Investigating PCA image reconstruction for pedestrian detection

Guilherme V. Carvalho, Lailson B. Moraes, George D. C. Cavalcanti, Tsang I. R.
Centro de Informtica (CIn)
Universidade Federal de Pernambuco (UFPE)
Recife, Pernambuco, Brasil
http://cin.ufpe.br/~viisar
{gvc,lbm4,gdcc,tir}@cin.ufpe.br

***Abstract*—The abstract goes here. DO NOT USE SPECIAL CHARACTERS, SYMBOLS, OR MATH IN YOUR TITLE OR ABSTRACT, AND I'M NOT SCREAMING!. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.**

***Keywords*-TODO; component; formatting; style; styling;**

I. Introduction

Pedestrian detection systems have been widely used and developed throughout computer vision history. Ranging from still image detection to automated car breaking systems, it is becoming a usual task in our lives.

In this paper we describe a pedestrian detection proposed by Malangon-Borja [1] that classifies an image based on how much its reconstruction is different from the original one, that is, it classifies based on the image reconstruction error. We analyze how it works and how its original performance can be improved, as well as its detection times, by using weights on the reconstruction errors and reducing the number of calculations made by the algorithm, respectively.

The remaining of the article is divided as follows. Section 2 describes a image reconstruction technique using PCA. Section 3 describes how to use image reconstruction to classify images. Section 4 presents our experiments and discussions. In section 5 conclusions are presented and we indicate areas of future research.

II. Image reconstruction with PCA

The idea of principal component analysis (PCA) seems to have been first proposed by Pearson in 1901 [2] and was later developed in many works during the last century, being now known as the Karhunen-Love extension, the Hotelling transform or proper orthogonal decomposition, depending of the field of application [2]. It was popularized in computer vision by Turk and Pentland, that in 1991 used PCA to propose the eigenfaces [3], a face recognition method still relevant and used in current days.

Given a set of samples, PCA yields a set of orthonormal vectors that can be used to linearly project the samples into a new space. This space maximizes the scatter of the projected samples (their variance) and minimizes the least mean square error, that is, it minimizes the difference between the projected sample and its reconstruction back to the original space [4]. This characteristic also makes PCA suitable for data compression and dimensionality reduction. The data can be anything, but in this work we are concerned only with images.

The formulation of PCA is as follows [1]. Consider a set of m images, each with r pixels of width and c pixels of height (i.e. of size $r \times c$ pixels). Each image I_i is represented by a column vector v_i of length rc. The mean of the set is defined by

$$\mu = \frac{1}{m} \sum_{i=1}^{m} v_i \tag{1}$$

and the covariance matrix C is given by

$$C = \sum_{i=1}^{m} (v_i - \mu)(v_i - \mu)^T. \tag{2}$$

PCA applies eigen-decomposition on C and keeps the k eigenvectors ($1 <= k <= rc$) corresponding to the k largest eigenvalues. These eigenvectors are called the *principal components* (PCs) of C. It is proven that a projection onto the space defined by these eigenvectors provides the optimal reduced representation of the data, minimizing information loss [4].

Let P be the matrix whose columns are the first k principal components of C. The projection p of an image u into this eigenspace is given by

$$p = P(u - \mu). \tag{3}$$

As the projection is reversible, we can try to recover the original version of a previously projected image. The result is known as the *reconstructed image* [1]. The reconstructed image u' of a projection p is

$$u' = P^T p + \mu = P^T P(u - \mu) + \mu. \quad (4)$$

This process can be interpreted as form of lossy compression and decompression: the reconstructed image usually is not equal to the original one, being just an approximation of it. For this reason, each reconstruction has an associated *reconstruction error*, which can be measured by

$$d = |u - u'| = \sqrt{\sum (u_i - u'_i)^2}. \quad (5)$$

In general, the more principal components we use to obtain a projection, the less information loss we have, what allows a more accurate reconstruction. Also, the more similar u is to the images that were used to produce P, the better the reconstruction will be for a fixed number of eigenvectors [1].

III. Classification using reconstruction

By definition, PCA looks for the set of PCs that best describe the distribution of the data that is being analyzed. Therefore, these PCs are going to preserve better the information of the images from which PCA was performed, or of those that are similar to them [1]. For example, consider a set of PCs obtained only from images that contains pedestrians. They must reconstruct better images of other pedestrians than any other type of images. Conversely, if we have a set of PCs obtained from images of anything except pedestrians, the reconstruction of pedestrian images must be poor.

Based on this fact, Borja and Fuentes propose a classifier for pedestrian detection that uses reconstruction errors as the classification criteria [1]. In addition to grayscale images, the classifier also use the corresponding computed edge images, since they characterize better the pedestrian silhouette and eliminate background variations.

In the training phase, PCA must be performed separately for four sets of images, what results in the following sets of PCs:

- The principal components P_{gp} and the mean μ_{gp} from a set of pedestrian grayscale images.
- The principal components P_{ep} and the mean μ_{ep} from a set of pedestrian edge images.
- The principal components P_{gn} and the mean μ_{gn} from a set of non-pedestrian grayscale images.
- The principal components P_{en} and the mean μ_{en} from a set of non-pedestrian edge images.

In the classification phase, when a new grayscale image has to be classified, the first step is obtain its edge image and perform four reconstructions: one for each set of PCs. Then, the reconstruction errors must be calculated and combined to produce the total error, that is the final classification score. If the total error is greater than or equal to zero, the image is classified as a pedestrian; otherwise, it is classifier as a non-pedestrian. The detailed procedure is described above.

1) Obtain the edge image e from grayscale image g using the Sobel operator.
2) Perform four image reconstructions:
   a) $r_{gp} = P^T P_{gp}(g - u) + \mu_{gp}$
   b) $r_{ep} = P^T P_{ep}(e - u) + \mu_{ep}$
   c) $r_{gn} = P^T P_{gn}(g - u) + \mu_{gn}$
   d) $r_{en} = P^T P_{en}(e - u) + \mu_{en}$
3) Calculate the corresponding reconstruction errors:
   a) $d_{gp} = |r_{gp} - g|$
   b) $d_{ep} = |r_{ep} - e|$
   c) $d_{gn} = |r_{gp} - g|$
   d) $d_{en} = |r_{en} - e|$
4) Compute the total reconstruction error, given by
$$d_t = d_{gn} + d_{en} - d_{gp} - d_{ep} \quad (6)$$
5) Classify the image according to:
$$\text{class}(g) = \begin{cases} \text{Pedestrian}, & d_t >= 0 \\ \text{Non-pedestrian}, & d_t < 0 \end{cases} \quad (7)$$

When we analyze how this classifier works, it is easy to understand the role of the PCs generated by the positive samples. As the images contains the recurrent pattern of the target object (pedestrians, in this case), PCA is able to capture it and better reconstruct similar images, i.e., images that contain the object, generating smaller reconstruction errors. The role of negative PCs was more obscure, tough. Since the negative images contain just random objects and does not represent a specific concept, what pattern is there to PCA capture? We initially thought that the errors of reconstructions obtained from the negative PCs had a negligible importance and therefore it would be possible to eliminate them without impacting the classification accuracy. However, our experiments show that is not the case: it turns out that these errors have a major importance on classification.

We found that the value of the negative PCs is not on the pattern PCA captures, but instead it lays out on the pattern PCA does *not* capture, i.e., the target object pattern. In this way, the negative reconstruction errors tend to be high for images that contains the target object while for the same images the positive reconstruction errors tend to be small, contributing to a high total error, what classifies the image correctly. For images that does not contains the target object, it is exactly the opposite. Thus, both components of total error (from positive PCs and from negative PCs) work together for the classification and that is why it is not practicable to construct a classifier that uses only positive reconstruction errors.

However, this does not mean that all the four components have the same relevance to classification. In the proposed classifier, each component contributes equally to the total error, but our experiments show that this relation is not true. Some components have more importance than others and

therefore have to be adjusted accordingly. So, we propose the *weighted total reconstruction error*, given by

$$d_w = w_{gn} d_{gn} + w_{en} d_{en} - w_{gp} d_{gp} - w_{ep} d_{ep} \qquad (8)$$

where w_{gn}, w_{en}, w_{gp}, w_{ep} are weights that adjust the relevance of each error to the classification (more relevant errors must have larger weights). They can be set manually or be determined by some optimization algorithm in relation to the training set. We use a genetic algorithm in our experiments.

It is also worth noting that the classifier can be made more flexible if we introduce a threshold parameter and compare the classification score (i.e. the total error) agains it, instead of keep it always fixed to zero. In this way, the user can adjust the classifier specificity, depending on his particular application.

IV. Experiments

The classifier was tested with the pedestrian detection task, that is, given a unlabeled image, the classifier must determine if it depicts or not a pedestrian. As said before, we need a set of pedestrians images and a set of non-pedestrian images to train the classifier. This training phase consists of computing image edges and applying PCA to the four image sets separately, what produces four sets of PCs. They will be used to perform the reconstructions in the classification phase.

The pedestrian images were obtained from the MIT CBCL pedestrian database [5]. It contains 924 color images of pedestrians in frontal or rear views, under different scene conditions. These images were converted to grayscale and had a part of the background cropped on the border, remaining with 45×105 pixels. Doing this, we reduce the non-relevant variation that exists among the images, what allows a better classification.

The non-pedestrian images were extracted from 120 assorted pictures that did not contain pedestrians. Each one was chopped into slices of 45×105 pixels and 5.000 slices were randomly chosen to compose the negative dataset. This dataset is more than four times larger than the positive dataset and that is related with the representation capacity of each one. The object detection problem is naturally asymmetric [6], since the positive examples represents a specific type of object while the negative examples represent the "rest of the world". That is why we need much more negative examples than positive ones.

To train the classifier, we selected 75% of the pedestrian images (693 samples) and the remaining images (231 samples) were used for testing; from the non-pedestrian images, we selected 80% of the images (4.000 samples) for training the classifier and the remaining images (1.000 samples) was used for testing. This particular scheme was chosen in a empirical way; it yielded the best results among

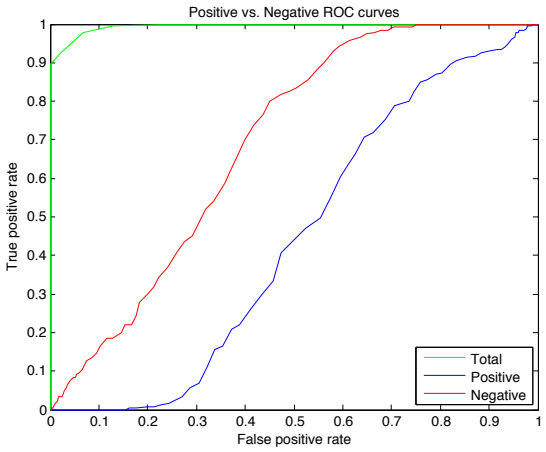

Figure 1. Comparison of positive and negative ROC curves. From left to right: the first curve (green) refers to a classifier that uses the total reconstruction error (equation 6) and has an AUC of 0.995; the second curve (red) refers to a classifier that uses only the reconstruction errors from negative PCs (d_{gn} and d_{en}) and has an AUC of 0.690; and the third curve (blue) refers to a classifier that uses only the errors from positive PCs (d_{gp} and d_{ep}) and has an AUC of 0.444. On the three classifiers, only the first 100 principal components are being used, that is, $k = 100$. Both positive and negative classifiers yields a poor result, far from the full classifier performance. It is surprising to see that the negative classifier even got a better result than the positive one.

the distributions we tested. The edge images are computed with x and y Sobel filters [7], which we combine to generate a single edge image. To implement the classifier and the auxiliary procedures, we used the MATLAB 2009b environment. The resulting code is available at [8].

We start our analysis by investigating the impact of reconstruction errors from the positive PCs and from the negative PCs to the classification. As discussed before, we initially questioned the role of negative PCs, since they are computed from completely uncorrelated images and do not represent a particular object. However, when we classify test samples using reconstruction errors only from the positive PCs or only from the negative PCs (but still using grayscale and edge images), it becomes clear that neither one can perform a satisfactory classification; both classifiers yields a very poor result. But when the errors are combined accordingly to the total reconstruction error expression, then we have an excellent classification. The receiver operating characteristic (ROC) curves of the three classifiers are shown in Figure 1 for $k = 100$. We get a similar pattern for all other values of k that were tested.

To understand why this happens, it is interesting to see how the reconstruction errors are distributed. In Figure 2, we plot the reconstruction error of each individual test sample in the three situations previously described: in the first plot only the errors from positive PCs are considered, in the second plot only errors from negative PCs are considered and in the third plot we used the total error. It is easy to see that the samples are mixed on the two first plots, indicating

that these criteria do not have discriminatory power to correctly separate the samples. Conversely, when the errors are combined, the distribution changes completely and we can notice that samples from different classes are in fact grouped, what makes possible the high classification accuracy we have gotten. Hence, the reconstruction errors from both positive and negative PCs contribute in a significant way to the classification and none of them can be removed.

On the other hand, we get very similar ROC curves when the classification considers the reconstruction errors from grayscale PCs and from edge PCs separately (now always using the errors from positive PCs and from negative PCs together). This in shown in Figure 3, which exhibit three ROC curves: one from the classifier that uses only grayscale errors, one from the classifier that uses only edge errors and one from the classifier that considers both (i.e. the total error), all using $k = 100$. Note that the ROC curves from the edge error classifier and from the total error classifier are nearly indistinguishable. In fact, their area under the curve (AUC) are practically equal.

Again, we plot the reconstruction error distributions, but now segregating errors from grayscale PCs and from edges PCs. These plots are shown in Figure 4, that also displays a plot of the combined errors distribution (i.e. grayscale and edge errors summed). At this time, we get a different scenario. All the three plots exhibit a clear boundary that separate samples from different classes. This indicate that there is a lot of redundancy between errors from grayscale PCs and from edge PCs and one of them can be safely removed from the classifier with just a minor impact on classification accuracy. Our results indicate that the errors from grayscale PCs generally yields a slightly worse result, as shown in the Table I. Therefore, it is possible to compose a classifier only with the edge errors and speed up the classification significantly (since two reconstructions will not have to be computed anymore). In general, such classifier drops the classification accuracy in less than 1% when compared with a classifier that uses the total error. Actually, the classifier based only on the edge errors sometimes performs even better in our experiments (when $k >= 200$), as also shown in Table I.

These findings also point out that, among all the four reconstruction errors, some are more relevant to the classification than others. However, on the total reconstruction error

Table I
AUCs FOR GRAYSCALE, EDGE AND TOTAL ERRORS CLASSIFIERS

k	Gray	Edge	Total
400	0.9611	0.9863	0.9744
300	0.9694	0.9877	0.9826
200	0.9768	0.9905	0.9877
100	0.9851	0.9944	0.9948
50	0.9905	0.9956	0.9972
25	0.9929	0.9913	0.9984

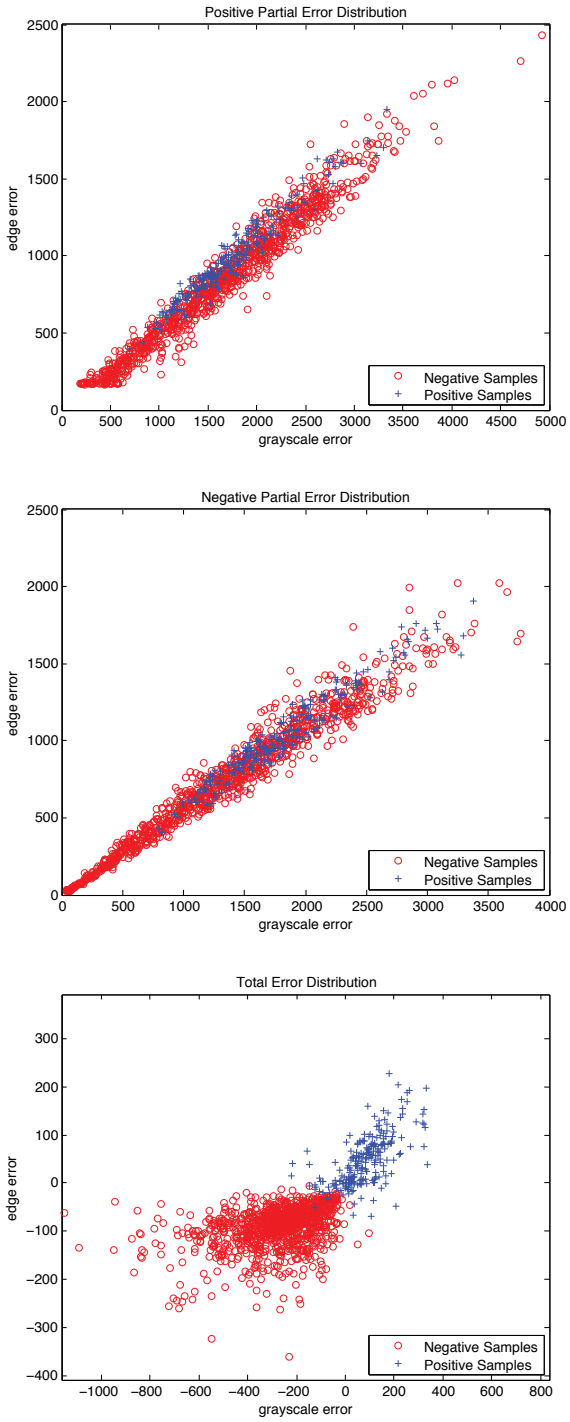

Figure 2. Comparison of reconstruction errors distributions. The first chart plots the errors of train samples when only the reconstructions from positive PCs are being considered. The second chart plots the errors only for the negative reconstructions. And the third chart plots the distribution of the total reconstruction error. Note that the errors are mixed on the first two charts; classify the samples based on them is a hard task. On the other hand, the third chart show well defined clusters of errors, what makes possible a good classification. This explains the curves of Figure 1.

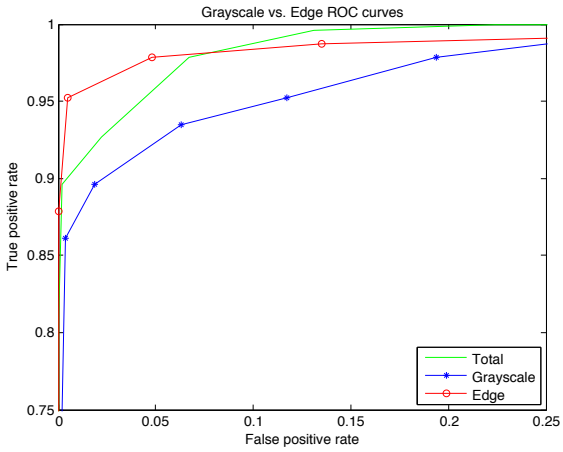

Figure 3. Comparison of grayscale and edge ROC curves. The green curve refers to a classifier that uses the total reconstruction error (equation 6) and has an AUC of 0.995; the red curve refers to a classifier that uses only the reconstruction errors from edge PCs (d_{ep} and d_{en}) and has an AUC of 0.994; and the blue curve uses only the errors from grayscale PCs (d_{gp} and d_{gn}) and has an AUC of 0.985. On the three classifiers, only the first 100 principal components are being used, that is, $k = 100$. Note that the scale is different of that from Figure 1; we approximated the upper left corner to better show the difference between the curves, that are very similar.

(equation 6) each error contributes equally. So, we use the weighted total reconstruction error (equation 8) to improve the classification accuracy. The problem with using weights is how to choose them. We can try some values and make adjusts manually, test after test. But we can also employ an optimization algorithm that looks for the values in an automated way. In this work, we use a genetic algorithm **??**. It initially chooses a set of random values and uses operations as crossover, mutation and selection to improve them during many iterations. These operations are inspired by evolutionary biology; hence the name genetic algorithms. The values are adjusted in relation to a fitness function. In our case, this functions aims to correctly classify the largest number of training samples using a given set of weights. The function can be fine-tuned in order to give more importance in finding more pedestrians despite the false detections or to minimize the number of non-pedestrians classified as pedestrians.

As this algorithm performs a local optimization and therefore is not optimum, we have run it five times for each value of k. We then have chosen the set of weights that gets the best classification rates among these five. Finally, we classify the test samples considering the weighted total reconstruction error. The produced ROC curves always have a larger AUC than the corresponding versions without weights for all values of k we tested. This is shown in Figure 5. The difference is more pronounced for large values of k and becomes small when it diminishes. Still, the AUC of the first classifier is always larger, what indicates that the weighted error is a better classification criteria than the non-weighted

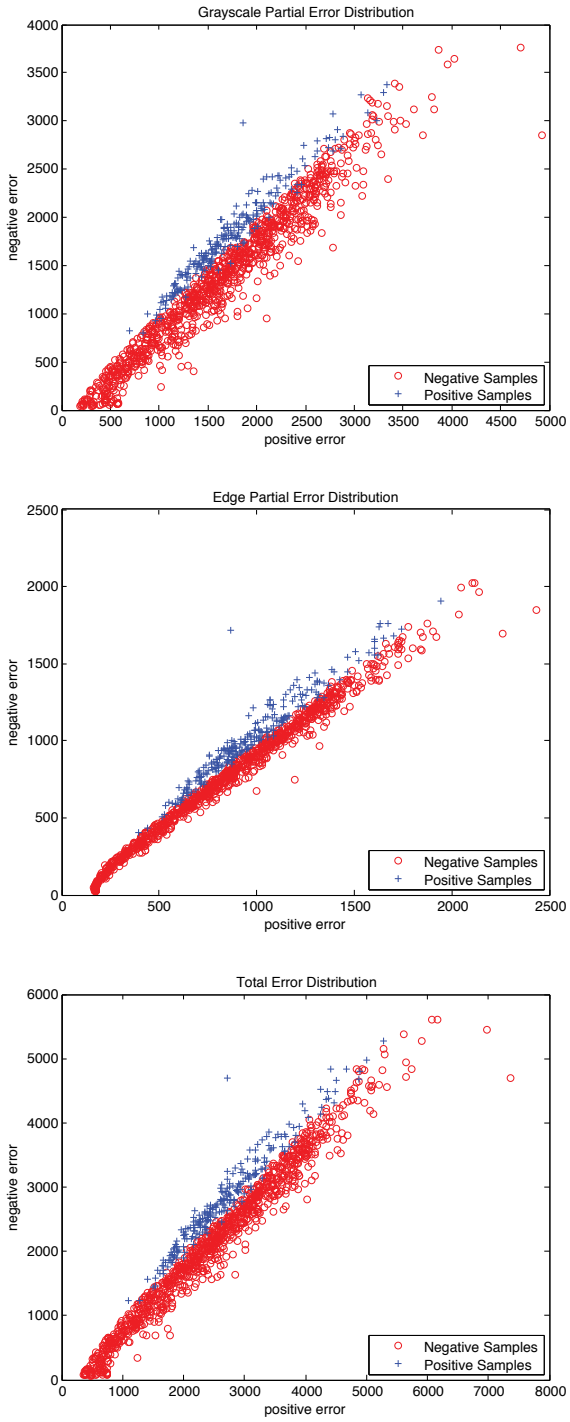

Figure 4. Comparison of reconstruction errors distributions. The first chart plots the errors of positive and negative train samples when only the reconstructions from grayscale PCs are being considered. The second chart plots the errors only for the edge reconstructions. And the third chart plots the distribution of the total reconstruction error. Note that the three charts show a clear boundary between errors from positive and from negative samples.

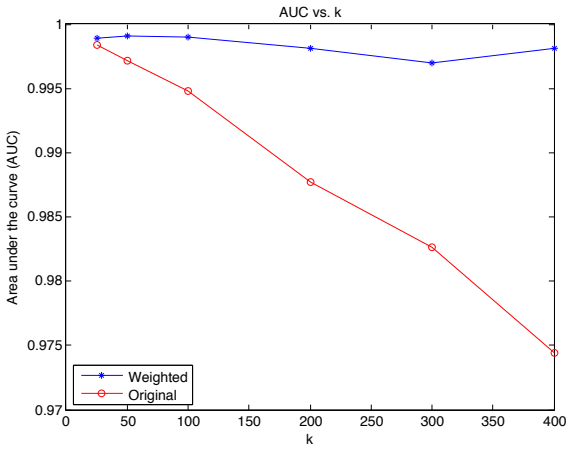

Figure 5. Comparison of weighted and original classifiers AUCs for different values of k. The AUCs of the weighted classifier are more stable when k varies.

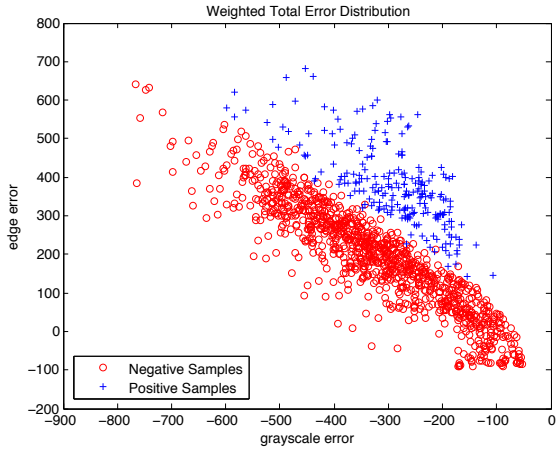

Figure 6. Distribution of the weighted total reconstruction errors for positive and for negative training samples, for $k = 100$. Compare it with the third chart of Figure 2. Here the boundary between errors from positive and from negative samples is much clearer.

one. As usual, we plotted the distribution of the weighted total reconstruction error, that can be seen of Figure 6. This time, we get a clear boundary between errors from samples of different classes.

V. Conclusion

In this work we analyzed a classifier that uses PCA image reconstruction for pedestrian detection. It uses four sets of principal components (PCs): two originated from grayscale and edge images that contains pedestrians, respectively, and two originated from grayscale and edge images that did not contain pedestrians, respectively. To classify an image, it uses these sets of PCs to perform four image reconstructions and compares the respective reconstruction errors to determine if the given image depicts or not a pedestrian.

To understand how the individual errors contribute to the overall performance, we compared two classifiers: one that uses only the reconstructions from the positive PCs and other that uses only the reconstructions from the negative PCs. In our experiments, both performed poorly when compared to the full classifier, indicating that positives and negatives reconstructions are indeed relevant to the classification. The crucial role of the reconstructions from negative PCs is particularly surprising, since non-pedestrian samples are not related and do not contain a specific pattern. Therefore, we conclude that the importance of the negative reconstructions is not on what they characterize, but instead on what they do not characterize. In this way, the negative reconstruction errors contribute to raise the total reconstruction error when the image does not contains a pedestrian, what makes possible a better classification.

We also investigated the relevance of grayscale and edge errors. In other experiment we compared the accuracy of a classifier that uses only grayscale reconstructions and other classifier that uses only edge reconstructions. This time, they performed very well, with a performance close to the that of the total error classifier. This suggest that there is a lot of redundancy between grayscale and edge information. The classifier based only on edge images got better results, indicating that edge information is more relevant to this classifier than grayscale information. In fact, the edge classifier performed better than the total error classifier itself for some number of PCs and, when the results were worse, the difference between their accuracy was less than 1%. For this reason, it is possible to speed up the classification time in a significant way by using only edge information and keep the classification accuracy virtually the same.

Since our experiments show that each error has a different relevance for classification, we proposed a classifier that uses a weighted version of the total reconstruction error. The problem with weights, though, is how to find the best values. To get an approximation in an automated way, we used a genetic algorithm with a fitness function that aims the best classification rates for the training samples. This classified performed better than the original version in our experiments for all numbers of PCs we tested.

Future work include testing this classifier with other pedestrian databases and verifying if the reconstruction approach is suitable to other detection problems in computer vision, such as face detection. We intend to develop the classifier into a complete detection system, capable of locate objects in large pictures. It is also possible to employ variations of PCA for reconstruction, like the more robust 2D-PCA [9].

References

[1] L. Malagón-Borja and O. Fuentes, "Object detection using image reconstruction with PCA," *Image and Vision Computing*, vol. 27, no. 1-2, pp. 2–9, 2009.

[2] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America A*, vol. 4, no. 3, pp. 519–524, 1987.

[3] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[4] J. Shlens, "A Tutorial on Principal Component Analysis," *Systems Neurobiology Laboratory, Salk Insitute for Biological Studies*, 2009.

[5] Center for Biological and Computional Learning, "CBCL Pedestrian Database #1," 2000. [Online]. Available: http://cbcl.mit.edu/software-datasets/PedestrianData.html

[6] X. Jiang, "Asymmetric principal component and discriminant analyses for pattern classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931–937, 2009.

[7] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[8] G. Carvalho and L. Moraes. Implementation Code for the PCA Reconstruction Classifier. [Online]. Available: http://github.com/lailsonbm/pca_reconstruction

[9] M. Li and B. Yuan, "2D-LDA: A statistical linear discriminant analysis for image matrix," *Pattern Recognition Letters*, vol. 26, no. 5, pp. 527–532, 2005.