The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

People Helping Robots Helping People: Crowdsourcing for Grasping Novel Objects

Alexander Sorokin Dmitry Berenson Siddhartha S. Srinivasa Martial Hebert

Abstract— **For successful deployment, personal robots must adapt to ever-changing indoor environments. While dealing with novel objects is a largely unsolved challenge in AI, it is easy for people. In this paper we present a framework for robot supervision through Amazon Mechanical Turk. Unlike traditional models of teleoperation, people provide semantic information about the world and subjective judgements. The robot then autonomously utilizes the additional information to enhance its capabilities. The information can be collected on demand in large volumes and at low cost. We demonstrate our approach on the task of grasping unknown objects.**

I. Introduction

Deploying autonomous mobile and dexterous robots in our homes presents a number of challenges: building flexible and inexpensive hardware, developing systems and algorithms to control the robot to achieve planned tasks, bridging the gap between human perception and the robot's world models, just to name a few.

There are numerous unsolved research challenges even for a specific task of cleaning up a room. One key challenge is the availability of accurate models for robustly and safely picking up objects. The models of the environment can be built by hand, they can be derived from CAD models or they can be constructed automatically. To automatically build the models, the robot must recover the geometry and find unique features for recognition. The robot must also determine what constitutes an object, how to call it and what to do with it.

We propose to design autonomous systems that rely on asynchronous human computation through crowdsourcing. In the long term, we would like to minimize human input. In the short term, we would like to maximize human input to go around hard AI problems like category-level object recognition, and enable scalable deployments of personal robots. For the cleanup task, whenever the robot encounters a novel object or an unknown situation, it will request detailed analysis of the situation and only take safe actions until the situation has been explained.

Historically, the failures of robot autonomy are mitigated through teleoperation. The operator constantly monitors one or more robots, drives the robot when necessary or provides high level goals. The defining characteristics of teleoperation is real-time presence of the operator and the use of a single operator per robot. In contrast, crowdsourcing allows hundreds of people teaching a single robot only when necessary.

We demonstrate our approach on the task of grasping novel objects using the framework illustrated in Fig. 1. We use state of the art modelling tools[1], [2], [3] to build the object model: SIFT-based model for recognition and pose estimation; a surface mesh for grasping. The modelling pipeline requires human-provided segmentations of objects of interest and provides multiple tuning parameters. We crowdsource image labeling, object clustering and selecting the final model to Mechanical Turk. With these tasks powered by people, we demonstrate that our robot can build models for novel objects and successfully manipulate them.

While the framework is conceptually simple, we encountered several research challenges.

Workers typically have limited engineering background, and very limited attention spans. Breaking down the complex novel object discovery problem into simpler subproblems that were easy to describe and reliably executable was a huge challenge. Workers often produced unusable output if the tasks were too complex or were described imprecisely. Constructing the correct interface for workers to use for image annotation proved to be another challenge.

A bigger challenge was automated quality control. Producing a 3D model of the object required good user input and the tuning of several algorithm parameters. The quality of the output was quite sensitive to both of those factors. We used a combination of averaging, grading, and a hierarchy of evaluators to automatically weed out bad user input and to automatically tune our algorithm parameters. This proved to be critical for grasping success.

Another challenge is system latency: the delay between sending out an image query and obtaining a segmented result. While it may seem at first sight that such a delay might be unacceptable for object grasping, we were convinced after extensive experiments that it was acceptable with proper scheduling. A personal robot typically has a long list of tasks it needs to perform. If it encounters an unknown object in its first task, it could send in the images for query, and move on to other tasks while awaiting a response. This is much like a human shopper at any store, picking up other objects in the list while waiting for the salesperson to show up.

This paper postulates a concept, that unlimited inexpensive human help is available online and can be harnessed to solve problems that are hard for robots but easy for humans. We believe that it is a first step towards greater adoption of crowdsourcing in mobile manipulation.

Alexander Sorokin is with the University of Illinois at Urbana-Champaign, syrnick@gmail.com

Dmitry Berenson and Martial Hebert are with the Robotics Institute, Carnegie Mellon University, dberenso,hebert@ri.cmu.edu

Siddhartha S. Srinivasa is with Intel Labs Pittsburgh, siddhartha.srinivasa@intel.com

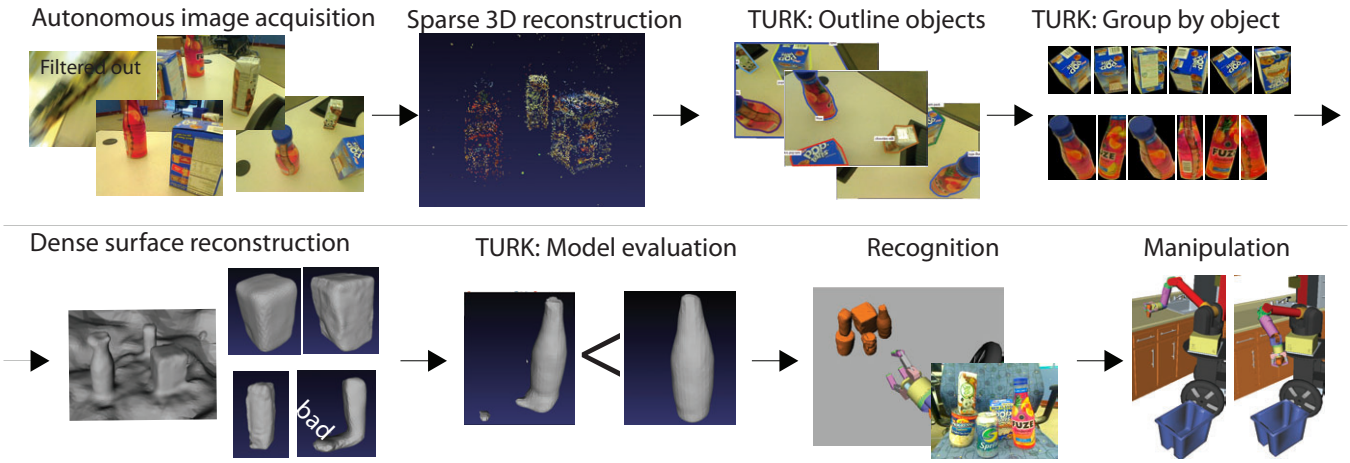

Fig. 1. Modeling pipeline starts with autonomous image acquisition. Blurry images are removed. The sparse 3D cloud is reconstructed using Bundler and valid images are selected for annotation. Workers provide object outlines and group the objects by type. Each object group generates several watertight models depending on the modelling parameters. Workers compare resulting models and only the best models are selected. The resulting models are evaluated on object manipulation task. A demo video of our system is available at http://peopleforrobotsforpeople.com/video/iros2010_movie.mov

II. Mechanical Turk

Amazon Mechanical Turk [4] is a marketplace for micro-tasks. Each task requires human judgment and provides some monetary reward. Each task defines what needs to be done, the user interaction, the quality requirements and who can work on it. Common tasks include content filtering, audio transcription and online inventory categorization. Most tasks require very short amounts of time to complete and provide payments in the range US $0.01-US $0.2.

The tasks are created by *requesters*, who request the services from *workers*. Workers are free to choose any task and complete it. Once the task is submitted by the worker, it becomes the responsibility of the requester to validate it. If the requester accepts the work, Amazon gives the payment to the worker on behalf of the requester. The requester has the option of rejecting the submission. In this case the worker receives no payment and the rejection is counted in the worker's statistics. Requesters are expected to accept all work performed in good faith and reject only malicious and negligent work.

The requester has additional control over who can work on their tasks. Mechanical Turk maintains a set of metrics for each worker: their task approval rate, how many tasks they have submitted, their location, etc. It is common to require that the approval rate be above 90%.

Mechanical Turk has very limited tools for quality assurance. The most powerful of them are qualification tests. The worker takes a test and receives a score of his or her performance. Unless the workers obtain a minimum score, they are blocked from performing the work. Even simple tests reduce the amount of spam and improve the quality of submissions. Finally, there are companies (e.g. CrowdFlower [5]) who provide commercial services to control quality on a large range on tasks. Their services charge a small fee over labor costs in exchange for quality guarantees. Robotics is currently too narrow of a domain to have commercial crowdsourcing solutions. We hope that as application of crowdsourcing becomes more established in robotics, it will be fully-supported by commercial vendors.

The main advantage of Mechanical Turk is the availability of a highly scalable on-demand workforce. There are thousands of people participating on the web site and hundreds of thousands of tasks are posted and get done daily.

A. Design constraints

Mechanical Turk has a number of limiting factors, that constrain what can be done and where it will be effective. First, all interactions on Mechanical Turk happen over the Internet. This delays any communication between the robot and the supervisor. These delays make very accurate real-time teleoperation impossible, which suggests that Mechanical Turk is more suitable for higher level tasks rather than low-level control.

Second, the system works particularly well for large volumes of simple tasks. Complex tasks must be split into simpler sub-problems that can be solved independently and in short time spans. Such structure however has an advantage, because each simple task presents a good target for automation. If an algorithmic solution becomes available, it can be used directly instead of human input. At the same time the human-powered application will generate necessary volumes of training and benchmark data. The small time-span of the tasks also requires the instructions to be short and easy to understand. As each worker will spend little time doing the tasks, they will also spend little time reading the instructions.

Third, workers on Mechanical Turk generally have no engineering background. They have varying command of English and varying levels of education. The tasks must thus be designed for the general public and rely only on a basic level of human abilities. Whenever specific skills are necessary, appropriate instructions must be given and the skills must be verified. For example, if strong command of English is required, a language test is appropriate. Alterna-

tively, worker location may be constrained to only English-speaking countries.

Fourth, Mechanical Turk is an open service with minimal bounds between the requester and the worker. The financial motivation of the workers is low. As a result, it is virtually impossible to guarantee that the data posted on Mechanical Turk will be kept private. At present, most tasks are open to any public observer. This requires special attention if Mechanical Turk is used in sensitive domains, such as outdoor surveillance. In the context of personal robotics, this will be a minor issue. We believe that advantages of effective autonomy will outweigh the privacy risks.

B. Task definition

To obtain human supervision, it is necessary to formalize the interaction with the user and build appropriate user interfaces. Building a new user interface for every new task is very expensive. An alternative solution is to use a general-purpose configurable tools that will accept formal specification of the interaction and generate the appropriate user interface automatically. Such tools are then used as building blocks to define specific human intelligence tasks.

Mechanical Turk [4] and Crowdflower [5] both provide task design interface and markup languages. They cover standard form-like user inputs: text boxes, multiple-choice and checkbox questions. However, these interfaces are not sufficient for annotation of objects in images.

We developed a general purpose annotation toolkit [6] that allows to obtain annotation of images with commonly used primitives: bounding boxes, object outlines, object segmentation masks, object attributes and text labels.

C. Quality control

Quality control on Mechanical Turk is an area of active research. The naive method to ensure quality annotations is to obtain multiple annotation and average. This methodology is necessary where subjective human judgment is required and where no definitive answer is possible. It has been shown [7], that very few workers can outperform an expert annotator at a fraction of the cost. In our tasks, we use multiple judgments in grouping and model evaluation to ensure completeness of coverage and robustness to individual errors. Averaging the annotations is not a universal tool. First, not all annotations can be averaged. Second, averaging requires at least 3 judgments to be collected. If all are correct, then much work is wasted.

One alternative to averaging is to use grading [8]. In grading, a worker looks at a small number of submissions (4-10) and assigns a numeric grade to each. In majority of tasks, grading is much simpler than the task itself. The amount of extra work is only a fraction of the actual work. Unfortunately, grading requires verified and trusted worker base. At a small scale of a few thousand tasks it will be difficult to establish.

At low volumes of tasks, it is possible to use a supervisor - a single dedicated and trusted grader. This approach would work for a few thousands of tasks per day. At higher volumes the grading will become a bottleneck. In such case the grading must be delegated to the Turk and the supervisor will only adjudicate the cases where grades disagree.

There is a relatively large number of spammers who intentionally violate the required protocol. They submit blank annotations or do random things in hope of receiving the payment. These workers are spotted automatically by looking at the metadata associated with the submissions. In particular, submissions with absolutely no work are rejected. Spammers are identified by multiple incorrect submissions. Once a worker is declared a spammer, all their submissions are automatically rejected and they are blocked from performing any future work.

Finally, some systems(e.g. Crowdflower [5]) use gold standard data to automatically evaluate workers performance. Such automated Q/A allows the system to determine how much a particular worker can be trusted.

III. System components

We used HERB [9] - a personal robotics platform. It is built on a Segway RMP200 platform. It has Barrett WAM arm, Barrett Arm, two onboard computers, multiple cameras and laser scanners. The robot is capable of autonomous navigation, obstacle and people avoidance, safe arm motion planning and grasping.

A. Image acquisition

To build the models of unknown objects, the robot collects data from 4 base locations around the table. At each location, arm motion is planned and safely executed to reach several requested views of the object. The images collected in each run are filtered by information content. We measure the amount of gradient energy and take a local maximum. This selection removes blurry images that significantly confuse the reconstruction algorithm.

B. Sparse 3D reconstruction

To build the models, we use MOPED modelling system [2] based on Bundler [1]. It creates an accurate reconstruction of SIFT feature 3D positions and 6 DOF camera poses. We provide camera calibration parameters to the reconstruction engine. The reconstruction obtained from a single camera has a single unknown - scale. When a large known object is present (such as a poster in figure 3), we use it to recover the scale. Once the calibration object is detected, we know the correspondence between reconstructed points in the scene and the model. This gives us the scaling factor of the reconstruction. When the calibration object is not available, we can use approximate camera locations measured from the robot arm configuration. Although the camera locations are not accurate enough for 3D modelling, the error in distance between the arm positions is sufficiently low to obtain accurate scaling of the 3D reconstruction.

The sparse reconstruction provides us with the list of images that were correctly registered. These images are submitted for annotation on Mechanical Turk.

Fig. 2. Annotation user interfaces: outlines, object grouping, model evaluation. Task costs and volumes used.

C. Image annotation and grouping

To obtain annotations on Mechanical Turk, we use open source annotation toolkit [6]. All annotation tasks were submitted to the server with a fixed cost per task(fig. 2). After the tasks were completed, worker submissions were manually validated.

To get object outlines, we require each worker to draw at least one outline in an image and type the name of that object. When no further outlines are possible, the worker marks the "all done" flag and the image is not considered for further annotation. The annotation interface and example results is shown in figure 2. Object labels are collected for future analysis. We currently use only object outlines to create 3D models.

As there are multiple objects in each view, we need to distinguish between different objects. Object labels provided by the workers are not consistent across different workers, so we need to associate the exemplar objects more directly. We use image grouping tasks to obtain explicit judgments about which objects are the same and which are different. We present 20 masked object images to a worker and ask them to place the objects in bins. Each masked image is automatically generated from the object outlines obtained at the previous round of annotations. Images placed into the same bin are considered the same object. When two images are placed into different bins, they are considered different objects. The output of this annotation is an object similarity graph: positive links connect similar instances, negative links connect dissimilar instances. We cluster objects using this sparse affinity graph and discard clusters with less than 5 members. Each cluster contains object masks that are used for dense 3D reconstruction.

D. Dense 3D reconstruction

MOPED models give us only 3D pose of the object and a sparse collection of 3D visual features associated with the object. For grasping, we need a surface model with normal information. We refine the sparse reconstruction into a dense model using Patch-Based Multi-View Stereo (PMVS) package [3]. The algorithm works like conventional stereo, except for the camera positions are arbitrary. The algorithm requires masks of the regions of interest and reconstructs only respective parts of the world. These were obtained from Mechanical Turk in Sec. III-C.

PMVS reconstructs a single oriented 3D patch at every K-th pixel in the image by minimizing photo-consistency errors between different camera views that would see that 3D patch. The photo-consistency is measured via correlation of re-projected patches. Each patch has size MxM pixels, which needs to be chosen appropriately. The algorithm performs multiple reconstruction passes relaxing the photo-consistency requirement at each pass. First, the most consistent patches are placed in 3D. Second, their neighbors are placed nearby if they satisfy less stringent consistency requirement. On the third pass, the consistency is relaxed once again and more patches are added. There are several parameters for to *tune*: K - the density of the reconstruction, σ - the scaling of images before the reconstruction occurs, M - the size of the patch for photo-consistency measurement, τ - the most stringent photo-consistency threshold. As no single combination of parameters always produces the best model, we use 16 different parameter settings to generate multiple models. Each model will be later judged and only the best model will be selected for the robot to use.

E. Meshing

The model obtained on the previous stage consists of a set of oriented patches. For grasp planning it needs to be converted into a trimesh. We use Poisson surface reconstruction algorithm [10], which we briefly summarize here. Oriented 3D patches form the vector field $\vec{V}$. This vector field can be seen as a gradient field for the indicator function χ of the surface. As shown in [10], the surface indicator function can be efficiently recovered as the solution of the Poisson problem: $\Delta\vec{\chi} = \nabla\cdot\vec{V}$. The water-tight triangular mesh is then extracted from the solution. The implicit representation of the surface efficiently handles the noise inherent to vision-based measurements of 3D point locations and normals.

F. Model evaluation

Each object has multiple models corresponding to different reconstruction parameters. Each model is textured and

rendered using Blender [11]. The videos of different models of one object are randomly shuffled and grouped in sets of 4. Each group is presented to the worker. The task is to assign a grade from 1(bad) to 10(perfect) to each model. The workers are explicitly instructed that the better-looking model must have higher score. All models of the object are randomly shuffled and presented 3 times. After all models are graded, the best model for each object is selected as the final model. As we will see in the experiments, the model evaluation step currently produces many errors.

G. Object recognition and pose estimation

We use MOPED system [2] for object recognition. The recognition module requires only a single calibrated camera and provides full 6DOF pose of detected object. MOPED extracts SIFT features in the image and matches them against the 3D models in the database. Once the feature correspondence is established, full 6DOF pose is recovered using RANSAC and verified. The recovered pose is highly accurate and GPU-based feature extraction gives real-time recognition with hundreds of models [2].

H. Grasping and manipulation

Once we obtain a model for the object and its pose in the environment we compute grasps for the object using the algorithm presented in [12]. This algorithm first samples the surface of the object and computes distances from surface points to the obstacles in the environment. These distances are used to inform the cost function of an optimizer, which quickly generates a set of grasps that are likely to be in force-closure and collision-free. The grasp set is then checked for collision and whether or not the grasp is reachable by the arm.

We then compute the force-closure score for all reachable and collision-free grasps and pick the highest-scoring grasp. We compute Inverse Kinematic (IK) for this grasp, which gives us the joint values of the arm that place the end-effector in the proper pose. These joint values are passed to a planning algorithm based on Rapidly-Exploring Random Trees (RRT) [13], which computes a collision-free path from the current configuration of the arm to the configuration given by IK. Once this path is executed, we close the fingers to grasp the object.

IV. Experiments

To validate the presented approach to grasping novel objects, we performed 6 rounds of reconstruction with varying difficulty: with and without a calibration target, single and multiple objects, with and without clutter. The ideal model must provide reliable recognition, accurate 6DOF pose estimation and successful grasping.

To evaluate if the models we build are useful for grasping, we validated whether the robot can detect the object, grasp it, lift it and drop it into the recycling bin. The objects were placed one at a time on the table and the robot arm was positioned so that the palm camera can see the object. The reconstructed object models were used for detection, grasp and motion planning. The grasps were generated using the method described in Section III-H, arm motions were planned using using RRTs. We measured two metrics: the overall task completion rate and the flawless execution rate. In the first case only the fact that the robot places the object in the bin is counted. In the second case, failure is declared if the robot grasps the object incorrectly or if the robot or the object touches any stationary objects in the environment.

We used common household items found in any grocery store: horizon chocolate milk box, soy on the go cappuccino drink box, box of pop tarts, salt, Progresso clam chowder can, bottle of Fuze, Sprite can and a medicine bottle.

To measure the efficiency of the qualification requirement, we run a smaller subsets of 140 grouping and 100 outline tasks. The qualification requirements are simple multiple choice tests measuring that the workers understand the instructions. The workers must take the test before they can work on the tasks.

A. Experimental results

Of the 6 rounds of reconstruction, 5 rounds were successfully reconstructed and produced 13 models (figure 3). One round of reconstruction failed, because the scene contained too little visual information with our current choice of features. Of the 13 models 2 were rated unusable (best model scored below 6) by Mechanical Turk. Most models posses visible defects on the surface, however they all closely follow the actual object surface. As a result they were sufficient for the grasp planning and successful grasping. A short video clip demonstrating our system in action is available at `http://peopleforrobotsforpeople.com/video/iros2010_movie.mov`.

We manually verified all Mechanical Turk submissions and only good submissions were used in the subsequent stages of the pipeline. The results of the evaluation are given in table I. Our findings are consistent with the literature [7], [14]: a single average worker produces rather poor results. By averaging 3-5 of them, we can obtain desired high accuracy (after filtering out trivial spam) at the price of higher annotation cost. We found that qualification requirements significantly improve the results (Table. I). Although 100% performance on the outlines seems too high to us, it corroborates our findings on a different task unrelated to this paper. On a simple task of providing boxes around people, we observed 10 errors out of 4000 tasks. In general we don't expect such high accuracy. We expect average good workers to produce 90%-95% of correct submissions once they fully understand the task requirements. The qualification-based model evaluation task is still work in progress. We are actively working towards for a completely automatic and verifiable quality assurance strategy.

Our main test was the success rate of the grasping and manipulation task. In 61.9% of the runs, the robot executed the tasked flawlessly: detection, grasp planning, grasping, lifting the object and placing it in the recycling bin. This rate excludes any errors: executing different grasp, touching the table with the object, dropping the object due to insufficient

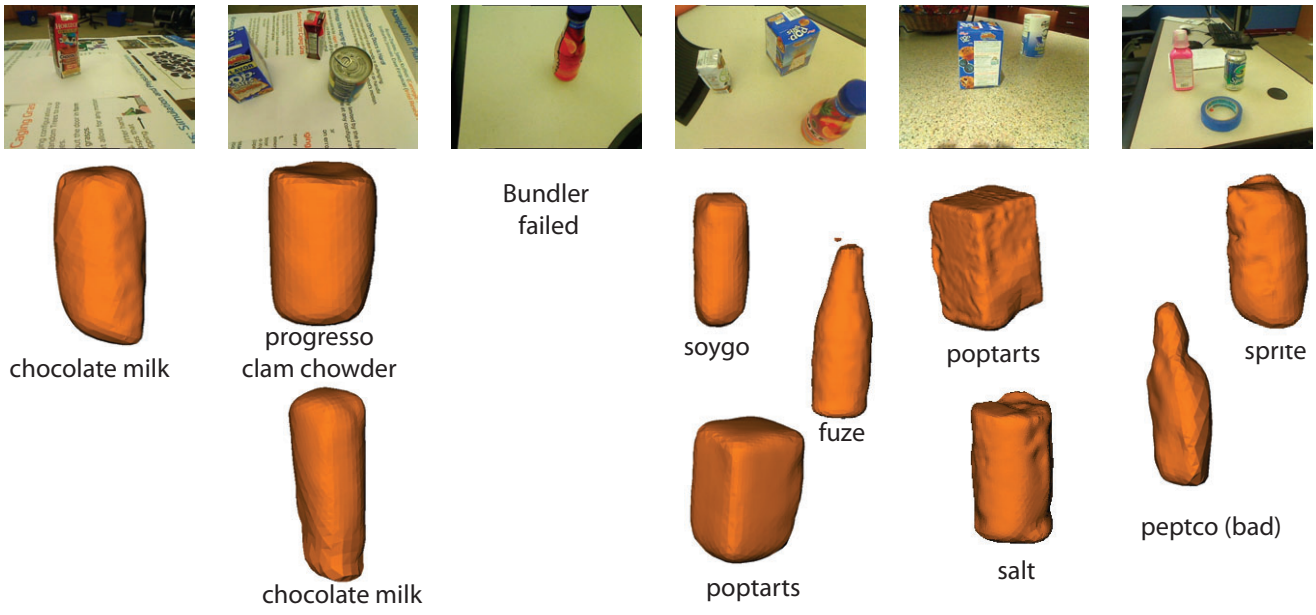

Fig. 3. Reconstructed models follow the shapes of the objects well. Visible defects are caused by (1) low camera resolution (640x480) (2) limited visibility at contacts with the table (3) featureless regions (peptco, duck tape), (4) non-planar and specular surfaces (e.g. sprite can)

TABLE I
MECHANICAL TURK SUBMISSION ACCURACY.

Task	Good	Minor errors	Bad	Requirements
Outlines	76.98%	2.44%	20.58%	none
	100.00%	0.00%	0.00%	qualification
Grouping	77.95%	11.81%	10.24%	none
	93.30%	6.67%	0.00%	qualification
Evaluation	46.99%	9.04%	43.98%	none

force. In 85.7% of the runs, the robot succeeded in placing the object in the recycling bin despite minor errors, such as sliding the object along the table. The smaller and more rounded objects were harder to grasp and manipulate. The worst object was the sprite can. It's rounded, metallic and reflective surface violated assumptions of the reconstruction algorithm resulting in a noisy model. Smaller models had more minor failures than bigger models. We believe the smoothing in Poisson reconstruction generates slightly bigger surface models than they should be. This would cause more problems for smaller objects than for bigger ones. We are looking into calibrating the reconstruction pipeline with known ground truth models.

V. CONCLUSION

Virtually unlimited amounts of human supervision are available through online work marketplaces. This supervision will greatly improve the robustness of deployed systems and allow us to sidestep hard AI problems on our way to autonomous robots. Rather than building a completely automatic system, we engineer our system to rely on massive human feedback. This shift in thinking opens up new challenges in algorithm design, user interface design, quality assurance policies and learning.

ACKNOWLEDGMENT

We thank Willow Garage, Gary Bradski, Caroline Pantofaru, Ethan Dreyfus, Mehmet Dogar, Mike Vande Weghe, Alvaro Collet, and Manel Martinez for helpful discussions. We thank Intel Labs Pittsburgh for sponsoring this project. We thank Yasu Furukawa and Michael Kazhdan for providing their reconstruction software. David Forsyth provided us invaluable advice and support.

REFERENCES

[1] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from Internet photo collections," *IJCV*, 2008.
[2] M. Martinez, A. Collet, and S. S. Srinivasa, "MOPED: A Scalable and low Latency Object Recognition and Pose Estimation System," in *ICRA*, Anchorage, 2010.
[3] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *PAMI*, 2009.
[4] "Amazon mechanical turk," http://www.mturk.com/.
[5] "Crowdflower," http://www.crowdflower.com/.
[6] "Web-based annotation toolkit for computer vision," http://code.google.com/p/cv-web-annotation-toolkit/.
[7] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks," in *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.
[8] "Castingwords," http://castingwords.com/.
[9] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe, "Herb: A home exploring robotic butler," in *Autonomous Robots*, 2009.
[10] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.
[11] "Blender," http://www.blender.org/.
[12] D. Berenson and S. Srinivasa, "Grasp Synthesis in Cluttered Environments for Dexterous Hands," in *Humanoids08*, December 2008.
[13] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, 2000.
[14] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *First International Workshop on Internet Vision at CVPR*, 2008.