# Flexible multi-classifier architecture for face recognition systems

*Alexandre Lemieux* and *Marc Parizeau*

Laboratoire de vision et systèmes numériques (LVSN),
Département de génie électrique et de génie informatique,
Université Laval, Ste-Foy (Qc), Canada, G1K 7P4.

## Abstract

*This paper presents face recognition results obtained using a multi-classifier system (MCS) with Borda count voting. Experiments were conducted on complete sections of the FERET face database with 4 different algorithms: embedded HMM, DCT, EigenFaces and EigenObjects. Particular classifier ensembles yielded almost 6% of improvement over the individual techniques. In order to facilitate experiments on classifier combinations and decision rules comparison, a flexible MCS software architecture based on object oriented principles is also presented. It allows runtime modifications to the algorithms employed and dynamical selection of classifiers. This architecture can be applied to any pattern recognition problem.*

**Keywords:** EigenFaces, HMM, Face Recognition, Multi-classifier, DCT.

## 1 Introduction

While face recognition algorithms become more and more sophisticated, robustness still remains an important issue in real life systems. Lighting, pose and occultations represent annoying effects that alter identification performance. Specific solutions to these problems have been proposed in the literature [3, 18, 1]. However, as each recognition technique shows strengths and drawbacks in different situations, a multi-classifier system (MCS) can use this rich information to increase the overall recognition rate. This kind of approach have been used in many areas of research [10, 2, 19, 5] and yielded remarkable improvements.

However, because of the various foundations of each technique, some difficulties arise in fusion mode. In fact, a certain uniformity between each method implementation is required to produce comparable results and to facilitates operations. To overcome these problems, a software architecture based on object-oriented principles is proposed. Gaining advantages from inheritance and polymorphism, a recognition engine is built to use any techniques, as long as their implementations respect the parent class interface. Moreover, a unique shared human database, containing all the informations related to the individuals to recognize, is shared among the different modules.

The proposed software architecture facilitates experimentations on classifier ensembles. Experiments are conducted on the FERET face database [15] using 4 algorithms, namely *EigenFaces* [16], *EigenObjects* [14], DCT [9] and HMM [13], which have not been combined together (to the authors knowledge) and are particularly interesting due to their good properties and performance. Finally, the FERET database, that contains over fourteen thousands images of more than twelve hundredth individuals, represents an appropriate resource for comparing face recognition algorithms.

This paper is organized as follows. Section 2 summarizes the related work on multi-classifier systems. Section 3 describes the proposed software architecture. A brief description of the face recognition algorithms is given in Section 4. Experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

## 2 Multi-classifier

Classifier ensembles have been shown very useful for improving performance in numerous problems ranging from handwriting character recognition [10] to speech recognition [2]. They also have been used efficiently for face recognition purposes [19, 5].

Parallel classifiers and hierarchical/multistage classifier represent the two major categories of MCS. While the multistage architecture tries to stabilize and/or refine decisions, the parallel configuration merges the classifiers output with a decision function (rule). The choice of the merging method is crucial and directly affects system performance. Several techniques based on votes [12, 5] can

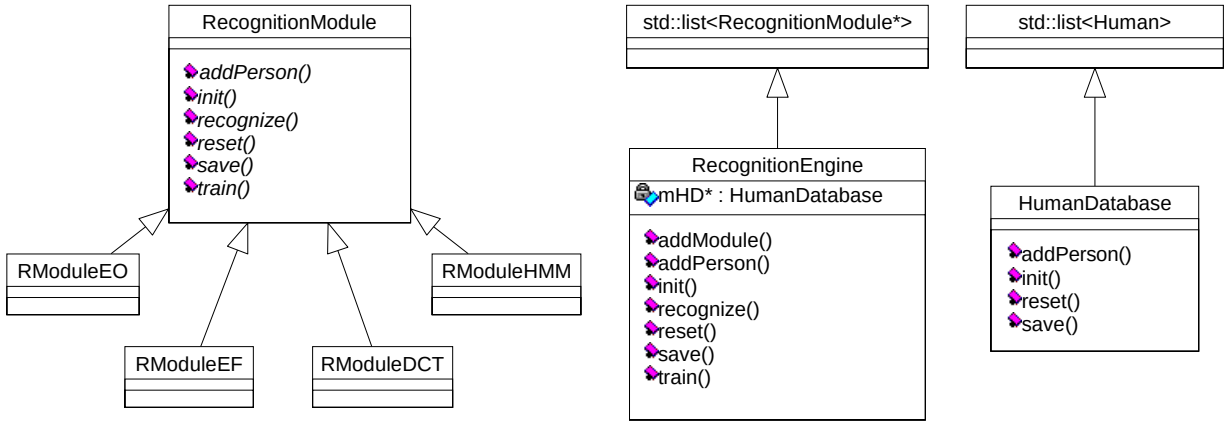


Figure 1: Multi-classifier software architecture.

be used but other methods like SVM or neural networks can also be employed to combine the classifiers raw output [19]. Other methods can be used to increase the classifiers performance like bagging [4] and boosting [8].

In this paper, the MCS architecture and experiments are implemented using parallel classifiers exclusively.

One limitation of MCS is that in some circumstances too many classifiers can yield misclassifications where a single one could have succeeded. Thus, dynamic selection of classifiers (DSC) tries to choose the best classifier for a specific test prototype [7]. An implementation of this kind of feature requires some specific software capabilities. In fact, design patterns [6] can be used to solve that problem and will be discussed in Section 3. These characteristics ensure that the system will be able to deal with any recognition module, whenever method switching occurs.

# 3 MCS software architecture

## 3.1 Implementation

As previously mentioned, the proposed software architecture aims at providing system flexibility, dynamic configuration, lower resource requirements and easier management of various recognition modules.

To achieve these goals, three components were designed to be assembled dynamically at runtime. The recognition engine, as the main component, is composed of several modules implementing different recognition algorithms. It also contains an important decision function used to generate identification results. Finally, all the information related to the known individuals are gathered in a database linked with the recognition engine. Figure 1 illustrates the system architecture with the dependance between each module.

**Recognition modules** The recognition modules must have a specific and standard interface known by the main engine and independent of the algorithm's implementation. This is accomplished by using the behavioral strategy pattern [6], based on abstraction. In fact, all classes are derived from the *RecognitionModule* abstract class that contains pure virtual functions. This mechanism ensure that all the functions will be implemented and available to the engine at any given time. The main interface consists of the *addPerson*, *init*, *train*, *reset*, *save* and *recognize* methods that can be commanded to the module.

**Recognition engine** The *RecognitionEngine* is the most important part of the system. First, it is derived from a list of pointers to *RecognitionModule* that are added (or removed) dynamically at runtime via an *addModule* method. The engine does not need to know the origin or type of each module, it only needs to know their basic functions (that information is provided by the abstract class). As the *RecognitionModule*, the *RecognitionEngine* possess some functions needed to interact with the different methods. It basically consist of the same list as in the *RecognitionModule*, except that for each one, the engine sequentially calls the corresponding function of each submodule. For example, if an individual has to be identified, the incoming image is sent to each module and a simple call to *RecognitionEngine::recognize* will produce a ranked list of class ids according to the decision method selected and the different recognition algorithms used.

**Human database** In a surveillance system, the number of individuals to recognize can reach high numbers,

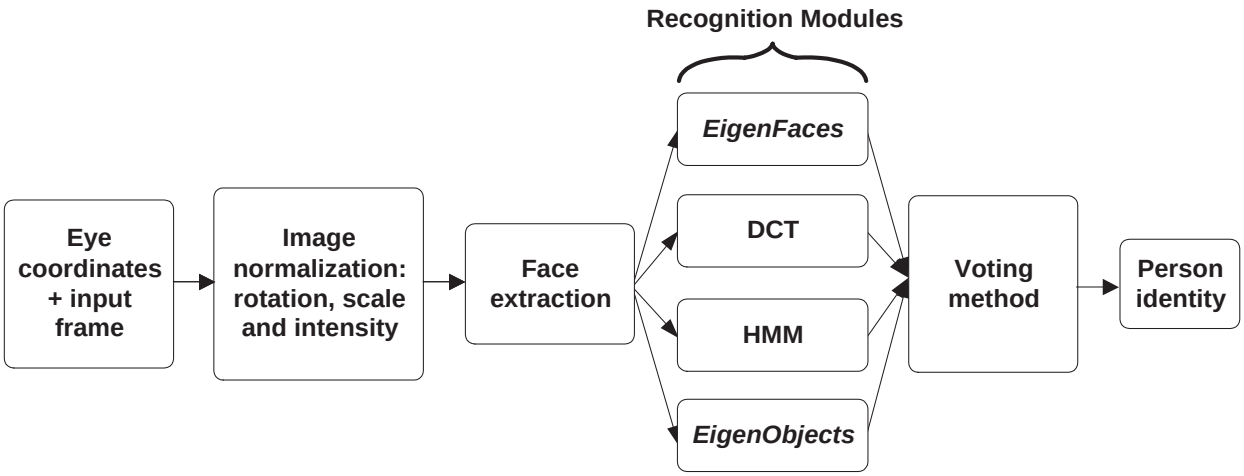


Figure 2: Face recognition experiments data flow diagram.

ranging from hundreds to several thousands. For each person, various informations need to be stored, like several images (original and normalized), id and specific informations. To reduce the impact of the large amount of resources required, the database must be shared between all the algorithms. For this reason, the *HumanDatabase* is initialized once and linked to the *RecognitionEngine*, thus becoming accessible to each module.

### 3.2 Strengths

The proposed MCS software architecture can provide many benefits. First, it enable intuitive and easy management of the modules at runtime, a feature particularly interesting for dynamic selection of classifiers applications. Moreover, because it is assembled dynamically, classifier combination experiments and decision function testing can be easily accomplished.

A distributed version of this architecture could also be implemented with small changes. In fact, the basic functions could be modified to include communication between the different modules and the main engine, thus enabling parallelism. This particular implementation would be well suited for managing recognition tasks with large databases in real-time.

## 4 Face recognition experiments

In order to experiment our MCS architecture with different face recognition algorithms, a simple application was developed in conjunction with a local face database. A screenshot of that system is illustrated in Figure 8. Image acquisition is done with a web-cam while preprocessing steps like background subtraction and skin color detection follow. The face detection task is carried out by a simple template matching technique. The overall data flow diagram is illustrated in Figure 2.

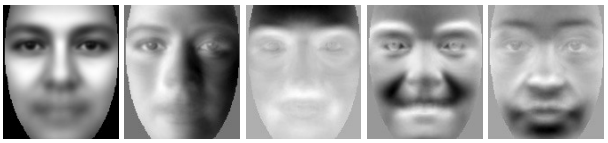

Figure 3: Average image + first 4 *EigenFaces*.

### 4.1 Techniques

Among the several methods that have been applied to face recognition [18], four were selected for the experiments. The *EigenFaces*, *EigenObjects*, DCT and HMM techniques were chosen and will be briefly described in the next paragraphs.

***EigenFaces* (EF)** The first method is the widely known *EigenFaces* technique [16] that followed previous work on Karhunen-Loève transform for face characterization [11]. It is based on dimensionality reduction via a principal component analysis of the faces contained in the learning set. Unknown faces are identified by first subtracting the mean face image from the input image and, second, by projecting that resulting face on the $N$ first *EigenFaces* (or eigenvectors). Figure 3 illustrates examples of an average face and the first 4 *EigenFaces*.

***EigenObjects* (EO)** The *EigenFaces* method can be further restricted to specific face features like eyes and nose (samples are illustrated in Figure 4). It has been shown [14] that these specific *EigenObjects* can improve

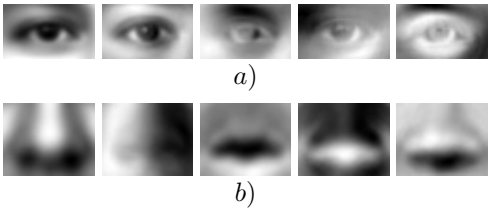

Figure 4: Average image and first 4 *EigenObjects* for a) the left eye and b) the nose.

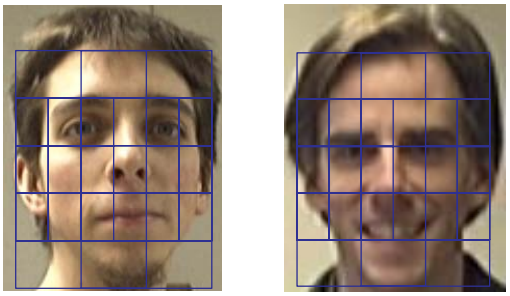

Figure 5: HMM initial segmentation.

face recognition over the original method. In fact, these mostly invariant face features (if not in occultation) are less affected by face expressions than the global representation of the face. However, there is more differences among faces than between eyes, thus confusion and misclassification errors are more likely to occur.

**Discrete Cosine Transform (DCT)** Recent work [9] on the use of DCT for face recognition has reported results similar to those obtained with the *EigenFaces* method. In fact, the two techniques share a closely related mathematical background. However, DCT seems less influenced by illumination variations than EF. Moreover, it was also shown to train faster and does not require complete retraining when new persons are added. In fact, only the corresponding coefficients are appended to the known representations.

**Hidden Markov Model (HMM)** The last method used in the experiments is the embedded Hidden Markov Model (HMM) [13]. This approach varies from the original technique by the fact that each state of an overall one-dimensional HMM can also be an HMM. Thus, the face is divided in horizontal regions (*super states*) containing sub regions (*embedded states*). The initial segmentation used in the experiments is illustrated in Figure 5. Observation vectors consist of DCT coefficients computed in the sub regions. The main drawback of this technique resides in the high computational cost needed for the training and testing phases.

### 4.2 Decision function

Among the several existing voting methods [17] that can be applied for the decision process, the Borda count [10] was shown to be relatively efficient. Each classifier first produces a ranked list of guesses in order of preference. Then, a mean rank is computed for each class and reordered in a list. The one with the highest rank (or smallest mean) wins.

## 5 Experimental results

### 5.1 Motivations

The experiments were conducted in order to verify important design decisions taken for an ongoing project. First, the multi-classifier advantages had to be verified over a local database and the FERET as well. Then, the choice of the recognition algorithms for classifier ensemble needed to be validated. In fact, an interesting point about the four selected techniques resides in their mutual characteristics. The *EigenFaces* and *EigenObjects* methods obviously shares the principal component analysis, but at different levels. The DCT and HMM techniques have also some common characteristics due to the DCT observations exploited by the HMM method. Finally, the PCA and DCT also share a common mathematical background [9], thus establishing a link between all four methods. The aim of our experiments is thus to determine the extent to which combining these methods can increase recognition performance.

### 5.2 FERET database

The FERET database was created in order to compare different face recognition algorithms [15]. It contains 14,126 images of 1199 persons with various expressions, illuminations, poses, etc. Predefined galleries (training set) and probe set (testing set) are provided with the database.

The results reported in this paper are based on two completely independent sets: the *FA* gallery (1,196 images) and the *FB* probe set (1,195 images). The gallery set contains one image per person for a total of 1,196 persons. The images contained in the probe set correspond to the same persons, but with different facial expressions.

### 5.3 Experimental protocol

As described earlier, 4 algorithms were integrated into the multi-classifier architecture. Some of the parameters used for each method are summarized in Table 1. All the images of the training and testing sets were normalized

| Algorithms | Parameters |
|---|---|
| ***EigenFaces*** | First 200 eigenvectors |
| ***EigenObjects*** | First 25 eigenvectors per feature (eyes+nose); total of 75 |
| **DCT** | 192 coefficients |
| **HMM** | 5 super states (3 6 6 6 3) embedded states |

Table 1: Face recognition algorithms parameters.

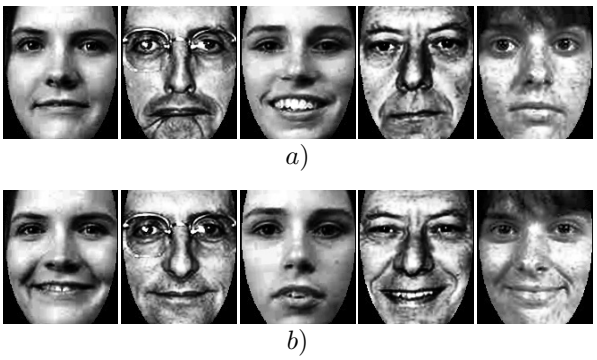

Figure 6: Sample normalized images from the FERET database. a) Training images and b) Testing images.

according to the following procedure (similar to the one used in the FERET baseline tests [15]):

1. Eyes centers were extracted and verified from the FERET files;
2. Images were rotated so that both eyes are perfectly aligned horizontally;
3. Images were scaled to set the distance between the eyes at 70 pixels exactly;
4. Images were cropped to size 130x150 (with the face centered and the eyes laying on row 45);
5. An oval mask was applied to eliminate hair and background;
6. The resulting image histogram is equalized.

Figure 6 illustrates some FERET images after normalization. The images illustrated in 6a) are taken from the training set while those in 6b) belongs to the probe set. It should be noted here that face expressions between training and testing sets are very different, thus making the recognition task somewhat difficult.

A modified Borda count metric was then used as the decision function. As describe earlier, the original method computes a mean rank for each individual. However, in the case of a classification error, a particular individual can receive a very low rank from only one recognition algorithm, thus being eliminated from the top guesses. For example, if an individual obtains ranks of 1, 2 and 550, the resulting mean rank drops to $553/3 = 184$ approximately. To overcome that particular problem, a maximum rank limit is established (10) to compensate for poor classifier performance. Thus, the mean rank for the previous example would be $1+2+10/3 = 4.3$ instead of 184.

Finally, all the algorithms were loaded dynamically at runtime and trained using the *FA* gallery set only. The classifiers were then tested on the complete *FB* probe set and combined with the Borda count voting method. It is important to note that the ranked lists of guesses provided by the EF, DCT and EO methods are generated by a nearest-neighbor algorithm using $L_1$ distance metric (city-block).

## 5.4 Experimental results

Figure 7 illustrates the different results obtained with the 4 individual algorithms and the 11 possible classifier ensembles. For each combination, a cumulative score is given for the three first positions (denoted *Top-1* to *Top-3*). These values represent the percentage of good identifications realized up to a certain level *N* of tolerance (*Top-N*).

While the scores are quite close between the individual classifiers (except for the *EigenObjects* method), the HMM clearly helps in the combination groups. In fact, the best 2-classifiers combinations contains an HMM and outperforms all others ensemble in the same group. It is also interesting to note that the *EF+HMM* still outperforms the 3-classifiers combination *EF+EO+DCT*. This clearly demonstrates the different forces between the classifiers; in fact, HMM seems to misclassify different individuals than the other techniques, thus gaining a lot from the combinations.

However, the fusion of the EF and DCT methods does not seem to be very efficient. The three ensembles containing this pair performs poorly comparing to others in the corresponding groups. These facts could infer that the EF and DCT techniques extract similar information, thus providing no advantage of a fusion. Moreover, the cumulative scores obtained for the *EF+HMM+EO* and *HMM+DCT+EO* ensembles verifies this assumption. In fact, the substitution of EF by DCT induces only a small difference in the recognition rate.

The results also suggest that the combinations of a PCA based method (ie: EF or EO) and the HMM technique provides good results. The ensembles containing these methods outperforms all others in the corresponding groups. The combinations *EF+HMM* and *EO+HMM* also provides outstanding *Top-2* performance, challenging the best 3-classifiers ensembles on the same level.

Finally, the 3-classifiers *EF+HMM+EO* is the overall winner with a small lead, but still yield a recognition rate increase of almost 6% over the individual methods

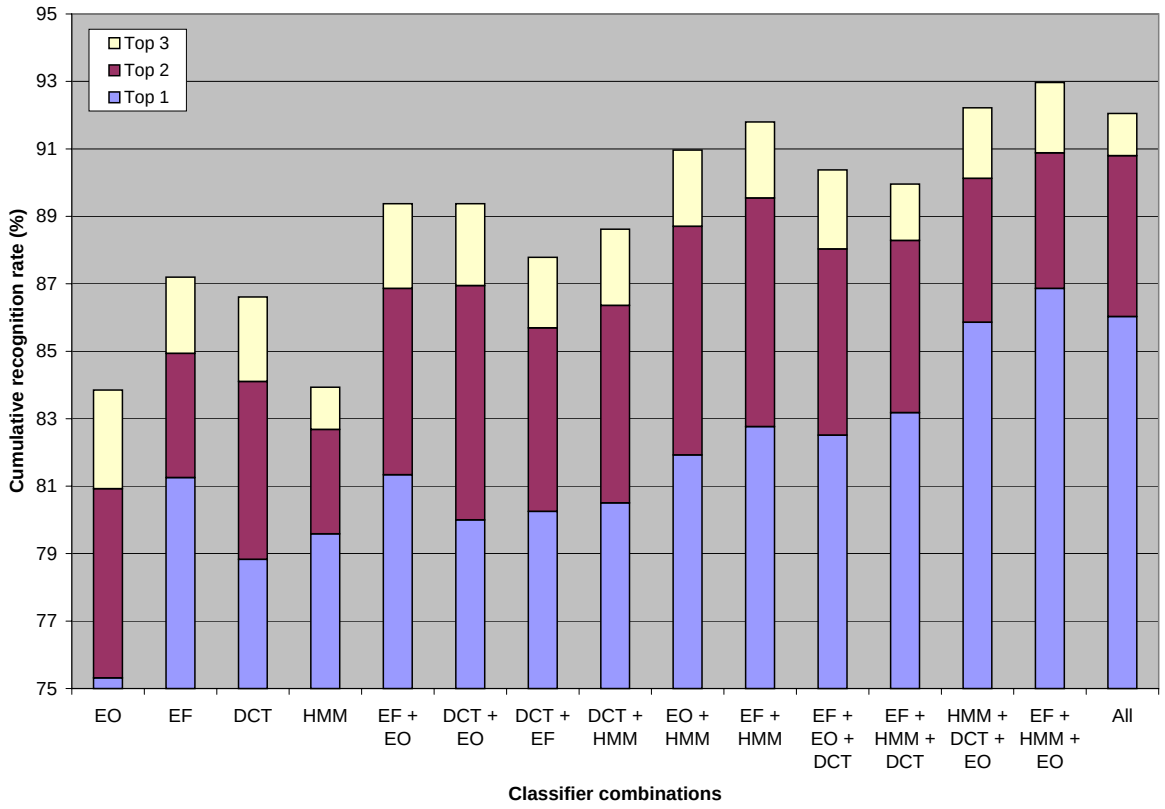


Figure 7: Cumulative score performance for different ensembles of classifiers on the FERET expression probe set (FB) containing 1,195 images. The recognition modules are abbreviated by EF (*EigenFaces*), DCT (Discrete Cosine Transform), EO (*EigenObjects*) and HMM (embedded Hidden Markov Models).

(*Top-1*). Moreover, this ensemble even outperforms the 4-classifiers (denoted *All*).

# 6 Conclusion

In this paper, a flexible and versatile software architecture for multi-classifier systems is proposed. A multi-classifier experiment using 4 different face recognition algorithms was conducted. Results show the advantage of classifier combinations over individual methods, by a recognition rate increase of almost 6% on a section of the FERET database.

Moreover, the embedded HMM seems to complement the other methods efficiently. In fact, almost all the combinations containing an HMM outperforms the others in the same group. Results also suggest that the combinations of the EF and DCT techniques in a multi-classifier system does not provide any benefit.

As future work, other algorithms could be implemented and different voting methods should be used. Dynamic selection of classifiers is also an interesting path to explore. Finally, experiments using an higher number of images per person could be realized to extend the conclusions obtained in this paper.

**Acknowledgements:** This research was supported by an NSERC–Canada grant to M. Parizeau.

# References

[1] Peter N. Belhumeur, Joao Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *PAMI*, 19(7):711–720, July 1997.


Figure 8: Multi-classifier prototype for face recognition. While the upper-left window represents the background subtraction results, the face detection process is illustrated in the upper-right window. After normalization, the face is presented to several classifiers (i.e. 3 in this example) and results are shown in the bottom window.


[2] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz. Fusion of face and speech data for person identity verification. *IEEE Transactions on Neural Networks*, 10(5):1065–1074, September 1999.

[3] David J. Beymer. Face recognition under varying pose. Technical Report AIM-1461, MIT AI Lab, 1993.

[4] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] Jacek Czyz, Josef Kittler, and Luc Vandendorpe. Combining face verification experts. In *16th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 28–31, August 2002.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, USA, 1994.

[7] G. Giacinto and F. Roli. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34(9):179–181, 2001.

[8] G.-Dong Guo and Hong-Jiang Zhang. Boosting for fast face recognition. In *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 96–100, 2001.

[9] Ziad M. Hafed and Martin D. Levine. Face recognition using discrete cosine transform. *International Journal of Computer Vision*, 43(3):167–188, July - August 2001.

[10] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. On multiple classifier systems for pattern recognition. In *11th International Conference on Pattern Recognition*, volume 2, pages 84–87, 1992.

[11] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(1):103–108, 1990.

[12] Josef Kittler, Mohammad Hatef, Robert P.W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-20(3):226– 239, 1998.

[13] Ara V. Nefian and Monson H. Hayes III. An embedded HMM-based approach for face detection

and recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume VI, pages 3553–3556, March 1999.

[14] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, WA, June 1994.

[15] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104, 2000.

[16] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[17] Merijn van Erp, Louis Vuurpijl, and Lambert Schomaker. An overview and comparison of voting methods for pattern recognition. In *Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 195–200, August 2002.

[18] W. Zhao, R. Chellappa, A. Rosenfeld, and P.J. Phillips. *Face Recognition: A Literature Survey*. http://citeseer.nj.nec.com/374297.html, 2000. 67 pages.

[19] Jie Zhou and David Zhang. Face recognition by combining several algorithms. In *16th International Conference on Pattern Recognition (ICPR)*, volume 3, pages 497–500, August 2002.