CPSC532c Project final report
A survey of techniques for object detection.

http://www.cs.ubc.ca/~talis/projects/cpsc532c/index.html

Jordan Reynolds
21357041
talis@cs.ubc.ca

December 19, 2004

Abstract

This paper presents an overview of two different methods of object detection within a scene. We then go on to discuss the implementation of a Boosting based object detector, as well as the development of a framework for testing and comparing several different forms of detector. We are motivated to create an efficient object detector as a way of solving the dynamic environment robotic control problem.

1 Introduction

Many of the current robotic navigational techniques attempt to build a detailed map of there environment, and use these maps to complete a goal. These internalized world models allow robotic systems to navigate within a space, and perform interesting tasks. These models however assume a fairly static world, significant changes in the environment render the internalized map useless. To this end purely reactive control systems have been developed based on a stateless model, these models do not internalize any information about the world, and can operate in completely dynamic environments. It is difficult however, to build reactive systems capable of performing interesting tasks. The stateless world model does not include any real sense of goal, and therefore require a higher level controller to plan and perform the required tasks. The problem then becomes building a high level controller capable of performing interesting tasks, that does not rely on a static model of the world.
One such system allows the high level controller to build a snapshot model of the world, which is flushed, and replaced in the next time slice, thus allowing for temporally localized world state models. To accomplish this, the controller must be able to identify objects in the current view of the world (eg it must be able to identify a mug if it is trying to collect mugs).
This paper investigates several methods for identifying objects within a single image snapshot of a scene. There are currently two approaches to this classification problem, the first asks if a patch belongs to a specific class, this is referred to as a patch based classifier. The second approach asks if a particular set of pixels could have been generated by a model of the object.

2 General process

Each of the following processes describes a functional mapping from a set of images into a model of how pixels occur within an image. The first step in all of these processes is to extract some specific information from the image and learn the structure of these features.

Most of the current approaches use some particular image transform, the most popular being the steerable pyramids and wavelet transforms.
Once we have extracted useful feature information from the transformed images, we apply some form of learning procedure to the features, and attempt to extract structural information. The resulting structure should describe some uniquely recognizable set of features from the underlying image, and therefore from the underlying object.
The methods discussed in the following section are primarily scale invariant when integrated into a full production system. None of the methods however are fully rotationally invariant as they stand, each requires training a different detector for each possible facing of the object.

3 Generative models

The first type of classifier to investigate is the generative model. In these approaches we attempt to learn statistically significant features of an object, and then recombine these features in a known way to synthesize a new image. These models can be reversed and used in classification mode, allowing the system to ask the question, what is the probability that this particular pixel was generated from this model. We will examine two different approaches to solving this problem.

3.1 Restricted Bayesian networks

This approach, presented by Schneiderman [7] [6] [5] [4] attempts to learn the structure of a Bayesian network to carry out the generative task. The problem of learning the structure of a Bayesian network is known to be NP-Hard, and therefore we restrict the structure of the final network to a known form in order to gain tractability.
As discussed in the previous section the first step is to extract feature information from the image, Schneiderman uses a three level wavelet transform to convert the input image to spatial frequency information. One then constructs a set of histograms in both position and intensity. The intensity values of each wavelet layer need be quantized to fit into a limited number of bins (for a 40×40 patch we would require $40 \times 40 \times 255 \times 10 = 4,068,000$ bins, the final factor of 10 coming from the number of levels in the wavelet transformed image). One problem encountered in the early implementation of this technique was the lack of high power frequency information in the images. With a linear quantization scheme the higher energy bins had primarily singleton values, this leads to a problem when a prior is introduced to the bin, as the actual count values are lost in the introduced prior. To correct this an exponential quantization technique was employed to spread the power evenly between all the bin levels.
Next the structure of the resulting network is learned. This is accomplished by examining all pairs of variables and minimizing the error function (1), where C_1 is the error of modelling each of the variables as independent, C_2 is the error of removing X_j from the pair, and C_3 is the error of not modelling either of the variables.

$$E_i(G) = \sum C_1(X_i, X_j) + \sum C_2(X_i, X_j) + \sum C_3(X_i, X_j) \tag{1}$$

A greedy search would then be performed over the set of all variables, in each pass we cluster nodes together that minimize this function. The result is a set of clusters of variables with strong interdependencies (eg they minimized the C_1 error). The next step performed by Schneiderman is purely one of efficiency, the resulting sets are dimensionally reduced by projecting them onto a smaller subset of vectors. This is only a viable approach if the the cascades presented in [6] is used[1]. Finally we build our network structure by removing nodes

[1]The cascade approach is performed over a sequence of scales, patches identified as not containing an object at a coarser scale are not re-examined at finer scale. This selects for models that do not necessarily a good detection rate, but selects for ones that have a good false negative rate

in the union of two sets, and boosting them to a higher level in our network, the result is a two layer net, with clusters being joined by single nodes. The final model selection is accomplished by scoring each of the previously generated models against a validation set.
This approach attempts to capture the interdependencies between pixels within the image, and produce a generative model. The resulting model however, makes a poor generative model, primarily due to the quantization step, the result of running a classifier in generative mode would be a poorly detailed, though model accurate image, that would likely not be human interpretable.

3.1.1 Restricted Bayesian network Pros.

- Generative models are seemingly able to handle greater interclass variations.
- Generative models allow for an easier integration of other knowledge in the form of a prior.

3.1.2 Restricted Bayesian network Cons.

- Greedy search for clusters is quite time and space intensive.
- Resulting generative model only re-generates quantized image data.
- Computation of patch class is quite expensive.
- The approach is still patch based, and relies heavily on the use of cascades for correct class identification, and speed of computation.

3.2 Cluster-based statistical models

The second generative model investigated was proposed by Rikert, Jones, and Viola [8]. In this approach textural information is learned and used for classification. We proceed by transforming the images, and then build a mixture of gaussian model based on the result of k-means clustering applied to the transformed image.
In the first step the image is transformed using a multi-directional steerable pyramid (in the case of the current framework, a six directional filter is used). The result of the pyramid is then compiled into a series of "parent" vectors composed of the first layer residue pixel, and the pixels from higher in pyramid resized without interpolation. This generates a $n \times m$ 26 dimensional vectors per image. For reasonably sized patches this quickly becomes intractable (150 40×40 patches generate 240,000 vectors).
Applying a greedy K-means clustering algorithm to these patches using the distance function given by (2). This clustering process is $O(n^2)$, where n is the number of nodes.

$$D(v_1, v_2) = \frac{\|v_1 - v_2\|}{\|v_1\| + \|v_2\| + 1} \tag{2}$$

We then iterate this process, using the local mean of a cluster for distance measurements, merging clusters as we proceed. The result is a set of clusters representing statistically close textural features from the original image. These clusters can now be represented as a series of gaussian distributions (eg each dimension of the vector has a population mean and variance).
Finally we need to choose from the collection of resulting distributions only those that differentiate an object from a non object. This is accomplished by greedily comparing each distribution too all members of a verification set, and eliminating those that cannot distinguish objects from non objects.
To utilize the resulting system we simply need to apply Bayes rule, thresholding the result

(the classifier equation is given in (3), where we calculate the probability parent vector v_i belongs to class C using our mixture of gaussian model).

$$\frac{\sum_{i=1}^{n} P(C|v_i)}{N} \tag{3}$$

3.2.1 Cluster based statistical model pros.

- As a generative model, produces quite good results[2]
- Can handle fairly significant interclass variations[3].
- The resulting classifier should be fairly scale and pose insensitive. As textual information does not necessarily rely on pose of the object, the detector may see enough recognizable texture to be able to classify the patch.

3.2.2 Cluster based statistical model cons.

- Training on large data sets quickly becomes intractable (150 80x80 patches takes approximately 7 days of CPU time to run first pass of K-Means clustering algorithm).
- Will not work well on objects with very little textural information (screens, mugs, smooth surfaces), or on objects with wildly varying textural information.
- Evaluation of (3) is quite slow, when the result only classifies a single object within a single class at a single scale.

4 Patch Classifier Models

We now turn our attention to patch based classifiers, that is classifiers that can only run in classification mode. These techniques follow closely with the pattern seen in the generative models, an image is transformed, some classifiable feature is extracted, and a learning algorithm is applied to attempt to model this structure.

4.1 Boosted Feature Detector

Currently the better part of classifier research is being directed towards boosted detectors as presented by Viola and Jones in [10], and in a modified form by Murphy, Torralba, and Freeman [3]. Boosting is a technique whereby a series of weak classifiers (classifiers only slightly better then random) are combined in a voting scheme to identify a patch. Once the feature vectors for a patch have been computed the application of a boosted classifier is quite quick, and if a cascade technique is applied near real time performance can be achieved[4].
Unlike the previous techniques we first begin by convolving the image with a series of filters and templates. The filters respond to interesting features within the image, and include edge, corner, delta, and Laplacian filters. We select regional responses to these filters using a set of templates, the result is similar to a wavelet transform, localizing filter response in power and space. The response of the filters can be represented by a pair of sufficient statistics, namely the L^2 norm or variance, and the L^4 norm or kurtosis. The result is a vector of size $(N_{filt} \times N_{temp} \times 2)$ which represents the response to all filters, with all templates applied, at both the L^2 and the L^4 norms. We now need to learn which combination of filters and

[2]based on experiments presented in [8]
[3]based on experiments presented in [8]
[4]Based on experiments presented in [10]

templates best segments an object from its surroundings.
The resulting vectors produced in the previous step are now passed into a boosting algorithm, the boosting function must identify which combination of filters and templates as well as weights best identify an object. In the simple boosting algorithm[5] as presented in [9]. In the learning process we wish to minimize the error function given by (4).

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i)) \tag{4}$$

In other words we only incur an error if our classifier G fails to correctly classify value x_i. As we proceed each vector is re-weighted depending on the success or failure of the previous iteration. Unsuccessfully classified patches are weighted more heavily, that is a detector capable of correctly classifying a hard to identify vector is given a higher weight. The goal is to assign weights (α_i values) to each of the classifiers, and then choose the n best for our classification task.
Once we have ranked and chosen our weak classifiers, we can begin classifying new vectors. This is accomplished by applying equation (5).

$$G(x) = sgn\left(\sum_{i=1}^{N} \alpha_i \cdot G_i(x)\right) \tag{5}$$

As in both of the previous cases the resulting classifier is only capable of identifying the presence of an object in a single patch at a single scale. To assemble a full object detector the patch classifier must be applied across the whole image, and across multiple image scales. Viola and Jones [10] suggest the use of cascades, an efficient process for applying classifiers across multiple scales. A simpler method would be to apply the classifier at a given scale, noting the most likely positions, re-scaling the image and repeating, identifying objects at locations with the strongest classifier response, above a given threshold. This second technique is applied to the detector discussed in the following section.

5 Experimental Results

The remainder of this paper deals with the development of a generalized framework for evaluating many different combinations of pre-processor steps along with several learning techniques, and specifically the full implementation of a boosting based detector, developed from the system presented in [3]. As we noticed during the evaluation of the previous three techniques, the learning process can be decomposed into a series of steps independent of the learning techniques used, this decomposition then served as the foundation for building a general framework.

1. Image pre-processing, Includes such tasks as patch extraction, image synthesis, and patch transformation.
2. Training set construction, Assembles the training data into a coherent set, incudes data segmentation.
3. Training data vectorization, In the previous three techniques the training set needs to be vectorizing before passing into training.
4. Training, Takes the vector sets, and applies the training algorithm.
5. Evaluation, The results of the training process are then verified.

[5]Here we discuss the adaBoost technique, though the experimental system applies a slightly different method, the underlying principals are similar

5.1 Sparsity of Data

As training an accurate detector requires a fairly large set of data, a method of synthesizing a large training set from a smaller subset of images needs to be devised. For this experiment positive patches were synthesized by simply flipping each of the image patches, this should provide more useful information for all of the detectors. The negative patches were generated by masking the object out of the parent image, and choosing n patches at random that did not contain the object of interest. These patches are fist chosen at the same spatial extent as the object within the image, if a patch cannot be found at this scale, we look for a smaller patch within the image. The result is a finite number of object images $(2 \cdot n)$ and a much larger number of non-object patches. For this experiment both 20 and 50 times more negative then positive patches were tested[6].

5.2 Boosted Detector

The boosted detector trained using the resulting framework was tested with a series of four different objects, with two different data set sizes, and two sets of spacial blocks. Due to time restrictions we were only able to produce a high quality classifier for a single object (screenFrontal), the remaining object classifiers were built using a smaller set of negative patches for every positive patch. Figure 1 shows the resulting ROC curves for three different system configurations[7]. The results of the screenFrontal detector are quite good, producing

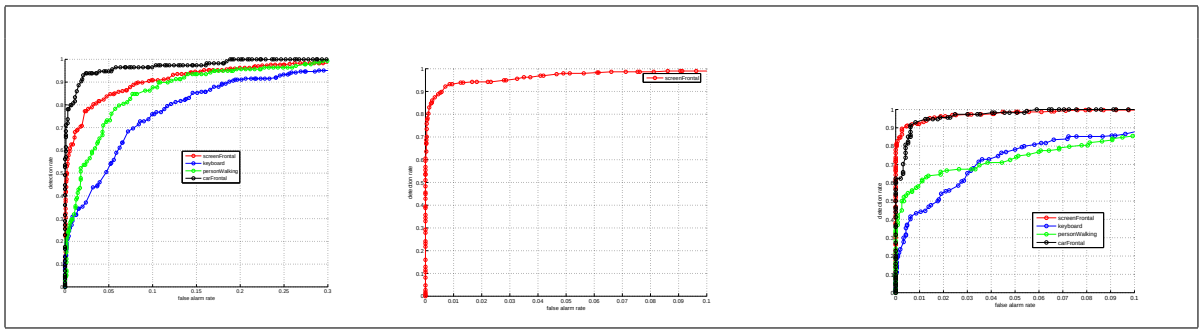

Figure 1: ROC curves for the resulting system, showing hit rate vs false alarm rate for a variable test size (all in the order of several thousand test patches), and all with 100 rounds of boosting. **left**)ROC curve for 4 objects trained with 20 negative per positive patches, using 23 spacial filters, **middle**)ROC curve for screenFrontal detector trained with 50 negative per positive patch, using 30 spacial filters **right**)ROC curve for 4 objects trained with 20 negatives per positive, using 30 spacial filters. (Please note the x axis scales change between plots)

a 90% detection rate with only a 0.5% false positive rate. An interesting behavior was noticed, increasing the number of non-object images did not have a drastic effect after about 20 negative patches per positive. It is hard to tell at this point if the lack of improvement of the resulting system is caused by lack of more information in the negative patches, or if it caused by the random patch generator creating multiple similar patches from the parent image. What did have a significant effect however, was the choice of spatial templates, by including an additional 7 filters (adding an additional $7 \times 13 = 91$ feature vectors) we saw a drastic improvement in the ROC characteristics. From this test we can infer that the selection of feature vectors is the most critical step in the learning process, and that even several thousand negative patches, and less then a thousand positive patches is sufficient to train a basic object detector. We should also note that the best characteristic curve comes from the car detector, which also has the smallest data set size.

[6]The negative patches were chosen at random from within training images

[7]please refer to http://www.cs.ubc.ca/∼talis/projects/cpsc532c/index.html for more detailed versions of these plots.

We now turn our attention to applying the detector to real world scenes. As we have discussed previously we will need to apply the detector at multiple locations, and across multiple scales to select objects from the scene. We build a gaussian pyramid to generate the images at multiple scales, classifying each resulting image as we go. The result is a set of confidences, and a set of class labels for each patch across all scales. We identify objects at locations with the highest thresholded confidences[8], and the size is determined by the patch size at the particular scale.
Figure 2 shows the results of running three of these classifiers over a pair of test images. Some errors in the resulting classifications were noted, but were caused by objects that structurally looked like our classifier[9]. To achieve a better classification, it would be worth iterating the training process, running the training, classifying a large set of test images, then including misclassified patches as negative examples in a re-training of the system. While more time consuming at training time, the resulting classifier would not be any larger, and this would not affect the time required to parse a new image.

Figure 2: Results of running the classifier over a pair of images. **left)** The keyboard detector run on a test image, the red box in the plot indicates the most likely detection, and the size of the box represents the scale at which it was found. **middle,right)** The results of the classifier run in the same way using the trained screen and car detectors respectively. (Note: The classifiers used were trained with 30 spatial blocks, and 20 negative patches per positive)

6 Conclusions and Future Work

We have shown that a boosted detector has quite good performance, even with a relatively sparse training data set. The resulting detector is able to scan an image, the current system however has issues of speed, the scanning and classifying of all patches within a small scene of 240×320 pixels takes on the order of 4 seconds per object. This is not acceptable, as we would normally wish to have a large number of objects detectable by a platform. Even using gist to selectively apply the detector we would want to be able to apply a dozen or so detectors in the same period of time.
Mounting the detector on a mobile robotics platform allows for only needing to re-apply the detector in areas local to objects we are certain about, and along newly exposed edges. It is also not necessary to be processing frames at the maximal camera rate, if we use sensors to track motion, and only re-classify along the motion boundaries we will only need to re-classify a full frame every few seconds to eliminate any long term errors. Using local contextual information, as presented in [3] also allows for reducing the burden of the classifier. If we know we are viewing a scene in a particular location, we can choose which filters to apply, for example if we are in a kitchen, we are not likely to see a car, so we can forgo applying the car detector.

[8] We do not need to threshold the confidences, but doing so allows the system to decide there is no object present, otherwise it will always classify a patch as containing the object.

[9] A picture frame looks a lot like a monitor viewed head on for example.

As we discovered from experimentation, the spacial and filter templates used has probably the greatest single influence on the performance of the system, and therefore is a good place to focus effort in an attempt to optimize the given boosting technique. as this transform basically produces the response of the image in location to a specific set of filters, similar information may be retrievable from the wavelet or steerable pyramid transforms. A great deal of evaluation time is spent on calculating the convolution of the image with the filters, and then computing the energy and kurtosis, any reduction in processing at this step will greatly speed up inference.
Another approach would be to build an iterative learner, that is able to use misclassified patches in a verification data set to learn a better classifier in a later step. The road block in implementing such a system is the large amount of data that would be required. Though as we are looking for misclassified patches, it would most likely be possible to simply re-parse the images from which we extracted the positive patches, looking for negative patches being labelled as positive.

References

[1] M Rubin Antonio Torralba, Kevin Murphy. Context-based vision system for place and object recognition. 2003.

[2] H. Schneiderman D. Hoiem, R. Sukthankar and L. Huston. Object-based image retrieval using the statistical structure of images., 2004.

[3] William Freeman Kevin Murphy, Antonio Torralba. Using the forest to see the trees:a graphical model relating features, objects and scenes, 2003.

[4] H. Schneiderman. A statistical approach to 3d object detection applied to faces and cars, 2000.

[5] H. Schneiderman. Learning statistical structure for object detection, 2003.

[6] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection., 2004.

[7] H. Schneiderman. Learning a restricted bayesian network for object detection., 2004.

[8] P. Viola T. Rikert and M. Jones. A cluster-based statistical model for object detection. 1999.

[9] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements Of Statistical Learning*. Springer, 2001.

[10] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision - to appear*, 2002.