CS228: Project Report
Boosted Decision Stumps for Object Recognition

Mark Woodward

May 5, 2011

1 Introduction

This project is in support of my primary research focus on human-robot interaction. In order to study human-robot interaction I require a method for detecting known objects in the robot's environment. In this paper I describe my implementation of a vision algorithm described in [1] which trains a classifier using GentleBoost and decision stumps. The algorithm averaged zero false negatives and 17.4 false positives per frame on the test set, with a simple extension this improved to zero false negatives and zero false positives on the test set. This algorithm is compared with an algorithm that uses PCA for dimension reduction and then trains a perceptron on the training set in the reduced space. This comparison algorithm averaged 0.1 false negatives per frame, and 122 false positives per frame on the test set, with the same extension the average false negatives dropped to zero and the average false positives dropped to 15.2 per frame. The false positive rate of the PCA + Perceptron algorithm is too high for actual use, but the boosted decision stumps algorithm looks promising.

2 Problem

Figure 1 shows my robotic platform and the object of interest in the study; a medicine bottle. In order to focus on the problem of human-robot interaction, a reliable object detection algorithm is required. For example, in one of my experiments, the robot moves objects around as the human gives feedback. In order

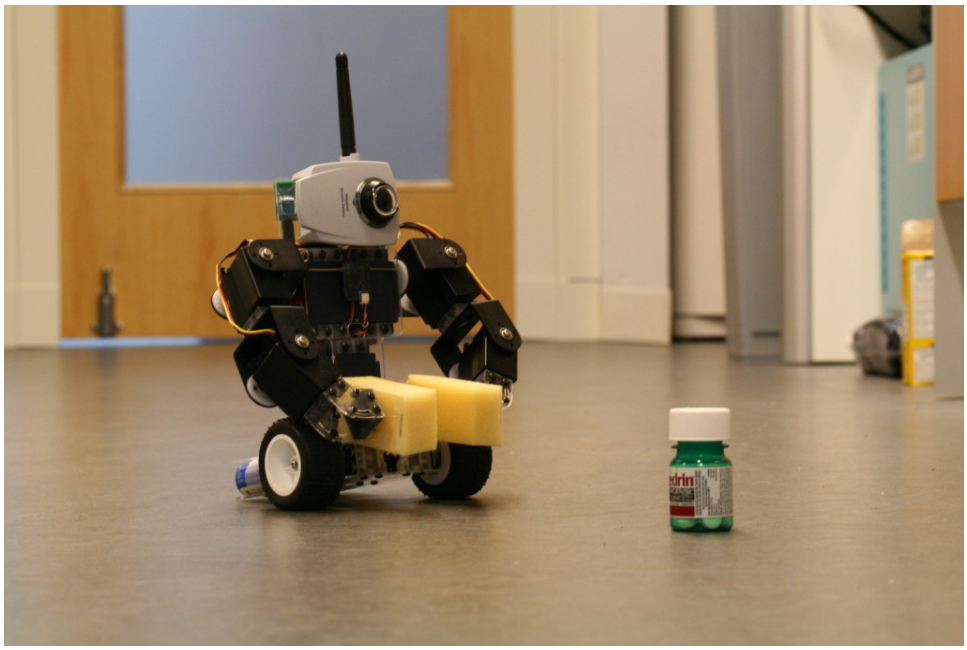

Figure 1: My Robotic platform and the and the object of interest, a medicine bottle

for the robot to move an object it must first detect the object in it's video stream. My CS228 project addresses the problem of detecting known objects in an image.

3 Data

The training examples are 32 x 32 pixel grayscale images. There are 50 positive images and 1000 negative images. Figures 2. and 3. show four positive images and four negative images respectively.

Figure 2: Four of the fifty positive training images (32 x 32 pixel).

Figure 3: Four of the one thousand negative training images (32 x 32 pixel).

4 Algorithms

4.1 GentleBoost with Decision Stumps

This algorithm is presented in [1] and [2]. Training consists of four steps:

1) Extract "patch" dictionary

2) Compute patch responses over training data

3) Learn GentleBoost classifier, with decision stumps

4) Trim the dictionary

4.1.1 Extract "patch" dictionary

From the positive training images (50 in my case), sample 2000 rectangular patches of width and length between 7 pixels and 14 pixels and record their location. These 2000 patches represent potential features of interest for the object in questions; for example, the edge between the white lid and dark bottle. Boosted decision stumps will be used to select patches that separate the training set. Figure 4 shows 35 of the 2000 patches. These are actually the 32 patches that GentleBoost selected for the 50 decision stumps.

Figure 4: The thirty five patches used in the fifty stumps selected by the boosting algorithm.

4.1.2 Compute patch responses over training data

Once the 2000 patches are extracted, we compute the response of the patches with each of the 1050 training images (50 positive, 1000 negative). This results in a 2000 dimensional vector for each of the 1050 training images. We use normalized cross-correlation to compute the "response" for a patch with an image. If p is the patch, l was the location of the patch when it was extracted, and I is the image, then the response of the patch for that image is

$$response = \frac{1}{\sum p} \sum_{x,y} (p(x,y) - \bar{p}) \times (I(l_x + x, l_y + y) - \bar{I})$$

where $\bar{p}$ is the mean of the pixels in the patch and $\bar{I}$ is the mean of the pixels in the image that the patch overlaps with.

4.1.3 Learn GentleBoost classifier, with decision stumps

The GentleBoost algorithm is described in [3]. Using decision stumps as the weak learner, each round of GentleBoost chooses a single patch to split on. GentleBoost fits the weak learner by weighted least squares of y_i on x_i with weights w_i, $y_i \in \{-1, 1\}$. Friedman contrasts GentleBoost with Adaboost in [3] and concludes that in addition to a nice statistical motivation, GentleBoost has fewer issues with numerical instability when implemented on a computer. Here is the GentleBoost algorithm with decision stumps.

Let N be the number of examples (1050 in our case). The strong hypothesis $H(x)$ is initialized to zero and the example weights w_i are initialized to 1. On each loop t of the GentleBoost algorithm we will choose to split on the patch p that minimizes

$$J_{wse} = \sum_{i=1}^{N} w_i (y_i - h_t(x_i))^2,$$

where

$$h_t(x) = a\delta(x^p > \theta) + b\delta(x^p \leq \theta),$$

and $\delta(statement)$ is 1 if *statement* evaluates to *true* and 0 otherwise. We can evaluate θ at the 1050 values of the response of patch p with the 1050 images. Given θ, the a and b that minimize the weighted squared error J_{wse} can be solved as follows:

$$a = \frac{\sum_i w_i y_i \delta(x_i^p > \theta)}{\sum_i w_i \delta(x_i^p > \theta)}$$

and

$$b = \frac{\sum_i w_i y_i \delta(x_i^p \leq \theta)}{\sum_i w_i \delta(x_i^p \leq \theta)}.$$

The stump $h_t(x)$ corresponding to the p, a, b, θ that minimizes J_{wse} is then added to the strong classifier $H(x) = H(x) + h_t(x)$, and the weights are updated $w_i = w_i e^{-y_i h_t(x_i)}$. In our case we loop 50 times, generating 50 decision stumps. The patches corresponding to the 50 decision stumps are shown in Figure 4. There are only 35 patches because some of the patches were used in more than one of the 50 stumps.

4.1.4 Trim the dictionary

GentleBoost constructed 50 decision stumps, with each stump corresponding to one of the 2000 patches. Thus, the strong classifier makes use of at most 50 of the patches (in our case 35 since some patches were reused). This speeds up classification, since you only need to compute the response of 50 patches with the test image, rather than the response of 2000 patches. In this way, boosting with decision stumps performs feature selection for us, which is the whole point of this approach.

4.2 PCA Reduction with Perceptron Algorithm

The PCA + Perceptron algorithm is pretty straight forward. We first run PCA on the 1050 training images, where each (32 x 32) image is treated as a 1024 element vector. We choose the 50 eigenvectors, corresponding to the top 50 eigenvalues, as the basis to reduce our images to.

Each example image is then projected onto the 50 eigenvectors, resulting in 50 features per example image. A perceptron is then trained on the reduced examples.

5 Results

Ten 1024x768 pixel frames, that were not involved in training procedure, are used to test the algorithms. Figure 5 shows 3 of these test frames. Each frame contains one medicine bottle. We test by creating 10 copies of each image, scaled by 0.8^i, $i = 0$ to 9, and sliding a 32x32 pixel window across and down, pixel by pixel, each of the images at each of the scales. The 32x32 pixel windows are passed to the two algorithms which classify the window as containing an image or not. I report the false positives and false negatives per image.

Figure 5: Three of the ten test frames. The medicine bottle in the left image is in the center of the image at the top edge of the carpet.

5.1 GentleBoost with Decision Stumps

False negatives=0, False positives=174 (mean=17.4, std=2.2).

As seen in Figure 6 there were many false positives, but the window with the highest confidence did correspond to the actual medicine bottle in all 10 test images.

5.2 PCA Reduction with Perceptron Algorithm

False negatives=1, False positives=1226 (mean=122.6, std=3.8)

On the training images in the reduced space, the perceptron algorithm was able to reach perfect classification, which was surprising to me (it did take about 30 minutes to run). I had thought that there would be no way that a linear classifier could classify 1050 50 dimensional examples. When run on the test set, in only one of the 10 test frames did the perceptron fail to identify the medicine bottle (false negative), which is pretty good. Unfortunately, there were tons of false positives (122.6 per frame). And the window with the best response did not correspond to the medicine bottle in any of the 10 test frames. See Figure 7.

6 Extension

The false positives for both of the above algorithms were too high for my uses, so I implemented an extension also used in [2], which addresses this problem. Given a trained classifier as above (boosted decision stumps or PCA+Perceptron), and a set of frames that do not contain a medicine bottle (for this we can crop out the medicine bottle from the training frames), we can run the classifier on these cropped frames to generate thousands of negative example images. We can then retrain the classifier on the original example images plus this new set. Figure 8 shows two of the fifty cropped images that are used in this technique. Below we report on the results of applying this technique to each of the algorithms.

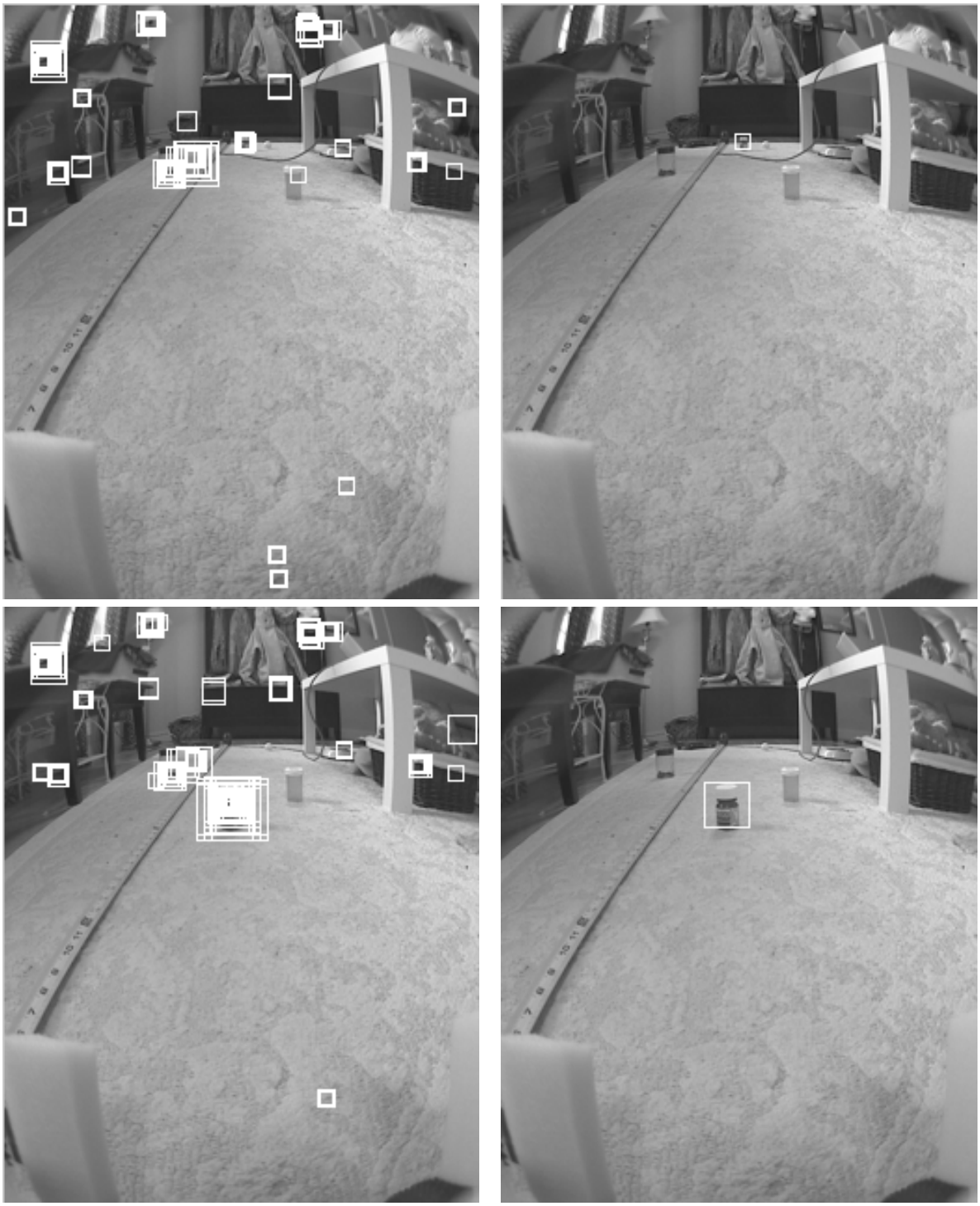

Figure 6: GentleBoost with decision stump matches and "best match" for two test frames. Matches are on the left and the best matches are on the right

6.1 GentleBoost with Decision Stumps

False negatives=0, False positives=0

When run on the 50 cropped frames, 6,086 additional negatives were generated. Once retrained, this technique performed perfectly on the 10 test images; see Figure 9.

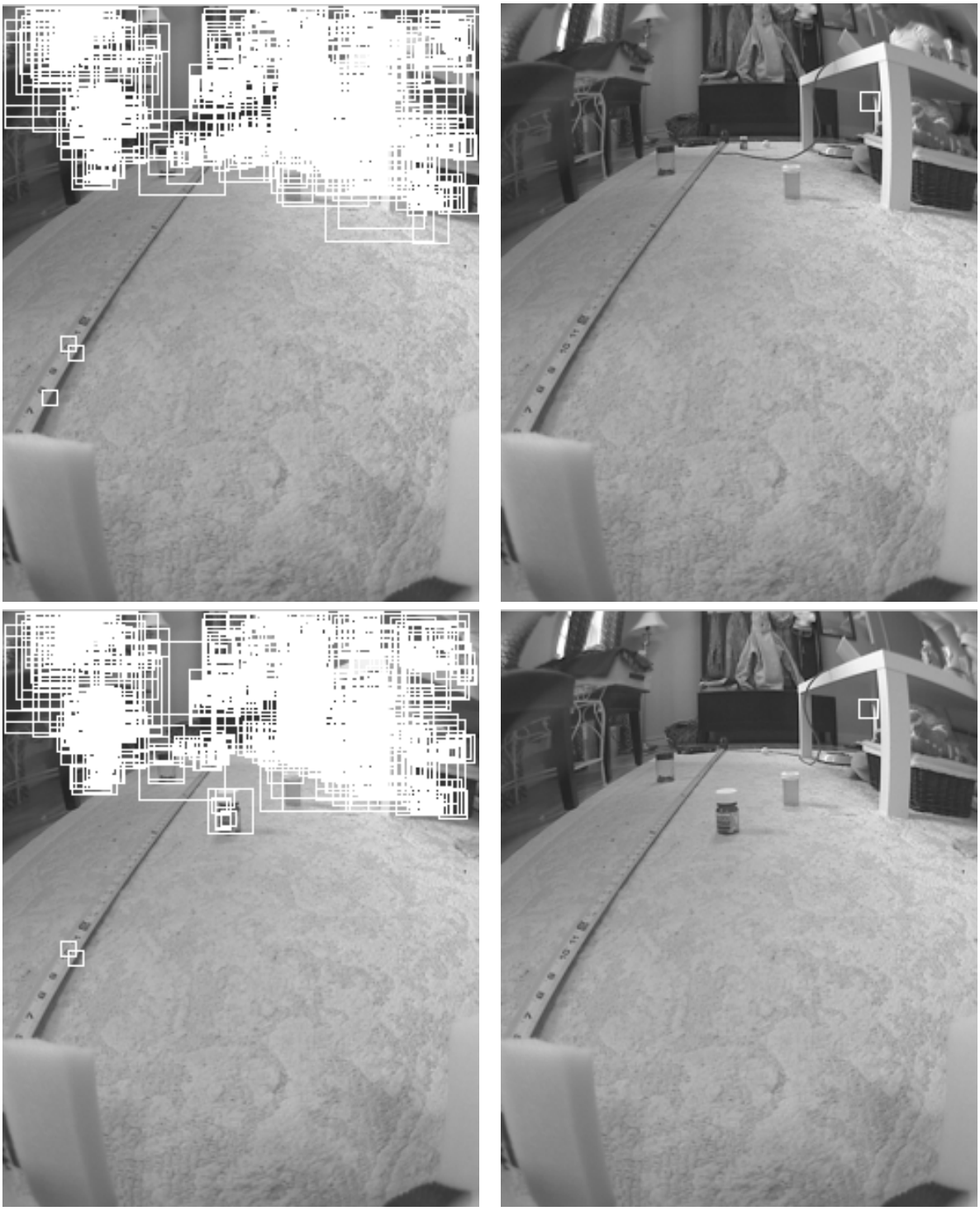

Figure 7: PCA reduction with perceptron algorithm matches and "best match" for two test frames. Matches are on the left and the best matches are on the right

6.2 PCA Reduction with Perceptron Algorithm

False negatives=0, False positives=152 (mean=15.2, std=2.8)

When run on the 50 cropped frames, $7,997$ additional negatives were generated. Here I limited the number of additional negative examples, as I would have had to significantly change the algorithm to handle

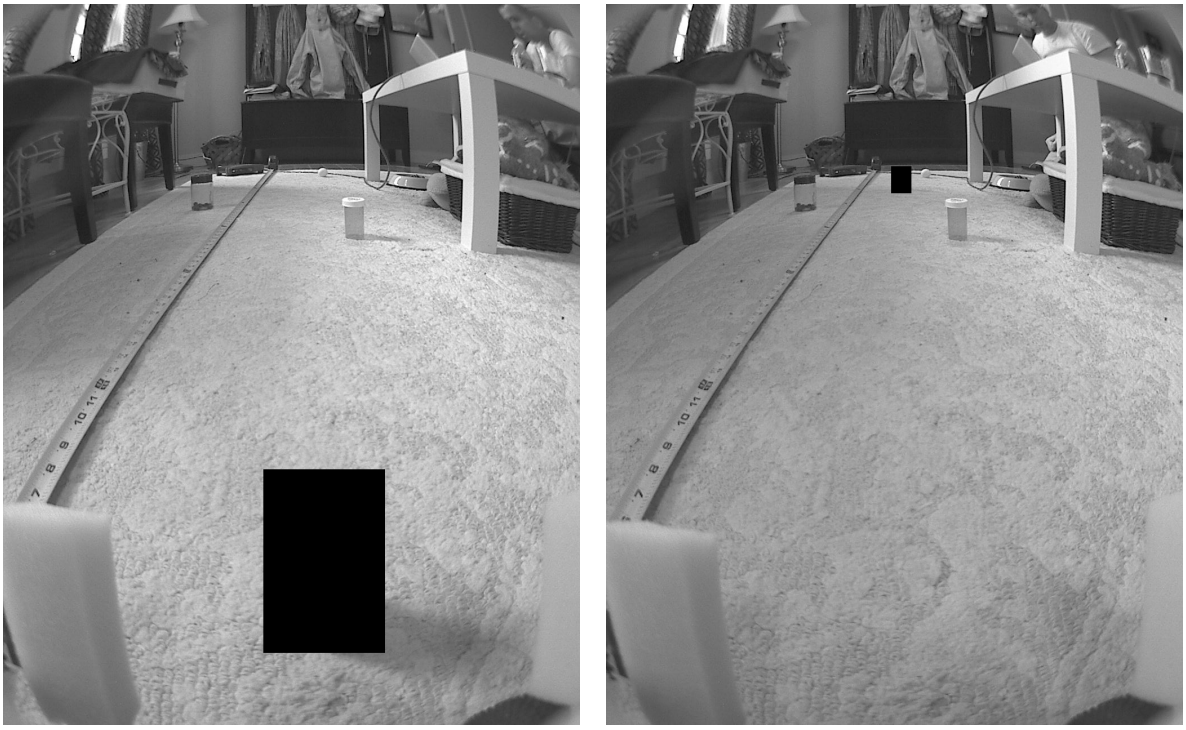

Figure 8: Two of the fifty training frames with the object of interest cropped out. Any matches in cropped frames are treated as additional negative (32 x 32 pixel) examples

all identified false positives in memory. Once retrained, there were now no false negatives (all medicine bottles were found) and the number of false positives was significantly reduced to 15.2 per frame. Unfortunately, the window with the best response still did not correspond to the medicine bottle in any of the 10 test frames. See Figure 10 for examples of the performance.

As without the extension, the perceptron algorithm was able to correctly classify all training images. This really surprised me that a linear separator could be found for 9057 examples (1050 + 7,997) in a 50 dimensional space.

7 Conclusion

I was very pleased with the boosted stump (with extension) algorithm's performance; you can't get better than perfect :). The results warrant further exploration for my application, with more complex training and test situations. I was also impressed with the performance of the perceptron algorithm, though it is not yet usable. Boosting the perceptron would definitely improve performance. Also, it is not surprising that PCA+Perceptron did worse, since the PCA reduction picks eigenvectors corresponding to variance in the examples, but not necessarily variance that exists between classes.

Overall I am pleased with the results, and I definitely have a new appreciation for the power of boosting and surprisingly the power of the simple perceptron.

References

[1] A. Torralba, *Sharing Visual Features for Multiclass and Multiview Object Detection*, NIPS 2007.

[2] M. Quigley, *High-Accuracy 3D Sensing for Mobile Manipulation: Improving Object Detection and Door Opening*, ICRA 2009.

[3] J. Friedman, *Additive Logistic Regression: a Statistical View of Boosting*, The Annals of Statistics 2000, Vol. 28, No. 2.

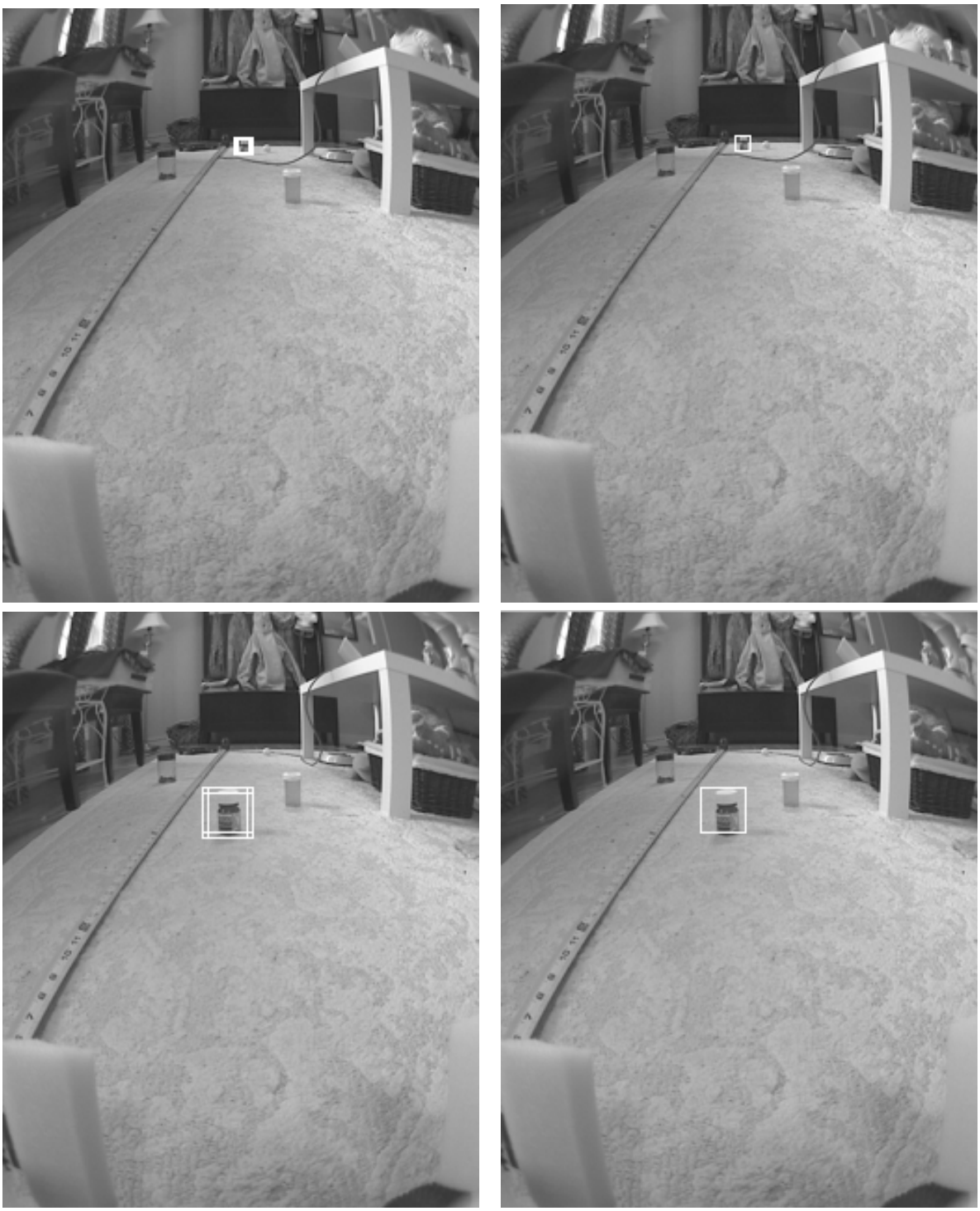

Figure 9: GentleBoost with decision stump (trained on auto-extracted negative examples from training frames) matches and "best match" for two test frames. Matches are on the left and the best matches are on the right

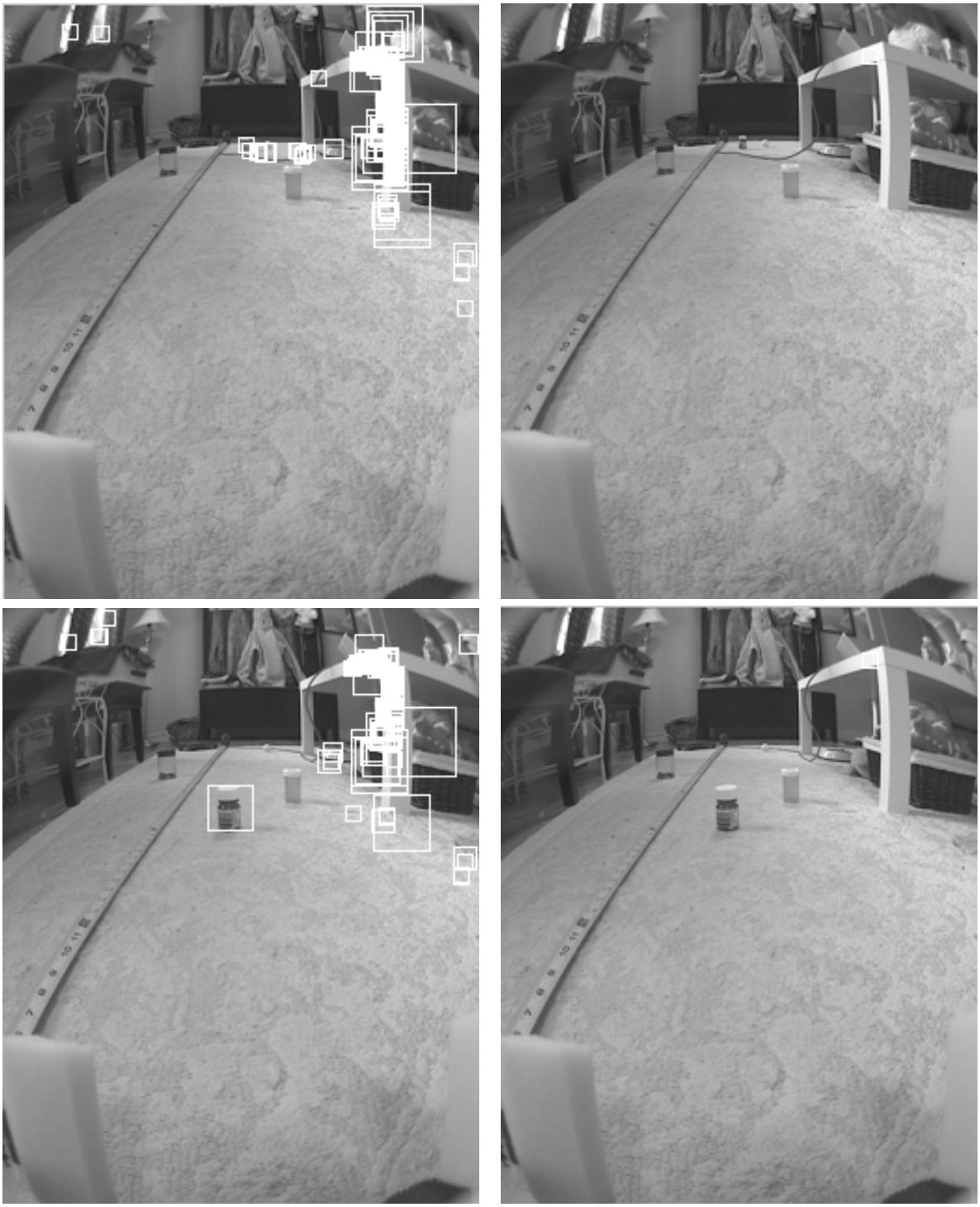

Figure 10: PCA reduction with perceptron algorithm (trained on auto-extracted negative examples from training frames) matches and "best match" for two test frames. Matches are on the left and the best matches are on the right (near the corner of the table)