

Computational complexity reduction in PCA-based face recognition

Włodzimierz M. Baranski, Andrzej Wytyczak-Partyka, Tomasz Walkowiak

Institute of Computer Engineering, Control and Robotics,
Wrocław University of Technology,
ul. Janiszewskiego 11/17, 50-372 Wrocław, Poland
Phone: +48-71-3202969 Fax: +48-71-3212677,
Marek.Baranski@pwr.wroc.pl

Abstract: *Face recognition is one of the most common recognition tasks. Algorithms have improved significantly over the last decade and nowadays give a satisfying recognition rate. In this work we will try to look at one of the classical face recognition algorithms [2] from a computational complexity point of view and try to suggest a way to reduce the complexity, so the algorithm could be successfully implemented in environments with seriously limited resources (mobile devices, embedded systems).*

Keywords: *pattern recognition, principal component analysis, computational complexity*

1 Introduction

Face recognition is one of the key aspects of human communication. We haven't, so far, developed any other way of recognizing individuals remotely. While it's certainty in human-human recognition is very high, face recognition does not seem to be the best way of identification for computers. Electronic devices generally do better with patterns like the iris or the fingerprints. However, in the pattern-recognition world, face recognition has always been an important topic and a lot of work has been done to improve the algorithms for solving that problem. The first work in this field is dated back in 1888 [1]. The author proposed a method, where the classification was based on deviations from the norm of curvature of the face profiles. One of the nowadays classical algorithms was presented in 1991 in [2] and based on the previous work in [3]. The math behind this algorithm is the Principal Component Analysis. Since then other methods were studied and better solutions have been found. Currently methods based on Support Vector Machines [12] [13] (a linear classifier that has been presented firstly in the 1960's) are gaining momentum, after they have proven to outperform neural networks in many solutions. We focus in this paper on a computational complexity problem and low resolution pictures so the algorithm could be successfully implemented in environments with seriously limited resources like mobile devices or embedded systems.

In the following paper we will briefly describe the original algorithm we base our work upon (Section 2). Later, the thesis will be presented. In Section 3 we will give the initial algorithm used to prove our thesis, the results of the tests, and we will suggest further algorithm's modification and show the recognition rate comparison with the others. Section 4 will compare the time results of the algorithms. The paper will conclude in Section 5 with our observing and future work suggestions.

2 Eigenfaces algorithm

2.1 Original algorithm

The original algorithm, we compared our results with, was an implementation of the well known eigenfaces algorithm [2]. The main idea behind this is the use of Principal Component Analysis for dimension reduction in the dataset. From PCA we get a set of eigenvectors (eigenfaces) that contribute to the creation of particular face images. Finally, in the learning process, each image from the learned-faces dataset is represented as a point, Ω , in the face-space, where the point's coordinates ω_i (in R^k , where k is the number of eigenvectors used) are the weights of each eigenvector's contribution in creation of the image:

$$\Omega = (\omega_1, \dots, \omega_k). \quad (1)$$

The recognition process consists of the same operation, which is called [2] – projection onto the face-space. The new point's coordinates are then compared to the known points' coordinates and the process concludes by finding the nearest neighbor of the tested point. This is, in the initial version, done by brute force searching through the dataset.

2.2 Complexity analysis

Computational complexity analysis of the eigenfaces algorithm shows, that the classification process might be one of the bottlenecks. Normally after projecting the face image onto the face-space, the next step is to search for it's nearest neighbor, by comparing the distances between the tested and the known points. A problem of complexity $O(n)$ has to be solved. We will propose a method for optimizing that stage of the algorithm. Using basic Euclidean geometry we will divide the points in face-space in to subsets by Binary Space Partitioning and place them in a Binary Search Tree (BST). Searching in a BST has a complexity of $O(n)$ only in the worst-case scenario, when all the nodes are linked in a list. The optimistic version (a balanced tree) is $O(\log(n))$ complex [10] [11]. Assuming an average case - the complexity of our program should lie somewhere between $O(n)$ and $O(\log(n))$, depending on the structure of the tree.

Our thesis is that the algorithm with classification based upon a search in the BST will have better time performance than the brute force search algorithm, while the recognition rate will remain over 50%.

3 Proposed modification and experiments

We placed the known points from the face space, Ω_j , in a Binary Search Tree, in a following manner. We took two most distant points – Ω_1 and Ω_2 from the known dataset and made them roots of two different binary trees. Then, from the remaining points, we picked those, which are closer to Ω_1 than to Ω_2 and assigned them to the Ω_1 tree. The remaining points were assigned to the Ω_2 tree. From those points, we again took two most distant ones, and repeated the procedure recurrently, thus creating a Binary Search Tree.

$$\forall j \in (3, \dots, n) \text{ dst}(\Omega_j, \Omega_1) < \text{dst}(\Omega_j, \Omega_2) \quad (2)$$

In order to test our theory we first ran a comparative test, in which we were feeding the algorithm with the same pictures it was trained on. As expected, the result was a 100% recognition of the pictures, both in brute force based and BST-based algorithms. This result was giving a hope for good results in the real test. In the first attempt, we tested the recognition rate in the ORL face database. The result was disappointing, successful recognition occurred in only ca 50% of cases. In the second attempt, we ran the test on our custom-made image database containing images of 30 subjects - Faculty of Electronics students at the Wroclaw University of Technology, 2 images per subject. We recorded a significant performance increase, on average a 60% recognition rate was achieved. The analysis of the distances between points in the face-space led us to a conclusion, that it is a common case for the distances between the nodes of the BST to be of the same order that the distance between the tested point and it's known equivalent. That led us to the following improvement in the BST building algorithm. This version will be called the BST+ algorithm, from now on. After picking two most distant points from the we modify the decision criteria of assigning a certain point to one of the Ω_1 or Ω_2 subtrees. We calculate both distances and if the difference between them exceeds some threshold (marked as A) we continue with the previous algorithm. Otherwise we place that point in both subtrees.

$$\forall j \in (3, \dots, n) \text{ dst}(\Omega_j, \Omega_1) < \text{dst}(\Omega_j, \Omega_2) \vee \text{dst}(\Omega_j, \Omega_1) - \text{dst}(\Omega_j, \Omega_2) < A \quad (3)$$

This modification gave an immediate boost of recognition rate by 10%.

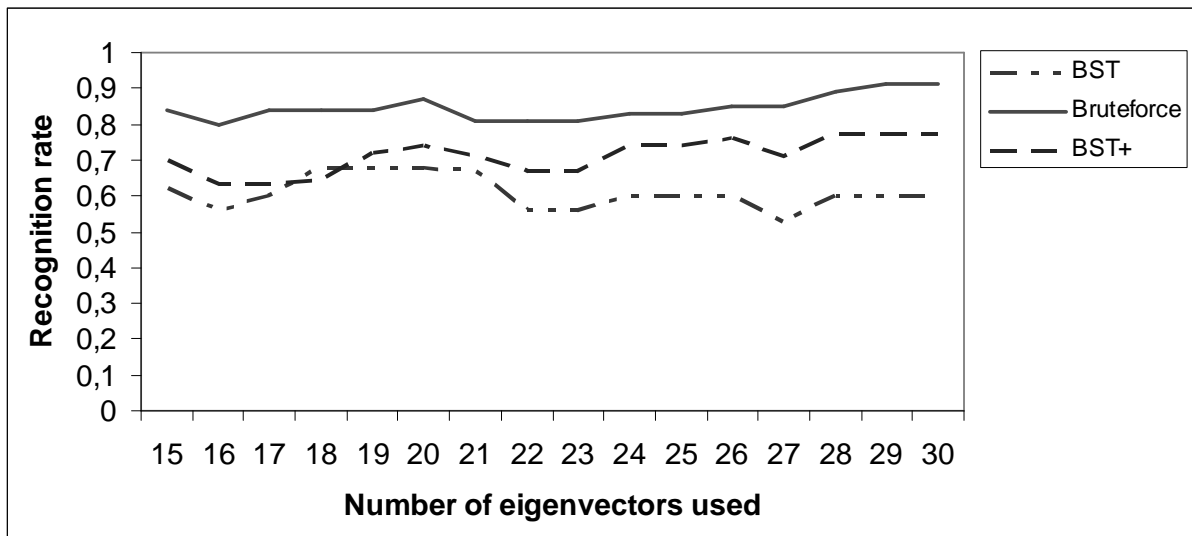


Fig. 1. Recognition rate comparison for the Eigenface algorithm and it's BST modifications. Top - bruteforce search, middle - BST+ search, bottom – initial BST version

The tests were performed using the bootstrapping method. We tested the algorithms with two datasets - the ORL database, and our custom-made database. In both cases, we used exactly one of the subject's images for algorithm training, and one more for recognition. The ORL database had 40 subjects, each photographed in 10 different versions, grayscale 112x92 px. Our database had 30 subjects, 2 versions per subject, grayscale, 250x300 px resolution.

4 Time performance

Although the BST algorithms didn't perform better in terms of recognition rate, we were sure they would outperform the original one in terms of time. In fact, in our tests, we recorded the time of both versions of the BST algorithm to be almost two times faster than the original 1. The BST+ version is slightly slower than the initial BST, due to increased size of the tree, but still gives the same order of time reduction.

The cost of building the tree was on average 0.7 s. The most time-consuming procedure was the initial vector generation, during the training - it took almost 50% of the whole execution time. The tests were performed on an AMD Athlon processor with a 1485 MHz Clock, under Matlab.

No of images	Bruteforce [s]	BST [s]	BST+ [s]
5	0.256	0.081	0.083
10	0.488	0.162	0.231
15	0.722	0.290	0.240
20	0.901	0.389	0.440
25	0.921	0.483	0.612

Tab. 1. Elapsed time for different versions of the recognition algorithm

5 Conclusions

We believe that using a BST for classification of face images is a good way for time-optimizing the process. Despite it's higher error rate it gives great savings in processing time of the device that will perform it. It seems obvious, that the increased recognition error is correlated to the dataset. In the ORL dataset, which is low-resolution the error proved to be significantly higher than in the other - higher-res dataset. Higher resolution of the images is followed by a higher resolution in the face-space, which makes classification easier for the algorithm.

Analyzing the BST for the ORL database, we noticed that the distance between the tested image and it's known-equivalent was usually of the same order as the distances between the nodes of the BST. That led us to an improvement in the BST algorithm and a 10% recognition rate increase. It is highly possible that further investigation in the BST performance could lead to better recognition rates allowing real time implementation in mobile devices.

The complexity of a search in a BST is $O(\log(n))$ only in the best case, which is when the tree is balanced. Due to nature of the dataset achieving a balance in the tree is troublesome and might not even make sense if we are able to accept the results given by the BST. The promising results of our experiment suggest that examination of other tree structures like Black and Red trees or AVL trees [11], which give a lower upper bound on the complexity, is purposeful.

References

- [1] F. Galton, Personal identification and description, In Nature, pp. 173- 177, June 21, 1888
- [2] M. Turk, A. Pentland, Eigenfaces for recognition, Journal of Cognitive Neuroscience, 3 (1), 1991a. URL: <http://www.cs.ucsb.edu/~mturk/Papers/jcn.pdf>
- [3] M. Kirby and L. Sirovich, Application of the Karhunen- Love procedure for the characterisation of human faces IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, pp. 831-835, Dec. 1990
- [4] J. Zhang, Y. Yan, M. Lades, Face Recognition: Eigenface, Elastic Matching, and Neural Nets, Proceedings of the IEEE, Vol. 85, No. 9, September 1997
- [5] V. Dinh, M. Nhat, S. Lee, Two-dimensional Weighted PCA algorithm for Face Recognition, The Fourth International Conference on Computer Recognition Systems (CORES'05), Poland, May 22-25, 2005
- [6] M. H. Yang, Kernel Eigenfaces vs. Kernel Fisherfaces: Face Recognition Using Kernel Methods, Proc. Fifth IEEE Int'l Conf. Automatic Face and Gesture Recognition (RGR'02), pp. 215-220, May 2002
- [7] W. S. Yambor, B. A. Draper, J. R. Beveridge, Analyzing PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures, Empirical Evaluation Methods in Computer Vision, H. Christensen and J. Phillips (eds.), World Scientific Press, Singapore, 2002
- [8] Ch. Ding, X. He, K-means Clustering via Principal Component Analysis, Proc. of Int'l Conf. Machine Learning (ICML 2004), pp 225-232, 2004

- [9] B. Heiseley, P. Hoz, T. Poggio, Face Recognition with Support Vector Machines: Global versus component-based Approach, *Computer Vision and Image Understanding*, Vol. 91, No. 1/2, 6-21, 2003
- [10] D. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Third Edition. Addison-Wesley, 1997
- [11] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001
- [12] G. Guo, S.Z. Li, and K. Chan, Face recognition by support vector machines, In *proc. IEEE International Conference on Automatic Face and Gesture Recognition*, 2000
- [13] P. J. Phillips, Support vector machines applied to face recognition, *Processing system* 11, 1999



Fig. 2. Images from the ORL database, original size. Top row – images used for recognition, bottom row – training images.

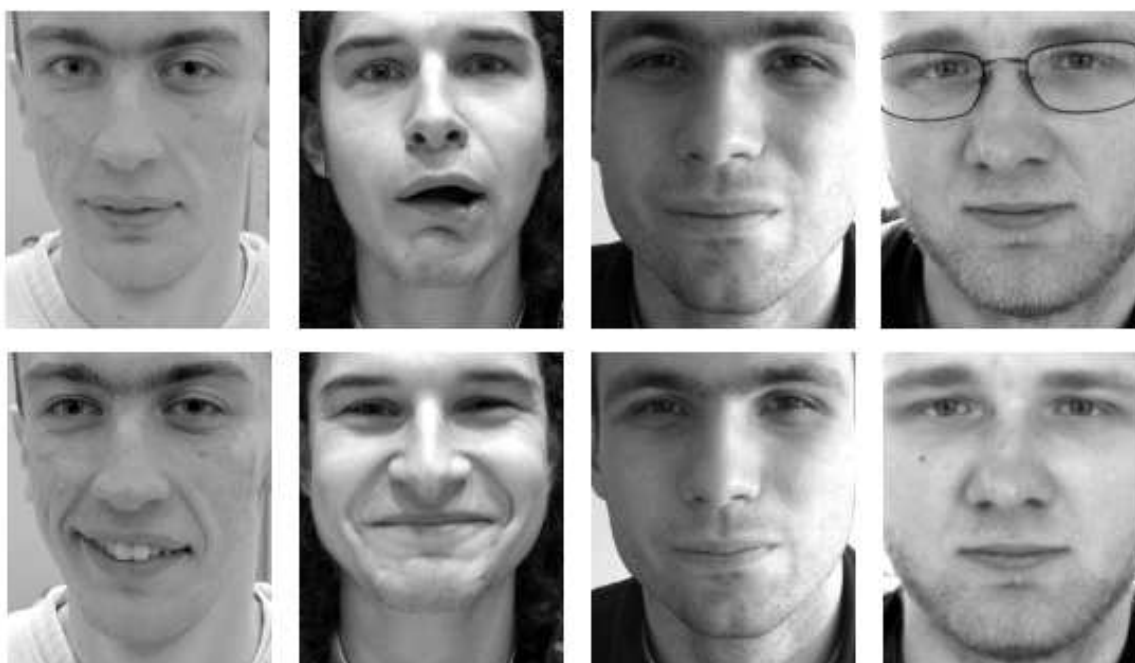


Fig. 3. Images from authors database, scaled to 50% original size. Top row – images used for recognition, bottom row – training images.