Autonomously Acquiring Instance-Based Object Models

John Oberlin and Stefanie Tellex

Abstract A key aim of current research is to create robots that can reliably manipulate generic objects. However, in many applications, general-purpose object detection or manipulation is not required: the robot would be useful if it could recognize, localize, and manipulate the relatively small set of specific objects most important in that application, but do so with very high reliability. The contribution of this paper is to automate this adaptation process by formalizing a grasping system as a hierarchy of bandit problems, enabling the robot to apply learning techniques to adapt itself to manipulate specific objects with increased accuracy. The robot performs best arm identification using a variant of Hoeffding races, enabling it to quickly find an optimal arm and then move on to the next object. Our approach also incorporates knowledge from a prior defined in terms of an ordering on the arms. We demonstrate that our bandit-based adaptation step significantly improves accuracy over a non-adaptive system, enabling a robot to quickly and autonomously acquire models for objects, and adaptively improve them through experience.

1 Introduction

2 Grasping Pipeline

3 Grasping System

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

Brown University

estimation. Section 3.6 describes our approach to enabling a robot to autonomously collect the data needed to perform grasping with this pipeline.

Our recognition pipeline takes 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. A visualization of data flow in the pipeline appears in Figure **??**. **JGO: This doesn't quite seem current:** For each module, we formalize its input, output, and reward function; each component can have multiple implementations which better for different objects. The following sections describe how we can use this pipeline to learn which implementation to use for specific objects; this learning dramatically speeds up performance.

3.1 Object Detection

The input of the object detection component is an image, I; the output is a set of candidate bounding boxes, B. For object detection we use a modified Canny algorithm which terminates before the usual non-maximal suppression step [1].

We start by converting I to YCbCr opponent color representation. Then we apply 5×5 Sobel derivative filters [3] to each of the three channels and keep the square gradient magnitude. We take a convex combination of the three channels, where Cb and Cr and weighted the same and more heavily than Y. After this we downsample, apply the two Canny thresholds, and find connected components. If a connected component is contained in another, we discared the contained component. We throw out boxes which do not contain enough visual data to classify. We generate a candidate bounding box for each remaining component by taking the smallest box which contains the component.

3.2 Object Classification

The object recognition module takes as input a bounding box, B, and outputs a label for that object, c, based on the robot's memory. This label is used to identify the object and look up other information about the object for grasping further down the pipeline.

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 [2] from all boxes of all classes and use k-means to extract a visual vocabulary of SIFT features [4]. We then construct a Bag of Words 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 [4]. We use kNN because our automated training process allows us to acquire as much high-quality data as necessary to make the model work well, and kNN supports direct matching to this large dataset.

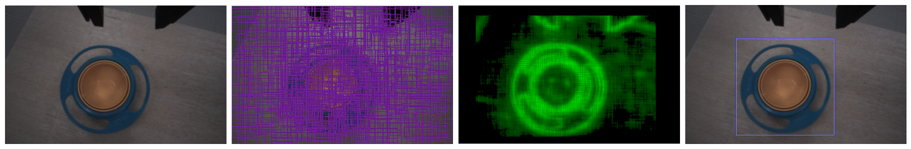
(a) Raw image from the camera. (b) Candidate bounding boxes. (c) Integral image objectness map. (d) Candidate bounding boxes.

Fig. 1 The object detection pipeline, showing a raw image from the camera, the integral image computed using objectness, and candidate bounding boxes.

3.3 Pose Estimation

We use the image gradient for object detection and pose estimation. During object detection, the gradient of the whole image is the first step in the Canny pipeline. For pose estimation, we require a crop of image gradient of the object at a specific, known pose.

As in bounding box proposal, we approximate the gradient using 5×5 Sobel derivative filters [3], but we use a different convex combination of the channels which focuses even less on the Y channel. Camera noise in the color channels is significant. To cope with the noise, we marginalize the gradient estimate over several frames taken from the same location, providing a much cleaner signal which matches more robustly. To estimate pose, we rotate our training image and find the closest match to the image currently recorded from the camera, as detected and localized via the pipeline in Section 3.1 and 3.2.

In order to match our template image with the crop observed at pick time, we remove the mean from and L^2 normalize the template and the crop. Removing the mean provides invariance to bias, and normalizing introduces invariance to scaling, which both help account somewhat for lighting.

3.4 Identifying Grasp Points

To identify a grasp points, we combine a depth map of the object with a model of the gripper. The depth map appears in Figure **??** and is acquired by moving the rangefinder on the arm through a raster scan over the object. The grasp model scores each potential grasp according to a linear model of the gripper to estimate grasp success. A default algorithm picks the highest-scoring grasp point using hand designed linear filters, but frequently this point is not actually a good grasp, because the object might slip out of the robot's gripper or part of the object may not be visible in IR. The input to this module is the 3d pose of the object, and the output is a grasp point (x, y, θ); at this point we assume only crane grasps rather than full 3d grasping, where θ is the angle which the gripper assumes for the grasp.

3.5 Closed-Loop Grasping

To grasp an object, we first scan the work area by moving the camera until the object is detected and recognized. Then we perform active visual servoing to move the arm directly above the object. Next, we perform orientation servoing using the pose estimation algorithm. Because these components are instance-based, they report position and orientation with high accuracy, enabling us to use a proportional controller (with no derivative or integral terms) to move the arm into position.

3.6 Autonomous Training

An object model in our framework consists of the following elements:

- cropped object templates (roughly 200), $t^1 ... t^K$
- depth map, D, which consists of a point cloud, (x, y, z, r, g, b).
- cropped gradient template, t_0

To train our model, the robot first moves the object to a known pose, then acquires images that are annotated with a pose as well as a cropped bounding box for training. Once the object has been moved to a known pose, we acquire the object model by moving the camera around the object, extracting bounding boxes from the resulting imgages, and storing the resulting crops.

Tellex et al. [5]

References

[1] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[2] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[3] Irwin Sobel. An isotropic 3x3x3 volume gradient operator. Technical report, Technical report, Hewlett-Packard Laboratories, 1995.

[4] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[5] S. Tellex, T. Kollar, S. Dickerson, M.R. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. AAAI*, 2011.