w(u, v))$  for 3-d positions  $(u, v, w)$ .

However because of camera positioning errors (as discussed before), we get  $\hat{C}(\hat{u}, \hat{v}, \hat{w}(\hat{u}, \hat{v}))$  instead of actual image  $C(u, v, w(u, v))$ . We again model the difference between  $(u, v, w)$  and  $(\hat{u}, \hat{v}, \hat{w})$  using an additive Gaussian:  $\hat{u} = u + \epsilon_u, \hat{v} = v + \epsilon_v, \hat{w} = w + \epsilon_w$ , where  $\epsilon_u, \epsilon_v \sim N(0, \sigma_{uv}^2)$ ,  $\epsilon_w \sim N(0, \sigma_w^2)$ . Now, for our class label  $z(u, v, w)$  in the 3-d image, we have:

$$\begin{aligned}
P(z(u, v, w) = 1|C) &= P(\hat{z}(\hat{u}, \hat{v}, \hat{w}) = 1|\hat{C}) \\
&= \int_{\epsilon_u} \int_{\epsilon_v} \int_{\epsilon_w} P(\epsilon_u, \epsilon_v, \epsilon_w) \\
&\quad P(\hat{z}(u + \epsilon_u, v + \epsilon_v, w + \epsilon_w) = 1|\hat{C}) d\epsilon_u d\epsilon_v d\epsilon_w
\end{aligned} \tag{6}$$

<sup>5</sup>In about 2% of the trials described in Section 6.2, grasping failed because the algorithm found points in the images that did not actually correspond to each other. (E.g., in one image the point selected may correspond to the midpoint of a handle, and in a different image a different point may be selected that corresponds to a different part of the same handle.) Thus, triangulation using these points results in identifying a 3-d point that does not lie on the object. By ensuring that the pixel values in a small window around each of the corresponding points are similar, one would be able to reject some of these spurious correspondences.

<sup>6</sup>The depths estimated from a stereo camera are very sparse, i.e., the stereo system finds valid points only for a few pixels in the image. (see Figure 6) Therefore, we still mostly rely on image features to find the grasp points. The pixels where the stereo camera was unable to obtain a depth are treated as regular (2-d) image pixels.

Here,  $P(\epsilon_u, \epsilon_v, \epsilon_w)$  is the (Gaussian) density over  $\epsilon_u, \epsilon_v$  and  $\epsilon_w$ . Now our logistic regression model is

$$\begin{aligned}
P(\hat{z}(\hat{u}, \hat{v}, \hat{w}(\hat{u}, \hat{v})) = 1|\hat{C}) &= P(\hat{z}(\hat{u}, \hat{v}, \hat{w}(\hat{u}, \hat{v})) = 1|x_s; \theta_s) \\
&= 1/(1 + e^{-x_s^T \theta_s})
\end{aligned} \tag{7}$$

where  $x_s \in \mathbb{R}^{918}$  are the image and depth features for the rectangular patch centered at  $(\hat{u}, \hat{v})$  in image  $\hat{C}$  (described in Section 3.3). The parameter of this model  $\theta_s \in \mathbb{R}^{918}$  is learned similarly.

Now to use a stereo camera in estimating the 3-d grasping point, we use

$$P(y_j = 1|C_i) = P(z_i(u, v, w(u, v)) = 1|C_i) \tag{8}$$

in Eq. 5 in place of Eq. 4 when  $C_i$  represents a stereo camera image with depth information at  $(u, v)$ .

This framework allows predictions from both regular and stereo cameras to be used together seamlessly, and also allows predictions from stereo cameras to be useful even when the stereo system failed to recover depth information at the predicted grasp point.

### 3.5 MAP Inference

Given a set of images, we want to infer the most likely 3-d location of the grasping point. Therefore, we will choose the grid cell  $j$  in the 3-d robot workspace that maximizes the conditional log-likelihood  $\log P(y_j = 1|C_1, \dots, C_N)$  in Eq. 5. More formally, let there be  $N_C$  regular cameras and  $N - N_C$  stereo cameras. Now, from Eq. 4 and 8, we have:

$$\begin{aligned}
&\arg \max_j \log P(y_j = 1|C_1, \dots, C_N) \\
&= \arg \max_j \log \prod_{i=1}^N P(y_j = 1|C_i) \\
&= \arg \max_j \sum_{i=1}^{N_C} \log \left( 1 - (1 - P(z_i(u, v) = 1|C_i))^{1/K} \right) \\
&\quad + \sum_{i=N_C+1}^N \log (P(z_i(u, v, w(u, v)) = 1|C_i))
\end{aligned} \tag{9}$$

where  $P(z_i(u, v) = 1|C_i)$  is given by Eq. 1 and 2 and  $P(z_i(u, v, w(u, v)) = 1|C_i)$  is given by Eq. 6 and 7.

A straightforward implementation that explicitly computes the sum above for every single grid-cell would give good grasping performance, but be extremely inefficient (over 110 seconds). Since there are only a few places in an image where  $P(\hat{z}_i(u, v) = 1|C_i)$  is significantly greater than zero, we implemented a counting algorithm that explicitly considers only grid-cells  $y_j$  that are close to at least one ray  $R_i(u, v)$ . (Grid-cells that are more than  $3\sigma$  distance away from all rays are highly unlikely to be the grid-cell that maximizes the summation in Eq. 9.) This counting algorithm efficiently accumulates the sums over the grid-cells by iterating over all  $N$  images and rays  $R_i(u, v)$ ,<sup>7</sup> and results in an algorithm that

<sup>7</sup>In practice, we found that restricting attention to rays where  $P(\hat{z}_i(u, v) = 1|C_i) > 0.1$  allows us to further reduce the number of rays to be considered, with no noticeable degradation in performance.



identifies a 3-d grasping position in 1.2 sec.

## 4 Robot Platforms

Our experiments were performed on two robots built for the STAIR (STanford AI Robot) project.<sup>8</sup> Each robot has an arm and other equipment such as cameras, computers, etc. (See Fig. 9 and 10.) The STAIR platforms were built as part of a project whose long-term goal is to create a general purpose household robot that can navigate in indoor environments, pick up and interact with objects and tools, and carry out tasks such as tidy up a room or prepare simple meals. Our algorithms for grasping novel objects represent perhaps a small step towards achieving some of these goals.

STAIR 1 uses a harmonic arm (Katana, by Neuronics), and is built on top of a Segway robotic mobility platform. Its 5-dof arm is position-controlled and has a parallel plate gripper. The arm has a positioning accuracy of  $\pm 1$  mm, a reach of 62cm, and can support a payload of 500g. Our vision system used a low-quality webcam (Logitech Quickcam Pro 4000) mounted near the end effector and a stereo camera (Bumblebee, by Point Grey Research). In addition, the robot has a laser scanner (SICK LMS-291) mounted approximately 1m above the ground for navigation purposes. (We used the webcam in the experiments on grasping novel objects, and the Bumblebee stereo camera in the experiments on unloading items from dishwashers.) STAIR 2 sits atop a holonomic mobile base, and its 7-dof arm (WAM, by Barrett Technologies) can be position or torque-controlled, is equipped with a three-fingered hand, and has a positioning accuracy of  $\pm 0.6$  mm. It has a reach of 1m and can support a payload of 3kg. Its vision system uses a stereo camera (Bumblebee2, by Point Grey Research).

We used a distributed software framework called Switchyard [Quigley, 2007] to route messages between different devices such as the robotic arms, cameras and computers. Switchyard allows distributed computation using TCP message passing, and thus provides networking and synchronization across multiple processes on different hardware platforms.

## 5 Planning

After identifying a 3-d point at which to grasp an object, we need to find an arm pose that realizes the grasp, and then plan a path to reach that arm pose so as to pick up the object.

Given a grasping point, there are typically many end-effector orientations consistent with placing the center of the gripper at that point. The choice of end-effector orientation should also take into account other constraints, such as location of nearby obstacles, and orientation of the object.

**5-dof arm.** When planning in the absence of obstacles, we found that even fairly simple methods for planning worked well. Specifically, on our 5-dof arm, one of the degrees of freedom is the wrist rotation, which therefore does not affect planning to avoid obstacles. Thus, we can separately consider planning an obstacle-free path using the first 4-dof, and deciding the wrist rotation. To choose the wrist rotation, using a



Figure 9: STAIR 1 platform. This robot is equipped with a 5-dof arm and a parallel plate gripper.



Figure 10: STAIR 2 platform. This robot is equipped with 7-dof Barrett arm and three-fingered hand.

<sup>8</sup>See <http://www.cs.stanford.edu/group/stair> for details.





Figure 11: The robotic arm picking up various objects: screwdriver, box, tape-roll, wine glass, a solder tool holder, coffee pot, powerhorn, cellphone, book, stapler and coffee mug. (See Section 6.2.)

simplified version of our algorithm in [Saxena *et al.*, 2007b], we learned the 2-d value of the 3-d grasp orientation projected onto the image plane (see Appendix). Thus, for example, if the robot is grasping a long cylindrical object, it should rotate the wrist so that the parallel-plate gripper’s inner surfaces are parallel (rather than perpendicular) to the main axis of the cylinder. Further, we found that using simple heuristics to decide the remaining degrees of freedom worked well.<sup>9</sup>

When grasping in the presence of obstacles, such as when unloading items from a dishwasher, we used a full motion planning algorithm for the 5-dof as well as for the opening of the gripper (a 6th degree of freedom). Specifically, having

<sup>9</sup>Four degrees of freedom are already constrained by the end-effector 3-d position and the chosen wrist angle. To decide the fifth degree of freedom in uncluttered environments, we found that most grasps reachable by our 5-dof arm fall in one of two classes: *downward* grasps and *outward* grasps. These arise as a direct consequence of the shape of the workspace of our 5 dof robotic arm (Figure 11). A “downward” grasp is used for objects that are close to the base of the arm, which the arm will grasp by reaching in a downward direction (Figure 11, first image), and an “outward” grasp is for objects further away from the base, for which the arm is unable to reach in a downward direction (Figure 11, second image). In practice, to simplify planning we first plan a path towards an approach position, which is set to be a fixed distance away from the predicted grasp point towards the base of the robot arm. Then we move the end-effector in a straight line forward towards the target grasping point. Our grasping experiments in uncluttered environments (Section 6.2) were performed using this heuristic.

identified possible goal positions in configuration space using standard inverse kinematics [Mason and Salisbury, 1985], we plan a path in 6-dof configuration space that takes the end-effector from the starting position to a goal position, avoiding obstacles. For computing the goal orientation of the end-effector and the configuration of the fingers, we used a criterion that attempts to minimize the opening of the hand without touching the object being grasped or other nearby obstacles. Our planner uses Probabilistic Road-Maps (PRMs) [Schwarzer *et al.*, 2005], which start by randomly sampling points in the configuration space. It then constructs a “road map” by finding collision-free paths between nearby points, and finally finds a shortest path from the starting position to possible target positions in this graph. We also experimented with a potential field planner [Khatib, 1986], but found the PRM method gave better results because it did not get stuck in local optima.

**7-dof arm.** On the STAIR 2 robot, which uses a 7-dof arm, we use the full algorithm in [Saxena *et al.*, 2007b], for predicting the 3-d orientation of a grasp, given an image of an object. This, along with our algorithm to predict the 3-d grasping point, determines six of the seven degrees of freedom (i.e., the end-effector location and orientation). For deciding the seventh degree of freedom, we use a criterion that maximizes the distance of the arm from the obstacles. Similar to the planning on the 5-dof arm, we then apply a PRM planner to plan a path in the 7-dof configuration space.

Table 1: Mean absolute error in locating the grasping point for different objects, as well as grasp success rate for picking up the different objects using our robotic arm. (Although training was done on synthetic images, testing was done on the real robotic arm and real objects.)

OBJECTS SIMILAR TO ONES TRAINED ON			NOVEL OBJECTS		
TESTED ON	MEAN ABSOLUTE ERROR (CM)	GRASP-SUCCESS RATE	TESTED ON	MEAN ABSOLUTE ERROR (CM)	GRASP-SUCCESS RATE
MUGS	2.4	75%	STAPLER	1.9	90%
PENS	0.9	100%	DUCT TAPE	1.8	100%
WINE GLASS	1.2	100%	KEYS	1.0	100%
BOOKS	2.9	75%	MARKERS/SCREWDRIVER	1.1	100%
ERASER/CELLPHONE	1.6	100%	TOOTHBRUSH/CUTTER	1.1	100%
OVERALL	1.80	90.0%	JUG	1.7	75%
			TRANSLUCENT BOX	3.1	75%
			POWERHORN	3.6	50%
			COILED WIRE	1.4	100%
			OVERALL	1.86	87.8%

## 6 Experiments

### 6.1 Experiment 1: Synthetic data

We first evaluated the predictive accuracy of the algorithm on synthetic images (not contained in the training set). (See Figure 7e.) The average accuracy for classifying whether a 2-d image patch is a projection of a grasping point was 94.2% (evaluated on a balanced test set comprised of the five objects in Figure 3). Even though the accuracy in classifying 2-d regions as grasping points was only 94.2%, the accuracy in predicting 3-d grasping points was higher because the probabilistic model for inferring a 3-d grasping point automatically aggregates data from multiple images, and therefore “fixes” some of the errors from individual classifiers.

### 6.2 Experiment 2: Grasping novel objects

We tested our algorithm on STAIR 1 (5-dof robotic arm, with a parallel plate gripper) on the task of picking up an object placed on an uncluttered table top in front of the robot. The location of the object was chosen randomly (and we used cardboard boxes to change the height of the object, see Figure 11), and was completely unknown to the robot. The orientation of the object was also chosen randomly from the set of orientations in which the object would be stable, e.g., a wine glass could be placed vertically up, vertically down, or in a random 2-d orientation on the table surface (see Figure 11). (Since the training was performed on synthetic images of objects of different types, none of these scenarios were in the training set.)

In these experiments, we used a web-camera, mounted on the wrist of the robot, to take images from two or more locations. Recall that the parameters of the vision algorithm were trained from synthetic images of a small set of five object classes, namely books, martini glasses, white-board erasers, mugs/cups, and pencils. We performed experiments on coffee mugs, wine glasses (empty or partially filled with water), pencils, books, and erasers—but all of different dimensions and appearance than the ones in the training set—as well as a large set of objects from novel object classes, such as rolls of duct tape, markers, a translucent box, jugs, knife-cutters, cellphones, pens, keys, screwdrivers, staplers, toothbrushes, a thick coil of wire, a strangely shaped power horn,

etc. (See Figures 2 and 11.) We note that many of these objects are translucent, textureless, and/or reflective, making 3-d reconstruction difficult for standard stereo systems. (Indeed, a carefully-calibrated Point Gray stereo system, the Bumblebee BB-COL-20,—with higher quality cameras than our web-camera—fails to accurately reconstruct the visible portions of 9 out of 12 objects. See Figure 6.)

In extensive experiments, the algorithm for predicting grasps in images appeared to generalize very well. Despite being tested on images of real (rather than synthetic) objects, including many very different from ones in the training set, it was usually able to identify correct grasp points. We note that test set error (in terms of average absolute error in the predicted position of the grasp point) on the real images was only somewhat higher than the error on synthetic images; this shows that the algorithm trained on synthetic images transfers well to real images. (Over all 5 object types used in the synthetic data, average absolute error was 0.81cm in the synthetic images; and over all the 14 real test objects, average error was 1.84cm.) For comparison, neonate humans can grasp simple objects with an average accuracy of 1.5cm. [Bower *et al.*, 1970]

Table 1 shows the errors in the predicted grasping points on the test set. The table presents results separately for objects which were similar to those we trained on (e.g., coffee mugs) and those which were very dissimilar to the training objects (e.g., duct tape). For each entry in the table, a total of four trials were conducted except for staplers, for which ten trials were conducted. In addition to reporting errors in grasp positions, we also report the grasp success rate, i.e., the fraction of times the robotic arm was able to physically pick up the object. For a grasp to be counted as successful, the robot had to grasp the object, lift it up by about 1ft, and hold it for 30 seconds. On average, the robot picked up the novel objects 87.8% of the time.

For simple objects such as cellphones, wine glasses, keys, toothbrushes, etc., the algorithm performed perfectly in our experiments (100% grasp success rate). However, grasping objects such as mugs or jugs (by the handle) allows only a narrow trajectory of approach—where one “finger” is inserted into the handle—so that even a small error in the grasping point identification causes the arm to hit and move the ob-

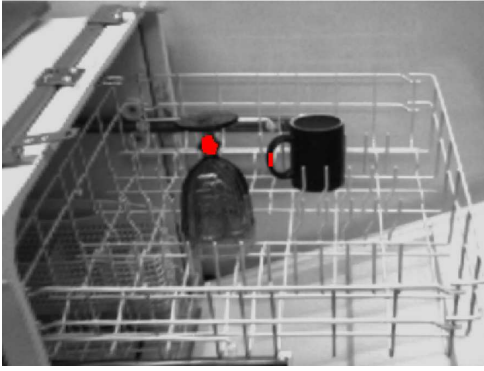


Figure 12: Example of a real dishwasher image, used for training in the dishwasher experiments.

ject, resulting in a failed grasp attempt. Although it may be possible to improve the algorithm’s accuracy, we believe that these problems can best be solved by using a more advanced robotic arm that is capable of haptic (touch) feedback.

In many instances, the algorithm was able to pick up completely novel objects (a strangely shaped power-horn, duct-tape, solder tool holder, etc.; see Figures 2 and 11). Perceiving a transparent wine glass is a difficult problem for standard vision (e.g., stereopsis) algorithms because of reflections, etc. However, as shown in Table 1, our algorithm successfully picked it up 100% of the time. Videos showing the robot grasping the objects are available at

<http://ai.stanford.edu/~asaxena/learninggrasp/>

### 6.3 Experiment 3: Unloading items from dishwashers

The goal of the STAIR project is to build a general purpose household robot. As a step towards one of STAIR’s envisioned applications, in this experiment we considered the task of unloading items from dishwashers (Figures 1 and 14). This is a difficult problem because of the presence of background clutter and occlusion between objects—one object that we are trying to unload may physically block our view of a second object. Our training set for these experiments also included some hand-labeled real examples of dishwasher images (Figure 12), including some images containing occluded objects; this helps prevent the algorithm from identifying grasping points on the background clutter such as dishwasher prongs. Along with the usual features, these experiments also used the depth-based features computed from the depth image obtained from the stereo camera (Section 3.3). Further, in these experiments we did not use color information, i.e., the images fed to the algorithm were grayscale.

In detail, we asked a person to arrange several objects neatly (meaning inserted over or between the dishwasher prongs, and with no pair of objects lying flush against each other; Figures 10 and 11 show typical examples) in the upper tray of the dishwasher. To unload the items from the dishwasher, the robot first identifies grasping points in the image. Figure 13 shows our algorithm correctly identifying grasps



Figure 13: Grasping point detection for objects in a dishwasher. (Only the points in top five grasping regions are shown.)

Table 2: Grasp-success rate for unloading items from a dishwasher.

TESTED ON	GRASP-SUCCESS-RATE
PLATES	100%
BOWLS	80%
MUGS	60%
WINE GLASS	80%
OVERALL	80%

on multiple objects even in the presence of clutter and occlusion. The robot then uses these grasps and the locations of the obstacles (perceived using a stereo camera) to plan a path while avoiding obstacles, to pick up the object. When planning a path to a grasping point, the robot chooses the grasping point that is most accessible to the robotic arm using a criterion based on the grasping point’s height, distance to the robot arm, and distance from obstacles. The robotic arm then removes the first object (Figure 14) by lifting it up by about 1ft and placing it on a surface on its right using a pre-written script, and repeats the process above to unload the next item. As objects are removed, the visual scene also changes, and the algorithm will find grasping points on objects that it had missed earlier.

We evaluated the algorithm quantitatively for four object classes: plates, bowls, mugs and wine glasses. (We have successfully unloaded items from multiple dishwashers; however we performed quantitative experiments only on one dishwasher.) We performed five trials for each object class (each trial used a different object). We achieved an average grasping success rate of 80.0% in a total of 20 trials (see Table 2). Our algorithm was able to successfully pick up objects such as plates and wine glasses most of the time. However, due to the physical limitations of the 5-dof arm with a parallel plate gripper, it is not possible for the arm to pick up certain objects, such as bowls, if they are in certain configurations (see Figure 15). For mugs, the grasp success rate was low because of the problem of narrow trajectories discussed in Section 6.2.

We also performed tests using silverware, specifically objects such as spoons and forks. These objects were often placed (by the human dishwasher loader) against the corners or walls of the silverware rack. The handles of spoons and





Figure 14: Dishwasher experiments (Section 6.3): Our robotic arm unloads items from a dishwasher.

forks are only about 0.5cm thick; therefore a larger clearance than 0.5cm was needed for them to be grasped using our parallel plate gripper, making it physically extremely difficult to do so. However, if we arrange the spoons and forks with part of the spoon or fork at least 2cm away from the walls of the silverware rack, then we achieve a grasping success rate of about 75%.

Some of the failures were because some parts of the object were not perceived; therefore the arm would hit and move the object resulting in a failed grasp. In such cases, we believe an algorithm that uses haptic (touch) feedback would significantly increase grasp success rate. Some of our failures were also in cases where our algorithm correctly predicts a grasping point, but the arm was physically unable to reach that grasp. Therefore, we believe that using an arm/hand with more degrees of freedom, will significantly improve performance for the problem of unloading a dishwasher.



Figure 15: Dishwasher experiments: Failure case. For some configurations of certain objects, it is impossible for our 5-dof robotic arm to grasp it (even if a human were controlling the arm).

#### 6.4 Experiment 4: Grasping kitchen and office objects

Our long-term goal is to create a useful household robot that can perform many different tasks, such as fetching an object

in response to a verbal request and cooking simple kitchen meals. In situations such as these, the robot would know which object it has to pick up. For example, if the robot was asked to fetch a stapler from an office, then it would know that it needs to identify grasping points for staplers only. Therefore, in this experiment we study how we can use information about object type and location to improve the performance of the grasping algorithm.

Consider objects lying against a cluttered background such as a kitchen or an office. If we predict the grasping points using our algorithm trained on a dataset containing all five objects, then we typically obtain a set of reasonable grasping point predictions (Figure 16, left column). Now suppose we know the type of object we want to grasp, as well as its approximate location in the scene (such as from an object recognition algorithm [Gould *et al.*, 2007]). We can then restrict our attention to the area of the image containing the object, and apply a version of the algorithm that has been trained using only objects of a similar type (i.e., using object-type specific parameters, such as using bowl-specific parameters when picking up a cereal bowl, using spoon-specific parameters when picking up a spoon, etc). With this method, we obtain object-specific grasps, as shown in Figure 16 (right column).

Achieving larger goals, such as cooking simple kitchen meals, requires that we combine different algorithms such as object recognition, navigation, robot manipulation, etc. These results demonstrate how our approach could be used in conjunction with other complementary algorithms to accomplish these goals.

#### 6.5 Experiment 5: Grasping using 7-dof arm and three-fingered hand.

In this experiment, we demonstrate that our grasping point prediction algorithm can also be used with other robotic manipulation platforms. We performed experiments on the STAIR 2 robot, which is equipped with a 7-dof arm and a three-fingered Barrett hand. This is a more capable manipulator than a parallel plate gripper, in that its fingers can form a large variety of configurations; however, in this experiment we will use the hand only in a limited way, specifically, a configuration with the two fingers opposing the third one, with all fingers closing simultaneously. While there is a large space of hand configurations that one would have to consider in order to fully take advantage of the capabilities of such a hand, identifying a point at which to grasp the object still remains an important aspect of the problem, and is the focus of the

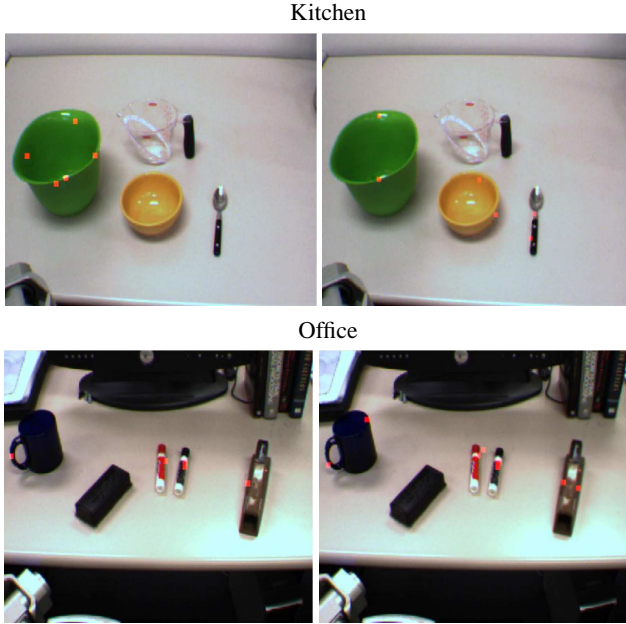


Figure 16: Grasping point classification in kitchen and office scenarios: (Left Column) Top five grasps predicted by the grasp classifier alone. (Right Column) Top two grasps for three different object-types, predicted by the grasp classifier when given the object types and locations. The red points in each image are the predicted grasp locations. (Best viewed in color.)

present experiment.

In particular, we asked a person to place several objects in front of the STAIR 2 robot. The bowls were placed upright at a random location on a table (with height unknown to the robot), and the plates were stacked neatly in a rack (also in a random location). Using our algorithm trained on a dataset containing the five synthetic objects described in Section 3.2, the robot chose the best grasp predicted from the image, and attempted to pick up the object (Figure 17). It achieved a grasping success rate of 60% for cereal bowls, and 80% for plates (5 trials for each object).

## 7 Conclusions

We proposed an algorithm for enabling a robot to grasp an object that it has never seen before. Our learning algorithm neither tries to build, nor requires, a 3-d model of the object. Instead it predicts, directly as a function of the images, a point at which to grasp the object. In our experiments, the algorithm generalizes very well to novel objects and environments, and our robot successfully grasped a wide variety of objects in different environments such as dishwashers, office and kitchen.

The ability to pick up novel objects represents perhaps a tiny first step towards the STAIR project’s larger goal of enabling robots to perform a large variety of household tasks, such as fetching an item in response to a verbal request, tidying up a room, and preparing simple meals in a kitchen. In the short term, we are working on applying variations of the



Figure 17: Barrett arm grasping an object using our algorithm.

algorithms described in this paper to try to enable STAIR to prepare simple meals using a normal home kitchen.

## Acknowledgments

We give warm thanks to Morgan Quigley for help with the robotic hardware and communication software. We also thank Justin Kearns, Lawson Wong, David Ho, Chioma Osundu, and Brad Gulko for help in running the experiments. This work was supported by the DARPA transfer learning program under contract number FA8750-05-2-0249, and by the National Science Foundation under award number CNS-0551737.

## Appendix: Predicting orientation

In [Saxena *et al.*, 2007b], we presented an algorithm for predicting the 3-d orientation of an object from its image. Here, for the sake of simplicity, we will present the learning algorithm in the context of grasping using our 5-dof arm on STAIR 1.

As discussed in Section 5, our task is to predict the 2-d wrist orientation  $\theta$  of the gripper (at the predicted grasping point) given the image. For example, given a picture of a closed book (which we would like to grasp at its edge), we should choose an orientation in which the robot’s two fingers are parallel to the book’s surfaces, rather than perpendicular to the book’s cover.

Since our robotic arm has a parallel plate gripper comprising two fingers that close in parallel, a rotation of  $\pi$  results in similar configuration of the gripper. This results in a discontinuity at  $\theta = \pi$ , in that the orientation of the gripper at  $\theta = \pi$  is equivalent to  $\theta = 0$ . Therefore, to handle this symmetry, we will represent angles via  $y(\theta) = [\cos(2\theta), \sin(2\theta)] \in \mathbb{R}^2$ . Thus,  $y(\theta + N\pi) = [\cos(2\theta + 2N\pi), \sin(2\theta + 2N\pi)] = y(\theta)$ .

Now, given images features  $x$ , we model the conditional distribution of  $y$  as a multi-variate Gaussian:

$$P(y|x; w, K) = (2\pi)^{-n/2} |K|^{1/2} \exp \left[ -\frac{1}{2} (y - w^T x)^T K (y - w^T x) \right]$$

where  $K^{-1}$  is a covariance matrix. The parameters of this model  $w$  and  $K$  are learnt by maximizing the conditional log likelihood  $\log \prod_i P(y_i|x_i; w)$ .

Now when given a new image, our MAP estimate for  $y$  is given as follows. Since  $\|y\|_2 = 1$ , we will choose

$$y^* = \arg \max_{y: \|y\|_2=1} \log P(y|x; w, K) = \arg \max_{y: \|y\|_2=1} y^T K q$$

for  $q = w^T x$ . (This derivation assumed  $K = \sigma^2 I$  for some  $\sigma^2$ , which will roughly hold true if  $\theta$  is chosen uniformly in the training set.) The closed form solution of this is  $y = Kq / \|Kq\|_2$ .

In our robotic experiments, typically  $30^\circ$  accuracy is required to successfully grasp an object, which our algorithm almost always attains. In an example, Figure 18 shows the predicted orientation for a pen.

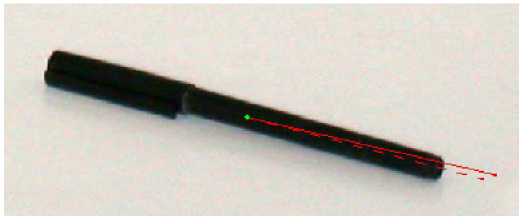


Figure 18: Predicted orientation at the grasping point for pen. Dotted line represents the true orientation, and solid line represents the predicted orientation.

## References

- [Bicchi and Kumar, 2000] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *International Conference on Robotics and Automation (ICRA)*, 2000.
- [Bower et al., 1970] T. G. R. Bower, J. M. Broughton, and M. K. Moore. Demonstration of intention in the reaching behaviour of neonate humans. *Nature*, 228:679–681, 1970.
- [Bowers and Lumia, 2003] D.L. Bowers and R. Lumia. Manipulation of unmodeled objects using intelligent grasping schemes. *IEEE Transactions on Fuzzy Systems*, 11(3), 2003.
- [Castiello, 2005] U. Castiello. The neuroscience of grasping. *Nature Reviews Neuroscience*, 6:726–736, 2005.
- [Coelho et al., 2001] J. Coelho, J. Piater, and R. Grupen. Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems*, 37:195–218, 2001.
- [Glassner, 1989] A. S. Glassner. *An Introduction to Ray Tracing*. Morgan Kaufmann Publishers, Inc., San Francisco, 1989.
- [Goodale et al., 1991] M. A. Goodale, A. D. Milner, L. S. Jakobson, and D. P. Carey. A neurological dissociation between perceiving objects and grasping them. *Nature*, 349:154–156, 1991.
- [Gould et al., 2007] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Meissner, G. Bradski, P. Baumstarck, S. Chung, and A. Y. Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.
- [Hsiao and Lozano-Perez, 2006] K. Hsiao and T. Lozano-Perez. Imitation learning of whole-body grasps. In *IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)*, 2006.
- [Hsiao et al., 2007] K. Hsiao, L. Kaelbling, and T. Lozano-Perez. Grasping POMDPs. In *International Conference on Robotics and Automation (ICRA)*, 2007.
- [Hueser et al., 2006] M. Hueser, T. Baier, and J. Zhang. Learning of demonstrated grasping skills by stereoscopic tracking of human hand configuration. In *International Conference on Robotics and Automation (ICRA)*, 2006.
- [Kamon et al., 1996] I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *International Conference on Robotics and Automation (ICRA)*, 1996.
- [Khatib, 1986] O. Khatib. The potential field approach and operational space formulation in robot control. *Adaptive and Learning Systems: Theory and Applications*, pages 367–377, 1986.
- [Kragic and Christensen, 2003] D. Kragic and H. I. Christensen. Robust visual servoing. *International Journal of Robotics Research (IJRR)*, 22:923–939, 2003.
- [Marotta et al., 2004] J. J. Marotta, T. S. Perrot, D. Nicolle3, P. Servos, and M. A. Goodale. Adapting to monocular vision: grasping with one eye. 104:107–114, 2004.
- [Mason and Salisbury, 1985] M. T. Mason and J. K. Salisbury. Manipulator grasping and pushing operations. In *Robot Hands and the Mechanics of Manipulation*. The MIT Press, Cambridge, MA, 1985.
- [Michels et al., 2005] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2005.
- [Miller et al., 2003] A. T. Miller, S. Knoop, P. K. Allen, and H. I. Christensen. Automatic grasp planning using shape primitives. In *International Conference on Robotics and Automation (ICRA)*, 2003.
- [Morales et al., 2002a] A. Morales, P. J. Sanz, and A. P. del Pobil. Vision-based computation of three-finger grasps on unknown planar objects. In *IEEE/RSJ Intelligent Robots and Systems Conference*, 2002.
- [Morales et al., 2002b] A. Morales, P. J. Sanz, A. P. del Pobil, and A. H. Fagg. An experiment in constraining vision-based finger contact selection with gripper geometry. In *IEEE/RSJ Intelligent Robots and Systems Conference*, 2002.
- [Pelossof et al., 2004] R. Pelossof, A. Miller, P. Allen, and T. Jebara. An svm learning approach to robotic grasping. In *International Conference on Robotics and Automation (ICRA)*, 2004.
- [Petrovskaya et al., 2006] A. Petrovskaya, O. Khatib, S. Thrun, and A. Y. Ng. Bayesian estimation for autonomous object manipulation based on tactile sensors. In *International Conference on Robotics and Automation*, 2006.
- [Piater, 2002] J. H. Piater. Learning visual features to predict hand orientations. In *ICML Workshop on Machine Learning of Spatial Knowledge*, 2002.
- [Platt et al., 2005] R. Platt, A. H. Fagg, and R. Grupen. Reusing schematic grasping policies. In *IEEE-RAS International Conference on Humanoid Robots, Tsukuba, Japan*, 2005.
- [Quigley, 2007] M. Quigley. Switchyard. Technical report, Stanford University, <http://ai.stanford.edu/~mquigley/switchyard>, 2007.



- [Saxena *et al.*, 2005] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Neural Information Processing Systems (NIPS) 18*, 2005.
- [Saxena *et al.*, 2006a] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng. Robotic grasping of novel objects. In *Neural Information Processing Systems (NIPS) 19*, 2006.
- [Saxena *et al.*, 2006b] A. Saxena, J. Driemeyer, J. Kearns, C. Osundu, and A. Y. Ng. Learning to grasp novel objects using vision. In *10th International Symposium of Experimental Robotics (ISER)*, 2006.
- [Saxena *et al.*, 2007a] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision (IJCV)*, 2007.
- [Saxena *et al.*, 2007b] A. Saxena, J. Driemeyer, and A. Y. Ng. Learning 3-d object orientation from images. Unpublished manuscript, 2007.
- [Saxena *et al.*, 2007c] A. Saxena, J. Schulte, and A. Y. Ng. Depth estimation using monocular and stereo cues. In *20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.
- [Saxena *et al.*, 2007d] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Learning 3-d scene structure from a single still image. In *ICCV workshop on 3D Representation for Recognition (3dRR-07)*, 2007.
- [Schneiderman and Kanade, 1998] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 1998.
- [Schwarzer *et al.*, 2005] F. Schwarzer, M. Saha, and J.-C. Latombe. Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. *IEEE Transactions on Robotics and Automation*, 21:338–353, 2005.
- [Shimoga, 1996] K. Shimoga. Robot grasp synthesis: a survey. *International Journal of Robotics Research (IJRR)*, 15:230–266, 1996.
- [Shin-ichi and Satoshi, 2000] T. Shin-ichi and M. Satoshi. Living and working with robots. *Nipponia*, 2000.