Autonomously Acquiring Models of Objects for Instance-Based Grasping

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. 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. 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.

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 combined with our proposed algorithm for autonomous collection of training data. 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. 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 un-annotated 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. Object Detection and Pose Estimation

We first describe our instance-based object detection and pose estimation pipeline, which uses standard computer vision algorithms combined to achieve a simple software architecture, a high frame rate, and high accuracy at recognition and pose estimation. This pipeline can be manually trained by an expert to reliably detect and grasp objects. Additionally, section III describes our approach to enabling a robot to autonomously train this pipeline by actively collecting images and training data from the environment.

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. Using instance recognition means we cannot

reliably detect categories, such as "mugs," but the system will be able to detect, localize, and grasp the specific instances for which it has models with much higher speed and accuracy.

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

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

We then apply the Canny edge detector with hysteresis [] to find the connected components of bright regions in the map $J(p)$, which correspond with high probability to objects in the image. We 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 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 [].

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.

Visual servoing tutorial paper: [**?**]

ST: Cite the visual servoing tutorial paper and talk about the connection to grasping rectangles.

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.

D. PID Controller

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

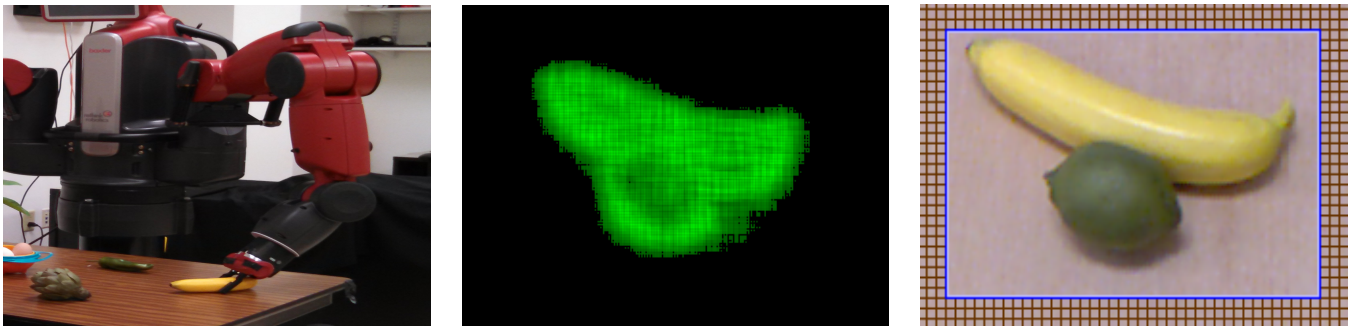

Fig. 1. The Object Detection Pipeline. Left: The raw image as viewed through the kinect. Center: The computed objectness map J. Right: Labeled object detections.

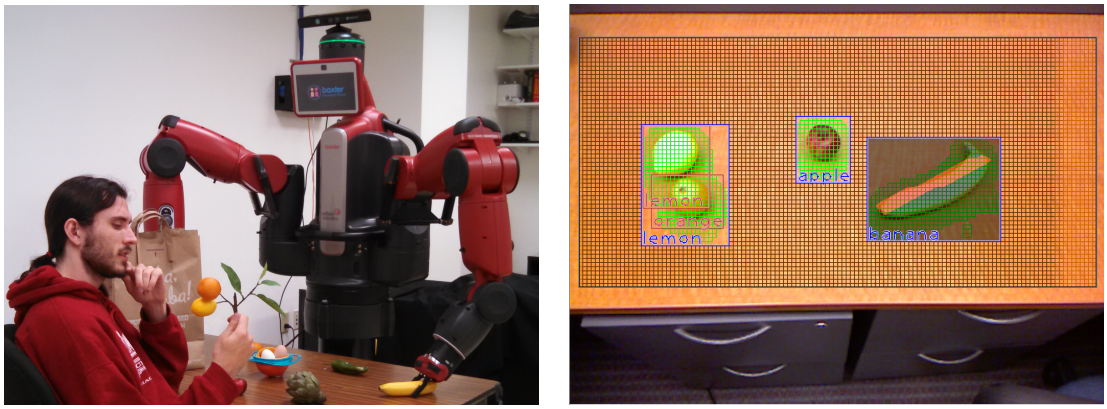

Fig. 2. Left: Baxter uses visual servoing to grasp an object. Right: The view through Baxter's wrist camera, showing the location of the target as well as the objective reticle.

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. ST: Find someone to learn PID coefficients. 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.

III. General Autonomous Training Algorithm

We teach the system by finding its weaknesses and training it to overcome them.

Our framework structures the training in a way that scales.

When the robot asks for help, we can ground ambiguous object examples to the right class and retrain the models online.

In the following algorithm design pattern, X is the set of ground truth labels for the specified problem, Y is the set of all possible observations for the specified problem, $D \subset X \times Y$ is the set of all possible training pairs observable in our problem instance, $\{T_i\}_{i \in \mathbb{N}}$ is a sequence of curated sets of training examples (where $\forall i \in \mathbb{N}, T_i \subset D$), $\mathcal{T}$ is the power set of D, $O : \mathcal{T} \to \mathbb{R}$ is the objective function that we are trying to optimize (which includes the execution of the physical actions the robot must take to evaluate it), $S : \mathbb{N} \to \mathbb{R}$ is the objective function history (i.e. $S(i) = O(T_i)$), $\mathcal{S}$ is the set of all possible objective function histories, $H : \mathcal{S} \times \mathbb{N} \to \{0, 1\}$ is a function which decides whether to ask for help at a specified time step, $\mathcal{C} = \mathcal{T} \times \mathcal{S} \times \mathbb{N}$ is the complete training state, $G : \mathcal{C}$ is a function which implements asking for and receiving help (e.g. auditing and grounding data under bad conditions or asking for a new collections strategy or a crucial example), $P : \mathcal{C} \to \{0, 1\}$ is a function that implements a stopping criterion and decides whether to continue training, and $L : \mathcal{C} \to D$ a function that proposes a new ground truth $x \in X$ and collects a corresponding observation $y \in Y$ to form a new training example $(x, y) = d \in D$.

Applying GATA to a problem involves defining the above quantities and implementing the high level algorithm in the figure [].

GATA

$t \leftarrow 1$
$c \leftarrow 1$
$T_1 = \emptyset$
$S(1) \leftarrow 0$
$i \leftarrow 1$
while c **do**
  if $H(S, i) == 1$ **then**
    $G(T_i, S, i)$
  else
    $d \leftarrow L(T_i, S, i)$
    $T_{i+1} \leftarrow T_i \cup \{d\}$
  end if
  $i \leftarrow i + 1$
  $S(i) \leftarrow O(T_i)$
  $c \leftarrow P(T_i, S, i)$
end while

Fig. 3. The high-level generalized automatic training algorithm (GATA) which we employ.

IV. Applications of GATA

A. Recognition Training

B. Pose Estimation Training

C. Grasp Training

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

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? For now we evaluate system performance, and leave user studies for future work.

A. Object Detection and Pose Estimation

For object Detection 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.

Data Collection Method	Success Rate
Expert Annotation	0.0
Dense Sampling	0.0
Hard Negatives Auto-Stopping	0.0

Fig. 4. Performance of our system on the object detection task.

Data Collection Method	Success Rate
Expert Annotation	0.0
Dense Sampling	0.0
Hard Negatives Auto-Stopping	0.0

Fig. 5. Performance of our system on the pose estimation task.

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 Control 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. Object Detection

We 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. Pose Estimation

C. Grasping

D. PID Control

VII. Related Work

?] created a mobile robot that explores the environment and actively plans paths to acquire views of objects such as doors. However it uses a fixed model of the object being detected rather than updating its model based on the data it has acquired from the environment.

Grasp Sampling Method	Success Rate
Expert Annotation	0.0
Uniform Sampling	0.0
Thompson Sampling	0.0

Fig. 6. Performance of our system on the grasping task.

Parameter Learning Method	Average Convergence Time
Expert Annotation	0.0
Constant Learning Rate	0.0
Decaying Learning Rate	0.0
Wide Scale Random Noise	0.0

Fig. 7. Performance of our system on the PID control task.

Methods for planning in information space [**? ? ?**] have been applied to enable mobile robots to plan trajectories that avoid failures due to inability to accurately estimate positions. Our approach is focused instead on object detection and manipulation, actively acquiring data for use later in localizing and picking up objects. **ST: May need to say more here depending on what GATA actually is.**

Early models for pick-and-place rely on has been studied since the early days of robotics [**? ?**]. These systems relied on models of object pose and end effector pose being provided to the algorithm, and simply planned a motion for the arm to grasp. Modern approaches use object recognition systems to estimate pose and object type, then libraries of grasps either annotated or learned from data [**? ? ?**]. These approaches attempt to create systems that can grasp arbitrary objects based on learned visual features or known 3d configuration. Collecting these training sets is an expensive process and is not accessible to the average user in a non-robotics setting. If the system does not work for the user's particular application, there is no easy way for it to adapt or relearn. Our approach, instead, enables the robot to autonomously acquire more information to increase robustness at detecting and manipulating the specific object that is important to the user at the current moment.

Visual-servoing based methods [**?**] **ST: Need a whole paragraph about that.**

ST: ?] seems highly relevant, could not read from the train's wifi.

Our approach is similar to the philosphy adopted by Rethink Robotic's Baxter robot, and indeed, we use Baxter as our test platform [**?**]. **ST: Haven't actually read this paper, just making stuff up based on Rod's talks. Should read the paper and confirm.** Baxter's manufacturing platform is designed to be easily learned and trained by workers on the factory floor. The difference between this system and our approach is we rely on the robot to autonomously collect the training information it needs to grasp the object, rather than requiring this training information to be provided by the user.

Robot systems for cooking [**? ?**] or furniture assembly [**?**] use many simplifying assumptions, including pre-trained object locations or using VICON to solve the perceptual system. We envision vision or RGB-D based sensors mounted on the robot, so that a person can train a robot to recognize and manipulate objects wherever the robot finds itself.

VIII. Conclusion

ST: First paragraph: contributions. What are the things this paper has done to advance the state of the art?

ST: Next paragraphs: future work, spiraling upward to more and more ambitiuos 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?