

# Report: Reducing the error rate of a Cat classifier

Raphael Sznitman

6 August, 2007

## Abstract

The following report discusses my work at the IDIAP from 06.2007 to 08.2007. This work had for objective to build a classifier for cat images, and focused on difficult images. Two approaches were implemented; one was using the Chamfer distance metric for matching, and the other learned models of our training images and used the Kullback-Leibler divergence as a metric of image comparison. Noise in the training and testing image proved to be an overwhelming problem, tainting our results. Control experiments, however indicated sound procedures.

## 1 Introduction

Object detection has long been a principle goal of computer vision, with face detection as a main objective. Recently, work has gone into detecting more complex objects ( i.e. Objects with various poses, colors and textures) with various techniques. Fleuret and Geman [1] use pose-indexed features to learn specific pose classifier in a coarse-to-fine tree, allowing for complexly shaped objects to be learned and detected. An application from this theoretical framework was a cat detector.

The work reported here specifically looked at improving the number of errors which this cat detector provided. Given, were cat images and a list of image positions where the cat classifier had classified true positive and false positive alarms. The goal of this work was to be able to determine if the given locations indeed were a cat or not. In other words, this work tried to reject some of the false positive alarm without affecting the true positive alarms. Ideally being able to reject half of the errors, would have been more than satisfactory, seeing that the original cat detector performed very well.

Two different methods were implemented to do this task. The first involved using the Chamfer distance transform, in order to match test images to templates (a training image). Similar techniques had successfully been used in hand and pedestrian detection [2] [3]. Another approach implemented used learned image models and used the Kullback-Leibler divergence to measure the similarity between models and test images. Both of these approaches are described and discussed in more detail in this report.

For the sake of clarity, let  $\mathbf{T} = (t^1, \dots, t^N)$  be a vector, where  $t^l$  is a training images and  $l \in (1, \dots, N)$ . Similarly  $\mathbf{X} = (x^1, \dots, x^M)$  is a vector such that  $x^k$  is a testing images and  $k \in (1, \dots, M)$ . In our case, the image sizes vary, but computations are performed on images of size  $150 \times 100$  pixels, where the

center of the images is the head location of a cat. This size was determined because 90% of cat head locations fall 50 pixels away from the image boundary.

## 2 Chamfer Distance

Our initial approach, using a Chamfer distance as a measure of similarity between images, is straight forward and similar to the work done by [2]. The general procedure is to take a test image, extract features from the image (i.e. Extract edges from images), perform a Chamfer distance transform on a feature image, and then match templates (i.e. Training images having gone through feature extraction) with the distance image. The feature extraction and distance transform are both pre-processes done separately from the template matching in order to speed up testing. A full experiment is when all test cases, are matched by all training images

### 2.1 Image Features

Both  $\mathbf{T}$  and  $\mathbf{X}$  undergo feature extraction at a pre-processing stage in order to reduce computational costs. The features extracted are the edges within the images. To do this, first a Gaussian filter is applied to blur an image, followed by two derivative filter. As a result of this, there are two new sets of training and test edge images,  $\mathbf{T}_e$  and  $\mathbf{X}_e$ .

The derivative filter is a 1D arrays with values (-1,0,1). The filter is passed in different directions in order to get the x and y directional derivatives.

The Gaussian filter is constructed by taking a 2D matrix of height and length  $6\sigma + 1$ . This generally provides better blurring on large values of  $\sigma$ . Consequently, good silhouette images are difficult to obtain because tuning  $\sigma$  is non-trivial. This is particular so for objects which have strong noise within the desired object, such as cats. Further discussion on this topic is covered later.

### 2.2 Distance Transformation

Each image in  $\mathbf{X}_e$  is transformed to form a new set of images  $\mathbf{X}_d$ . The transformation is such that for each pixel in an image of  $\mathbf{X}_e$ , we calculate the euclidean distance to the nearest edge pixel (i.e. a pixel with value 0). To calculate the euclidean distance, we search for edges symmetrically in all directions, looking at the nearest neighbors first and expanding the neighborhood until an edge pixel is found. This is just a depth first search on the pixels. More optimal implementations should be looked at, as the one implemented is fairly costly.

### 2.3 Matching

Once  $\mathbf{X}_d$  and  $\mathbf{T}_e$  have been computed, matching a template to a test can be performed quickly. Let  $x^k$  be an image in  $\mathbf{X}_d$ , and  $t^l$  be an image in  $\mathbf{T}_e$ . Note that there is no searching in this procedure, because the exact location of the heads of training images were given. That is, given a point on the image, true positive test cases were positioned less than 25 pixels away from the true heads location. The two image heads are then superimposed onto each other at their

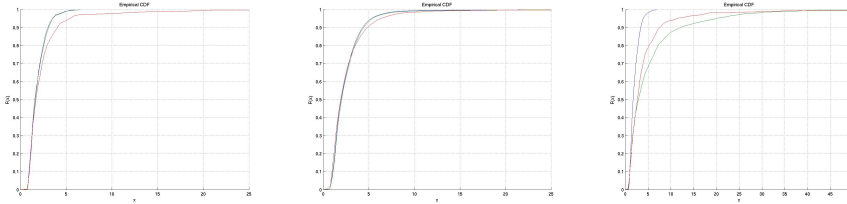


Figure 1: The Cumulative Distribution Functions for experiments I(left),II(center) and III(right). In each of the graphs, the blue line depicts  $\min(\Delta(x^k, t^l))|Y=1$  and the red  $\min(\Delta(x^k, t^l))|Y=0$

respective locations. Matching a test image to a template then becomes:

$$\Delta(x^k, t^l) = \sum_i \sum_j x_{i,j}^k I(t_{i,j}^l) \quad (1)$$

where  $I(\bullet)$  is the indicator function and  $i$  and  $j$  refer to pixel positions. If one regards the images as vectors, then this is simply the dot product of two vectors. That is, for every pixel, we sum the distance values which corresponding to an edge location of the template. If  $x^k = t^l$  then naturally  $\Delta(x^k, t^l) = 0$ , and so one can see that the more similar a template is to the test, the smaller  $\Delta(x^k, t^l)$  will be.

## 2.4 Experimentation

Three important experiments were performed using this approach. For the first experiment(I), after performing feature extraction and distance transforms, all the test images were matched against every template.

In the second experiment(II), a rotation of  $\theta$  was performed on the template image centers, for all values of  $\theta = [-20^\circ, 20^\circ]$ . The experimental procedure was the same as in the first experiment otherwise. This test was to account for rotational differences when matching the templates.

Lastly, the third experiment(III) was our control where all the true positive test cases were performed identically as in experiment I but random image location were chosen for other test cases. The random locations were chosen such that they did not fall too closer than 75 pixels from the image boundary.

In both the first and second experiment, there were 1500 training images and 898 test cases.

## 2.5 Results and Discussion

In all three experiments, the best 5% were then collected from both classes (True positive and False Positive)for data analysis. More formally, let  $Y$  be the real class of a test case, where  $Y = 1$  if a cat is present, otherwise  $Y = 0$ . We looked at the distribution of  $\min(\Delta(x^k, t^l))| Y = 0$  and also  $\min(\Delta(x^k, t^l))| Y = 1$ , where  $t^l$  is a template image and  $x^k$  is a test case. The  $\min(\bullet)$  refers to the minimum 5% to allow for more robust results.

From the first two experiments described above, results were inconclusive. In both I and II, the approach was not able to differentiate true positive points

from false positive ones. That is that the distribution of  $\min(\Delta(x^k, t^l)) | Y = 0$  and  $\min(\Delta(x^k, t^l)) | Y = 1$  were the same. This can be seen on the CDF curves in Figure 1(Left) and 1(Center).

From III however, we can observe that both distributions are separable. That is, when we test on random locations in an image, we can successfully distinguish cats from non-cats. This can be seen on Figure 1(Right). This also provides some indication that the general approach should work.

There are a few interesting points which can be observed from these results. First, that accounting for image rotations does not bring a benefit to our procedure. In fact, it does not make any difference compared to the original images. One could attempt to add multiple scales of the image as well, but its benefits are even less clear now.

Another interesting aspect is that expected results were seen from our control experiment (III). That is, detecting the presense of cats and non-cats location is manageable when the non-cat locations are picked randomly. This implies that the false positive test cases provided by the original cat classifier, are particularly difficult compared to an average point. It indicates that at some level some information is accounting for why the false positive location of test cases are particularly difficult to dissociate from true positive test cases. This is also what we expected because the original cat detector was not able to correctly detect in those cases.

After these results had been observed, a new experiment was performed. In this case, a hand full of images from  $\mathbf{X}_e$  were cleaned in order to only have outline edges of a cat. Because, cleaning the images requires a lot of time, only a few were done ( $\sim 10$ ). The procedure was run on these "Clean" images, and results were collected as was done in prior experiments. When the test images were "Clean" the distributions of  $\min(\Delta(x^k, t^l)) | Y=0$  and  $\min(\Delta(x^k, t^l)) | Y=1$  were indeed separable. Because only a few test cases were available, these results are not shown but yet tell us much. Results like these indicate that there is there is too much noise in the images for proper detection. Similarly, this undersized set of results confirm this approach is viable with low levels of noise and functions in principle.

From our experimentation, it appears as though the problem of noise is an important one for this approach. Observing the images from  $\mathbf{X}_e$  and  $\mathbf{T}_e$  reveal that either too many from the background edges and edges within the cat are present, or not enough to make out the overall shape of the cat. The problem seems to grow when both the test image and the template suffer from this noise.

### 3 Kullback-Leibler Divergence

In this approach, we begin by partitioning each image in both  $\mathbf{X}$  and  $\mathbf{T}$ , into blocks. A histogram of the gradient direction is then constructed in reference to each block for all images. Using the Kullback-Leibler divergence, the N most similar template images are found for each template. For each template, the average and standard deviation of each block is calculated using the N closest images found earlier. Finally, a test case is passed in a Gaussian model, using the template statistics. Below is a more detailed descriptions of these steps.

### 3.1 Partitioning Images and Histograms

All images in  $\mathbf{X}$  and  $\mathbf{T}$  are partitioned into 150 blocks of 10 x 10 pixels. That is, for any given image, we create  $B = (b_0, \dots, b_{149})$ , where  $b_i$  is a 10 x 10 pixel block. Performing this on all images in  $\mathbf{X}$  and  $\mathbf{T}$ , provides us with a vector  $\beta_x = (B^0, \dots, B^N)$  and  $\beta_t = (B^0, \dots, B^M)$

Applying the same edge detector from section 2.1, we create orientation gradient histograms. That is,  $b_i$  references a specific block of an image but is a histogram with 8 bins.. From the edge detector, the gradient direction of a pixel is calculate for a given block. This pixel votes towards one of eight possible directions, such that all pixels in a block form a histogram of the gradient direction. All histogram bins are initialized with one vote in order to avoid computational problems with the Kullback-Leibler divergence. Similarly, pixels which vote must be above a specific gradient threshold. This threshold is a hyperparameter and is difficult to set.

In some cases, given head locations in the real images were to close to the image boundary, not providing a full 150 x 100 pixel rectangle. For that reason, some blocks in images of  $\mathbf{X}$  and  $\mathbf{T}$  are informative. This being said, even bins in uninformative blocks have been initialized.

Having transformed an image in a set of histograms allows for comparisons of probabilities using Kullback-Leibler divergence.

### 3.2 N-Nearest Images

Here we are interested in finding, for each training image, which N (N=25) other training images are most similar to it. More specifically, for an image  $B^i$  in  $\mathbf{T}$ , find the 25  $B^t$ , where  $t = (0, \dots, M)$ , that minimize:

$$f(B^i, B^t) = \sum_b KL^2(B_b^i, B_b^t) \quad (2)$$

where  $KL(\bullet)$  is the Kullback-Leibler Divergence and is defined as:

$$KL(A, B) = \sum_i A(i) \log \frac{A(i)}{B(i)} \quad (3)$$

In this case, our histograms can easily be changed into probabilities and so calculating the divergence is trivial. For this we create the vector  $N = (n_0, \dots, n_M)$ , such that  $n_i$  is a vector which holds the indexes of the 25 closest images to  $i$ . In order to find the 25 closest images, we used Quicksort. Just as in our first approach, matching the two images required to superimpose the two given head locations on each other. Because of uninformative blocks, we also have to normalize  $f(B^i, B^t)$  in order to only account for important blocks. Normalizing  $f(B^i, B^t)$  is done by:

$$f(B^i, B^t) = \frac{1}{\beta} \sum_b KL^2(B_b^i, B_b^t) \quad (4)$$

where  $\beta$  is the number of relevant blocks used in this comparison. In order to reveal if a block is relevant we use the entropy of the histogram. Defined as:

$$H(A) = - \sum_i A(i) \log(A(i)) \quad (5)$$

relevant blocks were those below a set threshold.

At this point, we can view which images were most similar to any given training image. This can provide some insight into whether or not things are performing as expected.

In principle many different measure of distances could be used to find the most similar images. The use of Kullback-Leibler divergence as a measure of similarity was chosen out of interest. The angle between two vectors (Histograms in our case) could just as easily been used.

### 3.3 Training Statistics

Once we have found which training images are similar to each other, we can build models of training images. In principle this reduces the dependency when only looking at one given image, by aggregating data from multiple similar images. This reduces the influence of noise from a specific image.

For this, we create two new vectors  $\boldsymbol{\mu}$  and  $\boldsymbol{\sigma}$  such that  $\boldsymbol{\mu} = (\mu_0^0, \dots, \mu_{149}^0, \mu_0^1, \dots, \mu_{149}^M)$  and  $\boldsymbol{\sigma} = (\sigma_0^0, \dots, \sigma_{149}^0, \sigma_0^1, \dots, \sigma_{149}^M)$ . For any given  $\mu_b^t$  and  $\sigma_b^t$  can be calculated by:

$$\mu_b^t = \frac{1}{N} \sum_n KL^2(B_b^t, B_b^{t_n}) \quad (6)$$

and

$$\sigma_b^t = \frac{1}{N} \sum_n (KL^2(B_b^t, B_b^{t_n}) - \mu_b^t)^2 \quad (7)$$

where in both cases we are calculating our statistics from the 25 images which were most similar.

### 3.4 Gaussian Model

When performing a test, we compare each block from the test case with a training image, using the Kullback-Leibler Divergence and at the same time normalizing the result with the statistics from that particular training image. The idea is to learn the model of each training image individually and use these models to test our data. More formally when comparing images  $x^k$  and  $t^l$  then

$$\Delta(x^k, t^l) = \frac{1}{Z} e^{-\sum_b \frac{(KL^2(B_b^k, B_b^l) - \mu_b^l)^2}{2(\sigma_b^l)^2}} \quad (8)$$

where  $Z$  is a normalizing constant in order to have a density. Responses are in the range from 0 to 1. In this model, the best possible response is a test case which is the same as the average of a template. This is in order to reduce the influence of the strong noise in the images.

### 3.5 Experimentation

Two experiments were conducted to test this approach. The first was a control check to see if the general procedure worked on a simplified case. The second experiment consisted on testing the cat images as we had done in section 2.4.

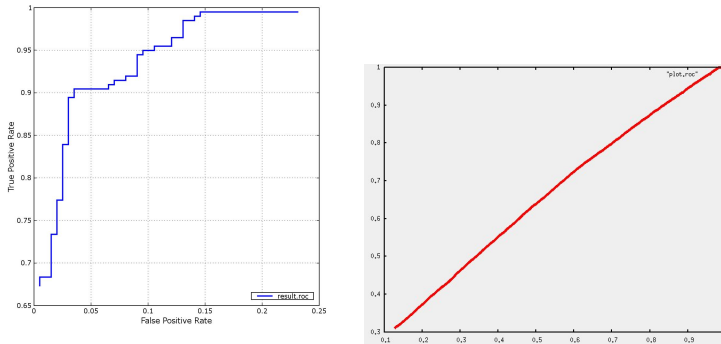


Figure 2: The Response Operator Curve for experiments I(left) and II(right).

As a control experiment(I), we create three types of synthetic images ( a vertical line, a horizontal line and a triangle ). We create 100 such images, in equal proportions, and each time placing random noise in the actual position of the object. In addition, random noise was added in blobs, to simulate the noise seen in the cat images. 6 test images were created by hand, 3 of them highly resembling an existing image (True Positive class), and 3 images with different shapes (False Positive class).

The control test involved testing each test case with each training image we had created. We retained the best 5% for analysis.

The second experiment(II) consisted in testing 898 test cases of true positive and false positives in images of cats and non-cats. There were 1500 training images to learn from. Again, we retained the best 5% for analysis.

### 3.6 Results and Discussion

From experiment I, our results were reassuring and generally positive. Figure 2 (left) is a ROC curve displaying the result from experiment I. In this control, the approach is able to correctly make out between the two true positives and the false positives. There is a clear difference in the distributions of the two classes, and that is why we can reject 95% of the false positives while keeping 90% of the true positives.

These result do also tell us something about our general procedure. First, we can see from the control experiment, that the general procedure works, and does provide a way to classify objects. To further justify this claim we should add more tests to this control to make sure that this is true, but the current result push to this conclusion.

An important observation not shown in the given results, are the images chosen when finding the N-Nearest neighbors (section 3.2). When viewing which images are chosen in this control experiment, one can see that the same type of objects are chosen to form the model of a given training image. This is precisely what was expected. That is, even though training images were different from each other, an image of a vertical line for example, chose other vertical lines as its closest neighbors.

Experiment II however show that the distribution of false positives in the same as the true positives, and that our detection process cannot make the difference between the true positive and false positive test cases. This ROC Figure 2 (right) curve shows that at no threshold can we significantly reduce the error rate. With the real training and test data our approach barely does better than randomly assigning classes.

There is one big issue which is quick to examine from this specific experiment concerning the N-Nearest neighbors (section 3.2). When observing which images are picked to create an image model, there is no noticeable similarity in the images chosen. That is, if a training image has a cat in a particular pose, the closest neighbors will not be visually similar.

To make sure that these observation were not due to our implementation, another similar experiment was conducted this time replacing the Kullback-Leibler Divergence for another metric ( Angle size between two vectors ). This change had no impact on the models created on a given training image.

This latter result suggests that the cause of these result stems prior to learning image models and most likely during feature extraction. Feature extraction is difficult in this case, not only because the data itself is complex, but because there are also two hyperparameters to adjust. The hyperparameters appear to be a lot more difficult to set than expected.

## 4 Conclusion

Even though our initial goal was not achieved, much has been learned from the two approaches above. It is now clear that the use of Chamfer distance works, but only in somewhat noisy environments. The Chamfer distance gave good results when only one of the two initial sources had noise, but failed when both signals carried large noise levels. Building models of training examples proved to be highly dependent on the amount of noise in images. Even though favorable results were found for synthetic data, cat images proved to be too noisy as well.

Perhaps the take home message of this report is how vital the initial image treatment and feature extraction are. As some of our data has shown, in the case of clean images, both of the approaches can be used to detect objects. Setting parameters properly is both non-trivial and time consuming. But these general result point out to two questions which deserve further investigation.

First, are the test cases presented particularly special? Due to the fact that the original cat classifier miss-classified these particular test cases, there is something different with these images which make them particularly difficult. For this reason, it would be interesting to conduct tests on easier examples and see how our implemented approaches fare.

Secondly, in the original cat detector, edges were also used as features. It would be interesting to observe the output of the edge detector on the test images in particular, and also use edge detector of the original cat detector ( as it is a little different ) to extract features.

## References

- [1] F.Fleuret and D.Geman. Focusing on cats (2007)

- [2] D.Gavrila, J.Giebel, M.Perception, D. Res and G. Ulm. Shape-based pedestrian detection and tracking. *Intelligent Vehicle Symposium IEEE*, volume 1, pages 8-14, (2002)
- [3] A. Thayananthan, B. Stenger, P. Torr and R. Cipolla. Shape context and chamfer matching in cluttered scenes. *Computer Vision and Pattern Recognition*, volume 1, pages 1-127, (2003)
- [4] M. Butt, P. Maragos. Optimum Design of Chamfer Distance Transforms. *IEEE Transaction on image processing*, volume 7, pages 1477-1484, (1998)

## Acknowledgements

I would like to thank my supervisor Francois Fleuret, for the opportunity to do this work and his guidance. I thank Johny Marietho as well for his help regarding technical issues.