Learning Object Location Predictors with Boosting and Grammar-Guided Feature Extraction

Damian Eads[1]
http://www.cs.ucsc.edu/~eads
Edward Rosten[2]
http://mi.eng.cam.ac.uk/~er258
David Helmbold[1]
http://www.cs.ucsc.edu/~dph

[1] Department of Computer Science
University of California
Santa Cruz, California, USA

[2] Department of Engineering,
Cambridge University,
Cambridge, UK

In this paper we present BEAMER: a system for unstructured object detection which takes in images and emits a list of (x,y) pairs denoting the locations of the detected objects. The system is designed to operate on greyscale images with small objects which have in many cases a similar appearance to the background. Some example data is given in Figure 1.

The system has three layers. The first layer uses a grammar which stochastically generates a rich set of image processing programs for extracting features from the image. The features are then post-processed to remove detections which are likely to be in error. We then use a localizing scheme to extract the object locations from a weighted sum of the post-processed features. All stages of the algorithm are trained: the post-processed features are selected and combined using AdaBoost [1], and the parameters of the post-processing and object detection are trained using cross-validation. The processing pipeline is shown in Figure 2

Feature extraction Features are extracted from the image using image processing programs generated stochastically using a generative grammar. The grammar specifies a number of low-level unary image processing operations, such as greyscale morphology, Gabor filters, median filtering, Haar-like features [2], and ways of combining them with binary operators such as addition, multiplication and so on. An example generated program operating on an image, I is:

$$\text{multiply}(\text{Laplacian}(\text{erode}(I)),\text{Gabor}(I))$$

Post processing Features are turned in to weak classifications using thresholding. However, the AdaBoost stage is attempting to learn a pixel classifier, and there are many circumstances where good pixel classification can lead to poor object detection and vice-versa. For instance detecting half of the pixels on all of the objects is a poor segmentation, but excellent detection. As a result, without post-processing, the pixel classification is often very noisy. So, after thresholding the individual features, either erosion, median filtering, dilation, no post-processing or region processing is performed. Region processing performs a 4-way connectivity flood-fill and turns sufficiently large areas in to abstentions. The parameters of the post-processing are learned in a validation stage.

Object detection The previous stages learn a pixel classifier, but the problem remains of how to turn a pixel classification in to a good set of object detections. The options we use are 1. connected components analysis, followed by centroiding, 2. blurring, then finding large local maxima and 3. finding the modes of a kernel density estimation (KDE) using the large local maxima as the input data. The best method and parameters are learned in a validation stage.

Validation and evaluation Evaluation of unstructured object detection is not well-defined, since it is not clear exactly what a true/false positive/negative is. It depends on the problem. Nevertheless these must be defined in order to learn parameters in a validation stage, and then evaluate the final system on unseen data. We have therefore defined three methods of scoring which are suitable for different tasks.

Cueing: In order to cue humans to objects in an image, detections must be nearby the object of interest, and multiple detections for one object are not penalized. Objects with no detections nearby are false negatives, and detections nearby no objects are false positives. All other detections are true positives.

Tracking: For the results to be useful in a tracking system, the detections must be nearby the objects of interest. However, multiple detections of one object are harmful, so these are counted as false positives.

Counting: For counting, detection positions need not be accurate.

Once the definition of what is a true positive etc had been established, the area under the ROC curve can be optimised.

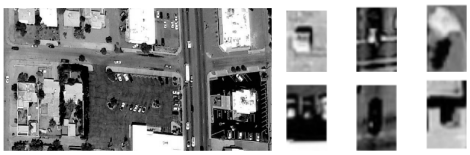

Figure 1: Some example data. Left: an aerial image of traffic. Right some difficult examples: (top) background, (bottom) cars.

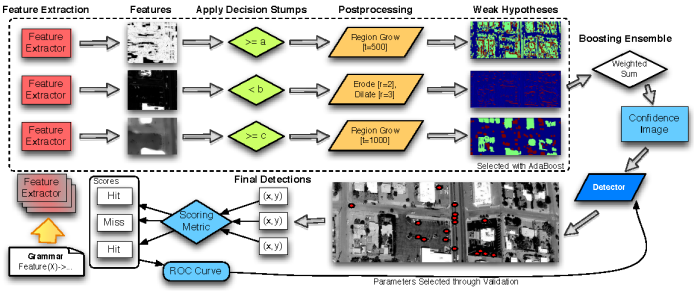

Figure 2: The Beamer object detection pipeline. The grey arrows show the flow of data during object detection.

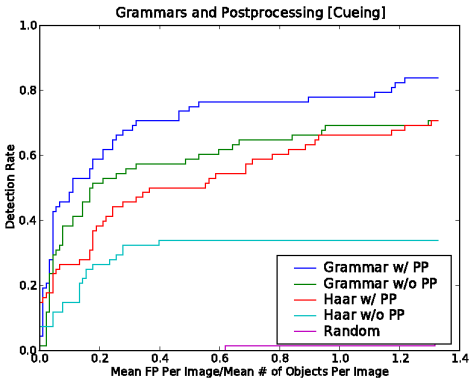

Figure 3: The figures show the performance of the system with ROC curves. Both plots show that best performance is obtained when our feature grammars and post-processing (PP) are used, compared to using Haar-like features. For comparison we show the effect of randomly scattering detections around the image. In this case, the system is optimized and evaluated using the 'cueing' metric.

A thorough experimental evaluation shows the value of our system. In particular, we perform tests with each subsystem omitted to show the value given by that particular subsystem. An example of some of the results, illustrating both the benefit of the feature grammars and post-processing is shown in Figure 3.

[1] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Int. Conf. Mach. Learning*, pages 148–156, 1996.

[2] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conf. Comput. Vis. Pat. Rec.*, 1:511, 2001.