Cloud-Based Robot Grasping with the Google Object Recognition Engine

Ben Kehoe[1] Akihiro Matsukawa[2] Sal Candido[4] James Kuffner[4] Ken Goldberg[3]

***Abstract*— Rapidly expanding internet resources and wireless networking have potential to liberate robots and automation systems from limited onboard computation, memory, and software. "Cloud Robotics" describes an approach that recognizes the wide availability of networking and incorporates open-source elements to greatly extend earlier concepts of "Online Robots" and "Networked Robots". In this paper we consider how cloud-based data and computation can facilitate 3D robot grasping. We present a system architecture, implemented prototype, and initial experimental data for a cloud-based robot grasping system that incorporates a Willow Garage PR2 robot with onboard color and depth cameras, Google's proprietary object recognition engine, the Point Cloud Library (PCL) for pose estimation, Columbia University's GraspIt! toolkit and OpenRAVE for 3D grasping and our prior approach to sampling-based grasp analysis to address uncertainty in pose. We report data from experiments in recognition (a recall rate of 80% for the objects in our test set), pose estimation (failure rate under 14%), and grasping (failure rate under 23%) and initial results on recall and false positives in larger data sets using confidence measures.**

I. Introduction

Consider the goal of a household robot that can reliably declutter floors, tables, and desks by identifying objects, grasping them, and moving them to appropriate destinations such as shelves, cabinets, closets, or trash cans. Errors in object recognition could be costly: an unwrapped chocolate bar could be mistaken for a cellphone and moved to the charging station, or vice versa—a cellphone could be placed in the trash can. Recognition in unstructured environments such as homes is challenging as the set of objects that may be encountered dynamically grows as our global economy designs new products at an increasing pace to satisfy consumer and shareholder demands.

The "cloud"—the Internet and its associated data and users—is a vast potential source for computation and data about objects, their semantics, and how to manipulate them [16] [22]. People upload millions of digital photos every day and there are several image labeling projects using humans and machine learning [37] [42] [45]. In this paper we propose an architecture that integrates Google's object recognition engine with open-source toolkits and a sampling-based grasping algorithm to recognize and grasp objects.

[1]Department of Mechanical Engineering; benk@berkeley.edu
[2]Department of Electrical Engineering and Computer Science; amatsukawa@berkeley.edu
[3]Department of Industrial Engineering and Operations Research and Department of Electrical Engineering and Computer Science; goldberg@berkeley.edu
[1–3] University of California, Berkeley; Berkeley, CA 94720, USA
[4]Google; 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA; {scandido, kuffner}@google.com

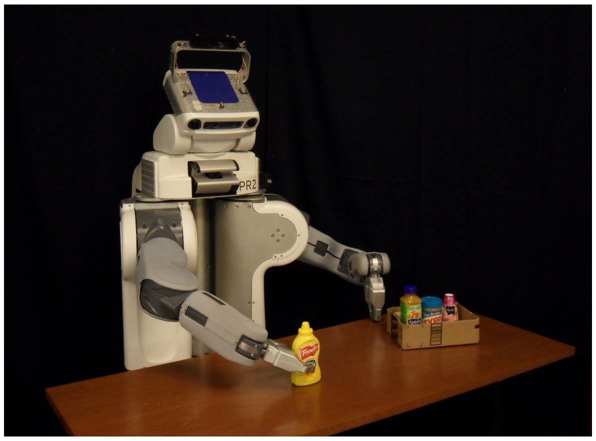
Fig. 1. After training, when an object is presented to the Willow Garage PR2 robot, the onboard camera sends an image to a Google server which returns a (possibly empty) set of recognized objects with associated 3D models and confidence values. For each object, the server also returns an associated set of grasps with associated confidence values. A set of measured 3D depth points is processed with this data locally to estimate object pose and select a grasp for execution or a report that the confidence values are insufficient for grasp selection. After executing a grasp, the robot assesses the outcome and stores results in the cloud server for future reference.

Although networked robotics has a long history [3] [20] [21], Cloud Computing facilitates massively parallel computation and real-time sharing of vast data resources. Cloud Robotics has potential to improve robot performance in at least five ways [19]: 1) Big Data: indexing a global library of images, maps, and object data [10] [14], 2) Cloud Computing: parallel grid computing on demand for statistical analysis, learning, and motion planning [9], 3) Open-Source / Open-Access: humans sharing code, data, algorithms, and hardware designs [6] [8] [46], 4) Collective Robot Learning: robots sharing trajectories, control policies, and outcomes, and 5) Crowdsourcing and call centers: offline and on-demand human guidance for evaluation, learning, and error recovery [13] [41].

The present paper considers how Big Data and Cloud Computing can enhance robot grasping. We train an object recognition server on a set of objects and link it with a database of CAD models and candidate grasp sets for each object, where the candidate grasp sets are selected using a variant on the quality measure from our previous work, where we studied how parallel computation in the cloud can facilitate computing of optimal grasps in the presence of shape uncertainty [27] [28]. In this paper, we extend the sampling based approach to consider 3D objects with uncertainty in pose.

We report two sets of experiments, the first with a set of

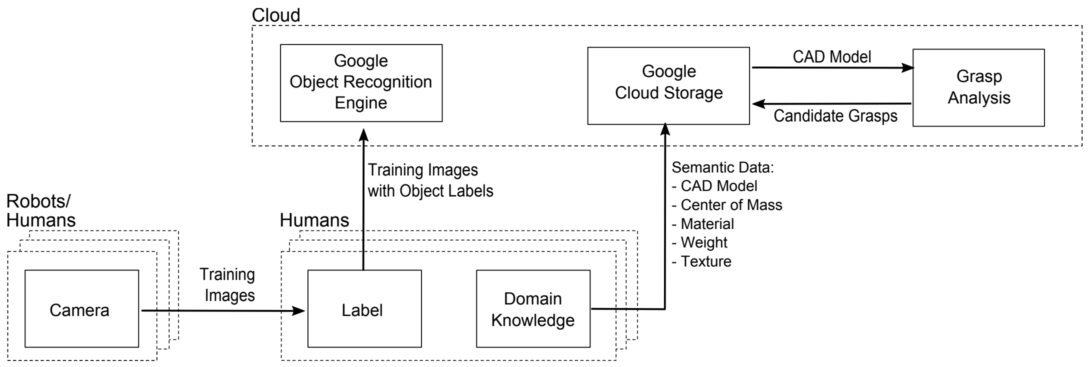

Fig. 2. System Architecture for offline phase. Digital photos of each object are recorded to train the object recognition server. A 3D CAD model of each object is created and used to generate a candidate grasp set. Each grasp is analyzed with perturbations to estimate robustness to spatial uncertainty.

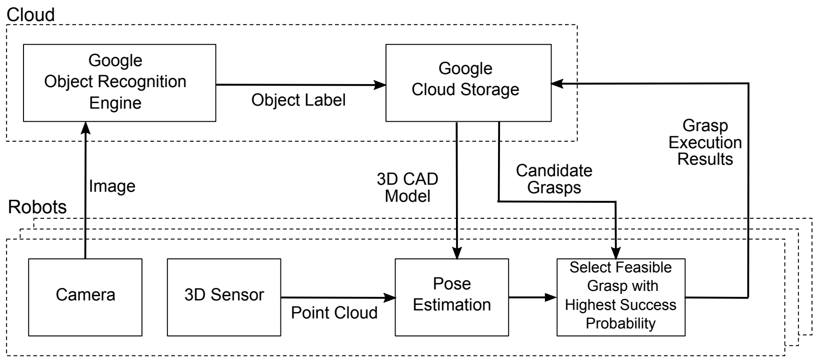

Fig. 3. System Architecture of online phase. A photo of the object is taken by the robot and sent via the network to the object recognition server. If successful, the server returns the stored data for the object. The robot then uses the measured 3D point set with the pressured 3D Mesh model to perform pose estimation, and selects a grasp from the reference set of candidate grasps. After executing the grasp, the robot assesses the outcome and stores results in the cloud server for future reference.

six household objects and the second with 100 objects. We used the Willow Garage PR2 robot [7] and created reference 3D mesh models and sets of candidate grasps, which were uploaded to the server.

II. Related Work

Goldberg and Kehoe [19] survey related work in Cloud Robotics and Automation. There has been significant progress in object recognition, from identifying features that are rapidly computable and invariant to translation, scale, and rotation, to learn the visual representation by incorporating semantic attributes and context information [17] [31] [34] [40] [44].

Researchers are working to improve both the scalability and accuracy of large-scale image recognition [26] [33] [35], making object recognition systems commercially viable. The purpose of this paper is to show how such a high-quality large-scale object recognition server can be incorporated into part of a cloud-based pipeline to improve grasping in robotics. Recent research has demonstrated the cloud's ability to enable information sharing between networked robots to accomplish tasks widely separated in time and space [24] [29] [30] [46].

There is substantial research on grasping [11]. While some research has looked at object recognition for grasping in isolation [25] [43], most work approaches it as a unified task. Approaches for object recognition for grasping include using local descriptors based on training images [15], and 3D model reconstruction involving 3D object primitives for pose estimation [23]. Saxena et al. [39] developed a method for calculating grasp points for objects based on images, where the grasps were learned from prior grasps for similar objects. This approach removed the need for a full 3D reconstruction of the object, but didn't take advantage of existing commercial object recognition systems.

III. Problem Statement

In the offline phase, the system considers a set of physical objects O: o_1 to o_{N_O}. For each object o_i, we use 3D sensing to obtain an associated reference 3D point set Θ_i and construct a 3D triangular mesh model v_i. We then use the Columbia University GraspIt! toolkit to pre-compute a set of candidate grasps $G_i = \{g_{i,k} \,|\, k \in [1, N_G]\}$ for each object and assign an associated confidence value $s_{G_{i,k}}$ to each grasp.

For each object o_i, we also capture a set of reference images at different viewpoints $\phi_{i,j}$ for $j \in [1, N_{o_i}]$. We define the training set of images $\Phi = \{\phi_{i,j} \,|\, i \in [1, N_O], j \in [1, N_{o_i}]\}$. Given this set, the Google object recognition engine applies machine learning methods to analyze the set. Also during the offline phase, semantic information about each object such as an identifier key, name, weight, surface properties such as friction, etc. can be stored in the cloud server.

The online phase uses confidence thresholds for image recognition, pose estimation, and grasping, denoted c_I, c_T, and c_G, respectively. In the online phase, when an object from the set O is presented to the robot, an image of the object ϕ and 3D point set Θ are taken, and the image is sent to the Google object recognition engine. The robot receives back a *match set* consisting of matched training images with associated confidence measures: $M = \{(\phi_{i,j}, s_{i,j}) \,|\, \phi_{i,j} \in \Phi\}$. We define the *match object set* as the set of objects for which at least one image was matched, along with the highest confidence score for each object: $M_O = \{(o_i, s_{o_i}) \,|\, \exists j : (\phi_{i,j}, s_i) \in M \wedge s_{o_i} > s' \, \forall j', s' : (\phi_{i,j'}, s') \in M\}$

If $|M_O| = 0$, this is called a *null recognition*. If $|M_O| = 1$, this is called a *single recognition*. If $|M_O| > 1$, this is called a *multiple recognition*. If the confidence $s_I < c_I$, we stop and report that the object cannot be identified. Otherwise, the system identifies the object as $o = \text{argmax}_{o_i}\, s_{o_i}$, with confidence $s_I = \max_{o_i} s_{o_i}$. If the identified object is correct and $s_I \geq c_I$, the trial is successful. If o is incorrect and $s_I \geq c_I$, it is a *false positive*. If o is correct but $s_I < c_I$, it is a *false negative*. If the recognition confidence is above threshold, we retrieve the associated 3D point set Θ_o and estimate object pose T with an associated confidence measure s_T. If the pose estimate confidence $s_T < c_T$, stop and report that the pose cannot be determined. Otherwise, the system uses the pose and associated pre-computed grasps G_o and grasp confidence values to select the feasible grasp g^* with the highest confidence s_G^*. If the confidence $s_G^* < c_G$, stop and report that no grasp is found. Otherwise, the robot executes the grasp, attempts to lift the object, uses the gripper state to estimate the success of the grasp, and stores the data and results.

IV. System Architecture

The system architecture of the offline phase is illustrated in Figure 2. The offline phase includes training of the object recognition server, as described in Section IV-A, the creation of object reference data as described in Section IV-B and the creation and analysis of the candidate grasp set as described in Section IV-C.

The system architecture of the online phase is illustrated in Figure 3. This phase begins when an object is detected by the robot system. It takes a photo and captures a 3D point cloud and sends this to the object recognition server, as described in Section IV-D. Online pose estimation and grasp selection are described in section Section IV-E.

A. Offline Phase: Object Recognition

Google Goggles is a popular network-based image recognition service accessible via a free app for Android and iPhone smartphones [2]. The app sends a photo of an unknown object or landmark to the server, which rapidly analyzes it to return a ranked list of descriptions and associated web links or a report that no reference can be identified (Figure 4).

We use a custom version of this system that runs on Google's production infrastructure. Our version can be

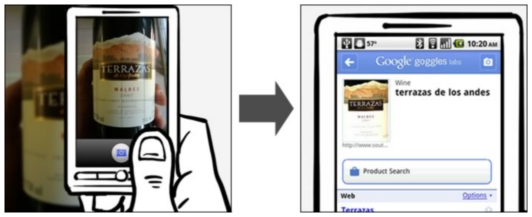

Fig. 4. A photo taken by a smartphone can be uploaded to the Google object recognition engine where it is analyzed, and results such as a list of relevant websites are returned to the user. We use a variant of this system where results determine object identity, pose, and appropriate grasp strategies.

trained on specific image sets and given a new image, returns the match set with confidence values. The server is exposed as two HTTP REST [18] endpoints—one for training, and one for recognition. The training endpoint accepts a set of 2D images of objects with labels identifying the object. The recognition endpoint accepts an image and returns a (possibly empty) set of matches. Each match is a stored image (from training) with its corresponding label and a confidence measure between 0 and 1.

B. Offline Phase: Object Model

For each object, we construct two 3D models: a point set Θ and a triangular mesh v. For our experiments, we selected one stable reference orientation for each object, and use two Microsoft Kinect sensors to scan a point set, which is filtered using tools from PCL, the Point Cloud Library [5] to define Θ, which is processed with surface reconstruction tools in PCL to create a reference 3D triangular mesh model. The 3D mesh model, reference point set, and candidate grasp sets are hosted on Google Cloud Storage [1], which is a multi-tenant and widely-replicated key-value store. Each object's data is associated with the same unique string used to train the object recognition server. From this key, a REST URL can be constructed to retrieve the data. In future work, we will explore alternative methods based on precise object geometry that may be used to compute stable poses on the planar worksurface and more accurate pose estimation and grasp generation.

C. Offline Phase: Robust 3D Grasp Analysis

The candidate grasp sets are generated using the Columbia University GraspIt! system [32]. GraspIt! takes as input the 3D triangular mesh model v and a model of the gripper that includes desired contact locations. We specify the built-in model of the Willow Garage PR2 parallel-jaw gripper. For each object model, GraspIt! generates a set of grasps that are feasible for a disembodied gripper. GraspIt! generates the set by randomly sampling a starting pose for the gripper in its open state surrounding the object, and then uses a simulated annealing method to iteratively improve the quality of the grasp [12]. This is repeated for a number of starting poses to produce a grasp set $G = \{g_k \,|\, k \in [1, N_G]\}$. In our experiments, 60 grasps were generated for each object. Each grasp g_k is evaluated by GraspIt! to estimate a grasp "quality" q_k as described in [12].

To estimate robustness to pose uncertainty, we use a variant of our previous sampling-based algorithm that models 2D shape uncertainty [27] and [28]. We extend that algorithm to model uncertainty in object pose as follows. Given the object's triangular mesh model v, we generate N_P perturbations in object pose by considering Gaussian distributions around the nominal position and orientation of the object. GraspIt! estimates grasp quality, $q_{k,l}$, for each perturbation. The weighted average of these values for a grasp over all perturbations, where the weights are the probability of a perturbation occurring, is used as the confidence measure for each candidate grasp:

$$s_{G_k} = \sum_l p(v_l) q_{k,l}$$

D. Online Phase: Object Recognition

In the online phase, the system submits an image to the Google object recognition server to retrieve the match set. After filtering matches below the confidence threshold, the best remaining match is taken. If there are no matches above the threshold, the robot stops and reports no matches found. Otherwise, the robot queries Cloud Storage for the reference data for the object. In the future, if no matches above threshold are found, the robot may take appropriate action such as moving its camera to obtain a better image.

E. Online Phase: Pose Estimation and Grasp Selection

If the object recognition server identifies the object with sufficient confidence, the reference data is used in the following steps. First, estimating the pose of the object using a least-squares fit between the detected 3D point cloud and the reference point set using the iterative closest point method (ICP) [36] [38]. We use the ICP implementation from PCL. The ICP algorithm performs a local optimization and therefore requires a reasonable initial pose estimate to find the correct alignment. We run ICP over a series of initial pose estimates. Ideally, the object data would include information about the stable poses of the object and these would be used as the initial pose estimates. We approximate this by using a fixed set of rotations for our pose estimates. We include 72 rotations about an internal vertical axis and for each of these rotations, we additionally include 8 rotations of 90° pitch down, to transform each object from an "upright" pose to a "horizontal" one.

Then, the initial estimate is computed by aligning the rotated reference point set to the detected point cloud such that the reference point set is on the work surface and the sides of the point cloud and point set are roughly aligned. For each initial pose estimate, the ICP algorithm generates an alignment and confidence score for that alignment, which is the sum of squared distances for all point correspondences it found. The alignment with the highest confidence score is chosen.

Using each estimated object pose, a candidate grasp is chosen from the candidate grasp set based on feasibility as determined by the grasp planner. The robot arm movement for the grasp then is planned using the inverse kinematics planner from OpenRAVE, a robotics motion-planning library [4]. Once the grasp is executed, success is determined based on the final position of the gripper jaws. The outcome data, including the image, object label, detected point cloud, estimated pose, selected grasp, and success or failure of the grasp, is uploaded to the key-value store for future reference.

V. Experiments Without Confidence Measures

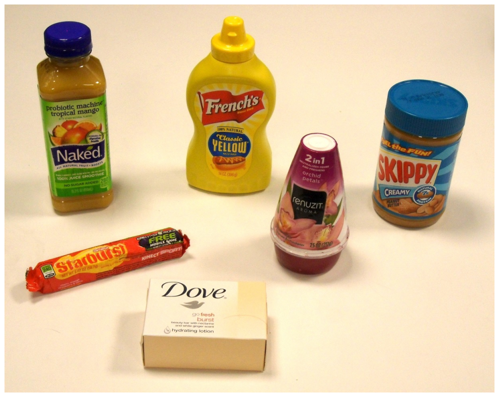

Fig. 5. The first set of objects used for the tests in Section V and Section VI-B. The objects were selected as representative of common household objects and are easily graspable by a parallel-jaw gripper.

We performed two sets of experiments. The first included a set of six objects and included end-to-end testing of image recognition, pose estimation, and grasping. The second set of experiments focused on evaluating the confidence measures for image recognition, using a larger set of 100 objects, and pose estimation, using the first set of objects. The confidence measure experiments are presented in Section VI.

We experimented with the set of six household objects shown in Figure 5. We used the Willow Garage PR2, a two-armed mobile manipulator. We selected these objects because they represent common object shapes and are graspable by the PR2's parallel-jaw gripper. The experimental hardware setup is shown in Figure 1. We used a robot-head-mounted ASUS Xtion PRO sensor, similar to a Microsoft Kinect, as our 3D sensor, and used the PR2's built-in high-definition Prosilica camera.

A. Object Recognition

We evaluated the performance of the Google object recognition server using a variety of training image sets.

We used the PR2's camera to capture 615 object images for training. We took images of objects in different poses against solid black and wood grain backgrounds, and under ambient florescent lighting and bright, diffuse incandescent light.

1) Test Results: We created 4 different training sets—a set of images randomly sampled from our pool (R), and three rounds of hand-selected training images (A,B,C). We trained the server on each set and used the remaining images in our pool to evaluate recognition performance. The hand-selected sets used human intuition about what would make a representative set of images.

Training Set	Size	Recall	Recall Rate	Training Time (s)	Recall Time (s)
R	228	307/387	0.79	0.45	0.29
A	92	247/422	0.59	0.40	0.29
B	52	215/422	0.51	0.39	0.28
A+B	144	317/422	0.75	0.40	0.29
C	49	199/422	0.47	0.39	0.30
A+B+C	193	353/422	0.84	0.40	0.29

TABLE I

IMAGE RECOGNITION PERFORMANCE FOR IMAGE TRAINING SETS. SET R WAS RANDOMLY SAMPLED. SETS A, B, AND C WERE HAND-SELECTED. THE AVERAGE CALL TIMES FOR TRAINING AND MATCHING A SINGLE IMAGE ARE GIVEN.

Table I shows the recall on the test set for the three training sets. We were able to achieve higher recall than random sampling through multiple rounds of hand-selected training images, but we were surprised to see that random sampling performed nearly as well (79% vs. 84%). Although there were many images for which the system was unable to make any identification (i.e., null recognitions), there were no false positives among the images we tested. For images where no object was recognized, such as those shown in Figure 6, lighting or the camera angle often obscured the text on labels.

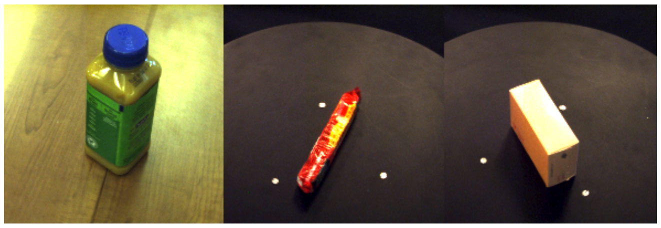
Fig. 6. Example images where no object could be identified.

B. Pose Estimation

Object	Total Trials	Failures	Failure Rate	Average Time (s)
Air freshener	15	2	0.13	7.4
Candy	15	0	0.00	1.4
Juice	15	1	0.07	10.2
Mustard	15	2	0.13	10.6
Peanut butter	15	2	0.13	2.1
Soap	15	0	0.00	3.6

TABLE II

POSE ESTIMATION RESULTS. WE MANUALLY DETERMINE FAILURE WHEN THE ESTIMATED POSE IS MORE THAN 5 MM OR 5 DEGREES FROM THE TRUE POSE.

We evaluated the system's pose estimation using 15 stable poses for each object. We manually declare failure when the estimated pose is more than 5 mm or 5° from the true pose. We observed that rotational symmetries of the object can cause the ICP algorithm to find a well-fitting but incorrect pose; most often this occurred with the estimated pose being inverted vertically from the true pose. For example, the shape of the mustard bottle is roughly symmetric above and below the waist of the bottle if the spout is disregarded. The ICP algorithm discards the spout this as part of its outlier rejection step, and produces a high quality score with an inverted pose for this object. We analyze this situation further in Section VI-B.

C. Grasping

Object	Candidate Grasp Set Size	Total Trials	Failures	Failure Rate
Air freshener	76	13	2	0.15
Candy	30	15	3	0.20
Juice	105	14	1	0.07
Mustard	61	13	3	0.23
Peanut butter	80	13	2	0.15
Soap	30	15	0	0.00

TABLE III

GRASP EXECUTION RESULTS. FOR CASES WHERE POSE ESTIMATION IS SUCCESSFUL, THE SYSTEM ATTEMPS TO GRASP AND LIFT THE OBJECT OFF THE WORKSURFACE. WE DECLARE FAILURE IF THE ROBOT DOES NOT ACHIEVE A GRASP OR DROPS THE OBJECT DURING LIFTING.

We evaluated grasping with cases where pose estimation is successful by having the system execute a grasp and attempt to lift the object off the worksurface. We declare failure if the robot does not achieve a grasp or drops the object during or after lifting. For some objects such as the air freshener and mustard bottle, small errors in pose estimation had a significant effect on grasp outcome. This is not surprising since in stable horizontal poses, the mustard bottle is nearly the width of the PR2's gripper opening. For the air freshener, the rounded and curved shape made it prone to rolling out of the gripper as it closed.

VI. EXPERIMENTS WITH CONFIDENCE MEASURES

We also studied the confidence measures generated by image recognition using a larger data set of 100 objects and 14,411 images. We also revisited pose estimation using the original data set from Section V.

A. Image Recognition

The second, larger data set included objects for which we only had images, not the physical objects. The data set consists of 100 objects, with approximately 140 images of each object. The set consists of photos of each object in a single stable pose, brightly lit against a white background. The images were taken at two low-elevation angles in 5° increments around the object, and from directly above in 90° increments. The confidence measure associated with image recognition is a match score returned by the Google server. This score, which falls between 0 and 1, is calculated based on a log likelihood passed through a transfer function that is used to maintain stability of the scores when the log likelihood formulation is updated. We randomly sampled a subset of images from the set for training and then tested all the remaining images. We repeated this procedure for sample set sizes ranging from 100 images to 7000. This larger data set provided us with conditions that did not exist in the smaller set used for end-to-end testing. For example, some

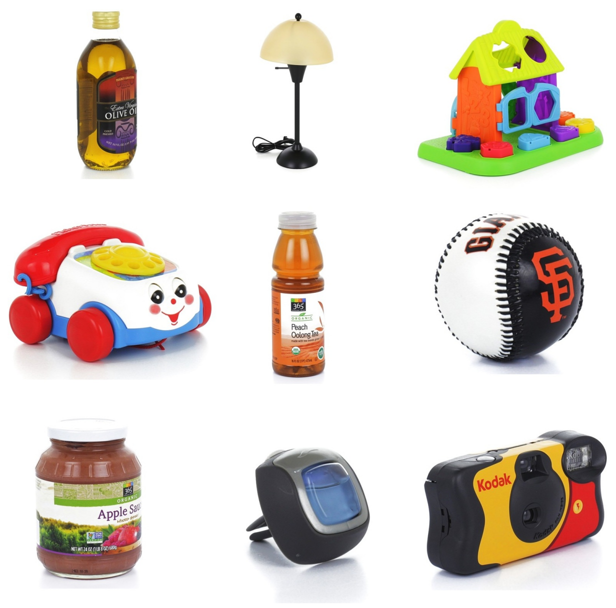
Fig. 7. Objects from the second data set used in Section VI-A. This set includes 14,411 images of 100 objects that are commercially available household products and toys. The images include photos of the object in a single pose, brightly-lit against a white background. The images were taken at two low-elevation angles in 5° increments around the object, and from directly above in 90° increments.

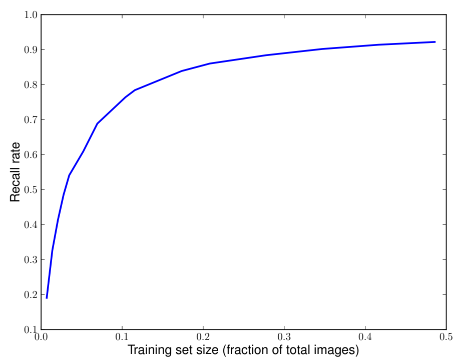

Fig. 8. Recall rate vs. training set size as a percent of total image size. The image set consists of 14,411 images of 100 different objects. The image set was tested by randomly sampling a number of images to train the object recognition server, and using the remaining images for testing. The recall rate is the fraction of the images tested that the object recognition server correctly identified.

of the objects were different models of the same products, differing only in color scheme or text. Other objects had similar shapes and colors.

The recall rate is plotted in Figure Figure 8, which shows much better results than in the first experiment. In Table I, set R was 37% of the total image set size, and resulted in a recall rate of 0.79. With our larger set, training with 35% of the images resulted in a recall rate of 0.90. The rate of false positives, which was below 1% for all training sets, is plotted in Figure 9.

Because the object recognition server returns multiple matches with confidences, we considered how this data might be used to recognize false positives. We trained the

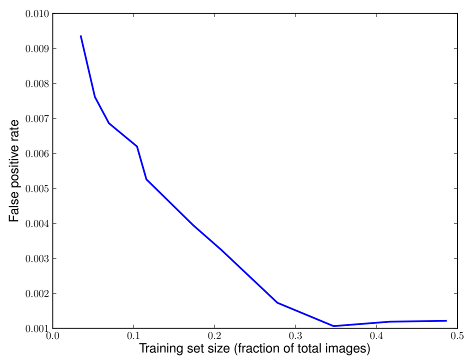

Fig. 9. False positive rate vs. training set size as a percent of total image set size. The image set consists of 14,411 images of 100 different objects. The image set was tested by randomly sampling a number of images to train the object recognition server, and using the remaining images for testing. The false positive rate is the fraction of images tested for which the object recognition server identified an object that was not correct. Note that the maximum false positive rate is under 1%.

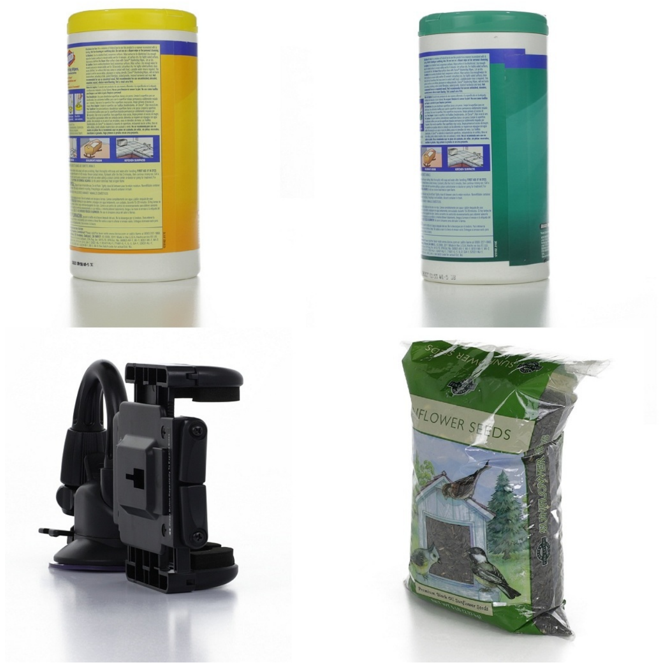
Fig. 10. Two examples of false positives, which occur less than 1% in our experiments. The images on the left are the measured images, and the images in the right column are what was matched.

system using 3000 randomly selected images, roughly 20% of our image set. We considered two separate cases: when only a single object is matched (a single recognition), and when multiple objects are matched (multiple recognition). In the single recognition case, the average score of a correct recognition was 0.49, whereas the average score of a false positive was 0.06. This suggests a threshold could be used to identify false positives. For example, if the maximum false positive score, 0.15, was used as the threshold, only 6% of correct recognitions would have been erroneously identified as false positives. When the server returns multiple possible object matches, the relative confidences of the different matches can be considered. In our experiments, we found that, on average, the second best object had a score 30% of the best object for a correct recognition, but for a false positive, the second best object had a score 79% of the best object. This also suggests a threshold could be used to identify false positives.

B. *Pose Estimation*

The confidence measure associated with pose estimation using ICP is the sum of the squared distances of corresponding points in the sensed and reference point clouds. This is calculated by the ICP algorithm. In our implementation, this value was used to rank ICP alignment solutions for different initial poses. Occasionally, incorrect alignments received high scores.

The sensed point cloud only includes one side of the object, whereas the reference model includes all sides. When properly aligned, there are occluded points on the reference cloud with no correspondences on the sensed point cloud. The ICP has thresholds that allow for these points to be filtered out so that they do not affect the score. However, this also allows incorrect alignments to receive good scores in some cases. In a baseline test of three objects where this occurred: the air freshener, mustard, and peanut butter as shown in Figure 5, 4 out of 10 pose estimations were found to be incorrect.

To address this, we extended our pose estimation algorithm to compute the aspect ratios of the aligned reference cloud and the sensed cloud. We first project the measured and reference point clouds onto the camera plane. For each of the resulting 2D point sets, we compute the second order moment to find the principal axis in the 2D plane. We reject the alignment if the angle between the principal axes is above a threshold (in our tests, we used $\pi/10$). Using this new method, 9 of 10 pose estimations were correct. In the failure case, the estimate was 180° from the correct pose. The shapes are very similar in this case, and the second order moment was not sufficient to detect it.

VII. Discussion and Future Work

This paper presents a system architecture, implemented prototype, and initial experiments and analysis for Cloud-based object recognition and grasping. Object recognition is performed in the cloud using a variant of the Google Goggles proprietary object recognition engine. We incorporated open-source software for pose estimation and grasping and introduce a sampling-based approach to pose uncertainty in 3D grasping. While we are encouraged by initial results, much remains to be done to allow such a system to be scaled up for many objects (and robots).

In our future work, we will seek to improve each aspect of the system. For image recognition, we will seek more details of the Google implementation and work to incorporate more sophisticated analysis of the confidence measures and match sets to reduce false positives and false negatives. For pose estimation, we will introduce precise CAD models of object geometry rather than relying on 3D point sets, and combine information about stable poses, surface patches, and color information to streamline pose estimation. We will also refine grasp analysis based on precise CAD models of each object, stable pose models, and sampling for robustness to uncertainty in object shape. We will also develop active approaches to low confidence values. For example, when the object recognition engine returns results with low confidence, the robot can move its camera to obtain subsequent images. Similar strategies can be developed to respond to low confidence in pose estimation and grasping.

VIII. Acknowledgments

We thank Pieter Abbeel, Dmitry Berenson, Arjun Baokar, Zoe McCarthy, Sachin Patil, Lumin Zhang, Chaitanya Gharpure, and Arshan Poursohi for advice and technical contributions. This work was supported in part by NSF Awards 0905344 and 1227406.

References

[1] Google Cloud Storage. http://cloud.google.com/products/cloud-storage.html.
[2] Google Goggles. http://www.google.com/mobile/goggles/.
[3] IEEE Society of Robotics and Automation Technical Committee on Networked Robotics.
[4] OpenRAVE. http://openrave.org/.
[5] PCL: The Point Cloud Library. http://pointclouds.org.
[6] rosjava, an implementation of ROS in pure Java with Android support. http://cloudrobotics.com.
[7] Willow Garage PR2. http://www.willowgarage.com/pages/pr2/overview.
[8] The African Robotics Network (AFRON). "Ten Dollar Robot" Design Challenge Winners. http://robotics-africa.org/design_challenge.html.
[9] Rajesh Arumugam, V.R. Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, F.F. Kong, A.S. Kumar, K.D. Meng, and G.W. Kit. DAvinCi: A Cloud Computing Framework for Service Robots. In *IEEE International Conference on Robotics and Automation*, pages 3084–3089. IEEE, 2010.
[10] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A Robot Path Planning Framework that Learns from Experience. *IEEE International Conference on Robotics and Automation*, pages 3671–3678, May 2012.
[11] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *IEEE International Conference on Robotics and Automation*, pages 348–353. IEEE, 2000.
[12] M. T. Ciocarlie and P. K. Allen. Hand Posture Subspaces for Dexterous Robotic Grasping. *The International Journal of Robotics Research*, 28(7):851–867, June 2009.
[13] Matei Ciocarlie, Kaijen Hsiao, E. G. Jones, Sachin Chitta, R.B. Rusu, and I.A. Sucan. Towards Reliable Grasping and Manipulation in Household Environments. In *Intl. Symposium on Experimental Robotics*, pages 1–12, New Delhi, India, 2010.
[14] Matei Ciocarlie, Caroline Pantofaru, Kaijen Hsiao, Gary Bradski, Peter Brook, and Ethan Dreyfuss. A Side of Data With My Robot. *IEEE Robotics & Automation Magazine*, 18(2):44–57, June 2011.
[15] Alvaro Collet, Dmitry Berenson, Siddhartha S. Srinivasa, and Dave Ferguson. Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation. In *IEEE International Conference on Robotics and Automation*, pages 48–55. IEEE, May 2009.
[16] Zhihui Du, Weiqiang Yang, Yinong Chen, Xin Sun, Xiaoying Wang, and Chen Xu. Design of a Robot Cloud Center. In *International Symposium on Autonomous Decentralized Systems*, pages 269–275. IEEE, March 2011.
[17] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–45, September 2010.
[18] Roy T. Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, May 2002.

[19] Ken Goldberg and Ben Kehoe. Cloud robotics and automation: A survey of related work. Technical Report UCB/EECS-2013-5, EECS Department, University of California, Berkeley, Jan 2013.

[20] Ken Goldberg, Michael Mascha, Steve Gentner, Nick Rothenberg, Carl Sutter, and Jeff Wiegley. Desktop teleoperation via the World Wide Web. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 654–659. IEEE, 1995.

[21] Ken Goldberg and Roland Siegwart, editors. *Beyond Webcams: An Introduction to Online Robots*. MIT Press, 2002.

[22] Eric Guizzo. Cloud Robotics: Connected to the Cloud, Robots Get Smarter, 2011.

[23] Y. Hirano, K. Kitahama, and S. Yoshizawa. Image-based Object Recognition and Dexterous Hand/Arm Motion Planning Using RRTs for Grasping in Cluttered Scene. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2041–2046. IEEE, 2005.

[24] Dominique Hunziker, Mohanarajah Gajamohan, Markus Waibel, and Raffaello D'Andrea. Rapyuta: The RoboEarth Cloud Engine. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6-10 2013 (to appear).

[25] Katsushi Ikeuchi. Generating an interpretation tree from a CAD model for 3D-object recognition in bin-picking tasks. *International Journal of Computer Vision*, 1(2):145–165, 1987.

[26] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *European Conference on Computer Vision*, pages 304–317, Marseille, 2008.

[27] Ben Kehoe, D Berenson, and K Goldberg. Estimating Part Tolerance Bounds Based on Adaptive Cloud-Based Grasp Planning with Slip. In *IEEE International Conference on Automation Science and Engineering*. IEEE, 2012.

[28] Ben Kehoe, Dmitry Berenson, and Ken Goldberg. Toward Cloud-based Grasping with Uncertainty in Shape: Estimating Lower Bounds on Achieving Force Closure with Zero-slip Push Grasps. In *IEEE International Conference on Robotics and Automation*, pages 576–583. IEEE, May 2012.

[29] James J. Kuffner. Cloud-Enabled Robots. In *IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, 2010.

[30] G. McKee. What is Networked Robotics? *Informatics in Control Automation and Robotics*, 15:35–45, 2008.

[31] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost. In *European Conference on Computer Vision*, Florence, Italy, 2012.

[32] A.T. Miller and P.K. Allen. GraspIt! A Versatile Simulator for Robotic Grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, December 2004.

[33] D Nister and H Stewenius. Scalable Recognition with a Vocabulary Tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168. IEEE, 2006.

[34] D. Parikh and K. Grauman. Relative Attributes. In *International Conference on Computer Vision*, Barcelona, Spain, 2011.

[35] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007.

[36] Szymon Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE Comput. Soc, 2001.

[37] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision*, 77(1-3):157–173, October 2007.

[38] Radu Bogdan Rusu. From Partial to Complete Models. In *Semantic 3D Object Maps for Everyday Robot Manipulation*, volume 85 of *Springer Tracts in Advanced Robotics*, pages 61–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[39] A. Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic Grasping of Novel Objects using Vision. *The International Journal of Robotics Research*, 27(2):157–173, February 2008.

[40] Zheng Song, Qiang Chen, Zhongyang Huang, Yang Hua, and Shuicheng Yan. Contextualizing object detection and classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1585–1592. IEEE, June 2011.

[41] A Sorokin, D Berenson, S S Srinivasa, and M Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2117–2122. IEEE, October 2010.

[42] Alexander Sorokin and David Forsyth. Utility Data Annotation with Amazon Mechanical Turk. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, number c, pages 1–8. IEEE, June 2008.

[43] George Stockman. Object Recognition and Localization via Pose Clustering. *Computer Vision, Graphics, and Image Processing*, 40(3):361–387, December 1987.

[44] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2010.

[45] Luis von Ahn. Human Computation. In *Design Automation Conference*, page 418, 2009.

[46] Markus Waibel. RoboEarth: A World Wide Web for Robots, 2011.