Learning to grasp objects with multiple contact points

Quoc V. Le, David Kamm, Arda F. Kara, Andrew Y. Ng

***Abstract*— We consider the problem of grasping novel objects and its application to cleaning a desk. A recent successful approach applies machine learning to learn one grasp point in an image and a point cloud. Although those methods are able to generalize to novel objects, they yield suboptimal results because they rely on motion planner for finger placements. In this paper, we extend their method to accommodate grasps with multiple contacts. This approach works well for many human-made objects because it models the way we grasp objects. To further improve the grasping, we also use a method that learns the ranking between candidates. The experiments show that our method is highly effective compared to a state-of-the-art competitor.**

I. Introduction

We consider the problem of robots cleaning a desk by grasping. This task poses many challenges: the variation in shapes and orientations of objects, the lack of complete 3D information, the occlusions between objects.

Recent approaches [1], [2] apply machine learning techniques to solve this problem. Their method generalizes well for novel objects because they do not rely on brittle hard-coded rules. The main shortcoming is that their system is built on top of a *pinch* grasp classifier [1] and relies on motion planner for finger placements. The method [2] extends pinch grasps in [1] to model power grasps. Although it tries to learn the position of the center of the end effector and the direction of the wrist, motion planning plays an important role for finger placements. Motion planners, unfortunately, only detect collisions and do not consider force stability.

In fact, when performing experiments with their methods, we notice that for many objects, pinch grasps [1] are not applicable. This is because human-made objects are designed to be reliably grasped with multiple contacts (two or more, using the thumb and other fingers).[1] Power grasps [2] are better but sometimes objects slip off the robot hand. This is because many configurations, despite being collision-free, are not stable for grasping.

We will illustrate the latter observation with a simplistic example (see Figure 1). Suppose we would like the robot to pick up a mug. The classifier by [2] will return the location at the red square as a good candidate because depth and visible light images have discontinuities at that location (Figure 1, left). Their method will use this candidate and query a motion planner for a good grasp configuration. Some grasp configurations, despite having no collision with objects, are not stable. For example, if the two fingers are placed at two locations specified by the blue circles, the motion planner does not detect collision, but the mug may slip out of the robot hand during grasping.

Ideally and hypothetically, their algorithm should return a place near the centroid of the object as a candidate and then, with high probability, the motion planner will place the fingers at the appropriate locations (see Figure 1, right). However, in this case, the location at the red square is rather indistinguishable in the depth and visible light images; and it is difficult to build a pattern recognizer to identify it.

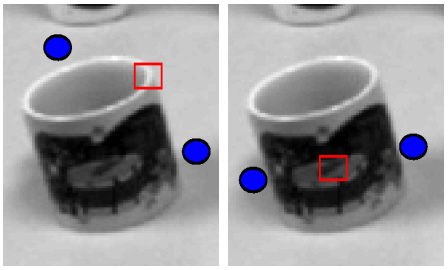

Fig. 1. Problems with the method in [2]. Left: best candidate returned by their classifier (red square) and finger locations (blue circles) that have no collision. These finger placements are not stable for grasping. Right: ideal finger placements (blue circles) and corresponding hypothesized candidate (red square). The red square, however, does not contain any useful information and thus cannot be found by the method in [2].

In this paper, we present a method that learns the contact points, i.e., the locations where the blue circles touch the objects. This method disambiguates the problem of fingertip placements and chooses the placements that are most stable. Our method uses a motion planner only for planning a trajectory that has no collisions. It also allows grasping objects at multiple contacts with pinch grasp as a special case. We will show that meaningful features can be extracted from these contact points. We also use supervised learning, or more specifically Support Vector Machines, to learn grasping concepts that generalize well for novel objects. Our experiments show that despite its simplicity, this idea significantly improves grasping accuracy.

With multiple contacts, most degrees of freedom (finger closure size and orientation) are fixed. Motion planning is used simply to avoid collisions and the number of configurations that the motion planning needs to search is quite small. This provides a big computation boost and starkly contrasts the approach in [2] where a motion planner is extensively used to search over 4 parameters (pitch, roll, yaw, finger closure size).

Quoc V. Le, David Kamm, Arda F. Kara, Andrew Y. Ng are with the Computer Science Department, Stanford University quocle@cs.stanford.edu, dkamm@stanford.edu, ardakara@stanford.edu, ang@cs.stanford.edu

[1]There are certainly cases that pinch grasps are preferred, such as grasping a pen. However, our robot hand (Barrett) cannot grasp such objects very well.

II. Previous Work

Early work in robot grasping assumed complete 2D or 3D models of objects. Under such assumptions, many types of grasps can be modelled and computed, for example force closure [3], [4], [5], form-closure [6], equilibrium grasps [7], [8], [9], stable grasps [10], compliant grasps [8]. More recent approaches use machine learning to combine more information for better grasping using SVMs [11] or for better controlling using reinforcement learning [12], [13].

The main drawback of those methods is that it is hard to extend them to real-world data where capturing complete 3D models for objects is difficult. For example, given a static scene, the back face of objects cannot be captured using a stereo camera. This realization leads further developments in robotic grasping where methods have to consider more realistic sensory data, for example, intensity images, point clouds, haptic feedbacks. With such sensory data, researchers have to take into account sensory noises and partial shapes.

Using local visual features, methods are proposed to find *planar* grasps, i.e., looking for 2D locations where the robot can place its fingertips [14], [15], [16], [17]. For non-planar grasps, a schema structured method is presented to deal with simple objects [18]. Also with schema structured learning, Platt et al. [19] proposes a method that assumes segmentation and fits ellipsoids to objects. Edsinger and Kemp [20] designed an algorithm to grasp cylindrical objects with power grasp using visual servoing.

Our method resembles methods proposed by Saxena et al. [21], and more specifically [2]. The early work [22] only considers intensity image data to learn the features. Later methods [2] include depth information and design a multi-stage processing which combines the method in [22] and motion planning. In detail, they use a single-point classifier to get a large number of candidates for grasping [23]. Then they apply a second classifier which combines geometric features to rank grasp configurations. These configurations are fed to a motion planner to determine the final grasp.

As mentioned earlier, there are two main issues with that approach. The first issue is accuracy and stability of grasps. As the method in [2] is divided into several stages, the errors get bigger after each stage. We also notice that because their method is based on a single-point classifier (one contact point) at the lowest level, the grasps are usually not stable. The second issue is due to computation, or more specifically, an extensive use of motion planning. This is because classifiers in early stages have a very low Area under the ROC score, the motion planner has to handle the chore of finding a good configuration for a large number of points.

III. A brief summary of our system

Our system makes use of both depth and visible light image data. To capture depth data, we used an active triangulation sensor [24] (see Figure 2). An important feature of this sensor is that it gives very detailed depth data (also called depth map or point cloud). However, the key ideas in this paper can be applied to depth data captured by other types of sensors such as stereo cameras.

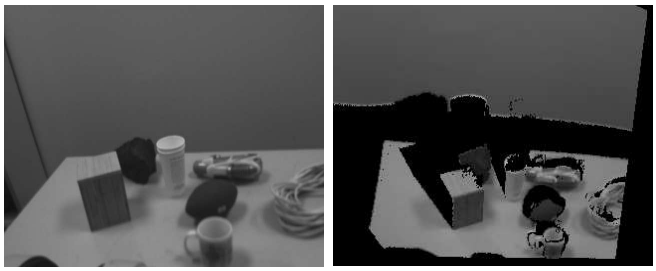

Fig. 2. Image data and depth data captured by our robot. Left: visible light image data. Right: depth data (note: missing back face of objects).

Given the depth map and intensity image data, our system has two main steps: ranking candidates and planning motion. Using the data, the robot first generates all grasp candidates for the scene.[2] Our system then extracts relevant features for each candidate. Based on the features, the robot uses a trained SVM model to compute a "score" for each candidate and rank the scores to get top 10 candidates for grasping. Finally, these top 10 candidates will be fed to a motion planner to remove any collision and inverse kinematics infeasibility. To grasp multiple objects, the robot has to execute the above two-step procedure many times.

The grasp candidates are generated from the edge images of depth and visible light images. The edge images are computed by the Canny edge detection algorithm. Using the distance between the contact points, we can prune any candidates with contact points that are too far apart. For instance, if the hand can only grasp objects of no larger than 10cm, we can prune away any three points that are more than 10cm apart.

To train the SVM model, we collected training data with human labels. During training time, for each grasp candidate, a human editor will choose to label as either "Bad" or "Good"or "Very Good". Here, "Bad" indicates that the candidate is impossible to grasp; "Good" indicates that the candidate might be possible to grasp; "Very Good" indicates that the candidate can be grasped very well. Sometimes, this may require the human editor to operate the robot in order to decide the labels. Given the labels, we used a ranking SVM algorithm which optimizes a measure that prefers better ranking for top candidates. Details regarding the features and the training method are explained in the following sections.

The system has several key novelties. First, instead of modelling power grasps around a point, our system learns the contact points themselves. Second, many new and intuitive features are designed to improve grasping results. Finally, our system uses a ranking method to achieve better results than a standard classifier.

Our method can work with two-contact-point candidates (for a two-fingered hand) or three-contact-point candidates (for a three-fingered hand). To simplify the language, we are going to use pairs for feature illustration.

Finally, we note that the algorithm works by first finding the candidate contact points, then extracting features based

[2]Here, each candidate contains contact points for all fingers. For example, if the robot hand has three fingers then we have 3 contact points per candidate.

on the contact points. Once all candidates are ranked, the fingers can be placed approximately close to the contact points.[3] Hence, in practice, there is a small difference between finger placement locations and contact points (see Figure 3). The main reason for this little discrepancy between finger placement and contact point is that contact points can be selected easily on the edge map.

Fig. 3. Edge map (white), contact points (green crosses), and finger placements (blue circles). This edge map is extracted from the mug picture in Figure 1. See text for more details.

IV. Features

In our system, we consider features from intensity image and depth map data. In this section, we will elucidate some of the most important features that have significant influences on grasping results.

A. Gradient angle features

The basic intuition for this feature is that when we grasp at multiple points, there is a correlation between the vector connecting the contact points and the gradient vectors at each contact point. Figure 4 shows that, in case of two contact points, to grasp objects well, the gradient vectors and the connecting vectors should form a straight line.

An important note here is that the gradient at a pixel location is usually noisy. Yet, the key insight is that the histogram of the gradients at a patch around this pixel can give interesting information. In this sense, our gradient angle features resemble SIFT features [25] and HOG features [26].

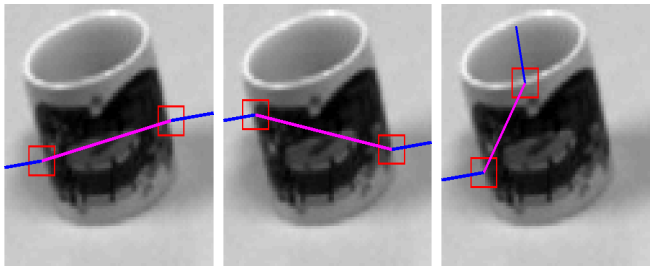

Fig. 4. Illustration of gradient angle features computed for grasping with pairs. The features are computed as the angles between the current gradient vector (blue line) and the vector connecting the current point with other contact points (magenta line). This feature is highly discriminative because it can tell apart between a good pair (left), a worse pair (middle) and a very bad pair (right). In case of the good pair, the blue lines and the magenta line are approximately in a straight line.

[3]In our experiments, finger placement locations are 1cm wider than the contact points.

To compute the gradient vectors, we first extract a 10x10 patch around each contact point in both depth and visible light images. Then we convolve each patch with an edge kernel and construct a histogram of gradients for a particular patch location using the idea in [26]. Using this histogram, we can extract the strongest gradient vector (the mode of the distribution). These steps are shown in Figure 5.

Given this mode gradient vector, we will use the angles formed between it and the connecting vectors to express the correlation mentioned earlier. These angles themselves are the features we are interested in. More explicitly, denote by g the mode gradient at the contact point and v a connecting vector from another contact point to the current contact point, the angle α is computed by

$$\alpha = \arccos \frac{g^\top v}{\|g\|\|v\|} \tag{1}$$

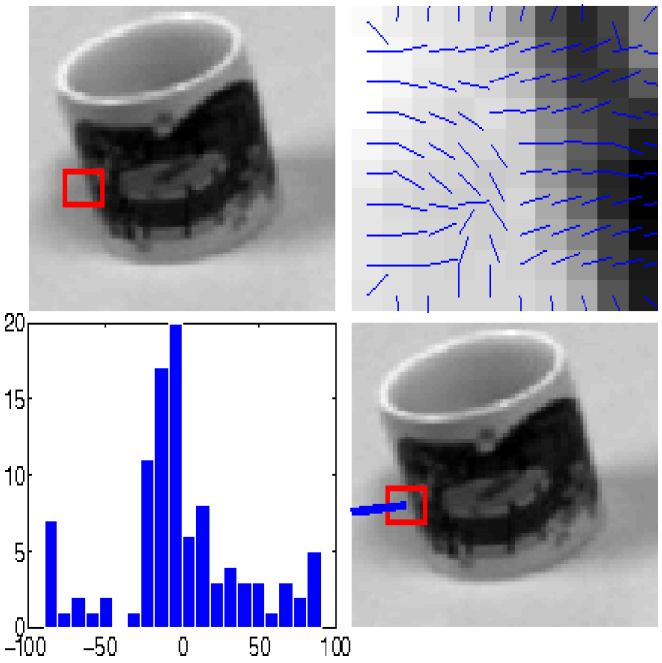

Fig. 5. Gradient computed at a contact point. Top left: an object with a contact point (red square). Top right: gradient angles at each pixel computed by convolving a Sobel edge kernel with the patch. Bottom left: histogram computed from the angles (-9^o is the mode of this distribution). Bottom right: the gradient (blue line) computed for the red square.

The main reason for the use of angles instead of a mere vector dot product is that angle is more robust to changes in lightning conditions: if the room is darker or brighter, the magnitude of the gradient vectors changes, while the angle does not.

In our system, we compute this feature for three types of edge detection algorithms (Prewitt, Roberts and Sobel) in both depth map and intensity image. For grasping with two-contact-point candidates (also called pairs), this gives 6 features for each contact point.

These features are our attempt to robustly capture **force stability** for grasps by incorporating local information (gradient at a contact point) and global information (vector con-

necting contact points). Note that these features are similar to the concept of surface normals. Yet, unlike surface normals, which can be unstable to compute near edges (computed by SVD, cf. [27]), our features are generally stable and less noisy.[4] Second, note that if we design a system with one contact point in mind, it is hard to discriminate between good grasps and bad grasps. For instance, two results (Figure 4 left and 4 right) have the same left point but they can turn out to be very good or very bad depending on the choice of the next contact point.

Gradient angle features are quite powerful and robust to illumination, rotation and translation. They, however, have several problems (see Figure 6). First, pairs that are on shadows, textures on the object can be considered as good pairs (Figure 6, left). This problem can be addressed by taking into account collision (sphere feature, see Section IV-D). The second problem is that pairs that belong to different objects may be misclassified as good pairs (Figure 6, right). This can be solved by i) knowing the distance between the two points (see Section IV-B) and ii) considering the depth variation from one point to another (see Section IV-C).

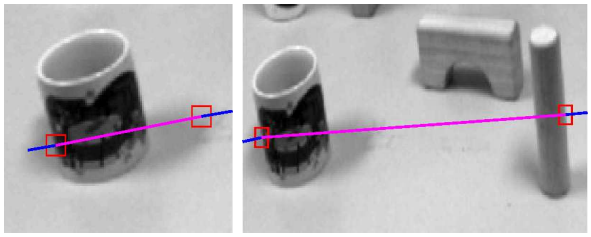

Fig. 6. Some problems with the gradient feature. Left: shadows and textures, this can be solved by sphere feature. Right: pairs in two different objects, this can be solved by distance and depth variation feature.

B. Distance features

In our system, we use distance between pairs to rule out pairs that are physically impossible for the robot hand to grasp (they are too far apart). Moreover, even in the case that the pairs are close, the hand may prefer to grasp pairs that are in some certain sizes. As an example, in the case of Barrett hand, it prefers to grasp pairs with distance 5cm than pairs with distance 10cm or 1cm, even though they may be both physically feasible. So our distance feature can be computed as follows

$$d = \|d_{pair} - d_{optimal}\| \tag{2}$$

where d_{pair} is the distance of the pair in depth image, and $d_{optimal}$ is the optimal grasp size of the robot hand. For d_{pair}, we used X-distance, Y-distance, Z-distance (in the robot's frame) and Euclidean distance (i.e. we have 4 distance features).

Another type of distance features we used is the distance from the base of the robot to the pair itself. Using this knowledge, the learner can learn to prefer pairs that are closer than pairs that are further away.

[4]The features are unstable in the case that the distribution is multimodal (patches are at corners). To solve this, we tried the idea of using top two gradients but this idea only slightly improved the results. So, to simplify the description we only use only one gradient.

C. Depth and pixel intensity variation features

As illustrated earlier, using gradient angle features alone, it is hard to know whether pairs belong to different objects or not. To address this problem, we can use the changes in depth as a feature. Figure 7 shows that if the pair belongs to different objects, there are a lot of changes in depth than if the pair belong to the same object.

To compute this feature, we use the depth of the pixels belong to the connecting line of the two contact points to compute summary statistics of the depth variation. Some relevant statistics are i) the variance of depths, ii) the maximum depth minus the minimum depth and iii) the number of times the line crosses depth discontinuities.[5]

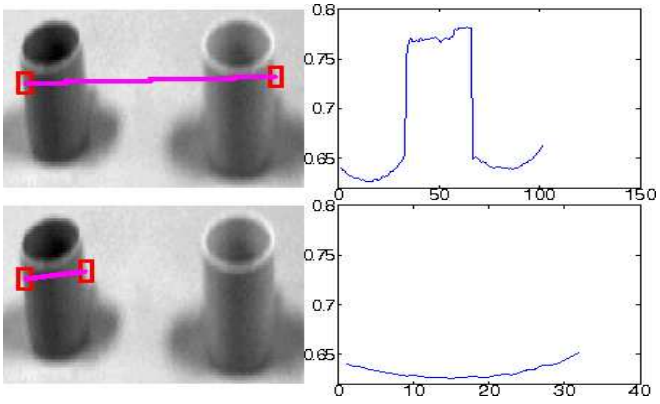

Fig. 7. Depth variation features: we can use the changes in the depth as a feature. Top row: a bad pair and corresponding change in depth taken from the connecting line (magenta). Bottom row: a better pair and the corresponding change in depth. Notice that the depths change more in the bad pair than in the good pair.

A similar idea can be applied to the pixel intensity of visible light image (see Figure 8).

One may view these features as a simple way to approximate segmentation. Segmentation, nevertheless, is avoided in our approach because it can be a lot harder to perform in a cluttered scene.

D. Sphere feature

This feature is a simple way to implement collision detection in the early stage of predicting. The goal of this feature is to differentiate pairs that are possible to grasp and pairs that are physically impossible to grasp. Pairs that are impossible to grasp are ones that if the fingers move there, there must be some collision.

Figure 9 shows a case when this feature is useful. This feature can only be computed in the depthmap. For this feature, we find the two spheres at the locations where the fingers should be if the contact points are used. The radii of the spheres are equal to the size of the fingertips. The features are computed by counting the total number of points in the

[5]We do not need complete 3D models to do this, only the front face of objects with surrounding environment is enough.

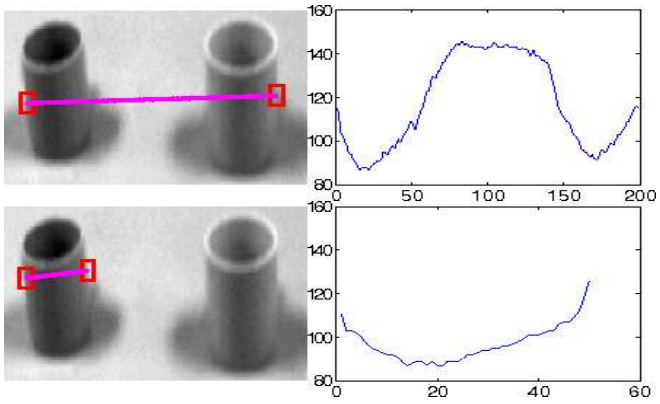

Fig. 8. Pixel intensity variation features: we can use the changes in the pixel intensity as a feature. Top row: a bad pair and corresponding change in pixel intensity taken from the connecting line (magenta). Bottom row: a better pair and corresponding change in pixel intensity. Notice that the pixel intensities change more in the bad pair than in the good pair. Note that there are more points in the x-axis of this figure than the previous figure because of missing depth readings.

point cloud inside the two spheres. The spheres that have zero point inside will potentially have no collision. However, due to noise in sensory data, there might be some points inside the spheres despite being good pairs. So we can only use this as a feature in a learning algorithm such that the algorithm can figure out a good threshold.

For example, the learner will learn that the case in Figure 9 is not graspable because the total number of points inside the two spheres is too high and there must be collision when the fingers get there.

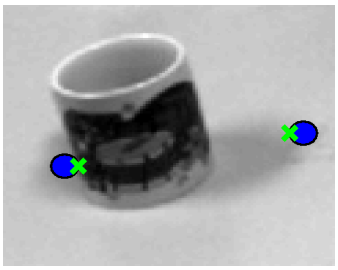

Fig. 9. Sphere feature: we can implement a simple form of collision detection in the early stage of predicting. The sphere features are the number of points inside the two spheres (in the depth map, but we visualize the spheres only in 2D). In this case, the number of points inside the left sphere is zero and the number of points inside the right sphere is over a hundred. The green crosses are the contact points.

E. Other features

In our algorithm, we also consider other features such as raw depth data, raw image pixel data. These features only slightly improve the algorithm.

V. Learning with Ranking Support Vector Machines

A. Why learning?

As can be seen from the previous section, all of our features are powerful but not perfect. To combine information provided by different features, one might consider to hard code rules to weight different features. In practice, we find that finding good hard-coded rules is usually difficult even with a lot of tweaking. A better way is to use learning algorithms to learn these grasping concepts from data.

B. Performance metric and learning method

In this section, we will discuss an accuracy metric and a learning method that are probably more relevant to the grasping task than previous work.

The proposed methods in [1], [2] employ a classifier to learn to classify a grasp point to be "Good" or "Bad", yet at prediction time the classifier is used to *rank* grasp points and then pick top points to grasp. There are two issues with this approach. First, if we would like to compare two sets of grasping results, we cannot use classification accuracy because the robot only considers top pairs to grasp. This leads to the second issue: optimizing classification accuracy is suboptimal because the learner has to make sure that all pairs are classified correctly even though top pairs are what matters.

We will take a full advantage of some recent developments in learning to rank and information retrieval. In such fields, a search engine has to rank search results and return top documents. In this particular setting, classification accuracy is not a meaningful measure and other ranking metrics are used to compare and optimize rankers. It has been shown that it is better to employ measures such as AUC (Area under ROC), Precision@k [28], or NDCG (Normalized Discounted Cumulative Gains) [29], [30], [31]. In this paper, we consider a ranking method that optimizes NDCG (first described in [31]). Here, we briefly survey the metric and the learning algorithm. Interested readers should refer to [31] for a detailed treatment.

Suppose we have a training set $\{x_{qi}, g_{qi}\}$ where $q = 1, \ldots, n$ indexes the scenes in the training set. x_{qi} is a vector describing the features for the i-th pair corresponding to scene q. g_{qi} is the numerical grade assigned to that pair (Bad=0/Good=1/Very Good=2).

Now, assume that for each scene we have $x_q = (x_{q1}, \ldots, x_{qm_q})$ which contains all features for grasping pairs of scene q. Also suppose that our ranker outputs ranking y_q (a permutation over $\{1, \ldots, m_q\}$) where $y_{qi} = r$ which means i-th pair has rank r.

The NDCG score for scene q is defined as [29]

$$NDCG_k(y, q) = \frac{1}{N_q} \sum_{i=1}^{k} D(y_i)\Phi(g_{qi}), \qquad (3)$$

where D is called the discount function $D(r) = \frac{1}{log(1+r)}$, $\Phi(g) = 2^g - 1$ and N_q is a normalization constant such that the optimal ranking based on the values of g_{qi} has score 1. k is called a truncation level (in the case of search engine $k = 10$). Intuitively the NDCG is a ranking evaluation criterion that puts strong emphasis at the topmost items, and between these items, there is a small decaying factor.

As an example, suppose if we have 100 pairs on the scene to grasp and if we misclassify top 5 pairs, we might just end up with a classifier with 95% classification accuracy;

whereas, if we use NDCG as the measure with $k = 10$, i.e., we care only about top 10 pairs, because Φ has an exponential component, any misranking of the top pairs will result in a bigger loss for $NDCG_{10}$.

Because NDCG focuses on ranking for top pairs, it is extensively used to measure and compare the performances of rankers or search engines. Methods that optimize this measure tend to perform well in practice.

The method in [31] optimizes the following objective

$$\min_{w,\xi_q} \quad \frac{\lambda}{2} w^\top w + \sum_q \xi_q$$

$$\text{s.t. } \forall q, \forall y \neq y_q, \; w^\top \Psi(x_q, y_q) - w^\top \Psi(x_q, y) \geq \Delta_q(y) - \xi_q$$

where $\Delta_q(y) = 1 - NDCG_k(y, q)$, $\Psi(x_q, y_q)$ is the features we have just described, and w is the parameter vector that we need to learn. This optimization objective is convex and thus can be solved efficiently via cutting plane and Hungarian Marriage algorithms (software is available in [32]). Note that, at prediction time, the scores for all pairs are computed as $w^\top \Psi(x_q, y_q)$, then a sort operation is sufficient to obtain ranking results.

To recapitulate, we will use NDCG as the 'offline' performance metric to compare different learners. In addition, we will optimize NDCG measure because it usually gives better ranking for the top pairs. We will show that optimizing such measures substantially improves the outcome of grasping.

VI. Multiple contact grasping

For the ease of presentation, we illustrate most main concepts and features in the case of grasping with two contact points. In practice, it is usually more stable to grasp objects with more than two-finger. For example, for a Barrett hand, grasping with three fingers are much more stable than with two fingers.

The method described above can be applied to the case of grasping with multiple contact points. For example, for the gradient angle feature, instead of having one angle for two contact points, we can have one angle for any pair of contact point in the case of grasping with three contact points. Likewise, for the distance feature, instead of having one distance feature, we will have three distance features; each corresponds to the distance between every pair of contact points. As a result, we will have more features and rely on the learning algorithm to learn the right combination of features that give the most stable grasping.

In fact, all of the experiments in the next section are performed with a robot with 3 fingers (i.e., three contact points).

VII. Descriptions of the robot

We performed our experiment on the STanford AI Robot (STAIR2). This robot has a 7-DOF arm (WAM, Barrett Technologies) and a three-fingered 4-DOF hand.

To capture depth data, the robot uses an active triangulation sensor which contains a laser projector and a camera [24]. The camera returns a 640x480 gray scale image. The active triangulation sensor returns a very dense depth map for most pixels in the image. This sensor, however, suffers from problems such as occlusions and noisy readings at the edges or shiny objects. Figure 2 shows data captured by the camera and the active triangulation system.

The whole system (arm, camera, laser) is calibrated by our recently proposed algorithm for joint calibration [33]. The average calibration errors of the entire system are often less than 5mm.

VIII. Experiments

We consider three sets of experiments. The first set of experiments is performed offline on a hold out test set with the purpose of verifying the performance metric and comparing different methods. In the second set of experiments, we compare our method and method in [2] when grasping novel objects. In the third set of experiments, we demonstrate the our method is effective for the task of cleaning up a table.

In our experiments, we consider grasping **triples** only, this means there are three contact points per candidate. The main reason is that our robot hand (Barrett) has three fingers.

A. Offline test

Our dataset contains 8 scenes and 420 grasping candidates. We split the dataset to 6 scenes with 336 candidates for training and 2 scenes with 84 for a hold-out test set. The training set is further split to training and validation set for model selection. All the data are collected with real objects: simple wooden blocks and boxes (see Figure 10).

Fig. 10. Training object examples.

This test is entirely offline, i.e. without robot execution. The goal is to determine the capability of the software component. We also would like to compare the generalization power of our method against previous approach. For this test, we will use NDCG as the performance measure. All methods are trained with linear models.

To make the two systems comparable, we label the grasping **triples** densely and any one-point result returned by [2] will query a motion planner to find three corresponding contact points. If the three contact points are close to any three contact points given by the label, we consider that as a successful triple.

In the first experiment, we would like to make sure that optimizing NDCG in the training set can give good performance in the test set. Table I shows that optimizing NDCG indeed gives a big improvement in ranking performance (5% increase in NDCG). Note that, although classification

accuracy gets worse when optimizing NDCG, this is not a big problem because classification accuracy considers suboptimal triples which are never used in the grasping. Also note that NDCG of 88.52 is considered to be very good (informally, every time we issue a query, 88% of the ranking results are good).

In the second experiment, we would like to confirm that grasping with multiple contact points improves grasping results compared to [2]. The results of the experiment (shown in Table II) show that it is indeed the case. This is because grasping with multiple contact points can give more discriminative features compared to grasping with one point and that combining multiple stages in [2] removes errors.[6]

TABLE I

ADVANTAGES OF USING RANKING SVMS: OPTIMIZING NDCG WILL RESULT IN BETTER RANKING PERFORMANCE.

Measure	Optimize classification accuracy	Optimize NDCG
Accuracy	93.67%	91.43%
$NDCG_{10}$	83.27%	88.52%

TABLE II

PERFORMANCE OF OUR METHOD AND PREVIOUS METHOD [2].

Measure	Previous method [2]	Our method
Accuracy	76.21%	91.43%
$NDCG_{10}$	72.04%	88.52%

B. Grasping novel objects with the STAIR2 robot

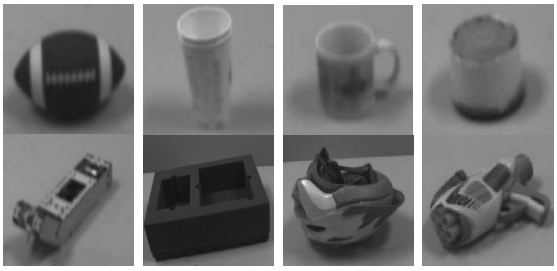

Fig. 11. Examples of objects we used in grasping-novel-objects experiment. Note that for some objects, it is hard to separate the object from the background.

In this test, we are interested in the task of grasping novel objects, the objects that never appear in the training set. The objects we consider in our experiment are slightly harder than objects used in [2]. For each object, we extensively performed 20 trials, and compared our method with method in [2]. Each trial is considered as success if the hand can grasp the object and move it to the bin. To speed up the method in [2], we hard coded the table height for their method such that the motion planner does not need to spend time looking at random locations on the desk. However, for our system, we did not hard-code the table height, which makes the problem harder for our algorithm. We reported the average success rates in Table III.

As can be seen from the table, our method significantly outperforms method in [2] for 8 out of 9 objects. The reason for the improvement is that our approach gives more stable grasps than [2] does.

TABLE III

PERFORMANCE OF OUR METHOD AND PREVIOUS METHOD [2] FOR THE TASK OF GRASPING NOVEL OBJECTS.

Object	Previous method [2]	Our method
Football	50%	70%
Cup	70%	85%
Mug	80%	90%
CD Holder	75%	95%
Wooden robot arm link	70%	80%
Foam (deformable)	70%	85%
Nerf gun	50%	75%
Helmet	90%	75%
Mean/Std	$69.37 \pm 13.74\%$	$81.87 \pm 8.42\%$

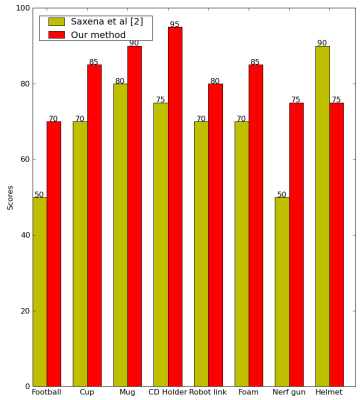

Fig. 12. Performance of two methods in bar chart.

For some objects, we were unable to match the performance reported in [2]. There are several possible reasons for this mismatch. The first reason is that although the objects have the same name, they look quite different and harder to grasp than those in [2]. Furthermore, in our experiments, the objects are placed in rather difficult locations and unless the predictor makes a very good prediction, the grasp is going to fail. Finally, unlike [2], we performed 20 trials per object, and this makes the statistics more stable than their experiments. For some objects, the performance of their method is higher than reported in [2] because our sensor has higher resolution.

In the introduction, we already elucidated the main intuition why grasps produced by [2] are unstable. This basic intuition indeed translated to their failure cases in the experiments. The main problem is that despite trying to learn features for power grasps, some of the features do not work well because the first stage gives bad pinch grasp predictions. Their method thus has to depend on motion planning to make the choice of finger placements. As explained earlier, this is problematic because the motion planning does not consider force stability.

Interestingly, the method in [2] outperforms our method in the case of helmets. In general, our method works for

[6]Note that, for method in [2] we optimized both NDCG and classification accuracy and found out that optimizing NDCG gives better ranking NDCG score for the test set. The results reported in Table II are obtained by optimizing NDCG in the training set.

observable triples: the triples that can be captured by a camera. Most of observable triples in the case of helmets are either too far apart or unstable to grasp. The method in [2] simply queries the motion planner and luckily most answers given by the motion planner are good. In this direction, we are considering a hybrid approach where both methods can play a role in the prediction.

Except from the helmets, most of our failures are mostly related to missing sensor readings due to occlusions (the camera cannot see the laser beam). A smaller faction of failures is due to the fact that sometimes the hand cannot hold the object strongly even though the predictions are good. There are also a few cases where the raw predictions are bad.

C. Cleaning up desks with the STAIR2 robot

Finally, we apply our method to clean desks that contain a few objects. In this setting, we use the counts of success/failure of the first attempt per object and use this as the performance metric. Unlike [2] where only an object is grasped from a cluttered scene, we would like to grasp all objects in a table. The results reported will be the average success rates for all objects.

We performed experiments with at tables with increasing difficulties: number of objects (ranging from 2 to 8) and different textures on tables. Figure 13 shows an example scene. Again, we would like to compare our method with the method in [2].

Fig. 13. An example scene which our STAIR2 robot tries to clean.

To make the comparison formal, we use similar desk setups for the competitive methods. For each algorithm, we performed 10 trials. And the rate of success of our algorithm is around 80% vs. 70% for the method in [2]. These statistics agree closely with the average performance reported in the previous section.

We also conducted several experiments with our algorithm only counting failures if the objects are moved out of the robot's vision. In this setting, our robot can autonomously clean up tables with 5 to 10 objects completely. We note that in our experiments, the objects are lying very close to and sometimes touching each other (see attached video).

In the attached video, we show some example cases that the robot cleans the table using our algorithm. More full length video sequences will be uploaded in the STAIR website http://stair.stanford.edu .

ACKNOWLEDGMENT

We thank John Duong Dang, Morgan Quigley, Josh Taylor, Lawson Wong and STAIR teams for the help with the project and the paper. Support from the Office of Naval Research under MURI N000140710747 is gratefully acknowledged.

REFERENCES

[1] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *IJRR*, 2008.
[2] A. Saxena, L. Wong, and A. Y. Ng, "Learning grasp strategies with partial shape information," in *AAAI*, 2008.
[3] V. Nguyen, "Constructing stable force-closure grasps," in *ACM Fall joint computer conference*, 1986.
[4] ——, "Constructing stable grasps," *IJRR*, 1989.
[5] J. Ponce, D. Sam, and B. Faverjon, "On computing two-finger force-closure grasps of curved 2d objects," *IJRR*, 1993.
[6] K. Lakshminarayana, "Mechanics of form closure," in *ASME*, 1978.
[7] J. Salisbury, "Active stiffness control of a manipulator in cartesian coordinates," in *IEEE Conference on Decision and Control*, 1980.
[8] ——, "Kinematic and force analysis of articulated hands," Ph.D. dissertation, Stanford University, 1982.
[9] M. Cutkosky, "Machanical properties for the grasp of a robotic hand," CMU, Tech. Rep., 1984.
[10] H. Hanafusa and H. Asada, "Stable prehension by a robot hand with elastic fingers," in *Seventh Inter. Symp. on Industrial Robots*, 1977.
[11] R. Pelossof, A. Miller, and T. Jebera, "An SVM learning approach to robotic grasping," in *ICRA*, 2004.
[12] K. Hsiao and T. Lozano-Perez, "Imitation learning of whole-body grasps," in *IROS*, 2006.
[13] K. Hsiao, L. Kaelbling, and T. Lozano-Perez, "Grasping POMPDPs," in *International Conference on Robotics and Automation*, 2007.
[14] E. Chinellato, R. Fisher, A. Morales, and A. del Pobil, "Ranking planar grasp configurations for a three-finger hand," in *ICRA*, 2003.
[15] J. Coelho, J. Piater, and R. Grupen, "Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot," in *Robotics and Autonomous Systems*, 2001.
[16] D. Bowers and R. Lumia, "Manipulation of unmodelled objects using intelligent grasping schemes," *IEEE Trans. on Fuzzy Systems*, 2003.
[17] A. Morales, E. Chinellato, P. Sanz, and A. del Pobil, "Learning to predict grasp reliability for a multifinger robot hand by using visual features," in *International Conference AI Soft Computing*, 2004.
[18] R. Platt, R. Grupen, and A. Fagg, "Improving grasp skills using shema structured learning," in *ICDL*, 2006.
[19] R. Platt, A. H. Fagg, and R. Grupen, "Learning grasp context distinctions that generalize," in *IEEE-RAS International Conference on Humanoid Robots*, 2006.
[20] A. Edsinger and C. Kemp, "Manipulation in human environments," in *IEEE/RAS International Conference on Humanoid Robotics*, 2006.
[21] A. Saxena, "Monocular depth perception and robotic grasping of novel objects," Ph.D. dissertation, Stanford University, 2009.
[22] A. Saxena, J. Driemeyer, J. Kearns, and A. Ng, "Robotic grasping of novel objects," in *NIPS*, 2006.
[23] L. Wong, "Robotic grasping on the Stanford artificial intelligence robot," 2008.
[24] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Y. Ng, "High accuracy 3D sensing for mobile manipulators: Improving object detection and door opening," in *ICRA*, 2009.
[25] D. Lowe, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision*, 2004.
[26] N. Dalai and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[27] A. Ng, A. Zheng, and M. Jordan, "Link analysis, eigenvectors, and stability," in *IJCAI*, 2001.
[28] T. Joachims, "A support vector method for multivariate performance measures," in *In Proc. Intl. Conf. Machine Learning*, 2005.
[29] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems*, 2002.
[30] C. Burges, R. Ragno, and Q. Le, "Learning to rank with nonsmooth cost functions," in *NIPS*, 2007.
[31] O. Chapelle, Q. Le, and A. Smola, "Large margin optimization of ranking measures," in *NIPS Worskop in learning to rank*, 2007.
[32] C. Teo, S. Vishwanathan, A. Smola, and Q. Le, "Bundle methods for regularized risk minimization," in *JMLR*, 2010.
[33] Q. Le and A. Ng, "Joint calibration of multiple sensors," in *IROS*, 2009.