

Reducing Uncertainty and Acquiring Knowledge Through Interaction

Rob Goeddel

Lauren Hinkle

James Kirk

Aaron Mininger

Abstract

When an agent is faced with uncertainty it has two major courses of action: it can try and explore on its own to gain more knowledge or it can consult an expert. We have created a image recognition system that attempts to utilize both sources of knowledge to improve its performance. Our system captures point-cloud data from an RGB-D camera and uses machine learning techniques to classify properties of objects including size, shape, and color. It is able to measure the amount of uncertainty for a given property and manipulate the object using a robotic arm in order to reduce that uncertainty. It also incorporates direct feedback from the user to further improve its classification performance and requests more information when it misclassifies an object. Users can easily teach the system new classes of properties on the fly. We demonstrate our system through a simple I-Spy game using toy blocks of various shapes and colors as objects.

1 Introduction

Agents that operate in environments that are partially observable or very noisy are often faced with large amounts of uncertainty. One option for the agent is to explore the environment to resolve some of that uncertainty. While this approach can be done autonomously without direct human involvement, there still may be fundamental limits to what information the agent is able to collect. A second option is for the agent to consult an expert to acquire missing knowledge, a technique known as active learning. This can present a large and rich source of knowledge but can be costly to acquire. Hybrid approaches enable an agent to explore the world on its own and ask an expert for knowledge only when it needs to. This hybridization leverages the strengths of each option and can lead to better performance. Of course, integrating them introduces new challenges such as how to tell when knowledge is unattainable through exploration or what constitutes an important question.

Our project involves one such attempt to utilize both sources of knowledge. Our system is tasked with identifying properties of various objects like color, shape, and size in order to play the children's game "I-spy". Unfortunately it is quite common for an object to be in a poor orientation and thus difficult for our system to classify. We use both recent history and information from our K-Nearst Neighbor classifier to determine when we are uncertain about an object. Instead of asking the user for help, the system first tries to get more information about the object by nudging it using a robotic arm. The goal is to change the orientation of the object in order to improve its classification performance. It will learn which classes are inherently uncertain (such as rectangles) and adjust accordingly. We also will ask for feedback from the user when an object is misidentified to gain more training data. Our system is flexible enough to allow users to teach new classes of properties during its operation.

This work is being incorporated in a larger project which includes teaching the agent new concepts and actions. Such a project relies on having symbolic information provided by our work through object segmentation and classification. The low level features provided by our system can be analyzed and combined in order to build new concepts such as object categories (like blocks and balls) or object properties (all bananas are yellow). Such reasoning and concept generalization is essential for agents that interact in novel situations.

The challenges involved in playing I Spy include object detection, feature extraction, classification, robot arm manipulation, and active learning. We discuss these further in Section 3.

2 Related Work

Object classification is a problem that appears throughout a variety of applications, ranging from law enforcement to manufacturing. While historic efforts have focused on the recognition problem, that is, finding a known object in a scene, more recent work has emphasized the more general problem of classifying objects into generic groups by type [5]. For example, we might be interested in identifying vehicles as bikes, motorcycles, trucks, cars, etc. Work by [9] and [3], among others, has shown that powerful classifiers may be constructed through boosting or similar simple feature-specific classifiers such as color, shape, etc.

Recent popularization of sensors that combine both image and depth information like the Microsoft Kinect inexpensively provide richer sensory information. The addition of depth can help disambiguate situations that would be very difficult with a normal camera image alone. Depth can make the determination of an objects underlying geometry much easier, leading to better identification [8]. Previous work has successfully incorporated such cameras (often called RGB-D cameras) with object recognition in indoors domains featuring common household objects [8, 6].

A different approach to this problem is learning words that describe different areas of the feature space, which can then be combined to describe classes of objects. Research in this area focuses on learning to map physically observable qualities to simple words with the intent of using these simple descriptors to learn more complex nouns and relationships. Of particular popularity is an approach focused on learning colors and shapes of small objects using sensory data [13, 10]. Additional approaches focus on learning entities in the environment through interaction with them [4]. These approaches all focus on interaction with physical objects to discover qualities about them.

Although the types of words learned in these works is similar, the approaches and end-goals vary. Semantic clustering [13], decision trees [4], and bag of word models [10] are all used in the learning algorithms. This prior work focuses mostly on visual classification and the corresponding vocabulary.

Though our work does not extend dramatically on this concept, our long term trajectory focuses more on the complex action space, previously less explored. This project lays the groundwork for developing an action-space, both for learning said actions as well as learning objects upon which to act. Additionally, we explore a novel way of extracting additional information from objects. We are unaware of previous work that uses interaction with objects to see them from new angles and thus obtain additional information to make guesses about object descriptions more confident.

3 Methodology

3.1 Object Detection and Tracking

In order to play I Spy, the agent must be able to point at objects matching the provided description as well as to nudge objects with low confidence in their descriptions in order to see them from a new angle and gather new information about them. In order for the agent to do this, it is necessary to have localized positions for individual objects in the scene. Motivated by changing the confidence thresholds for trustworthiness of descriptors, objects are tracked over time.

The RANSAC algorithm [2] is used to discover the dominant plane in each image, isolating most of the objects in the scene. Further segmentation is performed to try to discover definitive object boundaries. This segmentation considers both the change in color and change in depth of neighboring pixels. Each pixel begins as its own segment, and the segments are iteratively combined with neighboring segments if the difference in color or the distance between them is less than a given threshold. From these segmentations we extract a feature vector for each object.

In order to adjust the confidence thresholds to determine which shapes are more certain (discussed below), it is necessary to track an object over time. In order to do this, the object must be tracked over time. This can be challenging, as the physical object may change location over time or may be occluded by the robot arm or another object. As each new frame appears, the segmented objects in it are compared to the segmented objects of the most recent frame. They are matched to an object from the previous frame if there exists an object with a similar mean color near the same physical location. Any object that does not have a match from the most recent frame is compared to objects from a history of frames. In this case, the objects is matched to a prior object that is nearly the same color which is closest to the current location of the object. This allows previously occluded objects to be matched with their prior selves, and nudged objects to locate where they were nudged from.

3.2 Feature Extraction

Once an object was segmented we extracted relevant features in order to classify color, size, and shape. We considered objects that were of a single color, therefore average red, green, blue, hue, saturation, and value levels across the pixels were used as a six-dimensional feature vector. For size we calculated the diagonal of the 3D axis-aligned bounding box and used that as one of our features. The second feature was the average distance of the points from the centroid. Having the point-cloud data from the Kinect allowed us to compute those features more accurately in three-dimensions and saved us from having to do more complicated depth estimates.

Correctly classifying shapes required much more complicated features. During this work we explored several options for extracting features from objects before settling on the method described in this paper. We experimented with ORB, a rotation invariant feature extractor similar to SIFT and SURF [11]. Additionally we tested several feature detectors from OpenCV, including their square detector [1]. We also explored using RANSAC to discover simple shape types (such as planes, curves, and spheres), in order to determine the number of faces and edges each object has. This work was based off the RANSAC algorithms of Schnabel et al [12]. Although these and other feature detectors are frequently used in object recognition and learning tasks, we ultimately chose to use our own Principal Component Analysis (PCA)-based method which resulted in more accurate guesses from the agent than the other methods.

To extract edge features we apply PCA in order to find the axis the points were most spread out along. For N pixels we compute the mean pixel $\mu = \frac{1}{N} \sum_{i=1}^N x_i$. Then we find the covariance matrix $\Sigma = \frac{1}{N} BB^T$ where B is an $N \times 2$ matrix with the i th column equal to $x_i - \mu$. The primary axis becomes the eigenvector with the largest eigenvalue. We then transform the image into a canonical representation by finding a bounding box for the image aligned with the axes and rotating and scaling it to a unit box. To extract features we took evenly-spaced points from the top and bottom edges of the box and calculated the vertical distance from each point to the image. This process is illustrated by ???. We used seven sample points along the top and bottom; experimental evaluation showed no significant improvement with additional samples. In addition, we calculated the ratio of the width and height before scaling to get an idea of how elongated the shape was.

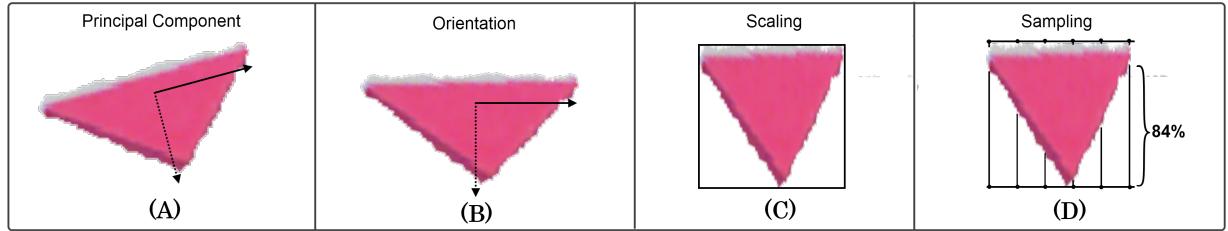


Figure 1: This figure illustrates using PCA to extract shape features. First the principal component is found (A), then the image is rotated and scaled accordingly (B, C). Finally, points are sampled along the top and bottom and for each the vertical distance to the image is recorded (D).

3.3 Classification

Initially we used SVM classification to validate our method, but we have since moved to a K-nearest neighbor approach. With the K-nearest neighbor algorithm it is easy to add another example to the training, without recreating the entire training model as was done with SVM classification. Additionally we no longer rely on third party software, namely liblinear [7] to do classification. Using our implementation of a K-nearest neighbor classification we also reported a confidence metric for each label based on the percentage of neighbors with a given label.

3.4 Confidence

One of the difficulties with reasoning about real objects is that the world is partially observable. Objects can have a dramatically different appearance at different orientations. Naturally when given such an object, a human may move their head around to gain more information and confidence as to what the object is. Our system has a fixed vision angle, but it is possible for it to manipulate objects to gain information on objects whose appearance has high orientation dependency. For this purpose a confidence threshold system was developed to learn what objects may have high uncertainty, and therefore would be good candidate for manipulation. When the system is queried for an object that it cannot currently see, it should manipulate an uncertain object with the highest probability of actually being the queried object.

All attributes are given the same default confidence threshold. When an object is correctly labeled and verified by the user, the thresholds for those labels are decreased. When an object

manipulation correctly reveals that it was previously mislabeled the associated label thresholds are increased. To determine the likelihood of a mislabel, the confidence of a label, calculated from the classification system, is compared to the threshold for that shape. If the confidence exceeds the threshold then it is reported as unlikely to be a mislabel. For example in our system the rectangle label is commonly misplaced on objects presented head on, so the learned threshold confidence for rectangle is very high.

3.5 Robot Arm Manipulation

3.6 Active Learning

equations

4 Experimental Results

4.1 Data Collection and Training

We acquired a large variety of children's foam blocks in many shapes, sizes, and colors to act as our objects for classification. A small subset of the blocks can be seen in Figure 2a. The foam of the blocks is an ideal material to work with as it. Some of these objects can be seen in Figure [figure](#). (1) has a smooth, matte finish, (2) comes in a variety of distinct colors, and (3) is deformable enough to be grabbed by the arm but still retain its shape. Additionally we collected more familiar objects to test our classifier on, demonstrating the ability of the system to abstract general shapes to real world objects.

Our system, seen in figure [Screenshot](#), segments objects found in the predefined play area and labels them with the most confident shape, size, and color. Additionally a user can train new shapes, colors, and sizes by simply clicking on that image and supplying the relevant label. This is sufficient for creating a large number of training samples, preferably at different positions and orientations. This method is not efficient for generating a large number of training samples of many different objects, so we collected a training dataset of footage of the foam blocks at numerous positions and angles throughout the play area which were then segmented and labeled with relevant descriptors.

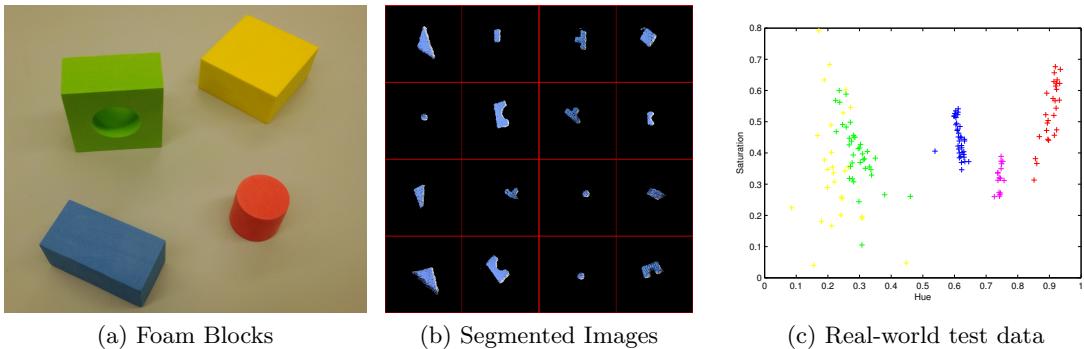


Figure 2

red	yellow	green	blue	purple
96.43%	92.00%	97.37 %	100.00%	88.57%

Table 1: This shows the result of OVA classification for five color types across 160 objects. Each test involved training on 80% of the points, with 20% left out for testing.

Real world data is extracted from toy foam blocks. The blocks themselves can be seen in (a). Using a segmentation algorithm, we extract the blocks from the Kinect data (b) and generate feature vectors for each object. Finally, we hand label the data for features such as color and shape for classification training. In (c), we see the actual distribution of our training samples in the hue and saturation dimensions.

Testing our system requires direct interaction and presentation of new objects or objects at new angles. Additional testing can be done by training the system with new shapes and colors and observing how quickly and effectively it can learn and apply the new information. Qualitatively it is easy to see whether the system is performing well. However to do a less subjective, quantitative evaluation of our methodology, we used the leave-one-out-cross-validation(LOOCV) method on our collected training data. This testing method mirrors the scenario where a novel object is presented to this system.

The results in Table 1 **todo** show LOOCV percentages for the shape, color and size. Unsurprisingly it has more difficulty with shape , which varies greatly by angle and position, and size, which is a more subjective attribute. It has particular problems with shapes that are presented head on, which tend to resemble rectangles. This is the primary motivation for developing a confidence threshold learning mechanism, which allows the system to determine which labels are inherently uninformative.

5 Conclusion

References

- [1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [2] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [3] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221–228. IEEE, 2009.
- [4] K. Gold, M. Doniec, C. Crick, and B. Scassellati. Robotic vocabulary building using extension inference and implicit contrast. *Artificial Intelligence*, 173(1):145–166, 2009.
- [5] D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert. Parts-based 3d object classification. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–82. IEEE, 2004.

- [6] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recognition combining rgb and depth information. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4007–4013. IEEE, 2011.
- [7] LIBLINEAR. <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>.
- [8] Z.C. Marton, D. Pangercic, R.B. Rusu, A. Holzbach, and M. Beetz. Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 365–370. IEEE, 2010.
- [9] M.E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1447–1454. Ieee, 2006.
- [10] D.K. Roy. Learning visually grounded words and syntax for a scene description task. *Computer Speech & Language*, 16(3-4):353–385, 2002.
- [11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.
- [12] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library, 2007.
- [13] D. Zambuto, H. Dindo, and A. Chella. Visually-grounded language model for human-robot interaction. *International Journal of Computational Linguistics Research (IJCLR)*, 1(3):105–115, 2010.