A Robot in the First Person

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

***Abstract*—Manipulating objects is an important task for robots that help people in the home, in factories, and in hospitals. General-purpose pick-and-place requires object recognition, pose estimation, and grasp planning; existing solutions cannot reliably recognize or pick up an object the robot has never encountered before []. However in many applications, general-purpose pick-and-place is not required: the robot would be useful if it could recognize and manipulate the small set of objects most important in that application, but do so with high reliability. To address this issue, we propose an architecture for perception and actuation that enables a robot to quickly and easily acquire instance-level descriptions of novel objects. The robot learns to recognize, estimate the pose, and grasp the object through active data collection as well as by asking a human annotator for specific supervision. Our approach converts the task of *category recognition* (pick up any mug) to *instance recognition* (pick up this mug), a much easier problem that conventional methods can solve with the right training data. By leveraging this first-hand experience with the object, naive users can quickly train our system to reliably localize and grasp the specific objects that they care about for their particular application. As a byproduct, our approach collects data on grasping and manipulation that we can use to train general-purpose models. Using our approach, a robot can interact with an object for ten minutes, and then reliably localize (90%) and manipulate it (90% successful grasps).**

I. Introduction

Robotic assistants will assist us at childcare, help us cook, and provide service to doctors, nurses, and patients in hospitals. Many such tasks require a robot to robustly perceive and manipulate objects. **ST: I would cut the previous two sentences, to avoid getting so specific about the robot so quickly. JGO: Rather than cut I decided to move it.**

Conventional systems are capable of perception and manipulation in a limited sense. Some systems require training by a human operator on an object to object basis, which is time consuming and can be difficult for a non-expert to perform [**? ? ?**]. There do exist some systems which do not require training on a per object basis, but they are computationally expensive and do not enjoy the highest accuracy or precision and have not been demonstrated for grasping [**?**].

To obtain the benefits of both approaches, we propose a system which trains itself to recognize and manipulate the specific objects it will need to use during future collaborations with humans. **ST: s/It is powerful because it/Our system JGO: ACK** Our system is powerful because it learns to identify and grasp on a per object basis. Our system is portable, convenient, and general because the expert knowledge it employs is built into the algorithms which it uses to train itself, requiring only basic interaction from a non-technical human collaborator.

ST: This paragraph feels a little misleading because we end by saying that these three parts are not new. I think it's important to headline and emphasize the part that is new. It might help to move the "Our contribution" sentence from the next paragraph up hear, and then discuss the three parts. JGO: ACK

Our contribution is an algorithm which allows a robot to autonomously train its subsystems, together with three applications of the algorithm to the tasks mentioned above. The first application is recognizing the category of an object and the second application is estimating the pose of the object, both of which we accomplish with simple and robust computer vision algorithms. The third application is grasping the object, which we accomplish with visual servoing techniques. Each of these components is well understood in its own right, and existing methods allow expert users to train systems to accomplish these tasks satisfactorily.

When we apply the algorithm to the recognition task, the robot trains the recognition system to discriminate between object instance categories. When we apply the algorithm to the pose estimation task, the robot trains the pose estimation system to determine which pose an identified object holds. When we apply the algorithm to the grasping task, the robot trains the grasping system to successfully and quickly pick and place the target object. **JGO: Quickly because perhaps the robot will tune its own PID controller. I will write under the assumption that it will and we can cut it if we don't get around to it.**

Crucially, our algorithm can recognize when it is doing a poor job at learning and asks a human collaborator to manually annotate information in those cases.

It works because our experiments tell us so. This is how well they work: . Thus we see an improvement over expert trained systems (cite usability results), and is an improvement over unannotated systems (cite success rates, the ability to generalize, and the low computational overhead since we don't crunch expensive features or do heavy classification at run time).

II. Related Work

In this section we discuss each of the tasks our system tackles, both at the level of sophistication we are using and at the state of the art. We address typical expert training and automated approaches that exist, together with usability concerns.

A. Object Recognition

JGO: Cover BING, SIFT, BoW, typical training pipelines, and RGB-D approaches popular in the robotics community. Our recognition pipeline takes RGB-D video from the robot, proposes a small number of candidate object bounding boxes in each frame, and classifies each candidate bounding box as belonging to a previously encountered object

class. Our object classes consist of object instances rather than pure object categories.

To generate condidate bounding boxes, we first apply the BING objectness detector [] to the image, which returns a set $\{B_i\}$ of thousands of approximate object bounding boxes in the image. This is a substantial reduction from the set of all bounding boxes in the image, but is still too large for us to process in real time. Besides, even good bounding boxes from BING are typically not aligned to the degree that we require. Therefore, we use integral images to efficiently compute the per-pixel map

$$O(p) = \sum_{B \in \{B_i\} s.t. p \in B} \frac{1}{Area(B)}.$$

We then apply the Canny edge detector with hysteresis [] to find the connected components of bright regions in the map $O(p)$, which correspond with high probability to objects in the image. We then form our candidate object bounding boxes by taking the smallest bounding box which surrounds each connected component. These bounding boxes make it easy to gather training data and to perform inference in real time, but at the expense of poorly handing occulusion as overlapping objects are fused into the same bounding box. It is possible to search within the proposed bounding boxes to better handle occlusion.

For each object c we wish to classify, we gather a set of example crops E_c which are candidate bounding boxes (derived as above) which contain c. We extract dense SIFT features [] from all boxes of all classes and use k-means to extract a visual vocabulary of SIFT features []. We then construct a BoW feature vector for each image and augment it with a histogram of colors which appear in that image. The augmented feature vector is incorporated into a k-nearest-neighbors model which we use to classify objects at inference [].

The use of SIFT features is motivated by the instance level nature of our task. State-of-the-art vision methods typically use HOG [] or CNN [] features, but that choice is motivated by category level recognition.

We use kNN because it is easy to rebuild online, which is a key property a classifier should enjoy if it is to interact with our framework in real time. State-of-the-art computer vision classifiers currently employ SVM's [] or other models which require expensive training. Using such a model would introduce a training step in the inside loop of our data collection process, which would be costly in either engineering or time. It is possible to use kNN during the online collection process and then train a stronger classifier in the background at higher latency, essentially introducing a cascading step in the data collection process.

B. Pose Estimation

JGO: Computer vision approaches, geometry and point cloud based approaches. Dieter Fox's automatic training pipeline (how well developed is it? we may need to sell our approach as being a general algorithm for allowing a robot to train arbitrary subsystems in order to differentiate ourselves.) We tackle the pose estimation problem using the same classification pipeline that we use for object recognition. We train a separate pose classifier for each object class. This time, the class assigned to each training example is the orientation from which the object is viewed in that example. During inference, we first determine the object class of a candidate bounding box, and once the class is known we apply the corresponding pose classifier to determine the orientation from which we are viewing the object. We combine this orientation with position information from the point cloud information derived from the D channel of the RGB-D video to form a full pose estimate.

C. Object Grasping

JGO: Including open and closed loop paradigms, learning specific and generic grasp models.

We use a dual rate PID controller in the sense that we use two sets of PID coefficients. The first set is for making large adjustments when the aim if off by a significant amount. The second set is for making small adjustments when aim is close to the target.

Our system is distributed and thus at times there is an appreciable amount of latency between communicating components. Care is taken to synchronize robot movement with object detection reports, allowing only a fixed amount of movement per report.

III. General Autonomous Training Algorithm

we teach the system by finding its weaknesses and training it to overcome these weaknesses. We provide a framework for structuring that training in an affordable, scalable way. we structure the training, the training structures the models.

"I see you're having trouble picking up gyroBowls. Lets think about it differently (initialize generic pose model) and practice gazing a bit (train the pose model)."

Also, grounding ambiguous object examples to the right class, retraining the models online.

IV. Applications of GATA

A. Recognition Training

B. Pose Estimation Training

C. Grasp Training

We consider a setting where a robotic arm with 7 degrees of freedom grasps objects with a parallel plate gripper which adds an additional degree of freedom, but much of what we discuss could be extended to other arms and grippers, for instance the universal jamming gripper []. **JGO: Moved this here from the first paragraph.**

This Grasp Rectangle business fits in nicely with the reticle.

0. Estimate the depth of the table by inspecting non-object locations. This helps decide when to close the gripper.

1. Servo orientation to the '0' orientation, or one of a few sparsely sampled keypoint orientations. Each viewing orientation (from the wrist) is tied to a grasping orientation.

2. Servo to a 'normal' scale. This is fixed, we don't want multiple scales running around.

3. Now instead of aiming at the center, aim at a proposed target offset from the center.

4. Once aiming is complete, save the image in the reticle, try to grasp, and save whether that grasp completed. This can be ascertained by the gripper position.

5. Calculate a density map for this

D. PID Parameter Training

JGO: Maybe a coordinate descent algorithm or wide-scale random noise search. Since we use a dual-rate controller, there are two separate sets of coefficients that we must train. We train the high-rate coefficients with the objective of getting the aim within the 'close' threshold. We train the low-rate coefficients with the objective of getting the aim within the 'hit' threshold. The training for the high and low-rate coefficients is analogous and happens independently, so without loss of generality we describe the training process for an arbitrary set of coefficients.

JGO: I imagine this has been done before so it would be good to find who did this. A single set of PID coefficients consists of K_P, K_I, K_D. In the inside loop of EM-like training, we randomly pick a coefficient K to train, fix the other two coefficients, and use a local search algorithm [] to find the optimal value of K conditioned on the fixed values of the other two parameters. This problem is not necessarily convex and so we run the inside loop of our algorithm until we have converged to a local minimum.

V. Experimental Setup

JGO: This is where we describe the experiments we performed. We are not trying to extend the state-of-the-art on our individual tasks. Rather, we are providing an interactive framework which will raise the maximum automated vision available to the average user.

Our system can be evaluated in two important ways. Firstly, how effective is the system at the tasks for which it is trained? Secondly, how accessible to users is the system?

A. Object Categorization and Pose Estimation

For object categorization and pose estimation, we constructed data sets on which we could evaluate our models. This involved hand annotating the ground truth for the images in the sets, which is a costly procedure which we are attempting to eliminate for future tasks. However, we cannot evaluate our system in a principled fashion without such a data set.

We demonstrate our method's success in this setting where we pay the cost to acquire the data so that we can trust our method and that cost need not be payed during future applications.

Probably uses confusion rates as the objective function.

Expert viewpoint collection (uses at least hard negatives)

Super dense sampling

uniform hard negative sampling with stopping criterion

B. Grasp Experiments

For grasp experiments, we conducted online trials in order to compute success rates. This uses grasp success rate as an objective function.

Expert annotation of grasps

Uniform grasp sampling

Thompson sampling

C. PID Experiments

For PID experiments, we conducted online trials in order to compute success rates. We use the time to convergence as the objective function

VI. Evaluation and Discussion

We could report the performance of the system as a function of user interactions.

We could report the performance of the system as a function of program lifetime.

Our representative set could consist of a block, a spoon, a bowl, a diaper, and a sippy cup. A *single cut video* showing multiple grasps of all objects is available here.

A. Expert Manual Training

We therefore establish a baseline for performance by training the system in a representative domain specific setting, which tells us how well it can perform on laboratory objects when trained by an expert. This represents the best that the system could be expected to perform.

B. Laboratory Automatic Training

How well does the automatic training system perform when trained in laboratory conditions?

C. Non-Expert In-The-Wild Training

We then go on to compare the performance of the system when trained to various degrees by naive and technical non-expert users.

We repeat the experiments with naive users in two settings. In the first, they train laboratory objects. In the second, they provide their own objects.

[**?**]

VII. Extensions

Right now, NODE runs on Baxter. We will port NODE to PR2 and other AH systems. GATA could be applied in other domains as well. What are some examples?