IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2019–10–01.
Submitted on 2019–07–18, accepted on 2019–09–27.
ISSN 2105–1232 © 2019 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2019.272

A Contrario Detection of Faces with a Short Cascade of Classifiers

Jose-Luis Lisani, Silvia Ramis

Universitat Illes Balears, Spain
{joseluis.lisani, silvia.ramis}@uib.es

Communicated by Gregory Randall *Demo edited by* Jose-Luis Lisani

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
Jose-Luis Lisani, Silvia Ramis, and Francisco J. Perales "A Contrario Detection of Faces: A Case Example" *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 2091-2118, 2017. https://epubs.siam.org/doi/abs/10.1137/17M1118774

Abstract

The a contrario framework has been successfully used for the detection of lines, contours and other meaningful structures in digital images. In this paper we describe the implementation of an algorithm for face detection published in 2017 by Lisani et al. which applies the a contrario approach to the computation of the detection thresholds of a classical cascade of classifiers. The result is a very short cascade which obtains similar detection rates than a classical (and longer) one at a much lower computational cost.

Source Code

The reviewed and documented source code and an online demo are available at the web page of this article[1]. Compilation and usage instructions are included in the README.txt file of the archive.

Keywords: face detection; a contrario method; Viola-Jones

[1] https://doi.org/10.5201/ipol.2019.272

Jose-Luis Lisani, Silvia Ramis , *A Contrario Detection of Faces with a Short Cascade of Classifiers*, Image Processing On Line, 9 (2019), pp. 269–290. https://doi.org/10.5201/ipol.2019.272

1 Introduction

The *a contrario* framework provides a statistical formulation of a perception principle that states that an observed structure should be considered perceptually meaningful only if it is rarely encountered in random data. This framework has been used successfully to detect contours and lines in images [3, 17, 18], modes in 1D histograms [4, 2], moving objects in video [14], changes in satellite images [13], etc. What we propose in this paper is to apply, for the first time to our knowledge, this approach to the detection of faces.

In 2001 Viola and Jones [16] proposed an algorithm for face detection that settled the basis of most current face detection methods. The algorithm detected faces in frontal position and was based on three elements: fast computation of Haar-like features from the input images; learning of the most discriminant features using the AdaBoost training procedure; and use of a cascade of classifiers to achieve high detection rates with low number of false detections.

Since this seminal work many improvements have been proposed, either for the computation of better image features [11, 9, 15], or using alternative learning techniques [20, 10, 7, 8]. A comprehensive review on face detection methods can be found in [22].

All of the above mentioned improvements have focused on the training step of the algorithm but little attention has been paid to the detection step. Jain et al. in 2011 [6] proposed to adapt the detection thresholds to the image contents, in such a way that *reliable* face detections could be used to detect other *difficult-to-detect* faces in the same scene. In [12] an improvement of the original Viola-Jones method was proposed, focusing in the detection step and using an *a contrario* approach. The authors showed that it is possible to improve the performance of the detector (i.e. increase the detection rates, keeping low the number of false detections and at a reduced computational cost) without the need of a long cascade of classifiers. This was achieved by replacing the fixed detection thresholds of the classifiers, learned in the training step, by adaptive thresholds particular to each input image. We describe in this paper the implementation of the algorithm presented in [12].

The paper is organized as follows: Section 2 describes the classical detection method proposed by Viola and Jones and introduces the basic definitions used throughout the text. Section 3 explains how the a contrario approach can be used to improve the results of a classical cascade. Section 4 contains the pseudo-codes describing the main steps of the proposed algorithm. Section 5 displays some experimental results and, finally, some conclusions are exposed in Section 6.

2 Analysis of the Viola-Jones Face Detector

This section provides a short overview of the Viola and Jones face detection method. We refer the reader to [19] for an in-depth analysis of the method.

A face classifier is a mathematical function that takes as input a portion of an image (typically a rectangular sub-image) and gives as output a numerical value (typically 1 or 0) indicating whether the sub-image contains or not a face.

Viola and Jones [16] defined a series of sub-image features (Haar-like features, see Figure 1) and used a learning set of frontal faces to train, with the Adaboost algorithm, a *strong* classifier that combined K of these features. They defined first a *weak* classifier as a function $h_k(x)$ computed at sub-image x and associated to a feature k. The feature value is obtained as the sum of intensity values in the 'white' feature mask minus the sum of intensity values in the 'black' feature mask. The different types of masks associated to each feature are displayed in Figure 1, and their position and size are chosen so that the set of all features of a given type covers the entire sub-image. The function $h_k(x)$ takes the value 1 if the feature value at x is above/below a learned threshold; otherwise $h_k(x) = 0$.

The *strong* classifier using K features (in opposition to the *weak* classifier that uses a single feature) is defined as

$$h(x) = \begin{cases} 1 & v_{det}(x) \geq T, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

with

$$v_{det}(x) = \sum_{k=1}^{K} \alpha_k h_k(x), \tag{2}$$

and

$$T = \frac{1}{2} \sum_{k=1}^{K} \alpha_k, \tag{3}$$

where x is a sub-image, K is the number of features of the classifier, $h_k(x)$ is the weak classifier associated with feature k and α_k is the *weight* of h_k in the final strong classifier. The detection threshold T is fixed and depends on the α_k values learned from the training set of images.

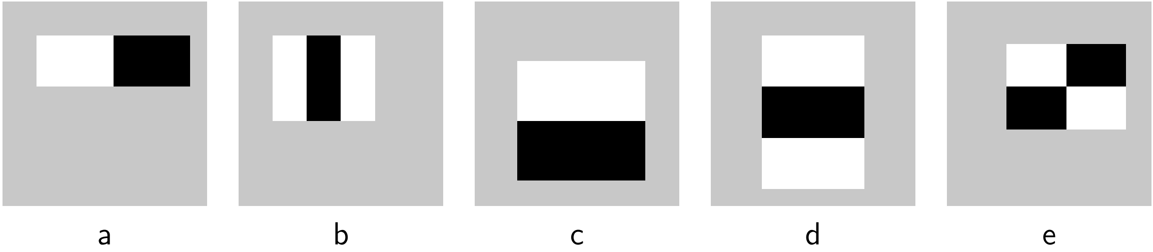


Figure 1: Haar-like feature masks used by the Viola-Jones detection method (figures from [19]).

Usually, the detection value v_{det} of the classifier is disregarded, since we are just interested in its binary response ('1' for faces, '0' for non-faces). In [12] the authors took a different approach. They analyzed the distribution of detection values (the set of detection values associated to all the tested sub-windows in a particular image[2]) for several strong classifiers with different numbers of features. These classifiers had all been trained using the same set of frontal faces and Haar-like features used by Viola and Jones in their original paper [16]. They found that:

1. The distribution of detection values in an image without faces tends to a normal distribution. This is exemplified in Figure 2. This figure displays the distribution of detection values for classifiers with increasing number of features (10, 20, 40, 80 and 200) for two images without faces. The image on the left is a pure Gaussian noise image with standard deviation $\sigma = 30$. The image on the right is a natural image. In both cases we observe that, as the number of features increases, the distribution of detection values tends to a normal distribution.

2. If the image contains faces, the distribution of values is again Gaussian, but the detection values corresponding to the sub-images containing the faces are much higher than the values of the Gaussian distribution. Figure 3 shows an example for an 80-features classifier. In this figure, the value T of the default detection threshold (Equation (3)) of the classifier is shown for reference. The red dots indicate the detection values for the sub-windows actually containing a face. A second example is shown in Figure 4. We observe that, as the number of features in the classifier increases, much farther away are the detection values corresponding to faces from the Gaussian distribution.

[2] In [12] all the sub-windows of sizes ranging from 20×20 to 220×220 pixels were tested. Moreover, flat image regions (sub-windows whose intensity standard deviation was below 20) were not considered.

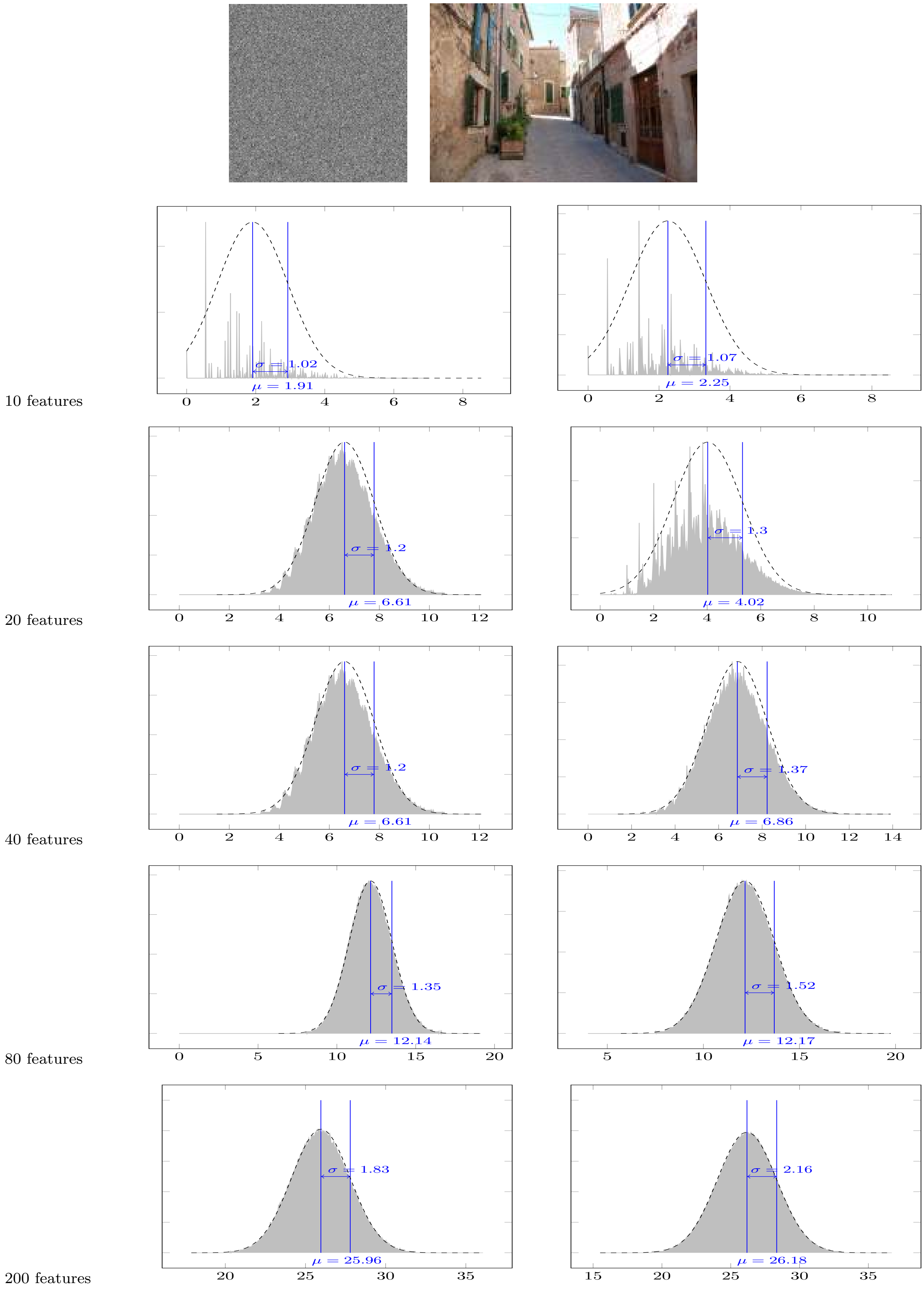


Figure 2: Distribution of detection values for classifiers with increasing number of features. From top to bottom: original image and histograms for classifiers with 10, 20, 40, 80 and 200 features. The mean μ and standard deviation σ of each distribution are shown, and the Gaussian function with the same mean and variance parameters is superimposed. For the left image a total number of 536402 sub-windows were checked by each classifier. For the right image 5170933 sub-windows were checked.

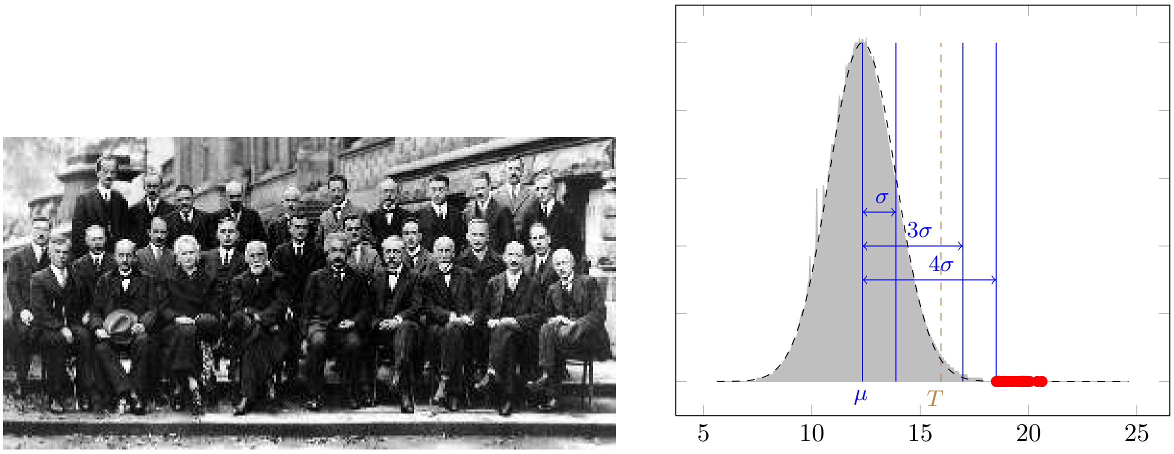


Figure 3: Input image and its histogram of detection values for a 80-features classifier. The red dots indicate the detection values for the sub-windows actually containing a face. T is the default detection threshold of the classifier as defined by Equation (3).

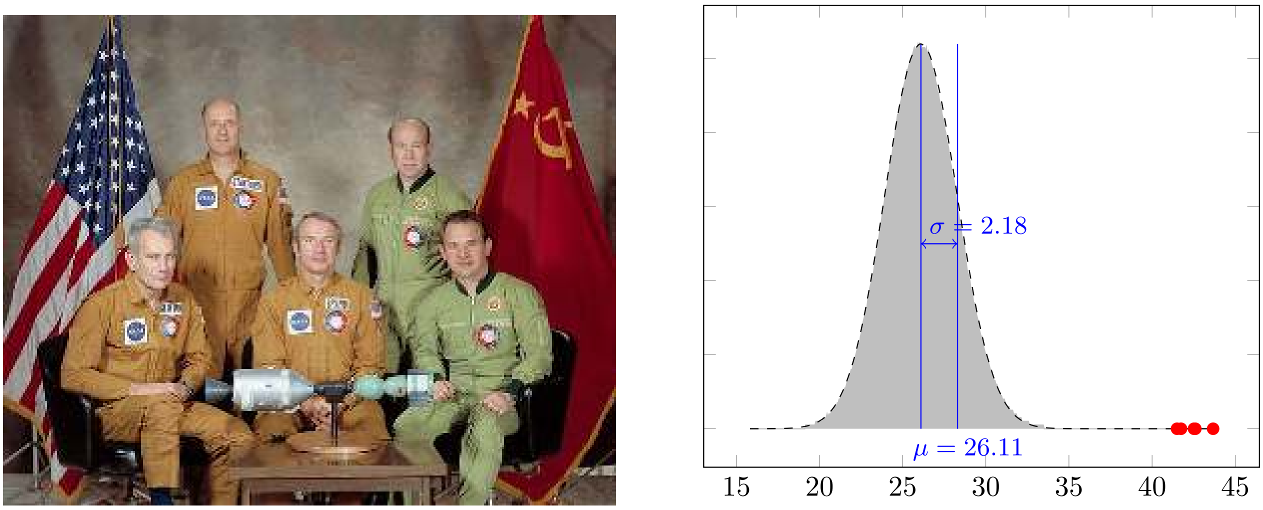


Figure 4: Left, original image. Right, distribution of detection values for a 200-features classifier. The red dots indicate the detection values for the sub-windows actually containing a face. A total number of 3426685 sub-windows were checked by the classifier.

These two observations lead the authors of [12] to devise a method for the automatic estimation of the detection thresholds, which is described in Section 3. For a further analysis of the Gaussianity of the distributions of detection values we refer the reader to the original paper.

3 Setting the Detection Thresholds Using an A Contrario Approach

As mentioned in the previous section, in the original Viola and Jones method the detection threshold determining the presence or absence of a face in a given sub-window is computed using Equation (3). This implies that the threshold is *learned* from the set of training images and that it is *fixed*, i.e. the same threshold will be applied to any image going through the classifier. The example in Figure 3 illustrates one of the shortcomings of this approach. In this figure, the position of the detection threshold is shown with respect to the distribution of detection values of one particular image.

Many of these values are above the threshold, so their corresponding sub-windows shall be (wrongly) detected as faces by the classifier, producing a large number of false positives. Moreover, the distributions in Figures 3 and 4 are clearly different, which indicates that they depend on the image content and that the use of a fixed threshold is too restrictive.

The method in [12] permits to *adapt* the detection threshold to the image being observed. This can be done by applying the *a contrario* detection principle to test the presence of a face in a sub-window against a noise or *a contrario* model where the face is not present. This is equivalent to performing the following hypothesis test:

$\mathcal{H}_0$ (null hypothesis): the sub-image **does not contain** a face
$\mathcal{H}_1$ (alternative hypothesis): the sub-image **contains** a face

The acceptation/rejection of $\mathcal{H}_0$ depends on a rejection threshold Θ and the *level of significance* α of the test is defined as

$$\begin{aligned}\alpha &= P(\text{rejecting } \mathcal{H}_0 | \mathcal{H}_0 \text{ is true}) = \\ &= P(v_{det} > \Theta | \mathcal{H}_0 \text{ is true}) = \\ &= P\left(\begin{array}{c|l} \text{accepting sub-image} & \text{the sub-image} \\ \text{as } \textit{face} & \text{does not contain} \\ & \text{a face} \end{array}\right) = \\ &= P(\text{False positive}),\end{aligned}$$

where v_{det} is the detection value associated to the sub-image, computed from (2). Following the discussion in the previous section, a Gaussian distribution of the detection values is assumed for the null hypothesis (i.e. the distribution of detection values for the non-faces sub-windows is Gaussian). This allows the computation of the level of significance in closed form. The mean μ and standard deviation σ of this Gaussian can be estimated from the empirical values of the histogram. The assumption here is that just a small fraction of the sub-windows in any image correspond, if any, to actual faces. Therefore, the actual distribution of detection values for the whole image corresponds, roughly, to the distribution of values under the null hypothesis.

The rejection threshold Θ can be written as a function of μ and σ: $\Theta = \Theta_s = \mu + s\sigma$, where s is a parameter. Then α can be expressed in terms of s

$$\alpha = P(\text{False positive}) = P(v_{det} > \Theta_s | \mathcal{H}_0 \sim \mathcal{N}(\mu, \sigma^2)) = P\left(\mathcal{Z} > \frac{\Theta_s - \mu}{\sigma}\right) = P(\mathcal{Z} > s), \tag{4}$$

where $\mathcal{N}$ denotes the Gaussian probability density function and $\mathcal{Z} \sim \mathcal{N}(0,1)$ is the standard Gaussian distribution with 0 mean and standard deviation equal to 1[3]. Remark that Θ_s is an *adaptive* threshold, since it depends on the detection statistics (μ and σ) of the input image.

A question that arises is which is the optimum value of the parameter s that guarantees such a low value of probability that no false positives are observed in the image. To answer this question we need first to establish the relation between the probability of false positives and the actual number of observed false positives. This relation is straightforward: if the number of tested sub-windows in the image is N, then the *expected number of false positives*, NFP can be computed as

$$NFP = N \times P(\text{False positive}). \tag{5}$$

[3] The relation between $\mathcal{N}(\mu, \sigma^2)$ and $\mathcal{Z} \sim \mathcal{N}(0,1)$ follows from a classical result on Gaussian distributions.

A criterion for the selection of the detection threshold is to compute the value of Θ_s that guarantees a value of NFP below some pre-defined upper bound $NFP_{\max}$. Combining equations (5) and (4) we obtain the value of the detection threshold as

$$\Theta = \mu + s^*\sigma, \tag{6}$$

with s^* such that $P(\mathcal{Z} > s^*) = \dfrac{NFP_{\max}}{N}$.

4 Description of the Method for Face Detection

Algorithm 1 describes the straightforward application of the conclusions of the previous section to the detection of faces using a single strong classifier. Basically, the algorithm computes the detection values for all the sub-windows of an input image, computes the mean and variance of these values and uses Equation (6) to determine which of these sub-windows contain a face. The only parameter of the algorithm is the allowed number of false positives. The process of extracting all the sub-windows of the image is detailed in Algorithm 3.

Algorithm 1: Face Detection with a Single Classifier and Adaptive Detection Threshold

```
Input      : input image $I$ (gray level), classifier data $\mathcal{C}$
Output     : image sub-windows containing a face $\mathcal{D}$
Parameters: maximum number of false positives $NFP_{\max}$

//Get all image subwindows (Algorithm 3)
$\mathcal{S}$ = GetSubWindows($I$)//Total number of subwindows = $N$
//Apply classifier to ALL subwindows and get detection values $V$
$V(\mathcal{S})$ = ApplyClassifier($\mathcal{S}, \mathcal{C}$)
//Compute mean and standard deviation of detection values
$(\mu, \sigma)$ = GetMeanStandardDev($V(\mathcal{S})$)
//Compute detection threshold (use Equation (6))
$\Theta = \mu + F^{-1}(\frac{NFP_{\max}}{N})\sigma, \qquad F(s) = P(\mathcal{Z} > s)$
//Get set of detected faces $\mathcal{D}$
$\mathcal{D} = \emptyset$
for $s \in \mathcal{S}$ do
    if $V(s) > \Theta$ then
        $s \rightarrow \mathcal{D}$ //Add sub-window to detections set
```

In [16] Viola and Jones proposed to use not just one strong classifier but several of them in a series configuration called **cascade of classifiers**. In this structure, the input of each classifier consisted of the positive detections of the previous one. The first stages of the classifier used few features so they only rejected the most obvious non-faces while the latest stages used many features to let through only the true faces. Moreover, each classifier was trained using the negative examples wrongly labeled by the previous classifier, so the detection thresholds became more and more restrictive. Besides the increase in the precision of the detection, the use of the cascade permits to speed-up the computation process, since most of the sub-windows shall be rejected by the first stages of the cascade, where only just a few features are checked.

In [12] several tests with a 200-features classifier[4] are performed and compared with the results of the cascade of 31 classifiers implemented in [19], and also with the OpenCV implementation of

[4] The classifier is trained using the same set of frontal 24×24 faces as in the original paper by Viola and Jones [16].

the Viola-Jones face detector. The detection rates obtained with the three methods are very similar, which implies that a single classifier endowed with an adaptive threshold performs as well as a full cascade of classifiers.

However, the performance of a face detector cannot be simply measured in terms of its detection rate, but also its computational efficiency must be taken into account. It is clear that a single classifier with 200 features must check ALL these features on all possible image sub-windows in order to produce its result. On the other hand, the use of a cascade of classifiers permits to reject most of the false positives in the early stages, which are composed by a small number of features. Therefore, even if the total number of features in the full cascade is big (up to 6000 features in the 38-stages original Viola-Jones cascade), the average number of features tested per sub-window is relatively small.

In order to preserve the detection rates of the single 200-features classifier at a much lower computational cost the authors of [12] propose a short cascade of 4 classifiers with 5, 10, 80 and 200 features respectively. As with any cascade the goal is to reject most of the sub-windows in the initial stages (in our case the first 3 stages) and then apply the 200-features classifier to a fraction of the sub-windows. The cascade is trained as proposed in the original Viola and Jones paper [16], using the same set of frontal 24 × 24 faces and, as negative examples for each stage, the false positives from the previous stage.

For detection, ideally the threshold adaptation principle exposed in Section 3 should be applied to all 4 classifiers. However, as illustrated in Figure 2, only when the number of features is large enough the Gaussian model for the distribution of detection values applies. Therefore, for the first two stages of the cascade (5 and 10 features respectively) the detection thresholds are set to allow a fixed percentage of sub-windows through the classifier. For the 5-features classifier all the sub-windows whose detection value is below the 80%-percentile are rejected (only the top 20% sub-windows are let through), while for the 10-features classifier all the sub-windows below the 95%-percentile are rejected (only the top 5% sub-windows are let through). For the 80-features classifier the threshold is set using Equation (6) with fixed value of $NFP_{\max}^{80\text{ feat}} = 100$. These values of the parameters are quite permissive, being the goal to preserve as many as possible of the true positives, which should be correctly classified by the last stage of the cascade. The only tunable parameter of the cascade is the maximum number of false positives for the 200-features classifier ($NFP_{\max}^{200\text{ feat}}$) in the last stage.

It must be noted that the practical implementation of the face detector described in the above paragraph requires the computation of the detection thresholds for the 4 proposed classifiers, which implies the knowledge of the distribution of detection values for each classifier. These distributions could be computed using the entire set of possible image sub-windows[5]. However, this would imply to check 295 ($= 5 + 10 + 80 + 200$) features on each sub-window. In order to reduce the number of computations we check all the sub-windows just for the 5-features classifier and then sub-sample the set of sub-windows to get the rest of distributions. If N denotes the total number of sub-windows, we use pN of them to obtain the detection histograms for the 10, 80 and 200-features classifiers, with p a fixed parameter which we have set to $p = 0.01 = 1\%$. Since N is typically of the order of millions the average of the $0.01N$ sampled detection values shall be very close to the population average [1].

Algorithm 2 describes the proposed detection method.

4.1 Computational Cost of the Short Cascade

For the set of fixed parameters described above it is possible to estimate the computational efficiency of the proposed cascade. Let N be the total number of image sub-windows, then the average number

[5] In the current implementation of the algorithm we consider all non-flat sub-windows (variance of intensity values above 20^2) of sizes ranging from 24×24 to 220×220 pixels.

Algorithm 2: Face Detection with a Short Cascade of Classifiers and Adaptive Detection Thresholds

```
Input      : input image $I$ (gray level), classifiers data $\mathcal{C}_5$, $\mathcal{C}_{10}$, $\mathcal{C}_{80}$, $\mathcal{C}_{200}$
Output     : image sub-windows containing a face
Parameters: maximum number of false positives $NFP_{\max}$

//Get all image sub-windows (Algorithm 3)
$\mathcal{S} = \mathrm{GetSubWindows}(I)$//Total number of subwindows = $N$

//Compute detection thresholds

//Apply 5-features classifier to ALL sub-windows and get detection values $V_5$
$V_5(\mathcal{S}) = \mathrm{ApplyClassifier}(\mathcal{S}, \mathcal{C}_5)$
//Compute 80% quantile of detection values for 5-features classifier
$\Theta_5 = \mathrm{GetQuantile}(V_5(\mathcal{S}), 0.8)$ //Detection threshold for 5-features classifier
//Randomly pick $0.01N$ sub-windows from S
$\mathcal{R} = \mathrm{SampleSubWindows}(\mathcal{S}, 0.01N)$
//Apply 10-features classifier to selected sub-windows
$V_{10}(\mathcal{R}) = \mathrm{ApplyClassifier}(\mathcal{R}, \mathcal{C}_{10})$
//Compute 95% quantile of detection values for 10-features classifier
$\Theta_{10} = \mathrm{GetQuantile}(V_{10}(\mathcal{R}), 0.95)$ //Detection threshold for 10-features classifier
//Apply 80-features classifier to selected sub-windows
$V_{80}(\mathcal{R}) = \mathrm{ApplyClassifier}(\mathcal{R}, \mathcal{C}_{80})$
//Compute mean and standard deviation of detection values for 80-features classifier
$(\mu_{80}, \sigma_{80}) = \mathrm{GetMeanStandardDev}(V_{80}(\mathcal{R}))$
//Compute detection threshold for 80-features classifier
$\Theta_{80} = \mu_{80} + F^{-1}(\frac{100}{N})\sigma_{80}, \qquad F(s) = P(\mathcal{Z} > s)$ //Use Equation (6) and $NFP_{\mathtt{max}} = 100$
//Apply 200-features classifier to selected sub-windows
$V_{200}(\mathcal{R}) = \mathrm{ApplyClassifier}(\mathcal{R}, \mathcal{C}_{200})$
//Compute mean and standard deviation of detection values for 200-features
classifier
$(\mu_{200}, \sigma_{200}) = \mathrm{GetMeanStandardDev}(V_{200}(\mathcal{R}))$
//Compute detection threshold for 200-features classifier
$\Theta_{200} = \mu_{200} + F^{-1}(\frac{NFP_{\max}}{N})\sigma_{200}, \qquad F(s) = P(\mathcal{Z} > s)$ //Use Equation (6)

//Get set of detected faces $\mathcal{D}$

$\mathcal{D} = \emptyset$
for $s \in \mathcal{S}$ do
    if $V_5(s) > \Theta_5$ then
        if $V_{10}(s) > \Theta_{10}$ then
            if $V_{80}(s) > \Theta_{80}$ then
                if $V_{200}(s) > \Theta_{200}$ then
                    $s \rightarrow \mathcal{D}$ //Add sub-window to detections set
```

Algorithm 3: Extract sub-windows of size 24 × 24 and normalized intensity variance from image

```
Input      : input image $I$ (gray level)
Output     : set of image sub-windows $\mathcal{S}$ of size 24 × 24 and normalized intensity variance
Parameters: minimum and maximum sizes of faces to be detected (default values:
            sizeMin=24 and sizeMax=220, respectively), minimum intensity variance of the
            sub-windows values (default: $V_{\text{min}} = 20^2$), and normalized variance value
            (default: $V_{\text{norm}} = 50^2$)
//Get set of zoom factors that rescale a sub-window of size $w \times w$ to $24 \times 24$
$\mathcal{Z} = \emptyset$
for $w \in \{sizeMin, \cdots, sizeMax\}$ do
    $\frac{24}{w} \to \mathcal{Z}$ //Add zoom factor to set

//Rescale input image and extract all subwindows of size $24 \times 24$
$\mathcal{S} = \emptyset$ //Set of all sub-windows
for $z \in \mathcal{Z}$ do
    $I_z = \text{RescaleImage(I, z)}$ //Rescale $I$ with scale factor $z$
    $w_z = width(I_z)$ //width of rescaled image
    $h_z = height(I_z)$ //height of rescaled image
    for $j \in \{0, \cdots, h_z - 25\}$ do
        for $i \in \{0, \cdots, w_z - 25\}$ do
            //Extract from $I_z$ sub-window of size $24 \times 24$ with top-left corner at $(i, j)$
            $s = \text{GetSubWindow}(I_z, i, j, 24, 24)$
            //Reject flat sub-windows
            if $Variance(s) > V_{min}$ then
                $\text{NormalizeVariance}(s, V_{\text{norm}})$
                $s \to \mathcal{S}$ //Add sub-window to set
                //For each sub-window, the values of $z$ and $(i, j)$ are stored for further
                processing (see Algorithm 5)
```

of checked features per sub-window is computed as

$$\mathrm{Avg} = \frac{T_f}{N}, \tag{7}$$

where T_f denotes the total number of checked features

$T_f =$	$5N+$	All the sub-windows go through the 5 features classifier
	$(10 + 80 + 200) \cdot 0.01N+$	A subset of 0.01N sub-windows is used to estimate the distribution of values for the 10, 80 and 200-features classifiers
	$10 \cdot 0.2N+$	20% of the sub-windows go through the 10-features classifier
	$80 \cdot 0.05N+$	5% of the sub-windows go through the 80-features classifier (this is an upper bound, since some sub-windows may be rejected by the previous classifier)
	$200 \cdot p'N$	A unknown (but very small) percentage p' of the sub-windows goes through the 200-features classifier

By replacing this expression in Equation (7), and taking into account that the unknown value p' is very small, we get an upper bound for the average number of checked features per image: $\mathrm{Avg} \approx 13.9$. For comparison, in [12] the average number of features checked by the 31-stages cascade of [19] is computed for different datasets, and the result is $\mathrm{Avg}_{31-stages} \approx 60$.

Speeding up the computation. A simple way of increasing the computation speed of the face detector is by replacing line 2 of Algorithm 3 (sub-windows extraction) by this new line:

$$\textbf{for } w \in \{\mathrm{sizeMin}, \mathit{zStep} \cdot \mathrm{sizeMin}, \mathit{zStep}^2 \cdot \mathrm{sizeMin}, \cdots\} \;\; \textbf{and} \;\; w \leq \mathrm{sizeMax} \;\; \textbf{do}$$

where $\mathit{zStep} > 1$ is a new parameter of the method.

With this modification the algorithm doesn't check all the sub-windows ranging from sizeMin × sizeMin to sizeMax × sizeMax pixels, but only a subset of them. As a consequence less computations are needed to get the final result. This is a strategy similar to the one used in [19] for the implementation of the 31-stages cascade. The downside of this modification is that some correct detections may be missed due to the incomplete sampling of the image domain. This may explain some missed detections in the results shown for the 31-stages cascade in the experimental section. Moreover, in our case a second drawback is that less data are used to compute the histograms of detection values, which results in less reliable detection thresholds. In the results shown throughout the paper this speed-up has not been applied, but it has been left as an optional parameter in the on-line demo accompanying the article.

4.2 Post-Processing of the Detection Results

As shown in Figure 6-left the raw output of Algorithm 2 is a set of squares that indicate the position of a detected face. Usually, for each true detection, many squares of similar sizes are found, centered around the detected face and forming a thick frame around it. Moreover, false detections do not usually exhibit such thick frames, meaning that the detection result is not very stable in this region of the image. All face detection methods apply some kind of post-processing to these raw results in order to get just one representative for each group of similar detections. The final output of our detector is obtained after the following post-processing steps:

- First, only *stable* detections are kept. Typically, the stability of a detection is measured in terms of the *thickness* of the group of detections it represents (i.e. the number of detections in the group). This criterion adds an additional parameter to the detector. For instance, in the OpenCV[6] implementation of the Viola-Jones detector, by default, a minimum of three detections are required to keep the group. In [19] a similar criterion is used, but the minimum required number of detections depends on the size of the detection windows. We get rid of this parameter by assessing the stability of the detection in a different way: we take advantage of the lack of left-right symmetry of many of the features of the 200-features strong detector (see Figure 5) and keep only the raw detections testing positive in a *mirror* version (horizontal flip) of the input image. This is illustrated in Figure 6-center. This post-processing, which we call *mirror filter*, is described in Algorithm 5.

- Second, the stable detections from the previous step are simplified according to the following rule: if two square regions intersect, and the intersection is important, we can consider that it is the same face, so we can keep the best scored one and discard the other. In particular, detections d_1 and d_2 are grouped together if

  $$\frac{A(d_1 \cap d_2)}{\min(A(d_1), A(d_2))} > 0.5, \tag{8}$$

  where d_i denotes a square detection (region of the image that tested positive for the detector) and $A()$ is the area operator. For each group of detections only the one with highest detection value is kept. Figure 6-right shows an example of the simplification provided by this grouping principle. Algorithm 6 describes this post-processing step.

The whole detection algorithm, including the post-processing steps, is described in Algorithm 4.

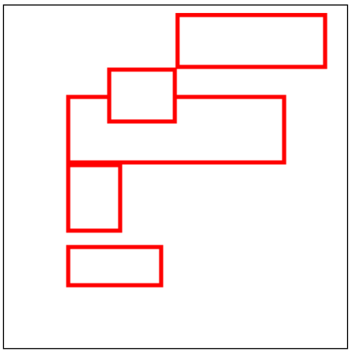


Figure 5: The image displays the location, in a 24×24 image, of 5 of the features with higher weights in the 200-features classifier used in our experiments. Only one of these features exhibits left-right symmetry.

[6] https://opencv.org/

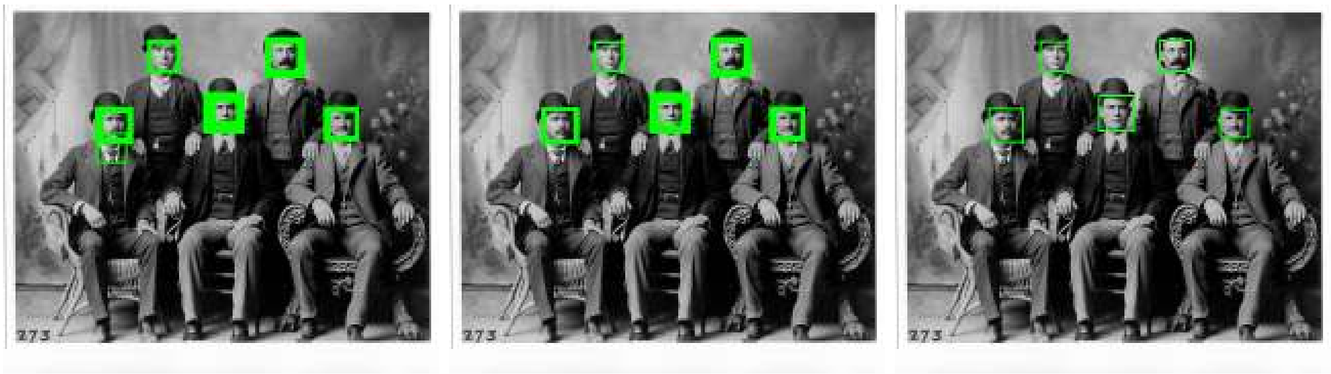


Figure 6: Left, output of the detector described in Algorithm 2 using $NFP_{\mathsf{max}} = 1$. Center, stable detections after applying Algorithm 5. Right, final detections after grouping and simplification with Algorithm 6.

Algorithm 4: Face Detection with a Short Cascade of Classifiers and Adaptive Detection Thresholds with Post-Processing

```
Input      : input image $I$ (gray level), classifiers data $\mathcal{C}_5$, $\mathcal{C}_{10}$, $\mathcal{C}_{80}$, $\mathcal{C}_{200}$
Output     : $\mathcal{S}^*$ image sub-windows containing a face
Parameters: maximum number of false positives $NFP_{\max}$

$\mathcal{S} = \text{GetRawDetections}(I, \mathcal{C}_5, \mathcal{C}_{10}, \mathcal{C}_{80}, \mathcal{C}_{200})$ //Algorithm 2
$\mathcal{S}' = \text{GetStableDetections}(\mathcal{S}, I, \mathcal{C}_{200})$ //Algorithm 5
$\mathcal{S}^* = \text{GroupAndSimplifyDetections}(\mathcal{S}')$ //Algorithm 6
```

Algorithm 5: Mirror filter

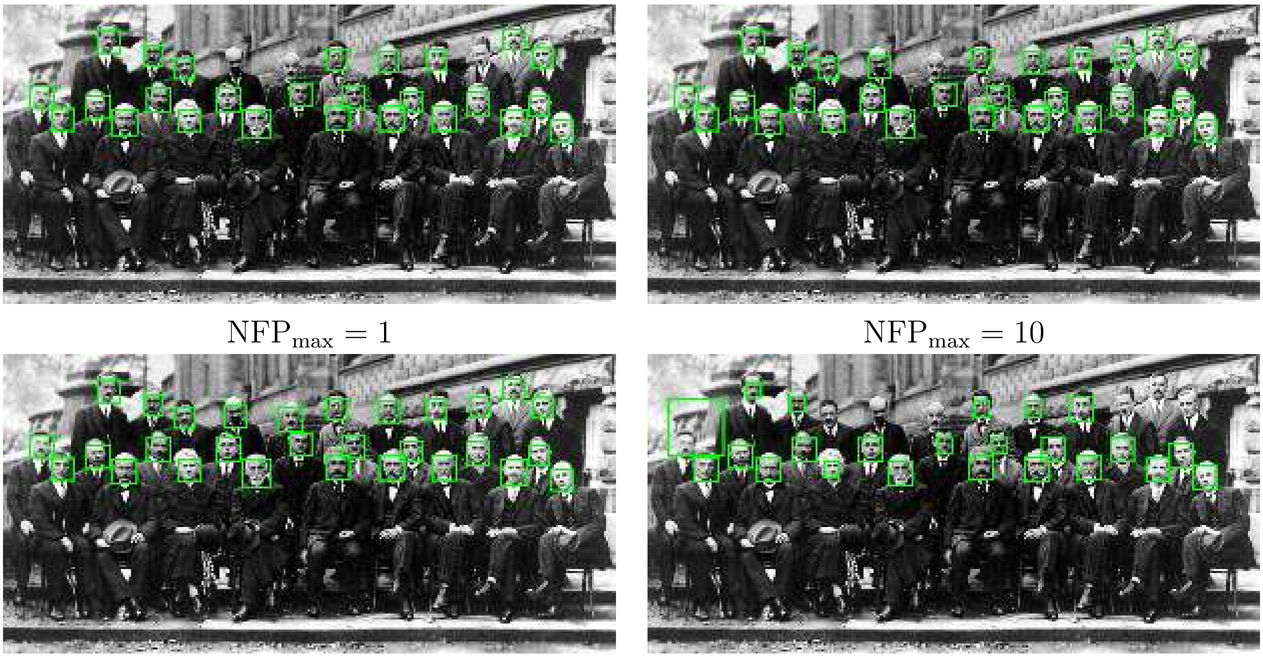
```
Input      : S set of sub-windows detected as faces by Algorithm 2, input image $I$, classifier
             data $\mathcal{C}_{200}$ and detection threshold $\Theta_{200}$
Output     : S' output set of sub-windows after rejection of un-stable detections
//For each sub-window $s \in \mathcal{S}$ the following information is known (see Algorithm 3):
//$z_s$: zoom factor applied to the original image to extract $s$
//$(i_s, j_s)$: position of the top-left corner of the sub-window in the rescaled image
$\mathcal{S}' = \emptyset$ //Initialize output detections set
for $s \in \mathcal{S}$ do
    $I_z = \text{RescaleImage}(I, z_s)$ //Rescale $I$ with scale factor $z_s$
    $w_z = width(I_z)$ //width of rescaled image
    $I'_z = \text{HorizontalFlip}(I_z)$ //Get mirror version of image
    $s' = \text{GetSubWindow}(I'_z, w_z - i_s - 24, j_s, 24, 24)$ //Get sub-window at mirror coordinates
    $v' = \text{ApplyClassifier}(s', \mathcal{C}_{200})$ //Get detection value for sub-window using 200-features
    classifier
    if $v' > \Theta_{200}$ then
        $s \to \mathcal{S}'$ //Add original sub-window to output detections set
```

Algorithm 6: Grouping and simplification of detections

```
Input  : S set of sub-windows detected as faces
Output : S' simplified set of sub-windows
//For each sub-window $s \in \mathcal{S}$ the following information is known:
//$(i_1, j_1)$, $(i_2, j_2)$: coordinates of the sub-window in the original image (top-left and
bottom-down corners)
//$v_{\texttt{det}}$: detection value for the 200-features classifier
$N = \text{Size}(\mathcal{S})$ //Number of input sub-windows
$\text{Sort}(\mathcal{S})$ //Sort sub-windows by decreasing detection value
//After sorting: $\mathcal{S}[0]$ has the highest detection value and $\mathcal{S}[N-1]$ the lowest
$\text{tags} = 1$ //Tags for each sub-window, initially set to 1 for all of them
for $k \in \{N-1, \cdots, 1\}$ do
    $d_1 = \mathcal{S}[k]$ //First sub-window
    $d_2 = \mathcal{S}[k-1]$ //Second sub-window
    $d_{12} = d_1 \cap d_2$ //Intersection of sub-windows (it might be an empty set)
    $A_{12} = \text{Area}(d_{12})$ //Area of the intersection
    $A_1 = \text{Area}(d_1)$ //Area of the first sub-window
    $A_2 = \text{Area}(d_2)$ //Area of the second sub-window
    $p = \frac{A_{12}}{\min(A_1, A_2)}$ //Equation (8)
    if $p > 0.5$ then
        //The area of intersection is relatively large
        //Keep $d_2$ (sub-window with higher detection value)
        //Tag $d_1$ for removal
        $\text{tags}[k] = 0$
$\mathcal{S}' = \emptyset$ //Initialize output detections set
for $k \in \{0, \cdots, N-1\}$ do
    if $tags[k] == 1$ then
        //Keep sub-window in output set
        $\mathcal{S}[k] \rightarrow \mathcal{S}'$ //Add sub-window to output detections set
```

5 Experimental Results

In Figures 7 to 9 the effect of varying the parameter of the method ($\mathrm{NFP}_{\mathrm{max}}$) is explored. We observe that the results are quite stable but that, as the parameter increases, more detections appear, although they are not always correct. For comparison, the result of the 31-stages cascade proposed in [19] are also shown. Let us remark that, as a way to improve the detection rates of the method, the author of [19] combines the results of applying the cascade to the original input image and to two slightly rotated versions of the same image. In order to make a fair comparison, we have modified the original code so that the 31-stages cascade is only applied on the original image.



$\mathrm{NFP}_{\mathrm{max}} = 1$

$\mathrm{NFP}_{\mathrm{max}} = 10$

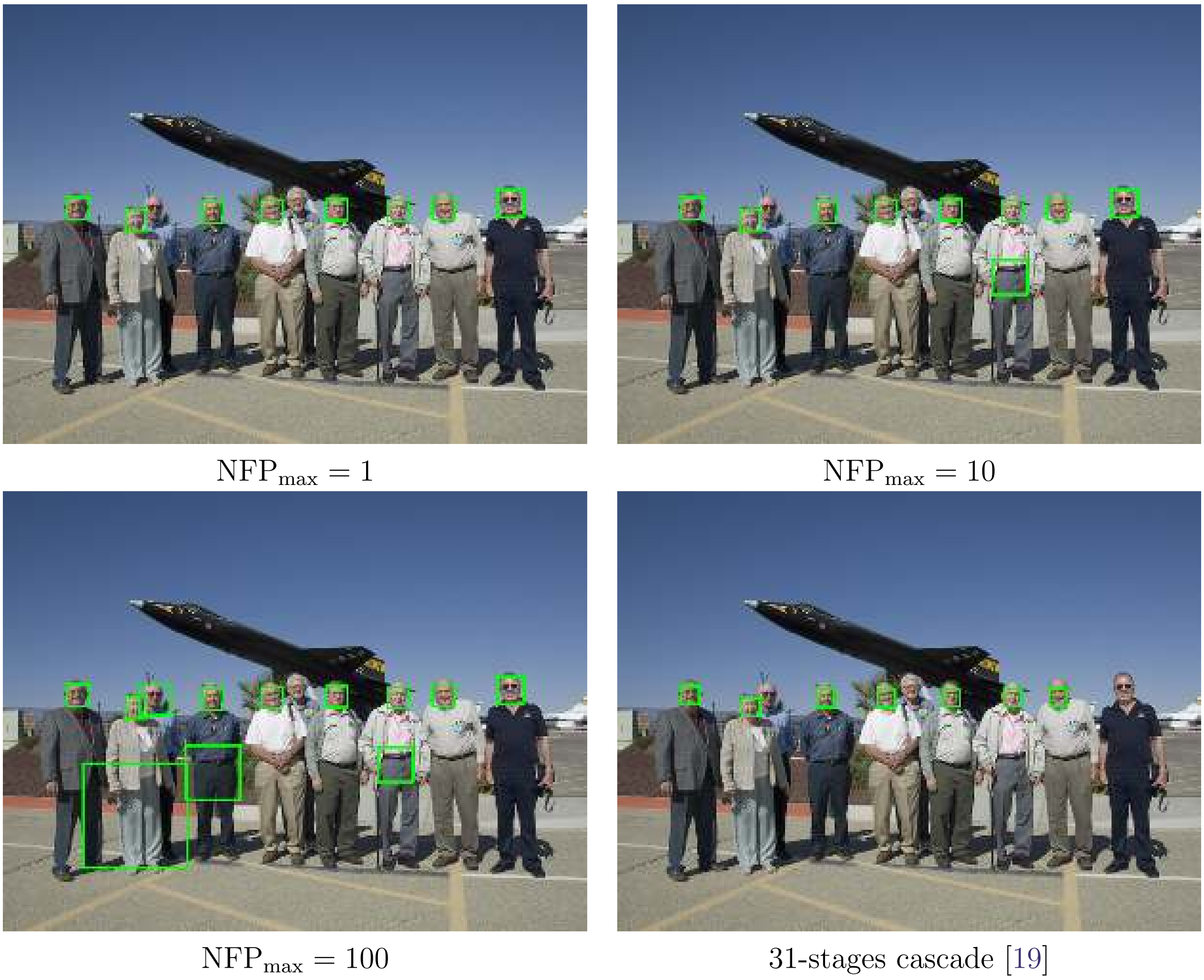
$\mathrm{NFP}_{\mathrm{max}} = 100$

31-stages cascade [19]

Figure 7: Results of the proposed method for different values of the parameter $\mathrm{NFP}_{\mathrm{max}}$. The bottom-right image displays the result of a full-cascade of classifiers without adaptive detection thresholds.

Figure 10 shows three examples of detection of frontal faces with $\mathrm{NFP}_{\mathrm{max}}$ fixed to 1. The results are compared with those of the 31-stages cascade from [19]. We observe that the use of adaptive detection thresholds permits to obtain good detection results with the short cascade, comparable to those of the large one.

Finally, some results on images containing “faces-in-the-wild” are displayed in Figure 11. These faces are not necessarily frontal and may be partially occluded. Remark that the classifiers (both in our method and in [19]) were trained with the same set of frontal 24 × 24 faces used in the original Viola and Jones paper [16], therefore, we cannot expect to get good results with this kind of images. However, the obtained results show a certain degree of robustness to partial occlusions and to variations in pose.



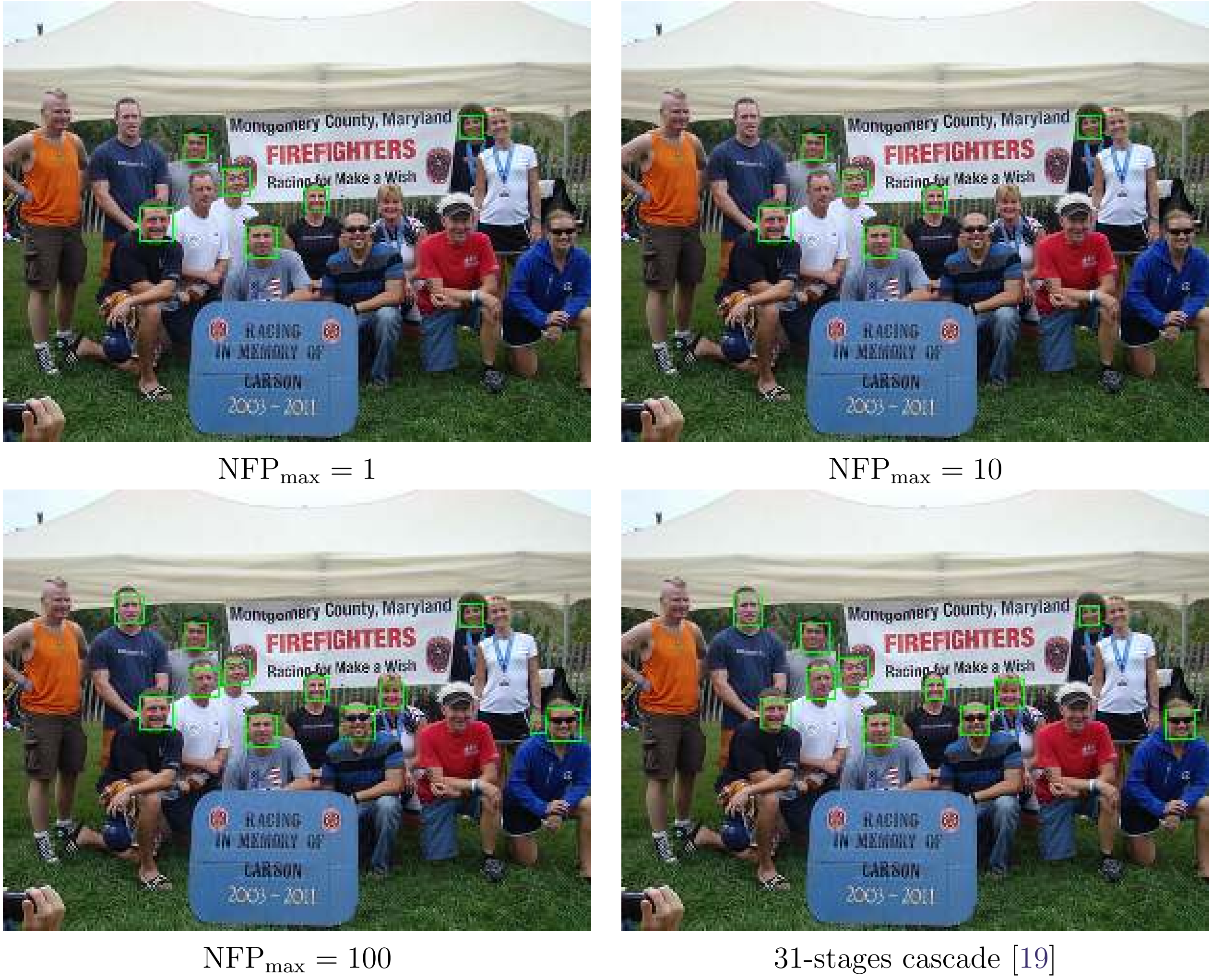
$\mathrm{NFP}_{\mathrm{max}} = 1$

$\mathrm{NFP}_{\mathrm{max}} = 10$

$\mathrm{NFP}_{\mathrm{max}} = 100$

31-stages cascade [19]

Figure 8: Results of the proposed method for different values of the parameter $\mathsf{NFP}_{\mathsf{max}}$. The bottom-right image displays the result of a full-cascade of classifiers without adaptive detection thresholds.



$\mathrm{NFP}_{\max} = 1$ $\mathrm{NFP}_{\max} = 10$

$\mathrm{NFP}_{\max} = 100$ 31-stages cascade [19]

Figure 9: Results of the proposed method for different values of the parameter $\mathrm{NFP}_{\max}$. The bottom-right image displays the result of a full-cascade of classifiers without adaptive detection thresholds.

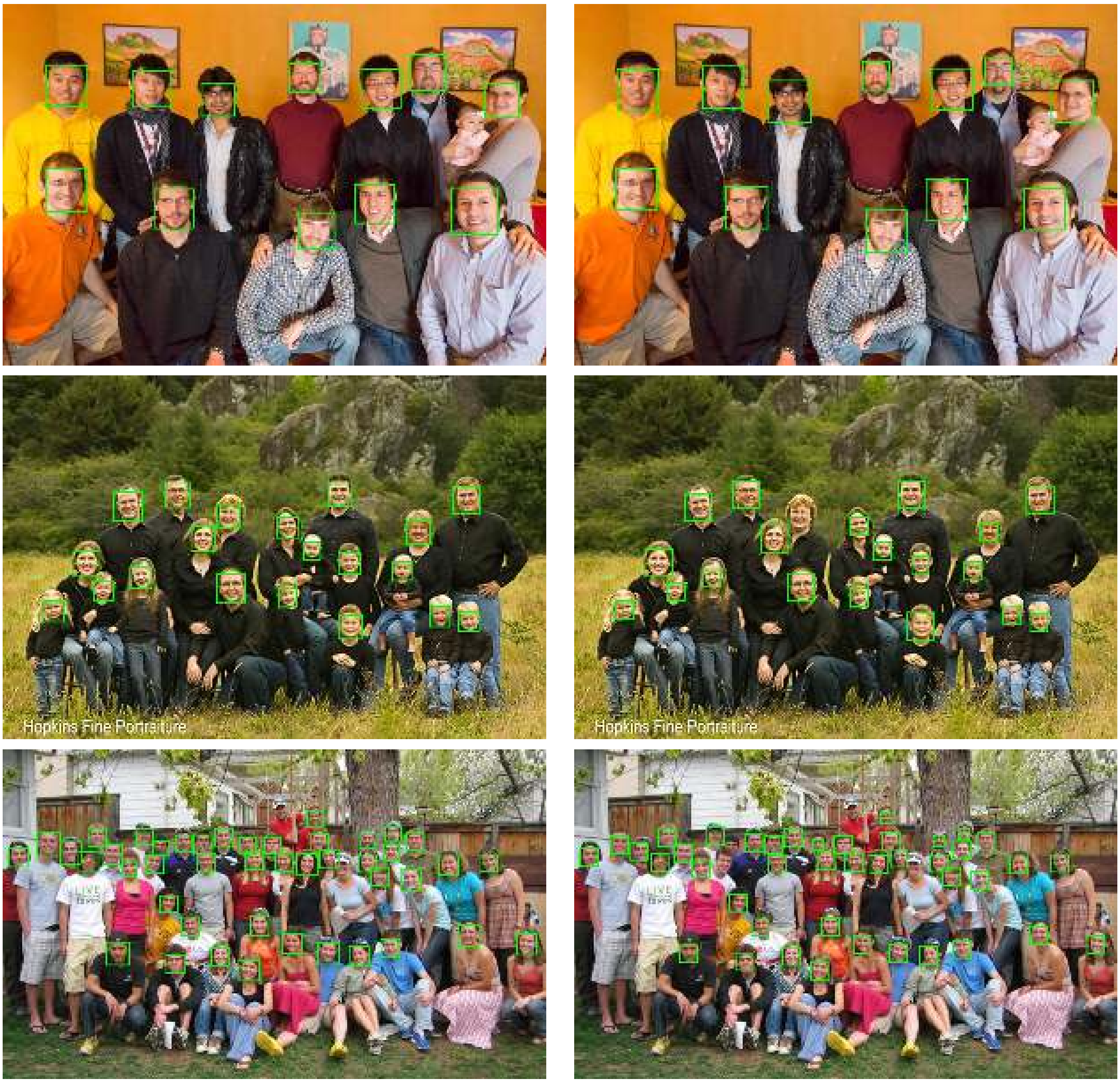


Figure 10: Results with frontal faces. Left, results of the proposed method for $\text{NFP}_{\text{max}} = 1$. Right, result of a full-cascade of classifiers without adaptive detection thresholds [19].

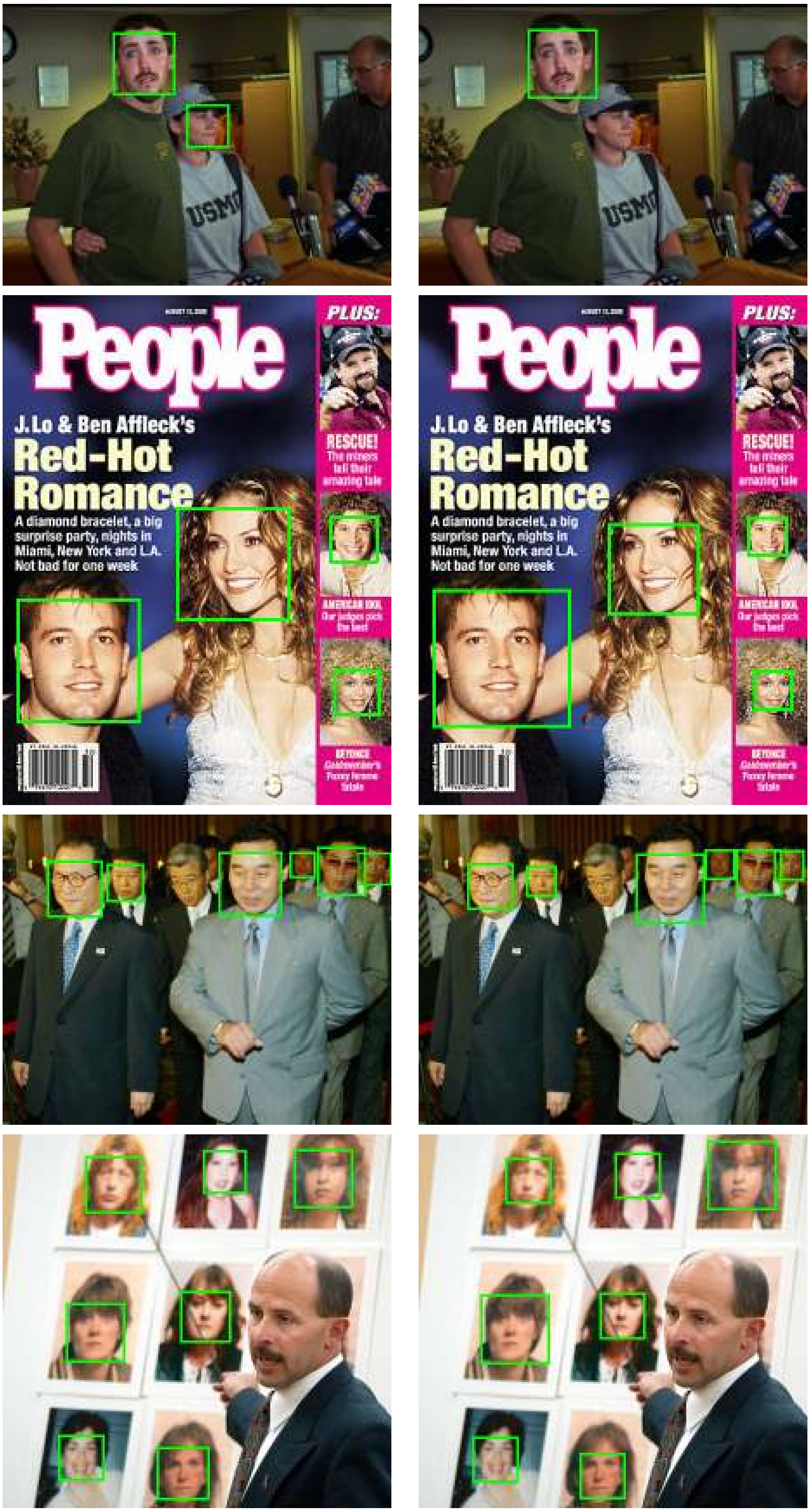

People
J.Lo & Ben Affleck's
Red-Hot Romance
A diamond bracelet, a big surprise party, nights in Miami, New York and L.A. Not bad for one week
PLUS:
RESCUE! The miners tell their amazing tale
AMERICAN IDOL Our judges pick the best

Figure 11: Results with “faces-in-the-wild”. Left, results of the proposed method for $\text{NFP}_{\text{max}} = 1$. Right, result of a full-cascade of classifiers without adaptive detection thresholds [19].

6 Conclusions

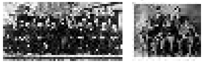
In this article we have described a method for the detection of frontal faces in digital images that improves the cascade of classifiers proposed by Viola and Jones in 2001 by incorporating adaptive detection thresholds computed using an a contrario approach. We have shown that the proposed method requires less computations than the original algorithm while providing similar detection rates. The adaptation of the method to the detection of faces in the wild (without restrictions on pose, robust to partial occlusions, etc.) shall be the subject of our future research.

Acknowledgements

The first author was supported by grants TIN2017-85572-P and DPI2017-86372-C3-3-R (MINECO / AEI / FEDER, UE). The second author benefited from the fellowship FPI/1828/2015 of the Conselleria d'Educació, Cultura i Universitats of the Govern de les Illes Balears under an operational program co-financed by the European Social Fund.

Image Credits

Miguel Colom, CC-BY


Public domain[7][8]

NASA[9][10]

from WIDER FACE: A Face Detection Benchmark [21][11]

from FDDB: Face Detection Data Set and Benchmark [5][12]

References

[1] M.O. Asadoorian and D. Kantarelis, *Essentials of Inferential Statistics*, University Press of America, 4th ed., 2004. ISBN 978-0761844518.

[2] J. Delon, A. Desolneux, J.L. Lisani, and A.B. Petro, *A nonparametric approach for histogram segmentation*, IEEE Transactions on Image Processing, 16 (2007), pp. 253–261. https://doi.org/10.1109/TIP.2006.884951.

[3] A. Desolneux, L. Moisan, and J.M. Morel, *Edge detection by Helmholtz principle*, Journal of Mathematical Imaging and Vision, 14 (2001), pp. 271–284. https://doi.org/10.1023/A:1011290230196.

[7] https://commons.wikimedia.org/wiki/File:Solvay_conference_1927.jpg

[8] https://www.publicdomainpictures.net/en/view-image.php?image=280567&picture=butch-cassidy-vintage-photo

[9] https://www.nasa.gov/topics/history/features/astp_gallery.html

[10] https://www.dfrc.nasa.gov/Gallery/Photo/People/Medium/index.html

[11] http://shuoyang1213.me/WIDERFACE/

[12] http://vis-www.cs.umass.edu/fddb/

[4] ——, *A grouping principle and four applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 508–513. https://doi.org/10.1109/TPAMI.2003.1190576.

[5] V. Jain and E. Learned-miller, *FDDB: A benchmark for face detection in unconstrained settings*, Tech. Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010. http://vis-www.cs.umass.edu/fddb/.

[6] V. Jain and E. Learned-Miller, *Online domain adaptation of a pre-trained cascade of classifiers*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2011, pp. 577–584. https://doi.org/10.1109/CVPR.2011.5995317.

[7] M. Köstinger, P. Wohlhart, P. Roth, and H. Bischof, *Robust face detection by simple means*, in Computer Vision in Applications Workshop (DAGM), 2012.

[8] A. Krizhevsky, I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks*, in International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, 2012, pp. 1097–1105. http://dl.acm.org/citation.cfm?id=2999134.2999257.

[9] S.Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum, *Statistical learning of multi-view face detection*, in European Conference on Computer Vision (ECCV), 2002, pp. 67–81. https://doi.org/10.1007/3-540-47979-1_5.

[10] R. Lienhart, A. Kuranov, and V. Pisarevsky, *Empirical analysis of detection cascades of boosted classifiers for rapid object detection*, in DAGM Symposium on Pattern Recognition, 2003, pp. 297–304. https://doi.org/10.1007/978-3-540-45243-0_39.

[11] R. Lienhart and J. Maydt, *An extended set of Haar-like features for rapid object detection*, in International Conference on Image Processing (ICIP), vol. 1, 2002, pp. 900–903. https://doi.org/10.1109/ICIP.2002.1038171.

[12] J. Lisani, S. Ramis, and F. Perales, *A contrario detection of faces: A case example*, SIAM Journal on Imaging Sciences, 10 (2017), pp. 2091–2118. https://doi.org/10.1137/17M1118774.

[13] J. L. Lisani and J-M. Morel, *Detection of major changes in satellite images*, in International Conference on Image Processing (ICIP), vol. 1, Sept 2003, pp. 941–944 vol.1. https://doi.org/10.1109/ICIP.2003.1247119.

[14] J. L. Lisani, L. Rudin, and A. Buades, *Fast video search and indexing for video surveillance applications with optimally controlled false alarm rates*, in IEEE International Conference on Multimedia and Expo, ICME'11, 2011, pp. 1–6. https://doi.org/10.1109/ICME.2011.6012151.

[15] M. Viola, M.J. Jones, and P. Viola, *Fast multi-view face detection*, in Proceedings of Computer Vision and Pattern Recognition, 2003.

[16] P. Viola and M. Jones, *Rapid object detection using a boosted cascade of simple features*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2001, pp. 511–518. https://doi.org/10.1109/CVPR.2001.990517.

[17] R. Grompone von Gioi, *A Contrario Line Segment Detection*, Springer, 2014. ISBN 978-1-4939-0575-1.

[18] R. Grompone von Gioi and G. Randall, *Unsupervised smooth contour detection*, Image Processing On Line, 6 (2016), pp. 233–267. https://doi.org/10.5201/ipol.2016.175.

[19] Y-Q. Wang, *An Analysis of the Viola-Jones Face Detection Algorithm*, Image Processing On Line, 4 (2014), pp. 128–148. https://doi.org/10.5201/ipol.2014.104.

[20] B. Wu, H. Ai, C. Huang, and S. Lao, *Fast rotation invariant multi-view face detection based on real adaboost*, in IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 79–84. https://doi.org/10.1109/AFGR.2004.1301512.

[21] S. Yang, P. Luo, C.C. Loy, and X. Tang, *WIDER FACE: A Face Detection Benchmark*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. https://doi.org/10.1109/CVPR.2016.596.

[22] S. Zafeiriou, C. Zhang, and Z. Zhang, *A survey on face detection in the wild: Past, present and future*, Computer Vision and Image Understanding, 138 (2015), pp. 1 – 24. https://doi.org/10.1016/j.cviu.2015.03.015.