Autonomous Generation of Bounding Boxes for Image Sets of the Same Object

Shui Hu, Jean Feng, Marc Rasi

December 10, 2010

1 Introduction

One of the challenges of computer vision research is to achieve accurate autonomous object recognition. This project focuses on the specific problem of finding bounding boxes of a given object in a set of pictures, each of which is guaranteed to contain this object. We run our experiments on the image database ImageNet, developed by Stanford's Vision Lab. In ImageNet, each "synset" is a group of about 1000 images of a certain object where the bounding boxes are labeled manually via Amazon Mechanical Turk. Since this method is not scalable, we propose an algorithm that generates bounding shapes and can partially eliminate the need for human labor. In this paper, we compare this new algorithm DetectRec, which relies on high-level image cues such as object shape, to our initial algorithm that relies on low-level cues like HOG.

2 An Initial Algorithm using Low-Level Cues

The initial algorithm we implemented relied on finding potential bounding boxes based on its "objectness" score and then learning HOG features to identify boxes that contained an object of the desired class. The Objectness measure, as described in the paper "What is an Object?" by Alexe, Desealers, Ferrari, [1] tries to quantify how likely an image window contains an object of any class based on multi-scale saliency, color contrast, edge density, and closed boundary characteristics.

Our initial algorithm is as follows:

1. Use Objectness measure to automatically collect bounding boxes for simple training images (single object against monochrome background) and generate HOG features for each box
2. Fit a Gaussian distribution for the training data
3. Repeat step 1 on test images, getting 50 candidate bounding boxes per image
4. Use the gaussian distribution to calculate the probability of each of the bounding boxes being correct bounds for the class

As we will show in our results and discussions, this algorithm did not perform very well, which suggests that the gaussian distribution did not fit the HOG data well, either because there was not enough data or because it is the wrong model to use in general. So we developed a new algorithm that follows the same general framework of finding general objects and then using high-level features to identify objects belonging to the specific class.

3 DetectRec Algorithm

DetectRec takes a high-level approach to the object localization problem. in essence, the algorithm only takes into account the object shape once given the object segmentations. An outline for the algorithm is as follows:

1. Learn generic shape models of an object class by running k-means on the object segmentations from the class.
2. For each image:
    a. Use hierarchical object detection (explained below) to find potential object.
    b. Select the object that is most similar to one of the generic shapes, and draw a tight bounding box around it.

3.1 Learning Shape Models

Given an object segmentation S, the algorithm first approximates S with a 20-gon P, which is further simplified to the vector (N, C, R) where

- N is the number of segments that are convex, concave, or straight relative to the object. A segment is a set of consecutive angles of P that have the same convexity (greater than, close to, or less than π) relative to P.
- C is a vector denoting which segments belonging to which class of convexity
- R is vector denoting the regularized version of each angle. Specifically, the regularized angle is the average of the angles belonging to that continuous section of the same convexity/concavity. If the segment is straight, the regularized version is π.

The intuition behind using an angle of vectors is that our shape model can then be made invariant to rescaling, reflections, and rotations. The vector (N, C, R) is useful since it captures the characteristics at different levels of generality, where N is the highest level and R is the lowest.

4 Shape Similarity

From this shape model, there are two ways we can measure shape similarity between two different shapes $(N^{(i)}, C^{(i)}, R^{(i)})$ and $(N^{(j)}, C^{(j)}, R^{(j)})$. First is simply the difference in number of segments, which we denote $ND^{(i,j)}$.

$$ND^{(i,j)} = |N^{(i)} - N^{(j)}|$$

The second measure of shape similarity combines the difference between convexity and angles in the two shape vectors. In order to make this metric invariant to rotations and reflections, we take the minimum distance over all possible rotations and reflections of one of the shape vectors. We denote this measure by $SC^{(i,j)}$ and define it as

$$\text{SC} = \min_{(C^{(*)}, R^{(*)})} \left(\sum_{k=1}^{20} 1\{C_k^{(*)} = C_k^{(j)}\} \right)^{-1} \left(\sum_{C_k^{(*)} = C_k^{(j)}} |R_k^{(*)} - R_k^{(j)}| \right)$$

where $(C^{(*)}, R^{(*)})$ ranges over all cyclic permutations and reversals of $(C^{(i)}, R^{(i)})$

We put these two measures together by multiplying them. Smaller $ND^{(i,j)} \times SC^{i,j)}$ denotes greater similarity. If two comparisons yield the same $ND^{(i,j)} \times SC^{(i,j)}$ (for example, many comparisons may have $ND = 0$), the one with smaller $SC^{(i,j)}$ is more similar. $ND^{(i,j)} \times SC^{(i,j)}$ takes into account both the difference in the matching segments of and the overall difference between two shapes. The value ND accounts for the overall similarity between the two shapes while SC measures the degree of difference in matching parts of the two shapes. The similarity score is also capable of identifying the similarities between a complete object and a partially occluded one.

5 Learning Generic Shapes

Given the set of training vectors containing the angles of 20-gon approximations, we cluster the vectors using k-means and return the centroids as shapes to use in finding bounding boxes. We use $ND \times SC$ as the distance metric for k-means.

To find the number of k-means clusters, we use the binary search algorithm. If there are m training examples, we start with m clusters. For each pair of centroids, if the inter-cluster $ND \times SC$ is below a certain threshold, we decrease the number of clusters. If the intra-cluster $ND \times SC$ between two vectors in one cluster is greater than a certain threshold, we increase the number of clusters.

The centroids correspond to generic shapes for the object class reflecting the intra-class variation due to angle of perception and variation within the object class itself.

Figure 1: Demonstration of the object segmentation algorithm at different grid resolutions

6 General Object Segmentation

Given an image, IM, this step produces a list, D, of candidate object segmentations.

First, use k-means to cluster the colors of IM to just two colors, producing a binary image, BIM. This step reduces noise because the background tends to blend into one color. Then run the following hierarchical object detector on the binary image.

1. Initialize an empty list, D, of objects.
2. Divide the black and white image BIM into an 8x8 grid.
3. Set the color of each grid cell to the most common color in the image in the grid cell.
4. Refine the shapes of objects in D by absorbing adjacent cells with the same color.
5. Add each contiguous set of cells of the same color to the list D of objects.
6. Repeat this process for finer grids (16x16, 32x32, 64x64, 128x128).

At the end, D gives us a list of potential objects in IM. These objects tend to be the more prominent objects in the image because they tend to have large regions of coherent color which corresponds to contiguous sets of cells of the same color.

7 Results

7.1 Analysis for First Algorithm

We ran the first algorithm on pictures from 5 different synsets (mangos, shoes, laptops, backpacks, and cameras). For each synset, our algorithm outputted a trained gaussian model for HOG features of bounding boxes containing objects. From the Objectness step of the algorithm, we get 50 candidate bounding boxes and then we classify bounding boxes as positive if they had an overlap of at least 0.75 with a true (manually labeled) bounding box and as negative if they had no overlap of 0.75 with a true bounding box. For both algorithms, overlap between two shapes A and B as

$$\frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}$$

We then classified all the bounding boxes using our gaussian model and plotted ROC curves. Figure 2 shows the ROC curves for the 5 different synsets.

We also tried a few other things to get a better understanding of how our algorithm was working. We first tried training on manually labeled data (as opposed to the automatic labeling we normally do in step 2 of our algorithm), which is illustrated by the green line in the mango ROC plot in figure 2.

To test our assumption that training on simple images produces better results than training on complicated images, we trained our gaussian model with image sets of both types. Figure 2 also shows the ROC curves from these tests.

We also tried training on positive examples from one synset and then testing on testing images from a different synset, to test if our algorithm was learning what one specific type of object looked like or if it was learning what objects look like "in general." Figure 3 shows the ROC curves from these tests.

Finally, we also plotted 30 ROC curves for testing on cameras after training on 10 through 300 training examples, to see how our algorithm performance changed with number of training examples.

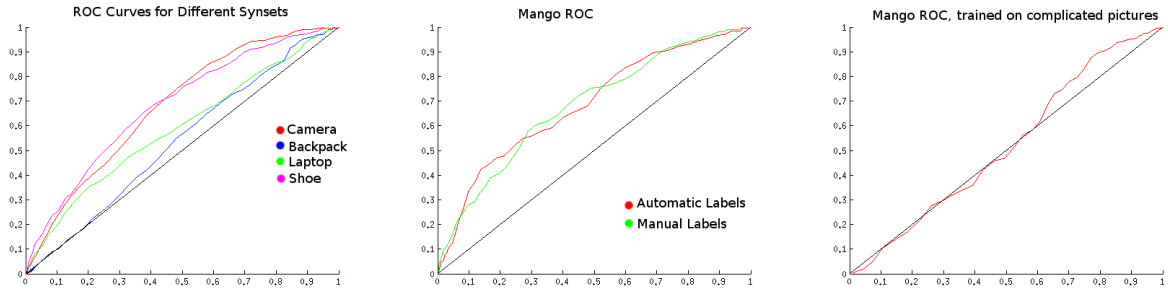

Figure 2: ROC curves on 5 different synsets. Also, training on complicated images.

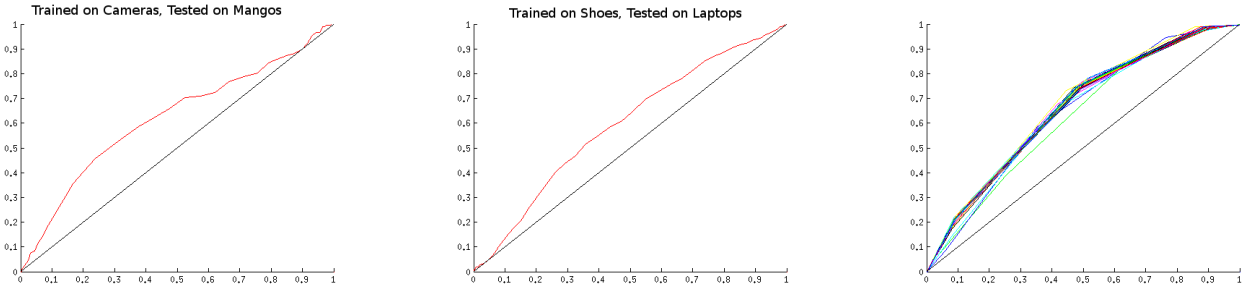

Figure 3: Left, center: Training on one synset and testing on a different synset. Right: ROC curves for varying number of training examples

7.2 Discussion for First Algorithm

We can see from our plain ROC curves on the individual synsets that our algorithm does do better than chance, which is good. So our algorithm is slightly helpful in picking out the positives and the negatives from the candidate boxes. But since there are so few positives in the candidate boxes to begin with (we found that about 2% of the candidate boxes were positives), we are still left with far more false positives than true positives. So we would like to know why the algorithm does not perform so well and what would be required to improve it.

Figure 3 show that performance does not increase very much as the number of training examples increases. The lowest green ROC curve is the ROC curve for 10 training examples, so our algorithm does improve a bit with number of training examples in the range of 10 to 20 training examples. But additional training examples do not significantly improve performance. This suggests three reasons why our algorithm is not performing so well.

1. The automatically labeled positive training examples might be bad because objectness does not find objects very well, even in simple pictures.
2. The positive examples might all be so similar that our algorithm does not learn very much when it sees new examples.
3. Our model might not fit the data very well.

Reason (1) is not the reason, as shown by the green line in figure 2. The green line shows how well our algorithm performs if we manually label positive examples, and it is not significantly different from the red line which corresponds to automatically labeled positive examples. So a combination of reasons (2) and (3) are probably making our algorithm perform poorly. Reason (2) is reasonable that there is not much variation in the "simple" images of objects that we are using for our training examples. Reason (3) is also likely since a single gaussian distribution probably does not model all the different possible ways an object can appear. Therefore, our algorithm performs more poorly on laptops and backpacks, which are deformable objects and look different from different angles, than on shoes, mangos, and cameras, which all look reasonably similar across all the images in the test sets (either because Amazon Turk users collect similar images or the object just looks the same from different angles).

Although reason (2) suggests that training a model only on simple images does not generate a robust model, our data does support the idea that, if we want to automatically generate bounding boxes without human intervention, we should start by training on the simple images. Specifically, if we run our automatic algorithm on complicated images instead of on simple images, we get the ROC curves shown in figure 2.

Figure 4: Cluster centroids for the shapes used in the DetectRec algorithm on the shoe synset. Different shoe shapes are distinctly visible.

The last thing we tried was to train on examples of one set and test on examples from a different synset, to see if our algorithm was learning the appearance of specific objects versus general objects. The ROC curves from this are shown in 3. By training on one synset and testing on a different one, this algorithm performed slightly worse than when we trained and tested on the same synsets. Yet, it was still better than chance. This suggests that this first algorithm is using image cues from the specific objects and characteristics of general objects. It is true that by training on shoes and testing on laptops, actually performed better than when we trained on laptops and tested on laptops. This just suggests that the algorithm wasn't able to learn anything specific to laptops, which supports reason (3): our model can't model all important features for complicated objects like laptops.

8 DetectRec Results and Analysis

We tested DetectRec on the shoe image synset. With an initial training set of 14 images and a test set of 100 images, DetectRec achieved 55% accuracy overall. Since classification is done in two steps, first finding objects and then classifying the object, we further broke down these results:

1. Given the image test set, the object of the desired class (shoe) was segmented in 71% of images.
2. Given a correct general-object segmentation, the object classifier then correctly identified the desired object in 77.46% of the images.

Objects detected by the object detector tended to be rather large and differed significantly in color from their surroundings. Over half of the images where the object detector failed to detect the correct object were cases where the correct object was adjacent to or occluded by another similarly colored object, so the object segmentation algorithm was unable to separate them. Therefore, DetectRec does suffer to some extent by forcing every image to be black and white.

We also tested the difference from adding more images. Using 30 training images, DetectRec found three representative shapes and achieved 70% accuracy overall for 50 test images. By breaking down these results again, we found that the general-object segmentation algorithm achieved 78% accuracy while the object-classifier accuracy rose to 89.74% (once again, this is given that the object was segmented correctly in the first place). This shows that even with a limited amount of data, the object-classifier will achieve a decent enough accuracy to at least warrant a collaboration between humans and computers to find bounding boxes for images for large databases such as ImageNet.

9 Conclusion

The absolute accuracy of our approach does not compare very well to the state of the art algorithms (Desealers, Alexe, and Ferrari achieve 70% accuracy by a similar metric to ours on a much more difficult test set in [2]). Our paper show the advantages and disadvantages to different approaches for object localization and can aid in generating bounding boxes for large image databases such as ImageNet.

References

[1] Alexe, Bogdan. Desealers, Thomas. Ferrari, Vittorio. "What is an Object?" Computer Vision Laboratory, ETH Zurich. IEEE Computer Vision and Pattern Recognition (CVPR), San Francisco, June 2010.

[2] Alexe, Bogdan. Desealers, Thomas. Ferrari, Vittorio. "Localizing Objects While Learning Their Appearance. Computer Vision Laboratory, ETH Zurich. European Conference on Computer Vision (ECCV), Crete, Greece, September 2010.