
ELSEVIER

Pattern Recognition Letters 23 (2002) 1459–1471

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Rejection based classifier for face detection

Michael Elad [*,1], Yacov Hel-Or [2], Renato Keshet

Hewlett-Packard Laboratories Israel, The Technion City, Haifa 32000, Israel

Abstract

Pattern detection problems require a separation between two classes, *Target* and *Clutter*, where the probability of the former is substantially smaller compared to that of the latter. In this paper we propose a new classifier that exploits this property, yielding a low complexity yet effective target detection algorithm. This algorithm, called the maximal rejection classifier (MRC), is based on linear successive rejection operations. An application of detecting faces in images is demonstrated using the MRC with encouraging results. © 2002 Published by Elsevier Science B.V.

Keywords: Pattern detection; Classification; Maximal-rejection; Face-detection; Generalized eigenvalue

1. Introduction

In target detection applications, the aim is to detect occurrences of a specific *Target* in a given signal. In this context, the non-*Target* samples are referred to as *Clutter*. In practice, the target detection problem can be characterized as designing a classifier $C(z)$, which, given an input vector z, has to decide whether z belongs to the *Target* class $\mathbf{X}$ or the *Clutter* class $\mathbf{Y}$. In example based classification, this classifier is designed using two training sets—$\widehat{\mathbf{X}} = \{x_i\}_{i=1,\ldots,L_x}$ (*Target* samples) and $\widehat{\mathbf{Y}} = \{y_i\}_{i=1,\ldots,L_y}$ (*Clutter* samples), drawn from the above two classes.

Since the classifier $C(z)$ is usually the heart of the detection algorithm, and is applied many times, simplifying it implies an efficient detection algorithm. Various types of example-based classifiers are suggested in the literature (e.g., Duda and Hart, 1973; Vapnik, 1995; Cortes and Vapnik, 1995; Baker and Nayar (1996)). One of the simplest and fastest are the linear classifiers, where $C(z)$ is based on a projection operation followed by a thresholding. The projection of z is performed onto a projection vector u, thus, $C(z) = f(u'z)$ where $f(*)$ is a thresholding operation (or some other decision rule). The support vector machine (SVM) proposed originally by Cortes and Vapnik (1995) and the Fisher linear discriminant (FLD) (see Duda and Hart, 1973) are two examples of linear classifiers. In both cases the kernel u is chosen in some optimal manner. In the FLD, u is chosen such that the Mahalanobis distance of the two classes after projection will be maximized. In the SVM approach the motive is similar, but the vector u is chosen such that it maximizes the margin between the two sets.

* Corresponding author.

E-mail addresses: elad@sccm.stanford.edu (M. Elad), toky@idc.ac.il (Y. Hel-Or), renato@hpli.hpl.hp.com (R. Keshet).

[1] Present address: The Computer Science Department (SCCM Program), Stanford University, Gates 2B, Stanford, CA 90305-9025, USA.

[2] Present address: The School of Computer Science, Interdisciplinary Center, Herzlia 46150, Israel.

0167-8655/02/$ - see front matter © 2002 Published by Elsevier Science B.V.
PII: S0167-8655(02)00106-X

In both these classifiers, it is assumed that the two classes have equal importance. In typical target detection applications the above assumption is not valid since the probability of z belonging to $\mathbf{X}$ is substantially smaller, compared to that of belonging to $\mathbf{Y}$. Neither the FLD nor the SVM exploit this property. Moreover, in both of these methods, it is assumed that the classes are linearly separable. in order to be able to treat more complex, and unfortunately, more common scenarios, non-linear extensions of these algorithms are required (see Duda and Hart (1973) or Cortes and Vapnik (1995)). Such extensions are typically at the expense of much more computationally intensive algorithms.

The maximal rejection classifier (MRC) is a linear-based classifier that overcomes the above two drawbacks. While maintaining the simplicity of a linear classifier, it can also deal with linearly non-separable cases. The only requirement is that the *Clutter* class and the convex hull of the *Target* class are disjoint. We define this property as convex separability, which is a much weaker condition than linear separability. In addition, this classifier exploits the property of high *Clutter* probability. Hence, it attempts to give very fast *Clutter* labeling, even if at the expense of slow *Target* labeling. Thus, the entire input signal is classified very fast.

The MRC is an iterative rejection based classification algorithm. The main idea is to apply at each iteration a linear projection followed by a thresholding, similar to the SVM and the FLD. However, as opposed to these two methods, the projection vector and the corresponding thresholds are chosen such that at each iteration the MRC attempts to maximize the number of rejected *Clutter* samples. This means that following the first classification iteration, many of the *Clutter* samples are already classified as such, and discarded from further consideration. The process is continued with the remaining *Clutter* samples, again searching for a linear projection vector and thresholds that maximizes the rejection of *Clutter* points from the remaining set. This process is repeated iteratively until a small number or non of the *Clutter* points remain. The remaining samples at the final stage are considered as *Targets*. The idea of rejection-based classifier was already introduced by Baker and Nayar (1996). However, in this work we extend the idea by using the notion of maximal rejection.

In order to demonstrate the behavior of the MRC, this algorithm is applied to the problem of detecting frontal and vertical faces in images. It is demonstrated that the MRC is a very efficient algorithm, requiring an effective computation of close to two convolutions of the input image per each resolution layer in order to reliably detect faces at all scales and all spatial positions.

2. The MRC in theory

Assume two classes are given in $\Re^n$, $\mathbf{X}$ and $\mathbf{Y}$ (*Target* and *Clutter* classes). It is required to discriminate between these two classes, i.e., given a point z drawn from one of these classes, we would like to label it correctly as either *Target* or *Clutter*. One important point, however, is that we know a priori that:

$$P\{\mathbf{X}\} \ll P\{\mathbf{Y}\}. \tag{1}$$

where $P\{\mathbf{X}\}$ is the a priori probability that an input signal will be a *Target*, and $P\{\mathbf{Y}\}$ is defined similarly. Based on this knowledge, we would like the classifier to give a decision as fast as possible (i.e., with as few operations as possible). Thus, *Clutter* labeling should be performed fast, even if at the expense of slow *Target* labeling.

Similar to other linear classifiers, we suggest to first project the sample z onto a vector u, and label it based on the projected value $\alpha = u^{\mathrm{T}}z$. Projecting the *Target* class onto u results with a probability density function (PDF) $P\{\alpha|\mathbf{X}\}$, defined as:

$$P\{\alpha|\mathbf{X}\} = \int_z P\{z|\mathbf{X}\}\delta(u^{\mathrm{T}}z - \alpha)\,\mathrm{d}z \tag{2}$$

where $\delta(x)$ is the Dirac's function. The term $P\{\alpha|\mathbf{X}\}$ defines the probability that a given input drawn from the *Target* class will obtain the value α after it has been projected onto u. Similarly, Projecting the *Clutter* class onto u results with a PDF $P\{\alpha|\mathbf{Y}\}$:

$$P\{\alpha|\mathbf{Y}\} = \int_z P\{z|\mathbf{Y}\}\delta(u^{\mathrm{T}}z - \alpha)\,\mathrm{d}z. \tag{3}$$

We define the following intervals based on $P\{\alpha|\mathbf{X}\}$ and $P\{\alpha|\mathbf{Y}\}$:

$$\begin{aligned} C_t &= \{\alpha | P\{\alpha|\mathbf{X}\} > 0, P\{\alpha|\mathbf{Y}\} = 0\} \\ C_c &= \{\alpha | P\{\alpha|\mathbf{X}\} = 0, P\{\alpha|\mathbf{Y}\} > 0\} \\ C_u &= \{\alpha | P\{\alpha|\mathbf{X}\} > 0, P\{\alpha|\mathbf{Y}\} > 0\} \end{aligned} \tag{4}$$

(t-Target, c-Clutter and u-Unknown). After projection, z is labeled either as a *Target*, *Clutter*, or *Unknown*, based on the following decision rule:

$$\text{Classifier}\{z\} = \begin{cases} \text{Target} & u^{\mathrm{T}} z \in C_t \\ \text{Clutter} & u^{\mathrm{T}} z \in C_c \\ \text{Unknown} & u^{\mathrm{T}} z \in C_u \end{cases} \tag{5}$$

Unknown classifications are obtained only in the C_u interval, where a decision cannot be made. Fig. 1 presents an example for the construction of the intervals C_t, C_c and C_u and their appropriate decisions. The probability of the *Unknown* decision is given by:

$$\begin{aligned} P\{\text{Unknown}\} &= \int_{\alpha \in C_u} P\{\mathbf{Y}\} P\{\alpha|\mathbf{Y}\}\, d\alpha \\ &\quad + \int_{\alpha \in C_u} P\{\mathbf{X}\} P\{\alpha|\mathbf{X}\}\, d\alpha \end{aligned} \tag{6}$$

Note that since $P\{\mathbf{X}\} \ll P\{\mathbf{Y}\}$, the contribution of $P\{\alpha|\mathbf{X}\}$ to $P\{\text{Unknown}\}$ is very small.

The above term is a function of the projection vector u. We would like to find the vector u which minimizes the "Unknown" probability. However, since this is a complex minimization problem, an alternative minimization is developed here, using a proximity measure between the two PDF's.

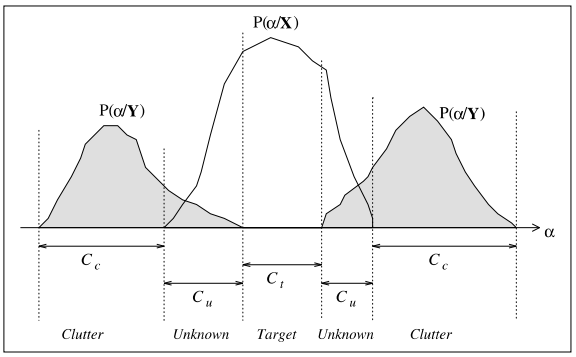

Fig. 1. The intervals C_t, C_c and C_u, for specific PDFs $P\{\alpha|\mathbf{X}\}$ and $P\{\alpha|\mathbf{Y}\}$.

If $P\{\alpha|\mathbf{Y}\}$ and $P\{\alpha|\mathbf{X}\}$ are far apart and separated from each other $P\{\text{Unknown}\}$ will be small. Therefore, an alternative requirement is to minimize the overlap between these two PDF's. We will define this requirement using the following expected distance between a point α_0 and a distribution $P\{\alpha\}$:

$$D(\alpha_0 \| P\{\alpha\}) = \int_\alpha \frac{(\alpha_0 - \alpha)^2 P\{\alpha\}}{\sigma^2}\, d\alpha \tag{7}$$

where σ is the variance of $P\{\alpha\}$ and the division by σ is performed in order to make the distance scale-invariant (or unit-invariant). Calculating the integral above, it is easy to verify that:

$$D(\alpha_0 \| P\{\alpha\}) = \frac{(\alpha_0 - \mu)^2 + \sigma^2}{\sigma^2} \tag{8}$$

where μ is the mean of $P\{\alpha\}$. Using this distance definition, the distance of $P\{\alpha|\mathbf{X}\}$ from $P\{\alpha|\mathbf{Y}\}$ can be defined as the expectation of the above distance:

$$\begin{aligned} D(P\{\alpha|\mathbf{Y}\} \| P\{\alpha|\mathbf{X}\}) &= \int_\alpha D(\alpha \| P\{\alpha|\mathbf{X}\}) P\{\alpha|\mathbf{Y}\}\, d\alpha \\ &= \int_\alpha \frac{(\alpha - \mu_x)^2 + \sigma_x^2}{\sigma_x^2} P\{\alpha|\mathbf{Y}\}\, d\alpha \\ &= \frac{(\mu_y - \mu_x)^2 + \sigma_x^2 + \sigma_y^2}{\sigma_x^2} \end{aligned} \tag{9}$$

where $[\mu_x, \sigma_x]$ and $[\mu_y, \sigma_y]$ are the mean–variance pairs of $P\{\alpha|\mathbf{X}\}$ and $P\{\alpha|\mathbf{Y}\}$, respectively. Since we want the two distributions to have as small an overlap as possible, we would like to maximize this distance or minimize the *proximity* between $P\{\alpha|\mathbf{Y}\}$ and $P\{\alpha|\mathbf{X}\}$, which can be defined as the inverse of their mutual distance. Note, that this measure is asymmetric with respect to the two distributions, i.e the proximity defines the closeness of $P\{\alpha|\mathbf{Y}\}$ to $P\{\alpha|\mathbf{X}\}$, but not vice versa. Therefore, we define the overall proximity between the two distributions as follows:

$$Prox(P\{\alpha|\mathbf{Y}\}, P\{\alpha|\mathbf{X}\}) = \frac{P\{\mathbf{X}\}\sigma_y^2 + P\{\mathbf{Y}\}\sigma_x^2}{\sigma_x^2 + \sigma_y^2 + (\mu_y - \mu_x)^2}. \tag{10}$$

Compared to the original expression in Eq. (6), the minimization of this term with respect to u is easier. If $P\{\mathbf{X}\} = P\{\mathbf{Y}\}$, i.e. if there is an even

chance to obtain *Target* or *Clutter* inputs, the proximity becomes:

$$Prox(P\{\alpha|\mathbf{Y}\}, P\{\alpha|\mathbf{X}\}) = \frac{\sigma_x^2 + \sigma_y^2}{\sigma_x^2 + \sigma_y^2 + (\mu_y - \mu_x)^2} \tag{11}$$

which is the cost function minimized by the FLD (see Duda and Hart, 1973). In our case $P\{\mathbf{X}\} \ll P\{\mathbf{Y}\}$ (Eq. (1)), thus, the first term is negligible in Eq. (10) and can be omitted. Therefore, the optimal u should minimize the resulting term:

$$d(u) = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_y^2 + (\mu_y - \mu_x)^2} \tag{12}$$

where σ_y^2, σ_x^2, μ_y and μ_x are all a function of the projection vector u. Minimization of this expression usually results in a small σ_x and large σ_y. This means that the projection of *Target* inputs tend to concentrate near a constant value, whereas the *Clutter* inputs will spread with a large variance (see e.g. Fig. 2).

For the optimal u, most of the *Clutter* inputs will be projected onto C_c, while C_t might even be an empty set. Subsequently, after projection, many of the *Clutter* inputs are usually classified, whereas *Target* labeling may not be immediately possible. This serves our purpose because for a *Clutter* input, there is a high probability that a decision will be made. Since these inputs are more frequent, this means a faster decision for the vast majority of the inputs.

The method we suggest follows this scheme: The classifier is applied *iteratively*, projecting and thresholding with different parameters at each iteration sequentially. Since the classifier is asymmetric, the classification is based on *rejections* (Fig. 3); *Clutter* inputs are classified and removed from further consideration while the remaining inputs are kept as suspected *Targets*. The iterations and the *rejection* approaches are both key concepts of the proposed scheme.

3. The MRC in practice

We return to Eq. (12) and find the optimal projection vector u. Thus, we have to express σ_y^2, σ_x^2, μ_y and μ_x as functions of u. It is easy to see that:

$$\mu_x = u^T\mathbf{M}_x, \quad \sigma_x^2 = u^T\mathbf{R}_{xx}u \quad \text{and}$$
$$\mu_y = u^T\mathbf{M}_y, \quad \sigma_y^2 = u^T\mathbf{R}_{yy}u \tag{13}$$

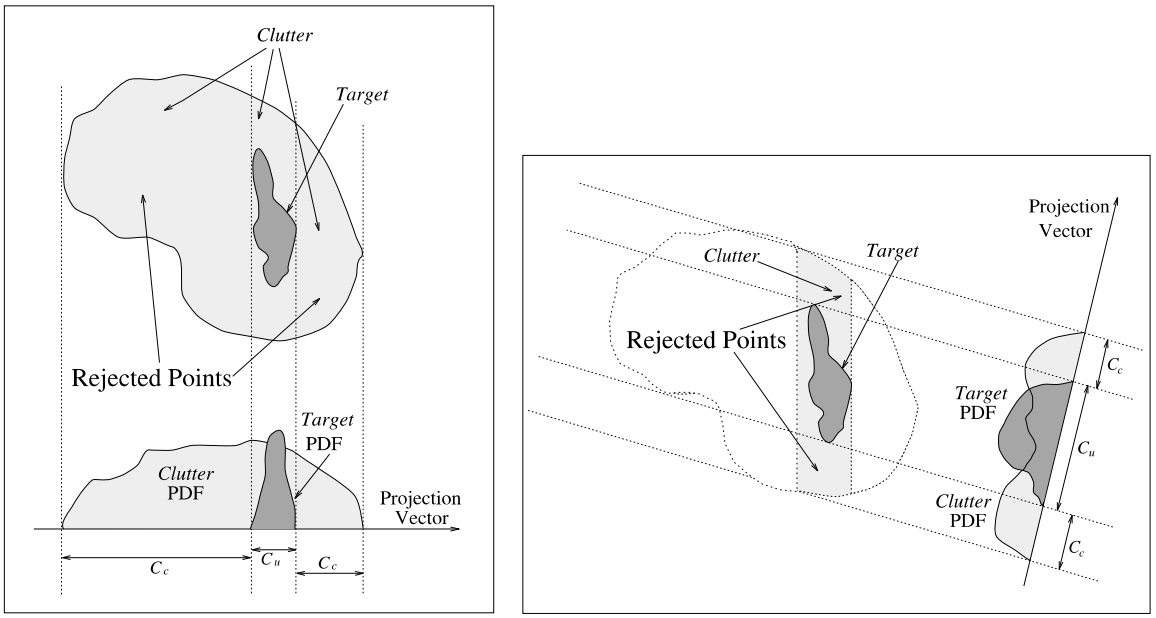

Fig. 2. First (left) and second (right) rejection stages.

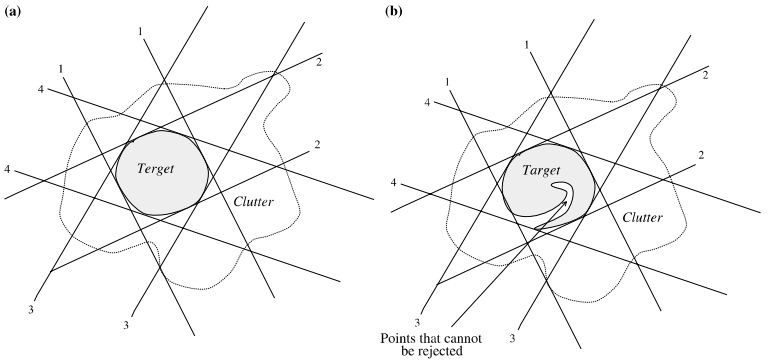

Fig. 3. (a) Rejection of *Clutter* inputs for a convex *Target* set. (b) Rejection of *Clutter* inputs for a non-convex *Target* set.

where we define:

$$\begin{aligned}
\mathbf{M}_x &= \int_z zP\{z|\mathbf{X}\}\,\mathrm{d}z \\
\mathbf{R}_{xx} &= \int_z (z-\mathbf{M}_x)(z-\mathbf{M}_x)^{\mathrm{T}}P\{z|\mathbf{X}\}\,\mathrm{d}z \\
\mathbf{M}_y &= \int_z zP\{z|\mathbf{Y}\}\,\mathrm{d}z \\
\mathbf{R}_{yy} &= \int_z (z-\mathbf{M}_y)(z-\mathbf{M}_y)^{\mathrm{T}}P\{z|\mathbf{Y}\}\,\mathrm{d}z.
\end{aligned} \tag{14}$$

As can be seen, only the first and second moments of the classes play a role in the choice of the projection vector u.

In practice we usually do not have the the probabilities $P\{z|\mathbf{X}\}$, $P\{z|\mathbf{Y}\}$, and inference on the *Target* or *Clutter* class is achieved through examples. For the two example-sets $\widehat{\mathbf{X}} = \{x_k\}_{k=1}^{L_x}$ and $\widehat{\mathbf{Y}} = \{y_k\}_{k=1}^{L_y}$, the mean-covariance pairs ($\mathbf{M}_x$, $\mathbf{R}_{xx}$, $\mathbf{M}_y$, and $\mathbf{R}_{yy}$) are replaced with empirical approximations:

$$\begin{aligned}
\widehat{\mathbf{M}}_x &= \frac{1}{L_x}\sum_{k=1}^{L_x} x_k \\
\widehat{\mathbf{R}}_{xx} &= \frac{1}{L_x}\sum_{k=1}^{L_x} (x_k-\widehat{\mathbf{M}}_x)(x_k-\widehat{\mathbf{M}}_x)^{\mathrm{T}} \\
\widehat{\mathbf{M}}_y &= \frac{1}{L_y}\sum_{k=1}^{L_y} y_k \\
\widehat{\mathbf{R}}_{yy} &= \frac{1}{L_y}\sum_{k=1}^{L_y} (y_k-\widehat{\mathbf{M}}_y)(y_k-\widehat{\mathbf{M}}_y)^{\mathrm{T}}
\end{aligned} \tag{15}$$

The function we aim to minimize is therefore:

$$d(u) = \frac{u^{\mathrm{T}}\widehat{\mathbf{R}}_{xx}u}{u^{\mathrm{T}}[\widehat{\mathbf{R}}_{xx}+\widehat{\mathbf{R}}_{yy}+(\widehat{\mathbf{M}}_y-\widehat{\mathbf{M}}_x)(\widehat{\mathbf{M}}_y-\widehat{\mathbf{M}}_x)^{\mathrm{T}}]u} \tag{16}$$

It is easy to show that u that minimizes the above expression satisfies:

$$\widehat{\mathbf{R}}_{xx}u = \lambda[\widehat{\mathbf{R}}_{xx}+\widehat{\mathbf{R}}_{yy}+(\widehat{\mathbf{M}}_y-\widehat{\mathbf{M}}_x)(\widehat{\mathbf{M}}_y-\widehat{\mathbf{M}}_x)^{\mathrm{T}}]u \tag{17}$$

and should correspond to the smallest possible λ. A problem of the form $Au = \lambda Bu$, as in Eq. (17), is known as the generalized eigenvalue problem (e.g., Duda and Hart, 1973; Golub and Loan, 1996; Demmel, 1997), and has a closed form solution. Notice that given any solution u for this equation, βu is also a solution with the same λ. Therefore, without loss of generality, the normalized solution $\|u\| = 1$ is used.

After finding the projection vector u that minimizes Eq. (16), the intervals C_{t}, C_{c}, and C_{u} can be determined. An input z is labeled as a *Target* or *Clutter* if its projected value $u^{\mathrm{T}}z$ is in C_{t} or C_{c}, respectively. Fig. 2 presents this stage for the case where C_{t} is empty, i.e. there are no inputs which can be classified as *Target*. Input vectors whose projected values are in C_{u} are not labeled. For these inputs we apply another step of classification, where the design of the optimal projection

vector in this step is performed according to the following new distributions:

$$P\{z|\mathbf{Y} \text{ and } u_1^{\mathrm{T}} z \in C_{\mathrm{u}}\} \quad \text{and} \quad P\{z|\mathbf{X} \text{ and } u_1^{\mathrm{T}} z \in C_{\mathrm{u}}\}$$

We define the next projection vector u_2 as the vector which minimizes the proximity measure between $P\{u_2^{\mathrm{T}} z|\mathbf{Y} \text{ and } u_1^{\mathrm{T}} z \in C_{\mathrm{u}}\}$ and $P\{u_2^{\mathrm{T}} z|\mathbf{X}$ and $u_1^{\mathrm{T}} z \in C_{\mathrm{u}}\}$. This minimization is performed in the same manner as before. Fig. 2 also presents the second rejection stage.

Following the second step, the process continues similarly with projection vectors $u_3, u_4, \ldots$, etc. Due to the proposed proximity measure minimization, it is expected that a large portion of the input vectors will be labeled as *Clutter* at each step, while following steps will deal with the remaining input vectors. Applying the cascade of classifiers in such an iterative manner ensures a good performance of the classification with respect to an accurate labeling and a fast classification rate.

Since we exchanged the class probabilities with sets of points, it is impractical to define the intervals C_{t}, C_{c}, and C_{u} using Eq. (4). This is because the intervals will be composed of many fragments each of which results from a particular example. Moreover, the domain of α cannot be covered by a finite set of examples. Therefore, it is more natural to define for each set, two thresholds bounding its projection values. As explained above, due to the functional that we are minimizing, in most cases the *Target* thresholds define a small interval located inside the *Clutter* interval (see Fig. 2). Therefore for simplicity, we define only a single interval $\Gamma = [T_1, T_2]$, which is the interval bounding the *Target* set, where we classify points projected outside Γ as *Clutter* and points projected inside Γ as *Unknown*.

In the case where the *Target* class forms a convex set, and the two classes are disjoint, it is theoretically possible to completely discriminate between them. More generally, if there are no *Clutter* inputs inside the convex hull of the *Target* set, exact discrimination is theoretically possible. This property is easily shown by noticing that we are actually extracting the *Target* set from the *Clutter* set by a sequence of two parallel hyper-planes, corresponding to the two thresholding operations. This constructs a parallelogram (see Fig. 3) that bounds the *Target* set from outside. Since any convex set can be constructed by a set of parallel hyper-planes, exact classification is possible. However, if the *Target* set is non-convex, or the two classes are non-convexly separable (as defined in the Section 1), it is impossible to achieve a classification with zero errors; *Clutter* inputs which are inside the convex hull of the *Target* set cannot be rejected. Fig. 3 presents such a case. Overcoming this limitation can be accomplished by a non-linear extension of the MRC, which is outside the scope of this paper.

In practice, even if we deal with a convex *Target* set, false-alarms may exist due to the the sub-optimal approach we are using, which neglects multi-dimensional moments higher than the second. However, simulations demonstrate that the number of false-alarms is typically small.

4. Comparison to other methods

The discriminant that is most related to the MRC is the FLD introduced by R.A. Fisher in 1936 (see Duda and Hart, 1973). The main idea is to find a projection vector that minimizes the *within-class* variances per each set, while maximizing the *between-classes* variance. For the two classes case, **X** and **Y**, this idea is expressed by maximizing the criterion:

$$\eta(u) = \left(\mu_y - \mu_x\right)^2 \Big/ \left(\sigma_x^2 + \sigma_y^2\right)$$

This criterion is exactly the one we get in Eq. (11), if we assume that the two classes have the same probability, i.e. $P(\mathbf{X}) = P(\mathbf{Y})$. The main advantage of MRC over the Fisher classifier is therefore the flexibility to deal with non-equally probable classes. This is quite important, not only with respect to the classification speed but also with respect to the effectiveness of the obtained projection vector. For example, if the means of the two classes coincide or nearly so, the Fisher classifier cannot give reasonable projection vectors since the numerator of $\eta(u)$ vanishes. Subsequently, MRC can apply successive rejections in an iterative manner, while Fisher discriminant applied iteratively will give poor performance.

Another classifier that draws a lot of interest recently is the support vector machine (SVM) proposed by Cortes and Vapnik (1995). The popularity of this classifier is based on the fact that it meets a theoretical limit bounding the classification error. Therefore, with respect to this theoretical bound, SVM is the optimal linear classifier. SVM chooses the projection vector u such that the margin between the *Target* and *Clutter* classes after projection is maximized. In a sense, this is a different way to define the proximity between the two distributions $P\{\alpha|\mathbf{X}\}$ and $P\{\alpha|\mathbf{Y}\}$. However, similar to the Fisher linear classifier, it deals with equally probable classes, so all the disadvantages listed above with respect to this point are valid here as well. Another drawback with SVM approach is that finding the optimal u requires to solve a quadratic programming problem. This procedure is complexity prohibited when the number of *Target* and *Clutter* points is more than several thousands.

The regularized multi-template matching (RMTM), proposed by Gotsman and Keren (1996) is a method close in spirit to the MRC. In this approach we assume that the *Target* class forms a subspace $\mathscr{T}$ embedded in some high dimensional linear space. It is also assumed that the *Clutter* class, $\mathscr{C}$ fills the residual space, i.e. the null space of $\mathscr{T}$:$\mathscr{C} = Null(\mathscr{T})$. The projection vector u in this approach is chosen to be the most "probable" vector in $\mathscr{C}$, where the probability is defined with respect to some a priori information on the *Clutter* class. Because RMTM was developed in the context of pattern detection in images, the probability of templates are assumed to be a measure of their spatial "smoothness" (see Jahne, 1995). The choice of the most probable vector in $\mathscr{C}$ as the projection vector relies on the fact that when applied to *Clutter* examples, it results with high values, while when applied to *Target* examples, it results with zero.

The similarity of RMTM approach to ours lies in the fact that *Target* and *Clutter* classes do not have equal status, and that *Target* samples are recognized by rejecting the *Clutters*. However, there are several important advantages in MRC compared to RMTM: First, while MRC uses actual training data from the *Clutter* set, RMTM uses an assumed information on this class. In addition, RMTM does not suggest an iterative rejection based classification. Instead, several vectors (orthogonal to the *Targets*, orthogonal between themselves, and the most smooth), are found and used in parallel. It is clear that these vectors are not optimal with respect to the maximum rejection rate. Another drawback in RMTM is the assumption that *Targets* and *Clutters* are orthogonal to each other. This assumption is not necessarily true and can degrade the results in many cases.

As to the computational complexity, RMTM is more efficient in the training part, as the variance and mean of the *Clutter* examples are not calculated. In the detection part MRC is much faster because of the optimal rejection approach, as opposed to the several parallel projections required by RMTM.

The concept of rejection-based classifier is not new and was already proposed by Baker and Nayar (1995, 1996). In their work, a composite rejector is suggested, which is constructed as an successive application of a core rejector. Their methodology refers both to the case where there are many sets (i.e., pattern recognition problem with many possible patterns), and to pattern detection, as in our case. In their work, the suggested core rejector is similar to that of Fisher's. Although this core rejector is not new, the importance of this work is in the establishment of a theoretical framework for rejection based classifiers. Our approach is based on a set of successive rejectors as defined in their work. The innovation of MRC, however, is by specifying the best rejector with respect to the maximal rejection rate.

5. Face detection using the MRC

The face detection problem can be specified as the need to detect all instances of faces in a given image, at all spatial positions, all scales, all facial expressions, all poses, of all people, and under all lighting conditions. All these requirements should be met, while having few or no false alarms and miss-detections, and with as fast an algorithm as possible. This description reveals the complexity of the detection problem at hand. As opposed to

other pattern detection problems, faces are expected to appear with considerable variations, even for detecting only frontal and vertical faces. Variations are expected because of changes in skin color, facial hair, glasses, face shape, and more.

Several papers already addressed the face detection problem using various methods, such as SVM (e.g., Vapnik, 1995; Osuna et al., 1997), Neural Networks (e.g. Rowley et al., 1997, 1998; Juell and March (1996)), and other methods (e.g. Sung and Poggio, 1998; Mirhosseini and Yan, 1995; Rajagopalan et al., 1997; Tankus et al., 1998). In all of these studies, the above list of requirements is relaxed in order to obtain practical detection algorithms. Following these works we deal with the detection of frontal and vertical faces.

In all these algorithms, spatial position and scale are treated through the same method: the given image is decomposed into a Gaussian pyramid with near-unity (e.g. 1.2) resolution ratio. The search for faces is performed in each resolution layer independently, thus enabling the treatment of different scales. In order to be able to detect faces at all spatial positions, fixed sized blocks of pixels are extracted from the image at all positions for testing. In addition to the pyramid part, which treats varying scales and spatial positions, the core part of the detection algorithm is essentially a classifier which provides a *Face*/*Non-Face* decision for each input block.

In this paper we propose the using the MRC for this task. In the face-detection application, *Faces* take the role of targets, and *Non-Faces* are the clutter. In a typical image having millions of pixels, it is expected to detect a few dozens of faces at the most, which means that picking a *Non-Face* block from the image is much more probable. This property is exploited by the MRC in order to obtain an efficient face-detection classifier. The MRC produces very fast *Non-Face* labeling (i.e. with a low computational cost), at the expense of slow *Face* labeling. Thus, on the average, it has a short decision time per input block.

The first stage in the MRC is to gather two example-sets, *Faces* and *Non-Faces*. As mentioned earlier, large enough sets are needed in order to guarantee good generalization for the faces and the non-faces that may be encountered in images. As to the *Face* set, the ORL data-base [3] was used. This database contains 400 frontal and vertical face images of 40 different individuals. By extracting the face portion from each of these images and scaling to 15×15 pixels, we obtained the set $\widehat{\mathbf{X}} = \{x_k\}_{k=1}^{L_x}$ (with $L_x = 400$).

As discussed in Section 3, the *target* class should be assumed to be convex in order for the MRC to perform well. The *target* class, containing frontal and vertical faces, is convex if and only if for every pair of faces drawn from it, their convex averages form also legitimate faces. One can easily imagine two faces that are not perfectly aligned, and therefore, when averaged, create a new block with possibly four eyes, or a nose and its echo, etc. To our help comes the fact that we are using low-resolution representation of the faces (15×15 pixels). This implies that even for such misaligned faces, the convex average appear as a face, and thus our assumption regarding convexity of the faces class is valid.

The *Non-Face* set is required to be much larger, in order to represent the variability of *Non-Face* patterns in images. We took 54 arbitrary images containing various textures, natural scenes, graphic images, etc. Common to all these images is that they contain no faces. Each of the 54 images was decomposed into a Gaussian pyramid with a 1.2 resolution ratio, thus creating 1290 images. Using the pyramids is beneficial both for enriching the *Non-Face* set, and for including multi-resolution versions of the same patterns in the data-base. Each possible block of 15×15 pixels in these 1290 images is considered as a candidate example of *Non-Face*. Thus, we have effectively collected more than 40 million *Non-Face* examples.

6. Improvements to the detection algorithm

6.1. Pre-processing

Given a block of pixels, the classifier is required to correctly classify it while being insensitive to

[3] http://www.cam-orl.co.uk/facedatabase.html: ORL database web-site.

variations, such as different lighting conditions, slightly different scale, spatial position, angle, additive noise, etc. For face blocks, the classifier should neglect the face background, which has no relevance to the classification result. All these and more requirements should be taken into account for a good generalization behavior, and making the applied classifier robust. One trivial, but practically close to impossible, method to meet the above requirements is to include in the example-sets all possible variations that are to be treated. This method is impractical since it requires a huge number of examples. Assuming that we could gather such example-sets and train on them, there is still no guarantee that the classifier will learn to "look" at the appropriate features, and to neglect the others.

An alternative to the above direct method is to apply pre-processing on the input block, and only then feed it to the classifier. For example, by removing the mean of the input block (or the best fitted 2D plane, as was used by Osuna et al. (1997) and Rowley et al. (1997, 1998)), we are actually removing some of the influence of the varying lighting conditions. This way, we are effectively using a classifier which is invariant to lighting conditions. This method is practical and typically used, and known to yield an overall improvement of the classification accuracy. However, applying a pre-processing on each input block has an additional computational cost.

When applying a linear classifier and settling with a linear pre-processing, these two operations can be combined into one, and in such cases, the additional pre-processing comes without computational cost. In order to see this property, let us denote the linear classifier of the input vector z as $\text{Class}\{z\} = T\{u^{\mathrm{T}}z\}$, where u is the projection vector, and $T\{\cdot\}$ is the thresholding operation. The pre-processing operation replaces z with Pz, where P is a square matrix which stands for the pre-processing operation. Thus, combing the pre-processing and the classification, the decision is obtained by $\text{Class}\{z\} = T\{u^{\mathrm{T}}Pz\} = T\{(P^{\mathrm{T}}u)^{\mathrm{T}}z\}$. Thus, instead of applying pre-processing on each input block z, the pre-processing is done only once on the projection vector u, and pre-processing is achieved with no computational cost.

Since the detection algorithm changes, a similar modification must be also applied in the training process. The same pre-processing must be applied on the two example sets before training. However, as we show here, this operation is done implicitly, rather than explicitly. In the MRC the example sets serve to compute the two mean-variance pairs. Thus, new mean-variance pairs should be computed using the pre-processed blocks $\{Px_k\}_{k=1}^{L_x}$ and $\{Py_k\}_{k=1}^{L_y}$. We denote these new mean-variance pairs as $\widetilde{\mathbf{M}}_x$, $\widetilde{\mathbf{R}}_{xx}$, $\widetilde{\mathbf{M}}_y$ and $\widetilde{\mathbf{R}}_{yy}$. Based on 15, we get:

$$\widetilde{\mathbf{M}}_x = \frac{1}{L_x}\sum_{k=1}^{L_x} Px_k = P\sum_{k=1}^{L_x} x_k = P\widehat{\mathbf{M}}_x$$

$$\widetilde{\mathbf{R}}_{xx} = \frac{1}{L_x}\sum_{k=1}^{L_x} (Px_k - \widetilde{\mathbf{M}}_x)(Px_k - \widetilde{\mathbf{M}}_x)^{\mathrm{T}} = P\widetilde{\mathbf{R}}_{xx}P^{\mathrm{T}}$$

$$\widetilde{\mathbf{M}}_y = \frac{1}{L_y}\sum_{k=1}^{L_y} Py_k = P\sum_{k=1}^{L_y} y_k = P\widehat{\mathbf{M}}_y$$

$$\widetilde{\mathbf{R}}_{yy} = \frac{1}{L_y}\sum_{k=1}^{L_y} (Py_k - \widetilde{\mathbf{M}}_y)(Py_k - \widetilde{\mathbf{M}}_y)^{\mathrm{T}} = P\widetilde{\mathbf{R}}_{yy}P^{\mathrm{T}}$$

Plugging these expressions into Eq. (16), the function that MRC minimizes is:

$$\widehat{d}(u) = \frac{u^{\mathrm{T}}[P\widehat{\mathbf{R}}_{xx}P^{\mathrm{T}}]u}{u^{\mathrm{T}}\{P[\widehat{\mathbf{R}}_{xx} + \widehat{\mathbf{R}}_{yy} + (\widehat{\mathbf{M}}_y - \widehat{\mathbf{M}}_x)(\widehat{\mathbf{M}}_y - \widehat{\mathbf{M}}_x)^{\mathrm{T}}]P^{\mathrm{T}}\}u} \tag{18}$$

which is equivalent to the minimization of the original function (16), subject to the constraint $u = P^{\mathrm{T}}u$.

One last question that remains is what kind of linear pre-processing operations could be beneficial. We can suggest several possibilities:

(1) *Removal of the mean*: Insensitivity to the mean corresponds to the ability to treat various kinds of lighting conditions, and different skin colors. Denoting a column of ones by $\mathbf{1}$, the vector mean is given by $\mathbf{1}^{\mathrm{T}}z/n$, where n is the vector dimension. The vector z without its mean is thus $z - \mathbf{1}\cdot\mathbf{1}^{\mathrm{T}}z/n = [I - \mathbf{1}\cdot\mathbf{1}^{\mathrm{T}}/n]z$.

(2) *Shaping the frequency representation*: Using 2D Discrete Cosine Transform (DCT), removal of

the $(0,0)$ entry is equivalent to removal of the block mean as suggested above. Removal of the $(0,1)$ and the $(1,0)$ entries stands for a removal of the best fitted smooth surface, which could be interpreted as lighting variations across the block. Removal of highest entries can be interpreted as low-pass filtering which removes too small details and noise. In all these cases, the block z is multiplied by the matrix D which applies the DCT, then multiplied by the diagonal matrix W which masks some of the entries out, or simply reduces their effect, and finally multiplied by D^{T} which does the inverse transform. The overall linear operation is therefore $D^{\mathrm{T}}WD$.

(3) *Masking out pixels*: By multiplying the vector z with a diagonal matrix M, which contains ones for non-masked pixels and zeros elsewhere, we can direct the classifier to neglect specific zones in the input blocks. This operation is important in order to remove the effects of the background in faces blocks (typically in the bottom left and right parts of the block).

In cases where several linear pre-processing steps are to be applied together, these pre-processing steps must be combined into one. More details about this projection combination process are given in Elad et al. (1999).

6.2. Algorithm speed-up

The MRC is a relatively very fast pattern detection algorithm. This is true because of the rejection obtained after each stage, which leaves smaller and smaller portions of the treated image for further consideration in later stages. However, this algorithm can be made faster yet by simplifying somehow the involved convolutions. This can be done in several ways:

(1) *Forcing the kernels to be separable*: By such a constraint on the minimizing vector of Eq. (16), instead of 225 multiplications/additions, only 30 such operations are required. Combining such a constraint results with more complicated optimization problem. As an alternative, the kernel can be computed as before, and rank-one filter can be extracted from it through singular-value-decomposition (SVD).

Having a separable kernel means that the convolution is done in two stages. The classifier efficiency gain can be made even higher by thresholding after the horizontal (1D) convolution. The required thresholds can be found by applying the horizontal kernel on the training faces, and finding the interval bounding all the faces. Using these thresholds, some of the input pixels are rejected already after the horizontal part, and thus, the vertical convolution is applied to fewer pixels in the image.

(2) *Masking entries in the kernel*: We have already mentioned masking as a measure to increase accuracy in detection results. The zero entries in the kernel also reduce the required number of multiplications/additions. Here we suggest to mask many of the entries, and this way further cut down the computational cost. Possible effective masks are (i) spreading the zero entries in a chess-board pattern; or (ii) leave only the eyes zone.

(3) *Reducing the block size*: We have chosen to use 15×15 pixels block size, which is an arbitrary choice. By choosing smaller block size, such as 7×7, we can cut down the required computations.

Given an image of size $N \times N$, applying the original 15×15 kernels requires 225 multiplications and additions per pixel. Applying 7×7 kernels should be done on a different layer of the pyramid, in order to detect faces of the same scale. Thus, the image size we operate on should be roughly $N/2 \times N/2$. Therefore, $49/4$ computations and additions are used per pixel. This means that the reduction in the overall number of computations is achieved by two means—the actual size of the kernel, and the fact that it corresponds to a different resolution layer in the pyramid.

All these ideas indeed reduce the computational cost, but may also degrade rejection rate and accuracy. Therefore, all the above methods are good algorithms as a fast candidate selection (FCS) stage, which is applied first, leaving a small number of candidate pixels as potential *Faces*. This part should work fast, and accuracy can be compromised, as long as no miss-detections are allowed. At the second stage, regular kernels should be applied for fine-tuning the results, even at the expense of a more complex algorithm.

7. Results

We trained the MRC for detecting faces by computing 50 sets of kernels $\{u_k\}_{k=1}^{50}$ and associated thresholds $\{[T_1^k, T_2^k]\}_{k=1}^{50}$, using the above described databases of *Faces* and *Non-Faces*. In our simulations, we have chosen to use the mean removal and the masking pre-processing stages. The following figures show the results obtained for several images. In all these examples, the first stage rejected close to 90% of the candidates. This stage is merely a convolution of the input image (at every scale) with the first kernel, u_1, followed by thresholding. For these examples, the complete MRC classification required an effective number of close to two convolutions per each pixel in each resolution layer. As can be seen in Figs. 4–6, there are few false alarms, which typically correspond to blocks of pixels having a pattern which may resemble a face. Generally speaking, the algorithm performs very well in terms of detection rate, false alarm rate, and most important of all,

Fig. 4. Face detection with the MRC—example 1.

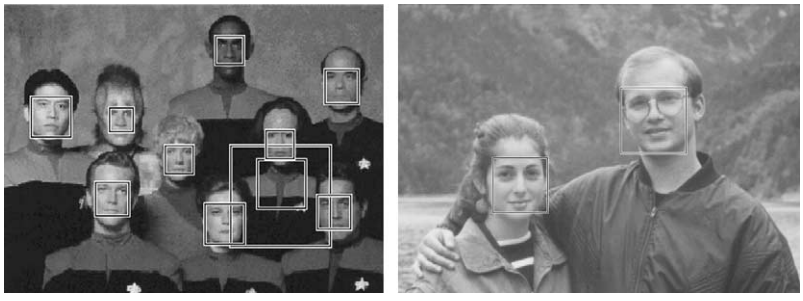

Fig. 5. Face detection with the MRC—examples 2 and 3.

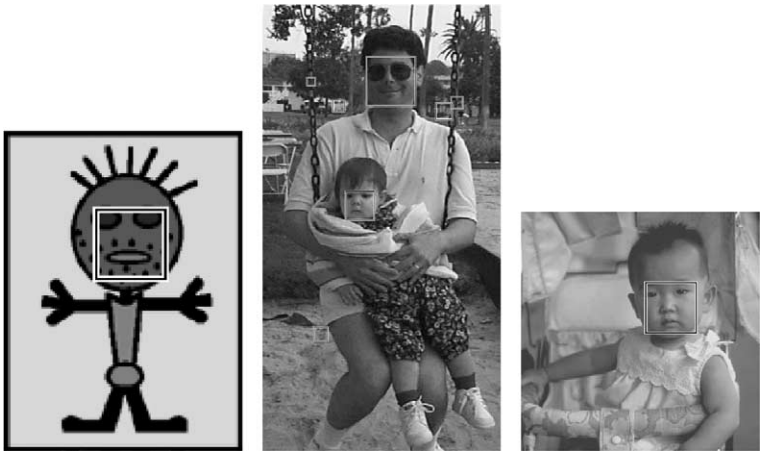

Fig. 6. Face detection with the MRC—examples 4–6.

computational complexity. Compared to Rowley et al. (1998) this MRC based face detection algorithm is found to be an order of magnitude faster and with roughly the same accuracy.

8. Conclusion

In this paper we presented a new classifier for target detection, which discriminates between *Target* and *Clutter* classes. The proposed classifier exploits the fact that the probability of a given input to belong to the *Target* class is much lower, than its probability to belong to the *Clutter* class. This assumption, which is valid in many pattern detection applications, is exploited in designing an optimal classifier that detects *Target* signals as fast as possible. Moreover, exact classification is possible when the *Target* and the *Clutter* classes are convexly separable. The FLD is a special case of the proposed framework when the *Target* and *Clutter* probabilities are equal. In addition, the proposed scheme overcomes the instabilities arising in the FLD in cases where the mean of the two classes are close to each other. An improvement of the proposed technique is possible by rejecting *Target* patterns instead of *Clutter* patterns in advanced stages, when the probability of *Clutter* is not larger anymore. The performance of the MRC is demonstrated in the face detection problem. The obtained face detection algorithm is shown to be both computationally very efficient and accurate. Further details on the theory of the MRC and its application to face detection can be found in (Elad et al., 1998, 1999).

9. Note added in proof

Recently, Viola and Jones (2001) suggested a successful rejection-based algorithm for face detection, similar in spirit to the MRC method proposed here. Their algorithm uses sub-linear weak classifiers combined via boosting, but as opposed to the MRC, maximal rejection is not exploited. In terms of performance these two methods are roughly equivalent both in terms of speed and accuracy. Further work is underway to study the interrelationship between these two methods and ways to further improve them.

References

Baker, S., Nayar, S.K., 1995. A theory of pattern rejection. Columbia University Technical Report CUCS-013-95.

Baker, S., Nayar, S.K., 1996. Pattern rejection. Proceedings 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Francisco, CA, USA, 18–20 June.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine Learning 20, 273–279.

Demmel, J.W., 1997. Applied Numerical Linear Algebra, first ed. SIAM—Society for Industrial and Applied Mathematics, Philadelphia.

Duda, R.O., Hart, P.E., 1973. Pattern Classification And Scene Analysis, first ed. Wiley-Interscience Publication, New York.

Elad, M., Hel-Or, Y., Keshet, R., 1998. Approximated invariant method (AIM): A rejection based classifier. Hewlett–Packard Technical Report—HPL-98-160.

Elad, M., Keshet, R., Hel-Or, Y., 1999. Frontal and vertical face detection in images using the approximated invariant method (aim). Hewlett–Packard Technical Report—HPL-99-5.

Gotsman, C., Keren, D., 1996. Regularized multi-template matching. Hewlett–Packard Technical Report—HPL-96-83.

Golub, G.H., Loan, C.F.V., 1996. Matrix Computations, third ed. Johns Hopkins University Press, Baltimore. MD.

Jahne, B., 1995. Digital Image processing. Springer-Verlag, Berlin.

Juell, P., March, R., 1996. A hierarchial neural network for human face detection. Pattern Recognition 29, 781–787.

Mirhosseini, A.R., Yan, H., 1995. Symmetry detection using attributed graph matching for a rotation-invariant face recognition system. Conf. Digital Image Computing: Tech. App., Australia.

Osuna, E., Freund, R., Girosi, F., 1997. Training support vector machines: An application to face detection. IEEE Conference on Computer Vision and Pattern Recognition, Puerto-Rico.

Rowley, H.A., Baluja, S., Kanade, T., 1997. Rotation invariant neural network-based face detection. Calnegie Mellon University Internal Report—CMU-CS-97-201.

Rowley, H.A., Baluja, S., Kanade, T., 1998. Neural network-based face detection. IEEE Trans. Pattern Anal. Machine Intell. 20, 23–38.

Rajagopalan, A.N., Kumar, K.S., Karlekar, J., Manivasakan, R., Patil, M.M., 1997. Finding faces in photographs. International Conference of Computer Vision (ICCV) 97, Delhi, India.

Sung, K.K., Poggio, T., 1998. Example-based learning for view-based human face detection. IEEE Trans. Pattern Anal. Machine Intell. 20, 39–51.

Tankus, A., Yeshurun, H., Intrator, N., 1998. Face detection by direct convexity estimation. TAU Technical Report.

Vapnik, V.N., 1995. The Nature of Statistical Learning Theory, first ed. Springer-Verlag, Berlin.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. Conference on Computer Vision and Pattern recognition 2001.