Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Haar-like features with optimally weighted rectangles for rapid object detection

Sri-Kaushik Pavani[a,b], David Delgado[a,b], Alejandro F. Frangi[a,b,*]

[a]Center for Computational Imaging and Simulation Technologies In Biomedicine (CISTIB), Department of Information and Communication Technologies, Universitat Pompeu Fabra, P. Circunvalacion 8, E-08003 Barcelona, Spain

[b]Networking Research Center on Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN), Barcelona, Spain

ARTICLE INFO

Article history:
Received 29 July 2007
Received in revised form 30 December 2008
Accepted 13 May 2009

Keywords:
Haar-like features
Object detection
Pattern rejection
Cascaded classifiers
Genetic algorithms

ABSTRACT

This article proposes an extension of Haar-like features for their use in rapid object detection systems. These features differ from the traditional ones in that their rectangles are assigned optimal weights so as to maximize their ability to discriminate objects from clutter (non-objects). These features maintain the simplicity of evaluation of the traditional formulation while being more discriminative. The proposed features were trained to detect two types of objects: human frontal faces and human heart regions. Our experimental results suggest that the object detectors based on the proposed features are more accurate and faster than the object detectors built with traditional Haar-like features.

1. Introduction

Object detectors aim at finding sub-regions of an image that contain instances of an object of interest. Many applications of object detection are challenging because high accuracy is required while images are evaluated at real-time speeds. Such applications include vehicle detection [26], where one needs to alert automobile drivers about possible accidents as soon as possible, and, surveillance tasks where every video frame needs to be checked in real-time for the presence of intruders [28].

Classifiers based on Haar-like features [18] have demonstrated to be considerably successful for use in object detection tasks. This is mainly due to the fact that they provide an attractive trade-off between accuracy and evaluation speed. Viola and Jones [27] proposed a popular object detection system based on these features. After Viola and Jones's contribution, many successful systems based on Haar-like features have been proposed [13,16,25,30].

In this paper, we propose modified Haar-like features whose rectangles are assigned optimal weights. These weights are optimal in the sense that they maximize their ability to discriminate object from clutter. Similar to the traditional Haar-like features, the proposed features can be used to form weak classifiers, which in turn can be boosted [21,22] and arranged in a rejection cascade architecture [3] to form an object detection system. In our experiments, three different techniques, brute-force search (BFS), genetic algorithms (GA) [5,7,8], and Fisher's linear discriminant analysis (FLDA) [6] were used to determine optimal weights.

The proposed features were trained for two object detection problems: detection of human frontal faces in digital photographs, and detection of cardiac structures from magnetic resonance imaging (MRI). The obtained results show that the object detectors based on the proposed features are more accurate and faster than the object detectors built with traditional Haar-like features.

The remainder of the paper is organized as follows. In Section 2, weak classifiers based on Haar-like features are introduced. In Section 3, algorithms to find optimal rectangle weights using BFS, GA and FLDA are presented. Later, in Section 4, the boosting procedure to combine weak classifiers to form strong classifiers is described. Section 5 describes the framework and the training procedure of the proposed object detector. A human frontal face and a heart detector were built based on the proposed object detection system. The results of their evaluation on real-life datasets are presented in Section 6. The paper is concluded in Section 7.

2. Haar-like features

Haar-like features are an over complete set of two-dimensional (2D) Haar functions, which can be used to encode local appearance of objects [18]. They consist of two or more rectangular regions enclosed in a template. The feature value f of a Haar-like feature which has k rectangles is obtained as in the following

* Corresponding author at: Center for Computational Imaging and Simulation Technologies In Biomedicine (CISTIB), Department of Information and Communication Technologies, Universitat Pompeu Fabra, P. Circunvalacion 8, E-08003 Barcelona, Spain. Tel./fax: +34 93 542 1451.

E-mail address: alejandro.frangi@upf.edu (A.F. Frangi).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.05.011

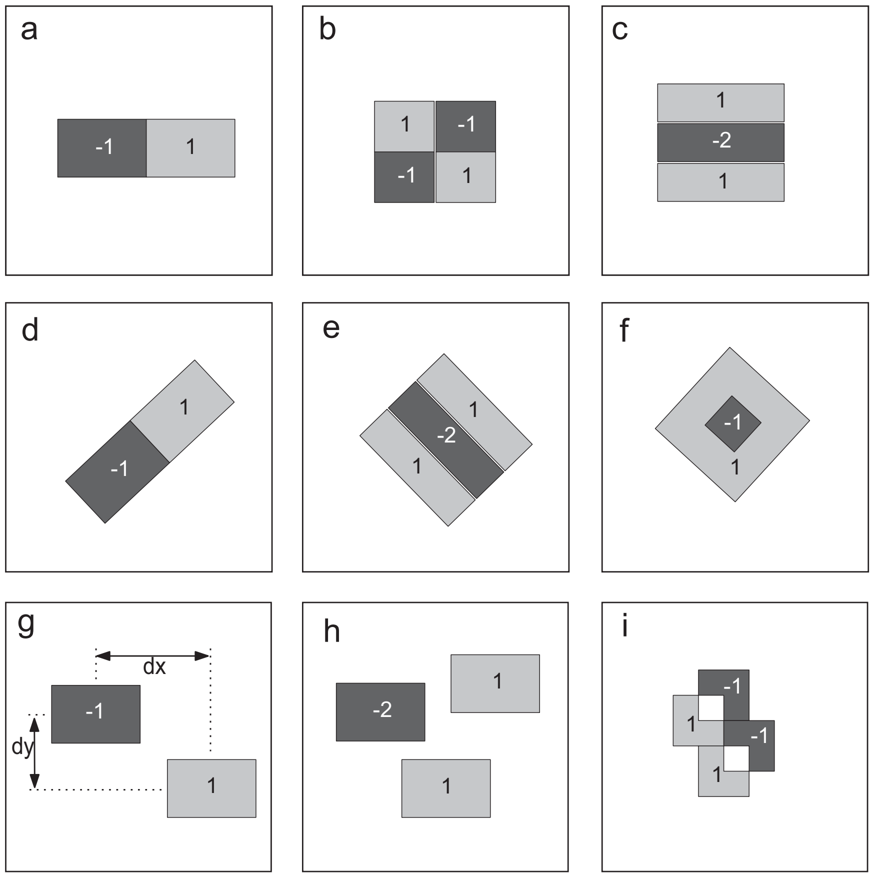


Fig. 1. Haar-like features are shown with the default weights assigned to its rectangles. (a) and (b) show Haar-like features proposed by Papageogiou et al. (c) shows a Haar-like feature with three rectangles introduced by Viola and Jones. (d–f) show Leinhardt's rotated features. (g) and (h) show Li et al.'s disjoint Haar-like features. (i) shows Jones and Viola's diagonal feature designed to capture diagonal structures in the object's appearance.

equation:

$$f = \sum_{i=1}^{k} w^{(i)} \cdot \mu^{(i)} \tag{1}$$

where $\mu^{(i)}$ is the mean intensity of the pixels in image $\mathbf{x}$ enclosed by the ith rectangle. Henceforth we will refer to the quantity μ as the *rectangle mean*. In (1), $w^{(i)}$ is the weight assigned to the ith rectangle. Traditionally, the weights assigned to the rectangles of a Haar-like feature are set to *default* integer numbers such that the following equation is satisfied:

$$\sum_{i=1}^{k} w^{(i)} = 0 \tag{2}$$

For example, the rectangles of a Haar-like feature as in Fig. 1(a) are assigned default weights 1 and −1. Similarly, the rectangles of a Haar-like feature as in Fig. 1(c) are assigned default weights 1, −2 and 1.

One of the main reasons for the popularity of the Haar-like features is that they provide a very attractive trade-off between speed of evaluation and accuracy. With a simple weak classifier based on Haar-like features costing just 60 microprocessor instructions, Viola and Jones [27] achieved 1% false negatives and 40% false positives for the face detection problem. The high speed of evaluation is mainly due to the use of integral images [27], which once computed, can be used to rapidly evaluate any Haar-like feature at any scale in constant time.

Since the introduction of horizontally and vertically aligned Haar-like features by Papageogiou et al. [18], many different Haar-like features have appeared in the literature [12,13,27]. Some of the features are shown in Fig. 1. They mainly differ in the number of rectangles and the orientation of the rectangles with respect to the template. Jones and Viola [10] introduced diagonal features to capture diagonal structures in object's appearance. Lienhart [13] enriched the features of Viola and Jones [27] with efficiently computable rotated Haar-like features. With the enriched set, they achieve a 10% lower false alarm rate for a given true positive rate. Li et al. [12] proposed Haar-like features with disjoint rectangles which were meant to characterize non-symmetrical features of an object's appearance.

2.1. Haar-like features with optimally weighted rectangles

In this paper, we argue that a Haar-like feature can be optimized for a given object detection problem by assigning optimal weights to its rectangles. The feature value of the proposed features is computed exactly as in (1), except that the default weights of its rectangles, $w^{(i)}$, are substituted with optimal values, $\hat{w}^{(i)}$. The proposed features maintain the simplicity of the traditional ones as in (1), while being more discriminative.

2.2. Weak classifiers

Weak classifiers label a sub-region of an image, $\mathbf{x}$, as belonging to either the object or clutter class by comparing f to a threshold θ

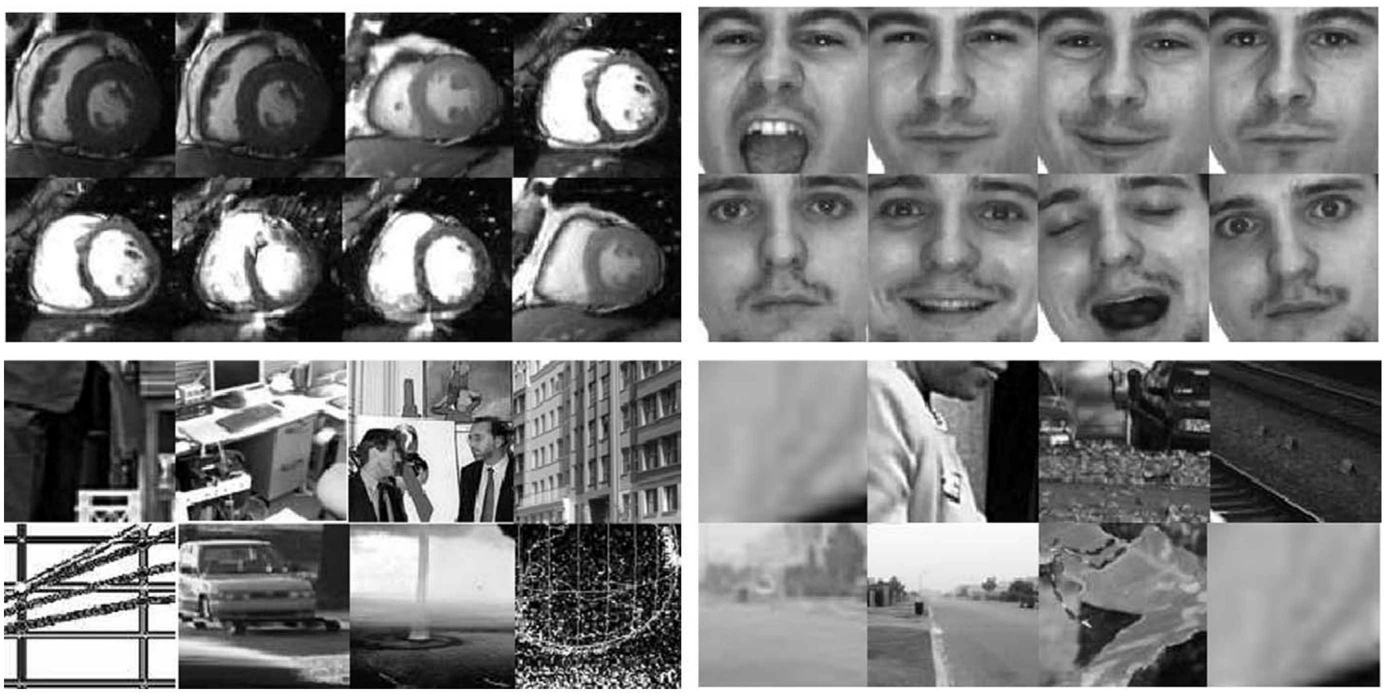


Fig. 2. The first row shows images from the heart and human frontal face image databases. The second row contains typical images from the clutter database.

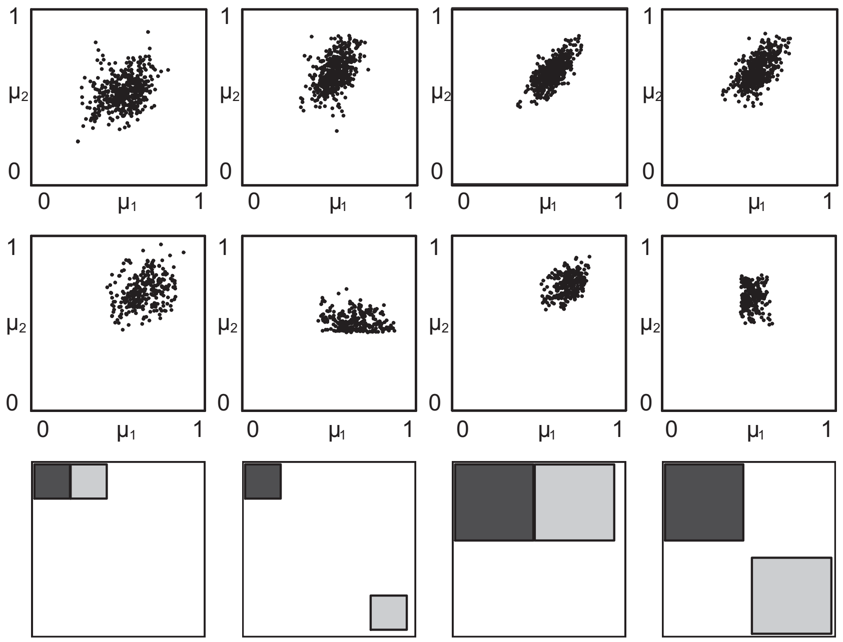


Fig. 3. Distribution of object points for different Haar-like features. The first and the second row show object point distributions in SRFS from human frontal face images and the heart images, respectively. The third row shows the Haar-like features that were used for the evaluation.

as in (3). They are called weak because they are expected to perform only slightly better than a random guesser:

$$h(\mathbf{x}) = \begin{cases} 1 \text{ object}, & f \cdot p \geq \theta \cdot p \\ -1 \text{ clutter} & \text{otherwise} \end{cases} \tag{3}$$

Here, $p \in \{1, -1\}$ is a polarity term, which can be used to invert the inequality relationship between f and θ.

2.3. Single-rectangle feature space (SRFS)

When an image is evaluated with a Haar-like feature with k rectangles, a vector of length k containing rectangle mean measurements can be generated. Using these measurements, the image can be represented as a point in a k-dimensional feature space—which we call single-rectangle feature space. Let the representation of an image belonging to the object class be called an object point, and similarly, the representation of an image belonging to the clutter class

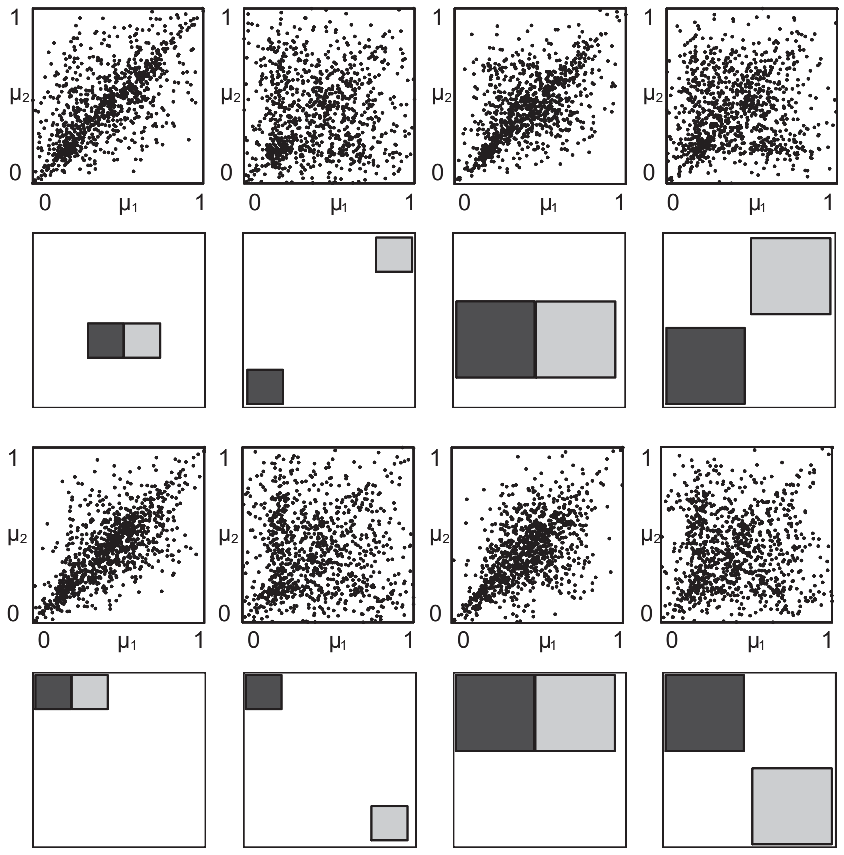


Fig. 4. Distribution of clutter points for different Haar-like features.

be called a clutter point. In the following, a discussion on the distribution of object and clutter points in a SRFS is presented. Based on this discussion, a comparison of the performance of weak classifiers constructed with traditional and the proposed Haar-like features is made.

For the purpose of this study, a set of images belonging to the object and clutter class were evaluated on Haar-like feature with two rectangles, thus generating a cloud of object and clutter points in a two-dimensional SRFS. The number of rectangles in the Haar-like feature was restricted to two for clarity of visualization.

Two different object databases were chosen for this study. One contained human frontal face images from the AR [14] database and the other contained short-axis magnetic resonance images of the heart. The clutter points were generated from a clutter database containing images that neither belonged to the face class nor the heart class. Some of the typical images in the object and clutter databases are shown in Fig. 2.

Each image was intensity normalized by dividing every pixel value by $2^n - 1$, where n is the number of bits used to represent each pixel value. This makes pixel values from any two images comparable as all of them vary from 0 to 1.

The distribution of object and clutter points in the 2D SRFS is shown in Figs. 3 and 4, respectively. As it can be observed, the object points form a more compact distribution than the clutter points irrespective of the size and the relative distance of the rectangles constituting the Haar-like feature. Compact distribution of object points results from making similar rectangle mean measurements. This can be understood from the fact that the images within the object class are strongly correlated to each other. The clutter points, however, are spread-out as the clutter images are very different from each other.

2.4. Performance of weak classifiers in SRFS

Consider a hypothetical 2D SRFS as shown in Fig. 5(a). It shows a compact distribution of object points in the midst of clutter points. As discussed earlier, each point in the SRFS is a representation of an image through its rectangle mean measurements ($\boldsymbol{\mu}$). The feature value of a Haar-like feature when evaluated on an image is the scalar product of the vector **w** that contains weights assigned to its rectangles, and $\boldsymbol{\mu}$. This has been mathematically expressed in (1). Geometrically, the computation of feature value can be understood as a product of magnitude of projection of $\boldsymbol{\mu}$ onto **w** and the magnitude of **w**, which is a constant.

Traditionally, the rectangles of a Haar-like feature with two rectangles are assigned default weights of 1 and -1. This yields a projection line that is parallel to the vector $\mu_1 - \mu_2$. To train a weak classifier as in (3), a scalar value θ is found such that object and clutter points are best separated. As shown in Fig. 5(b), θ divides the SRFS into two regions; the object region, coded white, and the clutter region, coded black.

Fig. 5(c) and (d) show classification using the optimal projection line. It is intuitive that by projecting object and clutter points to an optimal line, better classification performance can be achieved. Note that the proposed features have the capability to capitalize on the absolute difference in the brightness between the object class and the majority of clutter class images. For example, when the object points lie on the top-right or bottom-left corners of the 2D SRFS, better classification can be obtained by setting optimal line parallel to the vector $\mu_1 = \mu_2$. Traditional features, however, can produce good classifiers only when the majority of the object points would lie in left-top or the right-bottom of the 2D SRFS.

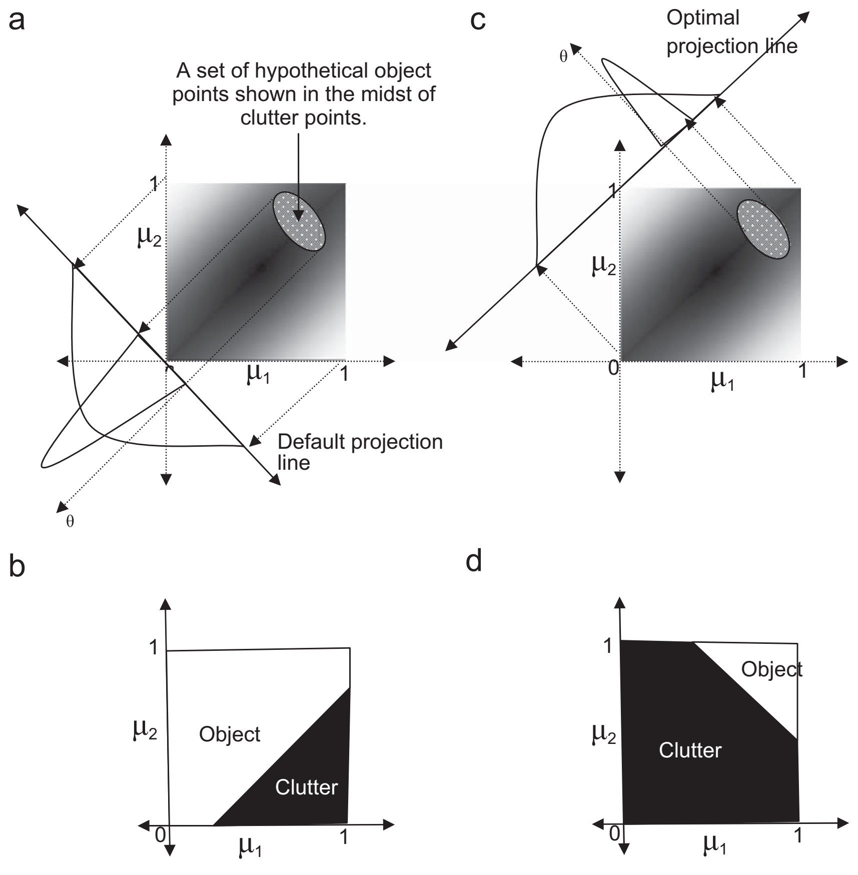


Fig. 5. A geometrical view of a weak classifier performance. (a) Weak classifiers built with traditional Haar-like features. (c) Weak classifier constructed with modified Haar-like features whose rectangles are assigned optimal weights. (b) and (d) show their respective partitioned SRFS.

3. Training weak classifiers

Training a weak classifier involves determining the following three parameters that give the least error on the training set of object and clutter class images:

(1) the threshold value θ,
(2) the polarity term p, and
(3) the weight to be assigned to each rectangle of a Haar-like feature $\hat{w}$.

The two terms, θ and p, are optimized using a brute-force search as performed by Viola and Jones [27]. Three different techniques were used to find optimal weights of the rectangles of the Haar-like features. The description of these techniques and the motivation for their use is described in the following sections.

3.1. Brute-force search

One obvious way to guarantee an optimal solution to the problem of finding optimal weights would be a brute-force search through the possible weights. The search space could be restricted to a finite number of possible values by quantizing the weight values and limiting them to a predefined range. If the weights assigned to rectangles are restricted to d values, then the number of distinct weight combinations for two, three and four rectangle Haar-like features are $\binom{d}{2}$, $\binom{d}{3}$ and $\binom{d}{4}$, respectively. For each of these weight combinations, optimal values of θ and p need to be determined, which makes brute-force search extremely time consuming. The search space can be reduced by increasing the quantization step and restricting the range. Such steps that reduce the resolution of the search space, however, may lead to assigning sub-optimal weights to the rectangles. Finding optimal weights using BFS is outlined in Algorithm 1.

Algorithm 1. Training of a single weak classifier using BFS.

```
  Input: Training images: $\mathbf{x}^{(i)},\ i \in \{1,\dots,n\}$
  Input: Training labels: $y^{(i)} \in \{-1,1\},\ i \in \{1,\dots,n\}$
  Input: Weights for each training image: $z^{(i)} \in \Re,\ i \in \{1,\dots,n\}$
  Input: Weak classifier with $k$-rectangle Haar-like feature: $h$
  Input: Possible weight values: $d$
1 Set $\varepsilon_{\min}$ to $\infty$.
2 for $t = 1$ to $\binom{d}{k}$ do
3     Set weights ($\hat{\mathbf{w}}$) to be assigned to the rectangles of $h$.
4     Find $\hat{\theta}$ and $\hat{p}$ that minimize the training error $\varepsilon$.
          $[\hat{\theta},\hat{p}] = \arg\min \varepsilon$
      where,
          $\varepsilon = \frac{1}{2}\sum_{i=1}^{n} z^{(i)}|h(\mathbf{x}^{(i)}) - y^{(i)}|$
5     if $\varepsilon < \varepsilon_{min}$ then
6         $\mathbf{w}^* = \hat{\mathbf{w}},\ \theta^* = \hat{\theta},\ p^* = \hat{p},\ \varepsilon_{\min} = \varepsilon$
  Output: Trained weak classifier: $h \to \mathbf{w}{=}\mathbf{w}^*$, $h{\to}\theta = \theta^*$, $h{\to}p{=}p^*$
  Output: Error of the weak classifier: $\varepsilon_{\min}$
```

3.2. Genetic algorithms

Given a specific problem to solve, the input to the GA is a set of solutions to that problem, called *genomes*, and a metric called a *fitness function* that returns a quantitative measure of the quality of a genome. GA, through a series of iterations, called *generations*, tries to improve the genomes through genetic operations such as *crossover* and *mutation*. The main advantages of GA are that it does not require gradient information and therefore it is capable of solving nonlinear problems with multiple local optima. Further, GA has been demonstrated to produce substantial improvement over random and local search methods [11]. Since GA evolves its solutions over many iterations, it tends to be computationally expensive. Yet, in comparison with brute-force search, it is faster. In practice, it was found that evolving 30 genomes over 100 generations yielded stable solutions.

In our problem of finding optimal weights to rectangles, the genome is an array, representing the weights to be assigned to the rectangles. A valid genome for a Haar-like feature with k rectangles would be an array of length k, whose elements can take real values. For a given genome and a Haar-like feature, our fitness function builds a weak classifier and returns the error ε made by the weak classifier on training examples. The lower the error returned by the fitness function, the better the genome represents the desired solution. The genetic optimization, as explained in Algorithm 2, is done over a series of N generations. In each generation, the population of genomes are improved through the application of crossover and mutation operators. The genome, along with the threshold and polarity values, that produce the least training error are chosen as the parameters to be assigned to the weak classifier h.

Algorithm 2. Training of a single weak classifier using GA.

```
   Input: Training images: $\mathbf{x}^{(i)},\ i \in \{1,\dots,n\}$
   Input: Training labels: $y^{(i)} \in \{-1,1\},\ i \in \{1,\dots,n\}$
   Input: Weights for each training image: $z^{(i)} \in \Re,\ i \in \{1,\dots,n\}$
   Input: Weak classifier to be trained: $h$
   Input: Number of generations: $N$
   Input: Size of each generation: $m$
1  begin
     Select initial population of genomes $\mathbf{g}_1^{(l)}, l \in \{1,\dots,m\}$.
     Set $\varepsilon_{\min}$ to $\infty$.
2    for $t = 1$ to $N$ do
3      forall $\mathbf{g}_t^{(l)}$ do
4        Set current genome as the weights to be assigned to
5        the rectangles of $h$
6            $h \to \mathbf{w} = \mathbf{g}_t^{(l)}$
         Find $\hat{\theta}$ and $\hat{p}$ that minimize the training error $\varepsilon$.
7            $[\hat{\theta}, \hat{p}] = \arg\min \varepsilon$
         where,
             $\varepsilon = \frac{1}{2}\sum_{i=1}^{n} z^{(i)} |h(\mathbf{x}^{(i)}) - y^{(i)}|$
8        if $\varepsilon < \varepsilon_{min}$ then
9          $\mathbf{w}^* = \mathbf{g}_t^{(l)},\ \theta^* = \hat{\theta},\ p^* = \hat{p},\ \varepsilon_{\min} = \varepsilon$
10     Reproduce genomes with the lowest training error in the
11     population to form new ones through crossover and mutation.
       Replace the worst genomes in the population with the best
       newgenomes.
12 end
   Output: Trained weak classifier: $h \to \mathbf{w} = \mathbf{w}^*$, $h \to \theta = \theta^*$,
   $h \to p = p^*$
   Output: Error of the weak classifier: $\varepsilon_{\min}$
```

3.3. Fisher's linear discriminant analysis

FLDA provides a closed-form solution to find a linear classifier that best separates two or more classes. Although it provides an optimal solution only when the classes are Gaussian with equal covariances [4], it is reasonably quick and provides good approximations to the optimal solution in the general case. Our motivation for choosing FLDA is as follows. Since object detection is basically a two class problem, FLDA might provide quick answer as it comes with a closed-form solution. This procedure directly outputs the optimal weights and, therefore, the optimal values of θ and p need to be computed only once. This makes FLDA much faster when compared to GA or brute-force methods.

Algorithm 3 describes a solution to find optimal weights using FLDA. The algorithm is presented with a database of object and clutter class training images with their corresponding labels. FLDA outputs a linear projection vector whose slope corresponds to the optimal weights assigned to the rectangles of the Haar-like feature. Once the optimal weights are found, the optimal values for threshold ($\hat{\theta}$) and polarity terms ($\hat{p}$) are found by searching exhaustively [27] over all possible solutions.

Algorithm 3. Training of a single weak classifier using FLDA.

```
  Input: Training images: $\mathbf{x}^{(i)},\ i \in \{1,\dots,n\}$
  Input: Training labels: $y^{(i)} \in \{1,-1\},\ i \in \{1,\dots,n\}$
  Input: Weights for each training image: $z^{(i)} \in \Re,\ i \in \{1,\dots,n\}$
  Input: Weak classifier to be trained: $h$
1 begin
    Evaluate the Haar-like feature in $h$ over all object and clutter
    training images. Each evaluation gives a $k$-dimensional vector
    $\boldsymbol{\mu}^{(i)}$. Compute between-class scatter [6]: $\boldsymbol{m}_1 - \boldsymbol{m}_2$ where $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$
    are the $k$-dimensional means of measurements made from $n_o$
    object class and $n_c$ clutter class training images respectively.
2
3       $\boldsymbol{m}_1 = \sum_{\forall i, \mathbf{y}^{(i)}=1} \frac{\boldsymbol{\mu}^{(i)}}{n_o}, \quad \boldsymbol{m}_2 = \sum_{\forall i, \mathbf{y}^{(i)}=-1} \frac{\boldsymbol{\mu}^{(i)}}{n_c}$
    Compute within-class scatter [6] : $\boldsymbol{S} = \boldsymbol{S}_1 + \boldsymbol{S}_2$
4       $\boldsymbol{S}_1 = \sum_{\forall i, \mathbf{y}^{(i)}=1} (\boldsymbol{\mu}^{(i)} - \boldsymbol{m}_1)(\boldsymbol{\mu}^{(i)} - \boldsymbol{m}_1)^t$
5     $\boldsymbol{S}_2 = \sum_{\forall i, \mathbf{y}^{(i)}=-1} (\boldsymbol{\mu}^{(i)} - \boldsymbol{m}_2)(\boldsymbol{\mu}^{(i)} - \boldsymbol{m}_2)^t$
    Compute optimal weights
6       $\hat{\mathbf{w}} = \boldsymbol{S}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$
    Find $\hat{\theta}$ and $\hat{p}$ that minimize the training error $\varepsilon$.
        $[\hat{\theta}, \hat{p}] = \arg\min \varepsilon$
    where,
        $\varepsilon = \frac{1}{2}\sum_{i=1}^{n} z^{(i)} |h(\mathbf{x}^{(i)}) - y^{(i)}|$
7 end
  Output: Trained weak classifier: $h \to \mathbf{w} = \hat{\mathbf{w}}$, $h \to \theta = \hat{\theta}$, $h \to p = \hat{p}$
  Output: Error of the weak classifier: $\varepsilon$
```

4. Boosting weak classifiers

Boosting is a meta algorithm that refers to a method of producing a "strong" classifier by additively combining a set of weak classifiers. The predictions from many weak classifiers are combined through weighted majority voting to produce the prediction of the strong classifier. For binary classification problems, the strong classifier has the form

$$H(\mathbf{x}) = \begin{cases} 1 \text{ object}, & \sum_{i=1}^{k} \alpha^{(i)} h^{(i)}(\mathbf{x}) \geq \Theta \\ 0 \text{ clutter} & \text{otherwise} \end{cases} \tag{4}$$

In (4), $h^{(i)},\ i=1,\dots,k$ are the k weak classifiers selected by boosting and $\alpha^{(i)},\ i=1,\dots,k$ are their corresponding weights. The *AdaBoost* [21,22], one of the popular boosting procedures, has been used in our implementation. The pseudocode of *AdaBoost* adapted to the object detection problem is shown in Algorithm 4.

The algorithm takes as input the object class training images $\mathbf{x}^{(i)},\ i=1,\ldots,n$, their labels $y^{(i)},\ i=1,\ldots,n$ and a set of weak classifiers $h^{(j)},\ j=1,\ldots,m$. The algorithm goes through a series of N iterations, and in each iteration a weak classifier is selected. In the first iteration, all the weak classifiers are trained according to Algorithm 2 or 3 and the weak classifier that produces the lowest error is selected. In each subsequent iteration, a weak classifier is selected such that the combined error of all the previously selected weak classifiers is minimized. This is performed by maintaining weights $z^{(i)},\ i=1,\ldots,n$ on the training images. The higher the magnitude of $z^{(i)}$, the greater is the need for $\mathbf{x}^{(i)}$ to be classified correctly. Initially, all the training images belonging to a class are assigned equal weights $z^{(i)}$. In each iteration, the value of $z^{(i)}$ is increased if $\mathbf{x}^{(i)}$ was misclassified by the selected weak classifier and decreased if $\mathbf{x}^{(i)}$ was classified correctly. This compels the algorithm to select a weak classifier that classifies the training samples with higher weights correctly in the subsequent iteration.

Once the weak classifiers and their corresponding weights determined through the boosting procedure, the threshold value of the strong classifier (Θ) is set such that the strong classifier has a maximum false rejection rate, β, on a validation set of object class images. The threshold value Θ is found such that the sum of weighted decisions of the selected weak classifiers is greater than Θ for at least $(1-\beta)q$ images in the validation set. Here, q is the total number of images in the validation set.

Algorithm 4. Boosting weak classifiers.

```
  Input: Training images: $\mathbf{x}^{(i)},\ i \in \{1,\ldots,n\}$
  Input: Training labels: $y^{(i)} \in \{-1,1\},\ i \in \{1,\ldots,n\}$
  Input: Validation object class images: $\mathbf{v}^{(i)},\ i \in \{1,\ldots,q\}$
  Input: Maximum false rejection rate: $\beta$
  Input: Set of weak classifiers: $\mathbf{h} = \{h^{(j)}\},\ j \in \{1,\ldots,m\}$
  Input: Number of weak classifiers per node: $N$
1 begin
    Set weights to each training image :
        $z_1^{(i)} = \begin{cases} \frac{0.5}{n_o}, & \mathbf{x}^{(i)} \in \text{object} \\ \frac{0.5}{n_c}, & \mathbf{x}^{(i)} \in \text{clutter} \end{cases}$
2   where, $n_o$ and $n_c$ are the number of object and clutter images,
    crrespectively.
    for $t=1$ to $N$ do
3       Apply Algorithm 1, 2 or 3 to train each weak classifier in $\mathbf{h}$
4       crand find the weak classifier, $h_t$, with minimum error, $\varepsilon_t$.
5       Compute the new weight for $h_t$ :
6           $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$
        Update the weights for each training image :
7           $z_{t+1}^{(i)} = \frac{z_t^{(i)} e^{-\alpha_t h_t(\mathbf{x}^{(i)})}}{Z}$
        where $Z$ is a normalization factor such that $z_{t+1}^{(i)}$ will be a
        crdistribution.
    Evaluate the selected weak classifiers on the validation set
    crand find $\Theta$ such that $\sum_{t=1}^{N}\alpha_t h_t(\mathbf{v}^{(i)}) \geq \Theta$ is true for $(1-\beta)q$
    crvalidation set images.
8 end
  Output: Strong classifier, $H(\mathbf{p})$, where $\mathbf{p}$ is a test image:
        $H(\mathbf{p}) = \begin{cases} 1 \text{ object}, & \sum_{t=1}^{N}\alpha_t h_t(\mathbf{p}) \geq \Theta \\ 0 \text{ clutter} & \text{otherwise} \end{cases}$
```

5. Training the object detector

This section describes how object detection systems can be built using boosted set of weak classifiers.

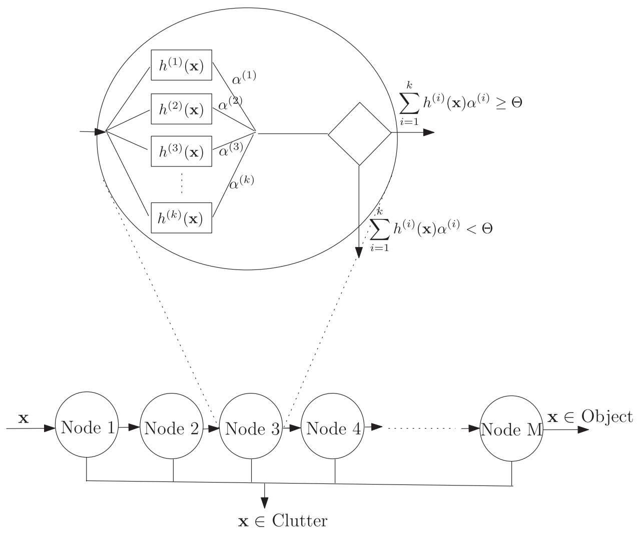


Fig. 6. A cascaded classifier structure consists of multiple nodes arranged in a degenerated decision tree fashion. The output of the node is the weighted majority of decisions of the individual classifiers present at that node. If an image is classified as clutter by a node, then that image is not processed in the successive nodes.

5.1. Framework of the object detection system

The framework of the proposed object detector is based on Baker et al.'s rejection cascade architecture [2,3]. A rejection cascade, as shown in Fig. 6, consists of serially connected nodes which label a test image as object or clutter. Each node contains a boosted set of weak classifiers. In Fig. 6, the *Node 3* of the rejection cascade has been expanded to show the k weak classifiers present in it.

The object detection proceeds as follows: a given test image is scanned at all positions and scales by the rejection cascade. When an image sub-region $\mathbf{x}$ is input to a node, it is classified by all the weak classifiers present in the node, and the weighted average of their decisions is output as the final decision of that node. Thus, each node labels an image sub-region as object or clutter. An image sub-region is labeled object only if it is labeled object by all the nodes of the cascade. On the other hand, if a sub-region is labeled clutter by a node, it is not further processed by any successive node. Thus, the rejection cascade tries to reject clutter as early as possible. Since a majority of image sub-regions belong to clutter [17,29], an object detection system based on rejection cascade architecture will be fast in scanning the entire test image.

The rejection cascade $H(\mathbf{x})$ can be expressed as

$$H(\mathbf{x}) = \begin{cases} 1 \text{ object}, & \prod_{i=1}^{M} H^{(i)}(\mathbf{x}) = 1 \\ 0 \text{ clutter} & \text{otherwise} \end{cases}$$

where $H^{(i)}(\mathbf{x})$ is a boosted set of weak classifiers as in (4), and M is the number of nodes.

5.2. Building the rejection cascade

The rejection cascade is built according to Algorithm 5. The algorithm takes in a database of object and clutter class images $\mathbf{x}^{(i)},\ i=1,\ldots,n$, their labels $y^{(i)},\ i=1,\ldots,n$ and a set of weak classifiers $h^{(j)},\ j=1,\ldots,m$. Given the training data, the algorithm iterates M times to construct the rejection cascade, where M is the desired number of nodes. In each iteration, a node $H^{(u)}(\mathbf{x})$ is built according to Algorithm 4. The clutter training images that were correctly classified by the current node are not considered to build the next node. This ensures that the next node to be built concentrates on rejecting the clutter images that were misclassified by the previously built nodes.

Algorithm 5. Building a rejection cascade of strong classifiers.

```
  Input: Desired number of nodes in the cascade: $M$
  Input: Training images: $\mathbf{x}^{(i)},\ i \in \{1,\dots,n\}$
  Input: Training labels: $y^{(i)} \in \{-1,1\},\ i \in \{1,\dots,n\}$
  Input: Set of weak classifiers: $\mathbf{h} = \{h^{(j)}\},\ j \in \{1,\dots,m\}$
1 begin
    for $u = 1$ to $M$ do
2       Build node $H^{(u)}$ according to Algorithm 4.
3       Remove $\mathbf{x}^{(i)}$ if $H^{(u)}(\mathbf{x}^{(i)}) = 0$ and $y^{(i)} = -1$.
4       Test the cascade with $u$ nodes on a test set of clutter images,
5       and add the sub-windows that were misclassified to the
        training set.
6 end
  Output: Rejection cascade, $H(\mathbf{t})$, of strong classifiers where
  $\mathbf{t}$ is a test image:
```

$$H(\mathbf{t}) = \begin{cases} 1 \text{ object}, & \prod_{i=1}^{M} H^{(i)}(\mathbf{t}) = 1 \\ 0 \text{ clutter} & \text{otherwise} \end{cases}$$

6. Experimental setup and results

The proposed features were trained for two object detection problems: detection of human frontal faces from digital photographs and detection of heart regions in short-axis cardiac magnetic resonance images. Detecting human frontal faces and heart regions is challenging as they exhibit high degree of intra-class appearance changes. Further, these objects are ideal for our study because applications involving detection of frontal faces and hearts benefit from real-time localization. Face detection is the first step in many computer vision systems and its output in real-time is indispensable for security systems or for human–computer interaction systems. Heart detection can be used as a precursor to complex heart segmentation techniques which in turn may help detecting any abnormality in the cardiac function. Since an MRI scan of heart produces a sequence of 3D images, and each image typically contains dozens of slices, heart regions in lots of images need to be detected and segmented before the pathology could be understood. Therefore, quick detection and segmentation algorithms would help in reducing the diagnosis time.

In the following experiments, we compare the speed and the accuracy of the object detectors constructed with the proposed features (optimal weights) over the traditional ones (default weights). Before the results are presented, the databases on which the object detectors were trained and tested are described.

6.1. The training datasets

To train the frontal face and heart detectors, two object databases (face and heart), and a clutter database were used. The frontal face database was composed of 5000 images. The images in this database were collected from publicly available facial image databases such as MIT–CBCL face database no. 1 [1], XM2VTS [15], Equinox [24], AR [14], and BioID [9]. The facial regions were cropped manually and resized to images of size 20×20 pixels.

The heart database consisted of 493 short-axis MR heart images. Similar to the face dataset, the heart regions were manually cropped and resized to images of size 20×20 pixels. To train the heart detector, 300 of the 493 images were used.

For the clutter image database, a total of 27,000 images were downloaded from the internet. These images did not contain either frontal face nor heart regions. For our experiments we generated approximately 4×10^9 sub-regions from these images.

6.2. The test datasets

The face detectors were tested on the MIT + CMU frontal face database (test sets A–C) [20]. This database consists of 130 images with 511 frontal faces. Two of the faces in this test set had to be flipped as they were initially upside-down. The remaining images were left untouched. Ground truth face location is available for every face in the database.

The heart detectors were tested on a set of 193 images. In all these images the heart regions were manually landmarked to obtain ground truth size and location.

In our experiments, we assumed that a face or a heart has been detected correctly if a positively detected sub-region satisfies the following two conditions:

(1) The size of the detected region is within $\pm 10\%$ of the size of the annotated face/heart.
(2) The distance between the center of the detected region and the annotated face/heart is not more than 10% of the size of the annotated face/heart.

These two conditions ensure that a positively detected sub-region of a test image contains a face/heart.

6.3. Limiting the number of Haar-like features

The number of ways in which rectangles can be arranged in a template to form Haar-like features is (almost) infinite [13]. Therefore, for practical reasons, all implementations of object detection systems that use Haar-like features need to limit the number of features that are used for training. In order to limit the number of Haar-like features, the following measures were taken:

(1) Only Haar-like features with two, three and four rectangles were considered.
(2) The template size of the Haar-like features was set 20×20 pixels.
(3) Lienhart and Maydt's [13] rotated rectangle features (see Fig. 1(d), (e) and (f)) were not considered.
(4) The inter-rectangle distances dx and dy for Li et al.'s [12] disjoint features as shown in Fig. 1(g) were discretized so that they were integer multiples of 100% of the rectangle size in corresponding directions.
(5) All the rectangles contributing to a single Haar-like feature had the same size.
(6) Rectangles with size less than 3×3 pixels were not allowed to form Haar-like features.

A total of 207,807 Haar-like features remained after removing the features that did not satisfy the above-mentioned conditions. Despite this reduction, we train more number of features than Viola and Jones [27], who used a total of 160,000 features.

6.4. Training object detectors

For every object detection problem (face and heart), four detectors were constructed. These four detectors basically differed on the type of features used to construct them. The features used were:

- Default weights: Traditional Haar-like features whose rectangles are assigned default weights.
- Optimal weights (BFS): Proposed features with optimal rectangle weights computed using BFS.
- Optimal weights (FLDA): Proposed features with optimal rectangle weights computed using FLDA.

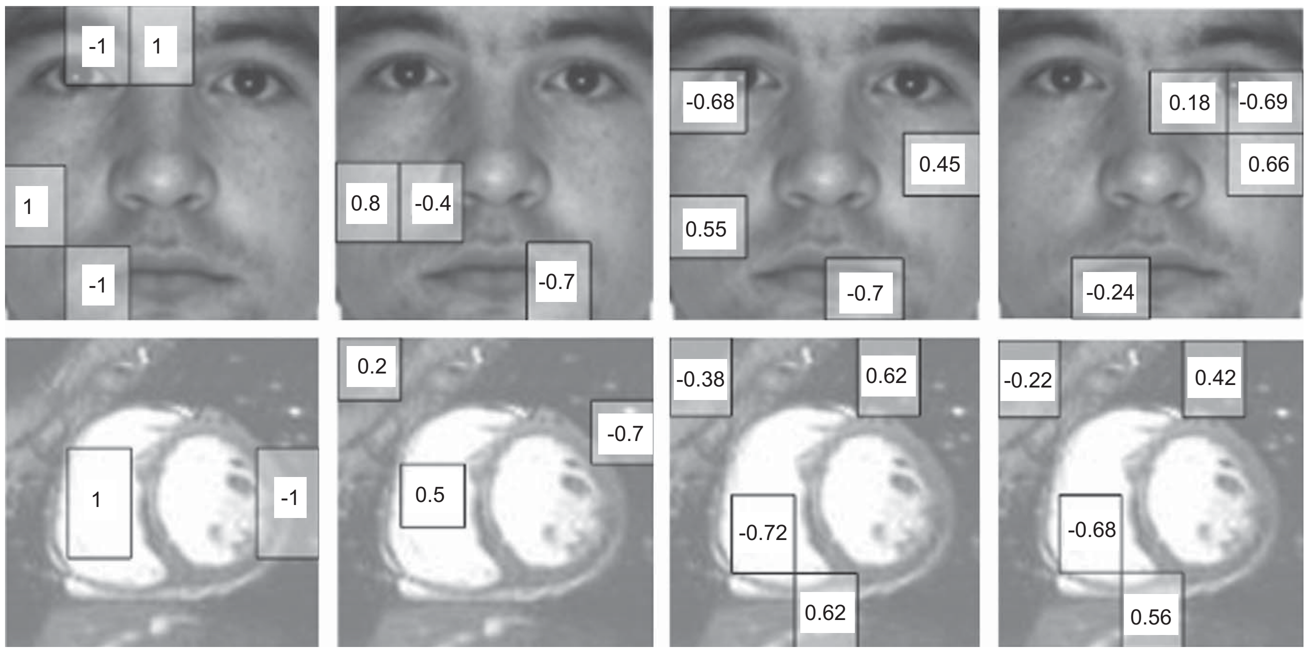


Fig. 7. Illustration of the Haar-like features, superimposed on a typical training images, that were selected in the first iteration of the AdaBoost procedure for different object detectors. The columns, from left to right, show features selected using different rectangle weighting strategies: default weights, optimal weights (BFS), optimal weights (GA) and optimal weights (FLDA), respectively.

- Optimal weights (GA): Proposed features with optimal weights computed using GA.

Each face detector had 36 nodes with a total of 1832 weak-classifiers distributed among them. The heart detectors had 16 nodes with 172 weak-classifiers. Corresponding nodes in every object detector had the same number of weak classifiers. The first ten nodes of the face detector and the heart detector were assigned only one weak classifier in order to reject as many clutter regions as possible with minimum computational effort. Every node of the rejection cascade was trained so that their false rejection rate on a database of object class validation set is 0.01 or lower. The Haar-like features that were picked in the first iteration of the AdaBoost procedure for different object detectors are illustrated in Fig. 7. The weight values that produced the minimum training error, which are superimposed on the rectangles, seem to be different from the default weights.

To compute optimal weights using BFS, the weight values were restricted to the range $[-1, 1]$ and quantized with a step size of 0.1. The GA parameters were set to the following values: population size =30, number of generations =100, crossover probability =0.8, and mutation probability =0.1. In our implementation, single point crossover was used, and mutation was effected by changing a randomly selected element of the genome with a random value selected in the interval $[-10\ 10]$.

The time required to training the face and heart detectors on a 2.33 GHz Intel Xeon Dual Core “woodcrest” processor with 8 GB RAM is shown in Table 1. Viola and Jones [27] report that the time required to train a 38-node detector was in the order of weeks in a 466 MHz AlphaStation XP900.

Table 1
Comparison of training time on a 2.33 GHz Intel Xeon Dual Core processor with 8 GB RAM.

Feature type	Face	Heart
Default weights	~ 2 days	~ 5 h
Optimal weights (BFS)	~ 22 days	~ 2 days
Optimal weights (GA)	~ 20 days	~ 2 days
Optimal weights (FLDA)	~ 10 days	~ 8 h

6.5. Comparison of accuracies of object detectors

The face detectors were tested on the MIT + CMU frontal face dataset, and Fig. 8 shows the obtained receiver operating characteristics (ROC) curves. The ROC curves were generated as described in [27] by adjusting the threshold (Θ) of the last two nodes of the object detector from $-\infty$ to $+\infty$. Note that in Fig. 8, the true positive rate is plotted against the number of false detections. The false positive rate can be obtained by dividing the number of false detections by 12, 378, 518, which is the total number of searched sub-windows in the MIT + CMU database.

Face detection results on MIT + CMU database from five other popular face detection systems are superimposed in Fig. 8. Rowley–Baluja–Kanade [20] used an ensemble of neural networks to learn face and clutter patterns. Their results are based on evaluating their detector on 130 images from MIT + CMU database. Roth–Yang–Ahuja [19] evaluated their SNoW based detector on all the images in MIT+CMU dataset except those containing line-drawn faces. Schneiderman–Kanade [23] proposed a probabilistic method to detect faces based on local appearance and principal component analysis. They report their results based on evaluating their detector on 125 images from MIT + CMU database.

Viola and Jones [27] report their results on evaluating their detector on 130 images from MIT + CMU database. Their detector had a cascade structure with 38 nodes, which in turn was built using Haar-like features with default weights. They also reported a modest improvement in detection results by using a simple majority voting scheme when each test image was evaluated using three differently built detectors. Note that the results reported under default weights refer to results from our own implementation of the Viola and Jones’s object detection procedure. We believe that the ROC curves obtained for default weights and those reported in [27] differ because of different image databases and different feature pool

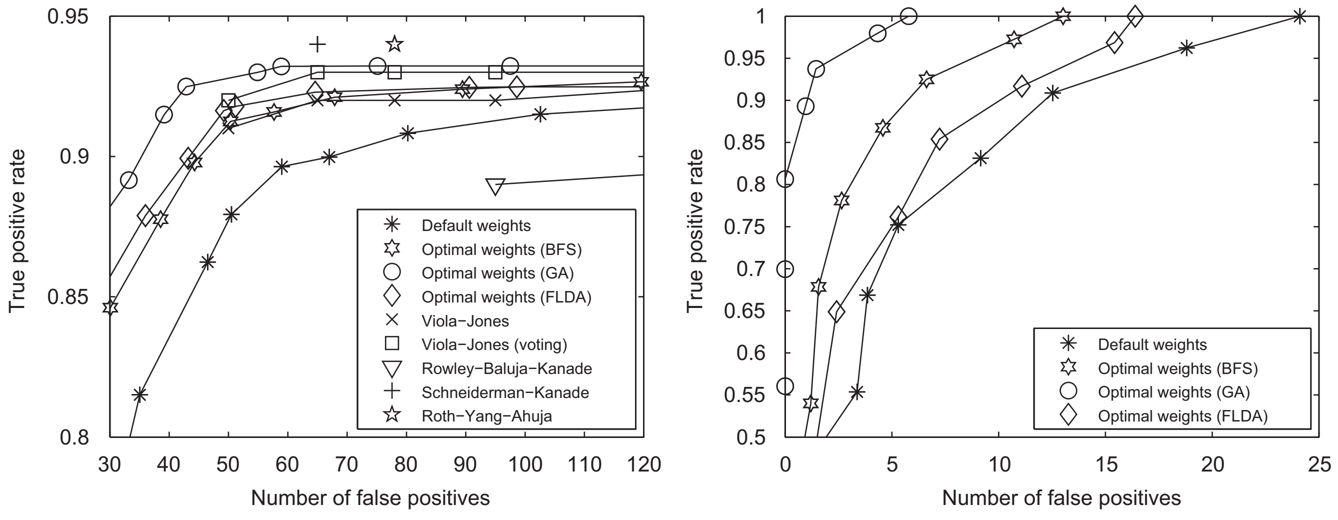


Fig. 8. Comparison of ROC curves for various face (left) and heart (right) detectors.

used in the training process. The image databases used in [27] are not publicly available or accessible in any way. Also, parameters such as how small or how big the rectangles forming the Haar-like features should be are not specified, whichmakes it difficult to recreate the feature pool exactly.

Fig. 8 shows the ROC curves obtained for the heart detector. The false positive rates for the heart detector can be obtained by dividing the number of false detections by 9, 842, 870, which is the number of searched sub-windows. As observed in the results from the face detector and the heart detectors constructed with optimal weights (BFS, GA and FLDA) were more accurate than that constructed with default weights. The only difference among the detectors was the strategy used to allocate the weights to the rectangles of Haar-like feature. All the other training parameters remained the same. Therefore, it can be concluded that the increase in accuracy was obtained by using the optimal weights alone.

For face and heart detection problems, it can be observed that the FLDA performs poorer than GA. We believe that there are two factors that contribute to this sub-optimal performance:

(1) The non-Gaussian nature of the object and clutter cluster of points in the SRFS (see Section 3.3).
(2) When FLDA is used, optimization of the three parameters $\hat{w}$, θ and p occur sequentially. That is, $\hat{w}$ is optimized first, then optimal values of θ and p are found for this particular combination of weights. This is sub-optimal in comparison to GA, where all three parameters are optimized simultaneously.

The accuracy of object detectors built with BFS are worse than those built with GA. However, it is expected that the performance of BFS would tend towards that of GA if the search range is increased and the quantization step is decreased. The values for the range ($[-1, 1]$) and quantization step (0.1) were chosen such that the time required for training object detectors using BFS and GA are similar.

The output of the face detector over various test images in MIT+CMU database are shown in Fig. 9. The face detector tolerates approximately 15° in-plane and out-of-plane rotations of face, and it is also tolerant to the race of the person. Fig. 10 shows the heart detection output on various test images. False positives occur when the underlying image is similar to that of the object of interest. For example, for the image on the top-left in Fig. 9, some of the collar regions have been labeled positive as they resemble a face. False negatives might have occurred due to extreme lighting conditions, poor image quality or due to insufficient resolution during search.

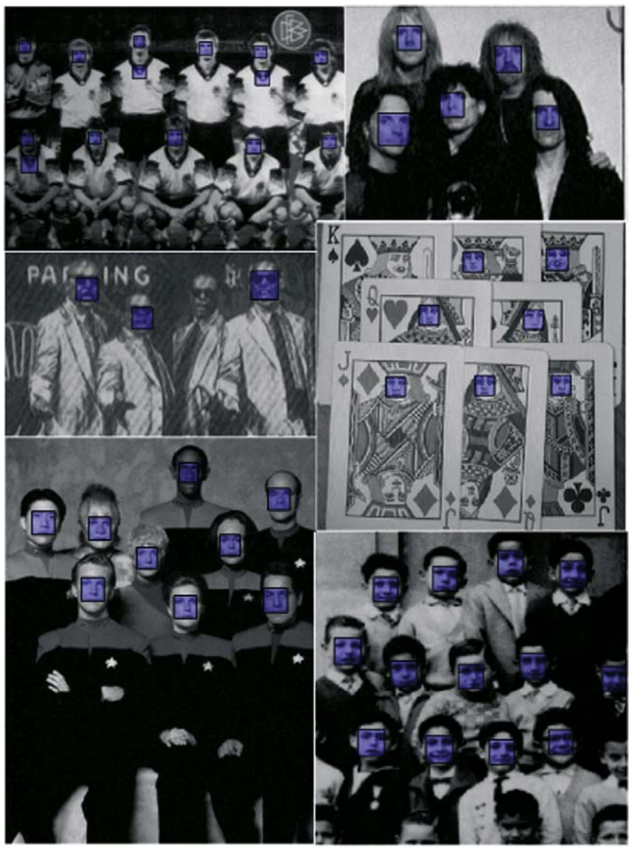


Fig. 9. Face detector output on various test images.

6.6. Comparison of speed of object detectors

As discussed in Section 5.1, the speed of an object detector is proportional to its efficiency in rejecting clutter images. The efficiency in rejecting clutter can be expressed as average number of nodes ($\bar{n}$)

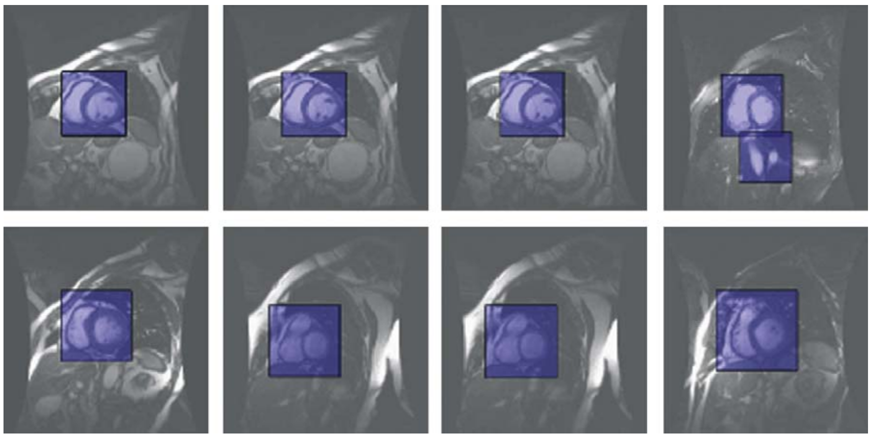
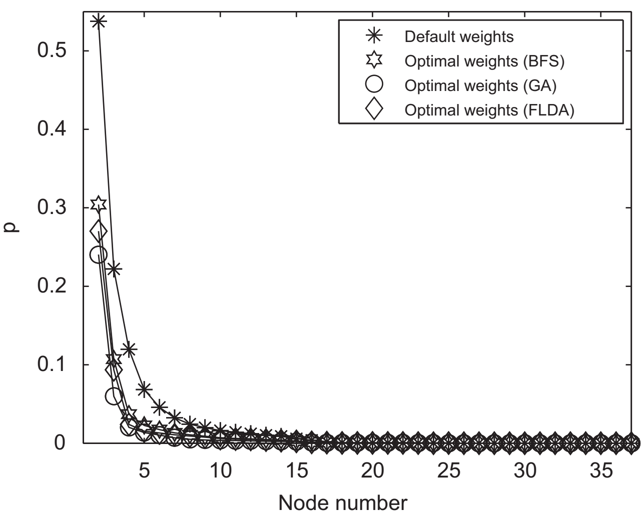


Fig. 10. Heart detector output on various test images.



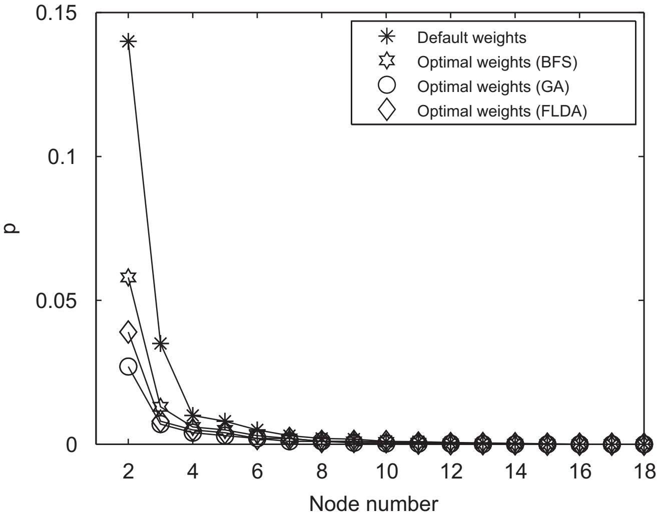


Fig. 11. False positive rate at a given node of the face (left) and the heart (right) detectors.

and the average number of weak classifiers ($\bar{c}$) required to reject a clutter image. The quantities $\bar{n}$ and $\bar{c}$ are computed as shown in the following equation, respectively:

$$\bar{n} = \sum_{i=1}^{n} p^{(i)} \text{ nodes} \tag{5}$$

$$\bar{c} = \sum_{i=1}^{n} c^{(i)} \cdot p^{(i)} \text{ weak classifiers} \tag{6}$$

Here, $p^{(i)}$ is the probability that a clutter image is evaluated by the ith node of rejection cascade. In (6), $c^{(i)}$ represents the number of weak classifiers in the ith node. An estimate of $p^{(i)}$ can be obtained as in (7) by testing the rejection cascade with N number of clutter images, and recording the number of clutter images evaluated ($e^{(i)}$) by the ith node:

$$p^{(i)} = \frac{e^{(i)}}{N} \tag{7}$$

Note that $p^{(1)} = 1$ because all the clutter images will be evaluated by the first node. Fig. 11 shows the estimated value of $p^{(i)}$ at each node of the rejection cascade for the face and heart detectors. They were obtained by testing the rejection cascades with 10,000 randomly selected clutter images. It can be noticed that the probability that a clutter image is evaluated in a node is consistently lower for the object detectors built with the proposed features than the traditional ones. This means that the object detectors built with the proposed features will better in rejecting clutter images and therefore, they will be faster in scanning through a test image. Table 2 tabulates $\bar{n}$ and $\bar{c}$ values for different rejection cascades. It can be observed that the rejection cascades built with the proposed features require less number of nodes and weak classifiers to reject an average clutter image than those built with traditional features.

The more number of weak classifiers ($\bar{c}$) required by the face detector to reject a clutter image on the average suggests that

Table 2
Comparison of $\bar{n}$, $\bar{c}$ and speed for different object detectors.

Weak classifier type	$\bar{n}$		$\bar{c}$		fps	
	Face	Heart	Face	Heart	Face	Heart
Default weights	2.14	1.22	5.06	1.74	~ 12	~ 15
Optimal weights (BFS)	1.55	1.09	3.01	1.37	~ 15	~ 20
Optimal weights (FLDA)	1.47	1.07	2.74	1.37	~ 17	~ 20
Optimal weights (GA)	1.38	1.05	2.35	1.23	~ 19	~ 20

weak-classifiers selected for face detectors are less powerful than those selected for the heart detector. This, in turn, suggests that faces are more difficult class to model using Haar-like features than hearts.

The average speed, in frames per second (fps), of object detectors (when tested on images that have one instance of object) is tabulated in Table 2. The face detectors were tested on images (of resolution 320×240 images) proceeding from a camera installed in a office scenario. On each frame, 8858 regions were searched for the presence of a face. The heart detectors were tested on 640×480 images, and on each image, 42,145 regions were searched. The reason why approximate times are reported in Table 2 is that the speed of the object detectors is not constant and it is dependent on the number of sub-regions that resemble an object. Sub-regions that resemble an object have high chances of being processed by most of the nodes of the detector and, therefore, they require more time to be processed. The speed of the object detectors increased approximately 1.5 times when the test images were uniformly white and contained no instances of object.

From (1), it can be noticed that the proposed features require additional floating point multiplications which make them slower to evaluate. However, since they reject more clutter than the traditional features, object detectors built with the proposed features are faster.

On a 700 MHz Pentium III processor, Viola and Jones [27] report that their implementation of the face detector (using Haar-like features with default weights) processes 384×288 image at the rate of 15 fps. Rowley–Baluja–Kanade [20] and Roth–Yang–Ahuja [19] do not specify the speed of their detector, however, Viola and Jones [27] independently tested Rowley–Baluja–Kanade detector and concluded that it is about 15 times slower than theirs. Schneiderman and Kanade [23] report that their detector can evaluate a 240×256 image in about 90 s on average using a Pentium II 450 MHz processor.

7. Conclusions

In this paper, we propose assigning optimal weights to the rectangles of the Haar-like features so that the weak classifiers constructed based on them give best possible classification performance. The optimal weights were computed in a supervised manner using three different techniques. (1) brute-force search, (2) genetic algorithms and (3) Fisher's linear discriminant analysis. This paper presents detailed experiments on two difficult object detection problems (frontal faces and human hearts), and our results indicate that by using the proposed features, better object detectors, both in terms of accuracy and speed of detection, can be constructed.

Acknowledgments

The authors would like to thank CETIR-Sant Jordi Cardiovascular Centre, Barcelona, Spain for providing the short-axis MR images. Special thanks to Sebastián Ordás and Löic Boisrobert for landmarking the heart regions. This work was partially funded by Grants TIC2002-04495-C02 and TEC2006-03617/TCM, from the Spanish Ministry of Education and Science, and Grants FIT-360000-2006-55 and FIT-360005-2007-9 from the Spanish Ministry of Industry. The Universitat Pompeu Fabra is a member of the Biosecure (IST-2002-507534) European Network of Excellence.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version of 10.1016/j.patcog.2009.05.011.

References

[1] CBCL Face Database # 1, MIT Center for Biological and Computation Learning ⟨http://www.ai.mit.edu/projects/cbcl⟩.
[2] S. Baker, Design and evaluation of feature detectors, Ph.D. Thesis, Graduate School of Arts and Sciences, Columbia University, September 1998.
[3] S. Baker, S.K. Nayar, Pattern rejection, in: CVPR '96: Proceedings of the Conference on Computer Vision and Pattern Recognition, June 1996, pp. 544–549.
[4] T. Cooke, Two variations on Fisher's linear discriminant for pattern recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2) (2002) 268–273.
[5] D. Delgado, L.H. Clemmensen, B. Ersboll, J.M. Carstensen, Precise acquisition and unsupervised segmentation of multi-spectral images, Computer Vision and Image Understanding 106 (2–3) (2007) 183–193.
[6] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley, New York, 2000.
[7] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[8] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, USA, 1992.
[9] O. Jesorsky, K.J. Kirchberg, R. Frischholz, Robust face detection using the Hausdorff distance, in: AVBPA '01: Proceedings of the International Conference on Audio- and Video-Based Biometric Person Authentication, Lecture Notes in Computer Sciences, vol. 2091, London, UK, 2001, pp. 90–95.
[10] M. Jones, P. Viola, Fast multi-view face detection, Technical Report MERL-TR2003-96, Mitsubishi Electric Research Laboratories, July 2003.
[11] K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Ph.D. Thesis, University of Michigan, 1975.
[12] S.Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, H. Shum, Statistical learning of multi-view face detection, in: ECCV '02: Proceedings of the European Conference on Computer Vision, Lecture Notes in Computer Sciences, vol. 2353, London, UK, 2002, pp. 67–81.
[13] R. Lienhart, J. Maydt, An extended set of Haar-like features for rapid object detection, in: ICIP '02: Proceedings of the International Conference on Image Processing, 2002, pp. 900–903.
[14] A. Martínez, The AR face database, Technical Report #24, Computer Vision Center, Barcelona, Spain, 1998.
[15] K. Messer, J. Matas, J. Kittler, J. Luettin, G. Maitre, XM2VTSDB: the extended M2VTS database, in: AVBPA '99: Proceedings of the International Conference on Audio and Video-Based Biometric Person Authentication, 1999, pp. 72–77.
[16] T. Mita, T. Kaneko, O. Hori, Joint Haar-like features for face detection, in: ICCV '05: Proceedings of the International Conference on Computer Vision, Washington, DC, USA, 2005, pp. 1619–1626.
[17] D. Mumford, B. Gidas, Stochastic models for generic images, Quarterly of Applied Mathematics LIV (1) (2001) 85–111.
[18] C.P. Papageorgiou, M. Oren, T. Poggio, A general framework for object detection, in: ICCV '98: Proceedings of the International Conference on Computer Vision, Washington, DC, USA, 1998, pp. 555–562.
[19] D. Roth, M. Yang, N. Ahuja, A SNoW-based face detector, in: NIPS '00: Proceedings of the Conference on Advances in Neural Information Processing Systems, 2000, pp. 855–861.
[20] H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1) (1998) 23–38.
[21] R.E. Schapire, A brief introduction to boosting, in: IJCAI '99: Proceedings of the International Joint Conference on Artificial Intelligence, San Francisco, CA, USA, 1999, pp. 1401–1406.
[22] R.E. Schapire, Theoretical views of boosting and applications, in: ALT '99: Proceedings of the International Conference on Algorithmic Learning Theory, London, UK, 1999, pp. 13–25.
[23] H. Schneiderman, A statistical approach to 3D object detection applied to faces and cars, Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, 2000.
[24] A. Selinger, D. Socolinsky, Appearance-based facial recognition using visible and thermal imagery: a comparative study, Technical Report 02-01, Equinox Corporation, March 1999.
[25] J. Šochman, J. Matas, Waldboost—learning for time constrained sequential detection, in: CVPR '05: Proceedings of Conference on Computer Vision and Pattern Recognition, Los Alamitos, USA, vol. 2, June 2005, pp. 150–157.
[26] Z. Sun, G. Bebis, R. Miller, On-road vehicle detection: a review, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (5) (2006) 694–711.
[27] P. Viola, M.J. Jones, Rapid object detection using a boosted cascade of simple features, in: CVPR '01: Proceedings of the Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA, 2001, pp. 511–518.
[28] P. Viola, M.J. Jones, D. Snow, Detecting pedestrians using patterns of motion and appearance, in: ICCV '03: Proceedings of the International Conference on Computer Vision, Washington, DC, USA, 2003, pp. 734–741.
[29] J. Wu, J. Rehg, M. Mullin, Learning a rare event detection cascade by direct feature selection, in: NIPS '04: Proceedings of the Conference on Advances in Neural Information Processing Systems, 2004, pp. 855–861.
[30] D. Zhang, S.Z. Li, D. Gatica-Perez, Real-time face detection using boosting in hierarchical feature spaces, in: ICPR '04: Proceedings of the International Conference on Pattern Recognition, Washington, DC, USA, 2004, pp. 411–414.

About the Author—SRI-KAUSHIK PAVANI is a Ph.D. candidate in Computer Science and Visual Communication in Universitat Pompeu Fabra, Barcelona. He received the M.S. degree in Electrical Engineering from the Texas Tech University in 2003, and the B.E. degree in Electronics and Communication Engineering from the Shanmugha College of Engineering, India in 2001. His current work is mainly focused on the development and implementation of object detection systems for access control and monitoring in intelligent environments. His research interests include statistical learning, pattern recognition, and mathematical modeling.

About the Author—DAVID DELGADO GÓMEZ was born in Madrid, Spain, on April 24, 1976. He received the M.Sc. degree in Mathematics from the Autonomous University of Madrid in 1999. In 2000, he worked as research assistant in the Knowledge Engineering Institute in Madrid at the same time he conducted the first two years of his Ph.D. studies in the Computer Science faculty of the Autonomous University of Madrid. In 2005, he completed the Ph.D. studies in the Department of Mathematical Modelling of the Technical University of Denmark. At present, he is working as researcher in the Computational Imaging Lab in the Pompeu Fabra University. His current research interests include machine learning, pattern recognition and statistical methods for image analysis in biometrics.

About the Author—ALEJANDRO FRANGI obtained his undergraduate degree in Telecommunications Engineering from the Technical University of Catalonia (Barcelona) in 1996. He pursued his PhD at the Image Sciences Institute of the University Medical Center Utrecht on model-based cardiovascular image analysis. During this period he was visiting researcher at the Imperial College in London, UK, and in Philips Medical Systems BV, The Netherlands. Dr. Frangi is Associate Professor at Universitat Pompeu Fabra (UPF) and ICREA-Academia Researcher. He currently leads the Center for Computational Imaging & Simulation Technologies in Biomedicine at the UPF composed of 45 members. He is Senior Member of IEEE and Associate Editor of IEEE Trans on Medical Imaging, Medical Image Analysis, the International Journal for Computational Vision and Biomechanics and Recent Patents in Biomedical Engineering journals. He has published over 45 journal papers and over 100 conference articles and book chapters. Dr. Frangi is foreign member of the Review College of the Engineering and Physical Sciences Research Council (EPSRC) in UK, is a recipient of the IEEE Engineering in Medicine and Biology Early Career Award in 2006, the Prizes for Knowledge Transfer (2008) in the Information and Communication Technologies domain and of Teaching Excellence (2008) by the Social Council of the Universitat Pompeu Fabra. Finally, he has recently been awarded an ICREA-Academia Prize by the Institució Catalana de Recerca i Estudis Avançats (ICREA).