Unsupervised, parts based object detection

Alex Kreimer
Department of Computer Science
Technion

September 15, 2013

Abstract

This project is submitted as part of the requirements for the Computer Vision seminar under the supervision of prof. Micha Lindenbaum.
We've built an algorithm that is capable of learning object appearance in a parts-based fashion and apply this knowledge to detect objects in novel images. The algorithm was tested using car image database that was built during the project.
The results show that the algorithm chose some number of relevant features (e.g. various car parts) along with some irrelevant features. The ROC of the classifier suggest that for robust object detection the algorithm should be improved

1 Intro

The goal of this project is to implement an unsupervised parts based object detection algorithm. The training algorithm is presented in section 2.

Features We treat square image patches as features. Size of a patch is a parameter to the algorithm. The results that are presented in this report are produced with patch size of $[41, 41]$

Distance from feature to image The distance between feature and image is defined as a minimum of distances between feature and any possible

square patch in the image. Note that the patch similarity measure function is a parameter to the algorithm. To produce the results presented here we've used the normalized cross correlation.

2 Training Algorithm

1. Designate a number of positive images as 'model' images. Choose candidate features from model images over a sparse grid. See Figure 1 for grid example.

2. For each feature, calculate its distance to all training images (both positive and negative).

3. For each feature, calculate the distance threshold that minimizes its training set classification error . See subsection 2.1 for details.

4. Select the most discriminative features from the set of all candidate features (the number of features is a parameter to the algorithm). See 2.2 for details

5. For each training image calculate a descriptor as explained in subsection 2.3 and train SVM.

2.1 Distance thresholds

We think of each feature as a simple classifier. If its distance to image is less than the threshold, the classification outcome is positive and negative otherwise. We accumulate feature distances to positive/negative images in a pair of histograms. Going over bin boundaries we find a threshold that minimizes the number of mis-classifications for our training set.

2.2 Feature selection

Since usually there's a large number of candidate features, we need to choose some small subset of them. One way to do this is a simple greedy algorithm that selects a feature(s) that had the smallest number of mis-classifications while calculating the thresholds. One disadvantage of such approach is that similar features may be chosen. We apply a variation of boosting procedure,

Figure 1: Model features over a grid

which begins with uniform weights for all training set images. At each iteration we double the weight of the mis-classified image, and half the weight of correctly classified images. When choosing the feature, its correct and incorrect classifications are weighted with their corresponding weights.

2.3 Image descriptors

For each image we build a descriptor vector. The length of a vector is a number of features chosen in subsection 2.2. Each coordinate correspond to a single feature and is 1 if the distance from the image to this feature is less than this features' threshold and 0 otherwise.

3 Detection Algorithm

For a novel image we calculate its descriptor exactly as stated in subsection 2.3 and test against the trained SVM.

(a) Image#1 (b) Image#2

Figure 2: Car images in the database

4 Training database

The project required training/test database. A collection of images was made that contained cars. The images were made using a hand held mobile device camera. Most of the images were made during the morning/noon hours which means lots of sun and some shades. Please see Figure 2 for an example of a random images from the database.

5 Various distance measures

During the testing we've experimented with different distance measures except ncc, which also include SAD, SSD. We didn't see a significant performance gain using any one of the metrics.

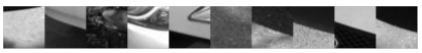

Figure 3: The features selected by the algorithm

6 Results

Selected features. One of the goals of the algorithm was to choose the most discriminative features, that separated well the images that have cars against the images that don't. The Figure **??** has patches that were selected by the algorithm. One may see that part of the images are indeed the parts of the car, while significant part is just 'noise'. I think that the reason for this is insufficient training data.

ROC To train and assess the performance of the SVM we used n-fold (n=10) cross-validation technique. See Figure **??** for the ROC. Cross-validation splits the data into the training set and the test set (usually 10% of the data is used for testing). The SVM is trained using the training set and tested using the test set. The process is repeated n times and the results are averaged to obtain the attached graph. One may see that the algorithm will detect about 85% of cars while producing 10% false positive results.

7 Conclusions

While the method performs at a level satisfactory for a small project in may be greatly improved in many respects. To name a few:

1. Better dataset should be prepared
2. More appropriate features should be used (allowing at least some tolerance to variability of views and lighting)

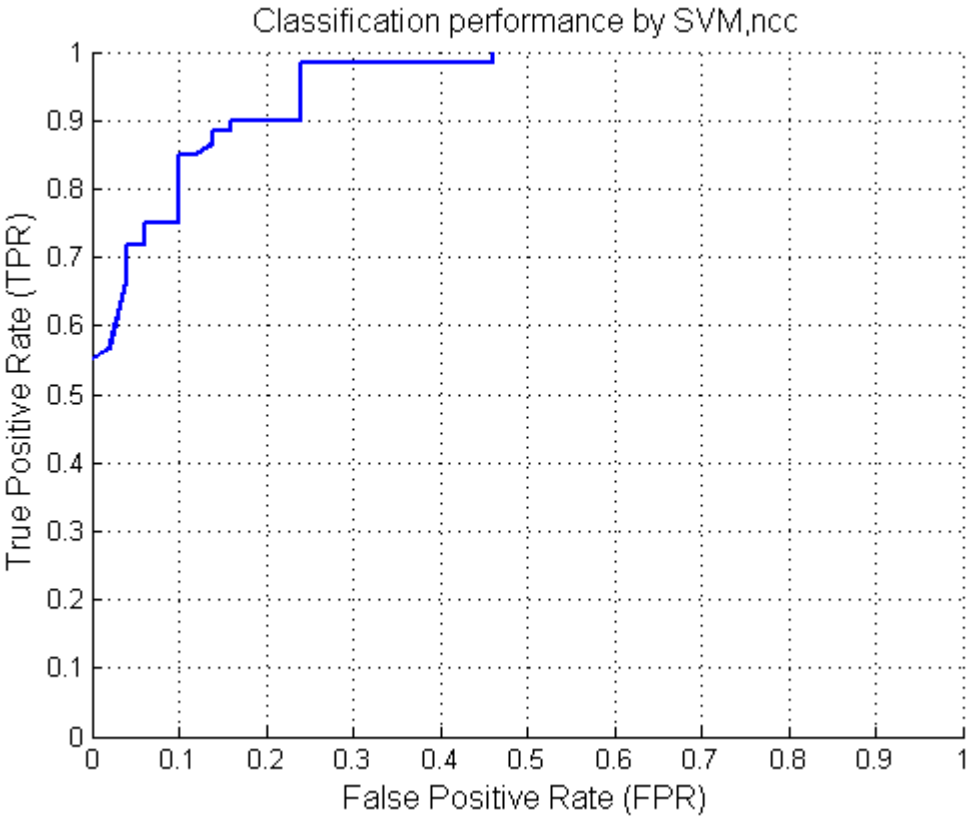
Classification performance by SVM,ncc
True Positive Rate (TPR)
False Positive Rate (FPR)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Figure 4: ROC