IDIAP RESEARCH REPORT

FACE DETECTION USING BOOSTED JACCARD DISTANCE-BASED REGRESSION

Cosmin Atanasoaei Chris McCool
Sébastien Marcel

Idiap-RR-02-2012

JANUARY 2012

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

Face detection using boosted Jaccard distance-based regression

Cosmin Atanasoaei, Christopher McCool, Sébastien Marcel
Idiap Research Institute
Martigny, Switzerland
cosmin.atanasoaei@idiap.ch

Abstract

This paper presents a new face detection method. We train a model that predicts the Jaccard distance between a sample sub-window and the ground truth face location. This model produces continuous outputs as opposite to the binary output produced by the widely used boosted cascade classifiers. To train this model we introduce a generalization of the binary classification boosting algorithms in which arbitrary smooth loss functions can be optimized. This way single output regression and binary classification models can be trained with the same procedure.

Our method presents several significant advantages. First, it circumvents the need for a specific discretization of the location and scale during testing. Second, it provides an approximation of the search direction (in location and scale) towards the nearest ground truth location. And finally, the training set consists of more diverse samples (e.g. samples covering portions of the faces) that cannot be used to train a classifier. We provide experimental results on BioID face dataset to compare our method with the sliding-windows approach.

1. Introduction

Face detection consists of finding the position of all the faces, if any, in images. Two components are usually required: a classifier and a search algorithm. The search (or scanning) algorithm forms sub-windows (or samples) at different locations and scales which are feed to the classifier. The sub-windows labelled as positive samples are considered as final detections. Usually a clustering algorithm (e.g. non-maxima suppression, averaging the overlapping regions, mean shift) is run on these detections to reduce the number of multiple detections.

Recently there has been a great interest in real-time face detection systems. Their speed depends mostly on the speed of the classifier to evaluate a sub-window. These systems are usually built using boosted classifiers [13, 14], because of their potential computational efficiency while providing state of the art performance. Another important factor is the speed to compute the features. The fastest features are evaluated in constant complexity at any location and scale, for example: Haar-like features [13] and MCT [6] or multi-block LBP codes [14].

The most popular and simple search strategy for face detection is the sliding-windows approach (we refer to this method as ***SScan***). The location and scale space is usually discretized using a fixed grid or a coarse-to-fine approach.

There are two problems with the *SScan* approach that we address in this paper. First, the discretization parameters are difficult to automatically adjust to the size of the image to scan and it clearly depends on the (unknown) number, on the size and on the distance between adjacent faces. Second, the classifiers cannot be trained with samples that cover just a part of the face. For example it is impossible to decide if a sample containing just half of the face should be considered as a positive or as a negative training sample. Therefore an uncertain region around the ground truth is formed during training [3, 2]. But these kind of samples consistently appear at testing time and there is no guarantee on the classifier's output in this situation.

A possible solution is to use regression to learn a richer information than just a label of a sub-window to test. There have been some previous work on this research direction. For example a boosted model is trained in [2] to predict the displacement of a facial feature patch from the ground truth. In [3] a regression approach is used to predict the eye positions. Our work follows this direction.

More specifically, we propose a new real-time face detection method that uses regression to guide the search. We train a model that predicts the Jaccard distance between a sample sub-window and the nearest ground truth face location. Then this model is used to search for faces in two steps. First we initialize of a set of potential detections with a coarse sampling and second we iteratively refine the most promising detections. We refer to this method as ***JScan***.

The main contributions of our paper can be summarized as follows:

- We train a model to learn how accurate a sub-window

is in both location and scale. This allows for arbitrary displacements and scale variations of the ground truth face locations sub-windows in the training samples. We then use this model for face detection without the need for a specific discretization of the search space.

- We propose a general formulation of boosting algorithms that is independent of the loss function and the specific classification or regression task to solve. This formulation allows for training binary classifiers and single output regressors with the same algorithm.

- An additional contribution is the proposed features that combine Multi-Block Local Binary Patterns and Modified Census Transform features.

This paper is organized as follows. The Section 2 presents a general view of the boosting algorithms for both classification and regression. Here we briefly introduce the sliding-windows approach to face detection. In Section 3 we describe our proposed method: the model that learns the distance of a sub-window to the ground truth face location and the associated search algorithm similar to sliding-windows. The experimental procedure and the comparative results with an equivalent complex boosted classifier are presented in Section 4. Finally we conclude and point to future work directions in Section 5.

2. Related work

In this section we present a general view of the boosting algorithms for both classification and regression (2.1). Then we briefly introduce the sliding-windows approach to face detection (2.2).

2.1. Boosting

Boosting [10] is a greedy method for learning a *strong classifier* as a linear combination of *weak classifiers*. This process is done iteratively in *boosting rounds*: a single new weak classifier is chosen and added to the combination. Each new weak classifier is usually trained to correct the mistakes made by the previous ones and to focus on the most challenging samples. Boosting can also be interpreted as a gradient descent algorithm in the functional space of the weak classifiers [9].

In this paper we focus on a more general formulation of boosting as a greedy optimization of the Taylor expansion of the loss function to optimize [9, 11]. This has the advantage of having the same formulation for both classification and regression, allowing for an easy and fair comparison between classification and regression methods for face detection.

More formally let $\boldsymbol{\chi}$ be the input signal space and $\{(x_n, y_n)_{n=1:N}\} \in (\boldsymbol{\chi} \times \mathbb{R})^N$ a set of N training samples. The targets $\{y_n\}$ to learn can be either binary labels $\{-1, +1\}$ or any other scalar for regression problems. The goal is to build a functional $f : \boldsymbol{\chi} \rightarrow \mathbb{R}$ to map the samples x_n to their targets y_n. At each boosting round t, $t \leq T$, its approximation is a linear combination of g_s *weak learners*: $f_t = \sum_{s \leq t} g_s$.

The criteria to choose f is to minimize a loss function of the form: $L(f_t) = \sum_n l(y_n, f_t(x_n))$. At each step $t+1$ the weak learner g_{t+1} is chosen to minimize:

$$g_{t+1} = \arg\min_g \sum_n l(y_n, f_t(x_n) + g(x_n)). \quad (1)$$

Taking the Taylor expansion of the loss, in the current f_t point, up to the second order, the optimization problem becomes:

$$\begin{aligned} g_{t+1} \quad = \quad & \arg\min_g L(f_t) + \\ & \sum_n \left(\frac{\partial l(y_n, f)}{\partial f} \mid_{f=f_t(x_n)} \right) g(x_n) + \\ & \frac{1}{2} \sum_n \left(\frac{\partial^2 l(y_n, f)}{\partial f^2} \mid_{f=f_t(x_n)} \right) g^2(x_n). \end{aligned} \quad (2)$$

Most boosting algorithms can be derived from this formulation. We distinguish between second order and first order boosting algorithms depending if they use or not the second order term in the Taylor expansion in Eq. 2.

The first order boosting algorithms perform a gradient descent in the functional space:

$$g_{t+1} = \arg\max_g \left| \sum_n \left(\frac{\partial l(y_n, f)}{\partial f} \mid_{f=f_t(x_n)} \right) g(x_n) \right|. \quad (3)$$

For example: "AdaBoost" [10] minimizes the exponential loss $l(y, f) = exp(-yf)$ and restricts the weak classifiers to the form $g_s : \chi \rightarrow \{-1, +1\}$, while "AnyBoost" [9] is a generic formulation for any loss functions. It can be noticed that the optimal weak learner g_{t+1} is known up to a scaling factor. This is the reason why g_{t+1} is typically scaled using the line-search algorithm, fixed steps or decreasing steps. In particular cases the optimal scale can be found analytically (e.g. AdaBoost).

The second order boosting algorithms use adaptive Newton steps to minimize the loss function. A well known algorithm of this type is "Gentle AdaBoost" [5, 11] which optimizes the exponential loss for the classification task.

Usually the normalized partial derivatives of the loss function are considered as *weights* associated to each sample (e.g. AdaBoost, Gentle AdaBoost). This allows, in certain conditions for the classification task, to interpret boosting as a greedy algorithm that concentrates the current weak

Classification:	$l_1(y,f) = exp(-yf)$
	$l_2(y,f) = log(1+exp(-yf))$
Regression:	$l_3(y,f) = \frac{1}{2}(y-f)^2$
	$l_4(y,f) = exp(y-f) + exp(f-y) - 2$

Table 1. Various loss functions for classification (l_1, l_2) and regression (l_3, l_4).

learner on the samples miss-classified by the previous weak learners.

The loss function depends on the specific problem to solve (see Table 1). For example, the classification task requires the two classes to be separated as far as possible: $l(y,f) = l(-yf)$, while the regression task needs a prediction as close as possible to the target: $l(y,f) = l(y-f)$.

2.2. Face detection using sliding-windows (SScan)

The Algorithm 1 presents the sliding-windows approach to face detection. Given a face classifier M that processes sub-windows of size $M^w \times M^h$, the algorithm searches for faces in the image I of size $I^w \times I^h$. The discretization of the location and scale space is governed by the dx, dy, ds parameters. The dx and dy coefficients are used to compute the displacement in location between two sub-windows, relative to the model size, for the scaled image I_s of size $I_s^w \times I_s^h$ by the s factor. If the classifier scores above a given threshold τ, then the detection $det = \{x, y, s\}$ is accepted in the final list D.

Algorithm 1 Face detection using sliding-windows.

```
dx ∈ (0,1), dy ∈ (0,1), ds ∈ (0,1), τ, M, I, D = φ
for s = 1 to s > 0 do
    scale the image: I_s ← I ⊗ s
    for x = 0 to x < I_s^w do
        for y = 0 to y < I_s^h do
            det = {x, y, s}
            if M(det) ≥ τ then
                D ← D ∪ det
            end if
            y ← y + dy * M^h
        end for
        x ← x + dx * M^w
    end for
    s ← s − ds
end for
return D
```

There are several problems with this method. First, the dx, dy, ds parameters are difficult to set a priori. They dependent on the size of the image to search and the number of and the distance between face locations. Second, the search algorithm uses a face classifier that is trained with roughly normalized samples. For example it cannot be trained with samples that cover just a part of the face. This is because it is impossible to decide if a sample containing just half of the face should be considered as a positive or as a negative training sample. Therefore an uncertain region around the ground truth is formed during training [3, 2]. The problem is that there is no guarantee on the output of the classifier for these kind of sub-windows that can appear during testing.

3. Proposed approach

In this section we introduce the features and the weak learner (3.1). Then we describe the training algorithm (3.2) using the framework presented in the previous section. Next we present the Jaccard distance (3.3) and how to use it for face detection (3.4).

3.1. Features and weak learner

A real-time face detection system requires features that are fast to compute at any location and scale. The first real-time system used Haar-like features [12], but LBP-based features became also very popular because they are robust to illumination changes [6]. Recently, the Multi-Block LBP features [14] have been shown to outperform both Haar-like features and LBP codes. Hence, in this work we use a new feature - the Multi-Block Modified Census Transform (MBMCT), that combines the multi-block idea proposed in [14] and the MCT features proposed in [6].

The MBMCT features are parametrized by the top-left coordinate (x, y) and the size $w \times h$ of the rectangular cells in the 3×3 neighbourhood. This gives a region of $3w \times 3h$ pixels to compute the 9-bit MBMCT:

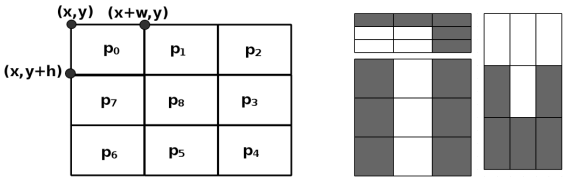
$$MBMCT(x,y,w,h) = \sum_{i=0:8} \delta(p_i \geq \bar{p}) * 2^i, \quad (4)$$

where δ is the Kronecker delta function, $\bar{p}$ is the average pixel intensity in the 3×3 region and p_i is the average pixel intensity in the cell i (see Fig. 1 (a)). The feature is computed in constant time for any parametrization using the integral image. Various patterns at multiple scales and aspect ratios can be obtained by varying the parameters w and h (see Fig. 1 (b)).

The MBMCT feature values are non-metric codes and this restricts the type of weak learner to boost. We use the multi-branch decision tree proposed in [14] as weak learner. This weak learner is parametrized by a feature index (e.g. dimension in the feature space) and a set of fixed outputs, one for each distinct feature value. More formally, the weak learner g is computed for a sample x and a feature d with:

$$g(x) = lut[x^d], \quad (5)$$

where lut is a look-up table with 512 entries a_u (because there are 512 distinct MCT codes) and d indexes the space



(a) (b)

Figure 1. (a) Multi-block MCT feature for image representation. (b) Examples of some patterns that can be obtained by varying the parameters w and h.

of x, y, w, h possible MBMCT parametrizations. The goal of the boosting algorithm is then to compute the optimum feature d and a_u entries.

3.2. Training

In this section we derive the second order boosting algorithm to train the multi-branch decision tree. We chose this formulation over the first order because generally the loss decreases faster using Newton-Raphson steps than using gradient descent steps. The Eq. 2 can be rewritten for a fixed feature d as:

$$g_{t+1}(d, a_u) = \arg\min_{a_u} L(f_t) + \sum_u a_u \left(\sum_{n, x_n^d = u} \left(\frac{\partial l(y_n, f)}{\partial f} \mid_{f=f_t(x_n)} \right) \right) + \sum_u a_u^2 \left(\frac{1}{2} \sum_{n, x_n^d = u} \left(\frac{\partial^2 l(y_n, f)}{\partial f^2} \mid_{f=f_t(x_n)} \right) \right) \quad (6)$$

or more compactly as:

$$g_{t+1} = \arg\min_{d, a_u} L(f_t) + \sum_u a_u L'_u + \sum_u \frac{1}{2} a_u^2 L''_u, \quad (7)$$

where L'_u and L''_u are the cumulated first and second order derivatives of the loss for the samples that have the feature d with the value u. It can be noticed that the quadratic optimization problem can be solved **separately** for each look-up-table entries a_u. The exact solution and the potential loss decrease Δ are:

$$a_u = -\frac{L'_u}{L''_u} = -\frac{\sum_{n, x_n^d = u} \left(\frac{\partial l(y_n, f)}{\partial f} \mid_{f=f_t(x_n)} \right)}{\sum_{n, x_n^d = u} \left(\frac{\partial^2 l(y_n, f)}{\partial f^2} \mid_{f=f_t(x_n)} \right)} \quad (8)$$

$$\Delta = -\sum_u \frac{(L'_u)^2}{2L''_u}. \quad (9)$$

The training algorithm is presented in Algorithm 2. At each boosting round t, the optimal weak learner g_{t+1} is chosen by evaluating each feature d and selecting the optimal one with the highest decrease Δ in the loss. Then g_{t+1} is added to the strong model f. It can be noticed that the algorithm is of linear complexity with the number of samples, features and boosting rounds. There are two important benefits: it performs feature selection and it can be used for any smooth loss function. In this paper we use this algorithm for both face classification and Jaccard distance-based face regression.

Algorithm 2 Second order boosting multi-branch MBMCT decision trees.

for $t = 0, f = 0$ **to** $t \leq T$ **do**
  $d^* = 0, \Delta^* = \infty, a_u^* = 0$
  for feature d **do**
    for feature value $u \in 0...511$ **do**
      compute L'_u and L''_u
    end for
    $\Delta = -\sum_u \frac{(L'_u)^2}{2L''_u}$
    if $\Delta < \Delta^*$ **then**
      $d^* \leftarrow d, \Delta^* \leftarrow \Delta, a_u^* \leftarrow -\frac{L'_u}{L''_u}$
    end if
  end for
  $f \leftarrow f + g_{t+1}(d^*, a_u^*), t \leftarrow t + 1$
end for
return f

3.3. Jaccard distance

The Jaccard distance [7] is a statistical method to measure the similarity between two sets A and B (see Eq. 10).

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}. \quad (10)$$

This can be extended to measure the overlapping between two rectangular regions. Then, $|A \cap B|$ and $|A \cup B|$ stand for the area of their intersection and reunion, respectively. We decided to use an approximation to the Jaccard distance that it is easier to compute in the case of face detection:

$$J_m(A, B) = 1 - \frac{|A \cap B|}{max(|A|, |B|)} \quad (11)$$

Let A be the ground truth face location and B a sub-window to evaluate. Then, the perfect detection corresponds to the distance $J_m(A, B) = 0$, while the background sub-windows corresponds to the distance $J_m(A, B) = 1$. The target y to learn (see Sections 2.1 and 3.2) for the particular sub-window B is $J_m(A, B)$.

3.4. Face detection using Jaccard distance-based regression (JScan)

We propose a regression-driven search algorithm for face detection. Instead of using a classifier, we train a model to learn a richer information: the Jaccard distance between a sub-window and the closest ground truth face location. An immediate benefit of using regression is that the model can be trained with sub-windows at any location and scale, which implies that no uncertain region is formed any more.

Assuming that such a model M is provided, the search algorithm becomes as presented in Algorithm 3. There are several significant differences compared to Algorithm 1. First, no dx, dy, ds discretization is needed any more. This is because the model predicts how far the current sub-window is from the true face location, which can be used to guide the search instead of some a priori fixed discretization parameters. Second, the proposed method is split in two stages: the initialization of potential locations (steps 8, 10 and 12) and the refinement (steps 14, 15) of these locations to minimize the Jaccard distance. It can be noticed that we refine only the sub-windows that are close to the ground truth. This has the benefit of concentrating the effort (evaluating sub-windows) in the most promising regions of the search space. Finally, we ignore the detections that are farther away than τ from the true location (step 16). This corresponds to eliminating false alarms (see Algorithm 1, step 7).

Initialization stage (steps 1-12)

The search for the optimal face locations is initialized using an uniform grid. The idea is *to sample such that all the faces to be included at least in halves in some sub-window*. This implies that we need to sample every $\frac{M^w}{2}$ and $\frac{M^h}{2}$ on the horizontal and vertical axis, respectively. The scale sampling factor is slightly more difficult to set. Let s be the current scale. Then a sub-window at this scale has the size of $\frac{1}{s}M^w \times \frac{1}{s}M^h$ relative to the original image. The difference in size between two sub-windows at consecutive scales s and $s' < s$ is: $(\frac{1}{s'} - \frac{1}{s})M^w \times (\frac{1}{s'} - \frac{1}{s})M^h$. To make sure all the faces are contained at least halved in some sub-window we set the conditions: $(\frac{1}{s'} - \frac{1}{s})M^w \leq \frac{M^w}{2}$ and $(\frac{1}{s'} - \frac{1}{s})M^h \leq \frac{M^h}{2}$. This implies $\frac{1}{s'} - \frac{1}{s} \leq \frac{1}{2}$. At the limit, it can be shown that we obtain the following relation for the scale variation: $s_n = \frac{2}{n+1}, n \geq 1$.

Refinement stage (steps 14 and 15)

Next the detections close enough to the ground truth locations are *refined* in two steps. Given that the Jaccard distance is isotropic, the optimal direction where the face location resides cannot be deduced from this information. Instead we sample *independently* each axis (horizontal, vertical and scale) with two values (left - right, down - up, bigger - smaller). In the first step we sample with the half, while in the second step with the quarter of the displacement used in the initialization. The sampling spacing is decreased because smaller steps are required as the detection get closer to the ground truth locations. Only the detections that are within 0.50 and 0.40 Jaccard distance from the ground truth are refined at the first and the second step respectively. It can be noticed that it pointless to have more than two refinement steps because the spacing resolution (divided by two at each step) reaches the limit.

For example, for the sub-window $(x, y, \frac{1}{s})$ we generate at the first refinement step the six sub-windows to refine: $(x \pm \frac{M^w}{4}, y \pm \frac{M^h}{4}, \frac{1}{s} \pm \frac{1}{4})$. Considering a model of the size 24×24, the initialization part process locations at every 12 pixels, while the refinement steps at every 6 and 3 pixels respectively.

Algorithm 3 Face detection using Jaccard distance-based regression.

1: $\tau \in (0,1), M, I, D = \phi$
2: **for** $s = 1$ **to** $s \geq max(\frac{M^w}{I^w}, \frac{M^h}{I^h})$ **do**
3: $\quad I_s \leftarrow I \otimes s$
4: $\quad$ **for** $x = 0$ **to** $x < I_s^w$ **do**
5: $\quad\quad$ **for** $y = 0$ **to** $y < I_s^h$ **do**
6: $\quad\quad\quad det = \{x, y, s\}$
7: $\quad\quad\quad D \leftarrow D \cup det$
8: $\quad\quad\quad y \leftarrow y + \frac{M^h}{2}$
9: $\quad\quad$ **end for**
10: $\quad\quad x \leftarrow x + \frac{M^w}{2}$
11: $\quad$ **end for**
12: $\quad \frac{1}{s} \leftarrow \frac{1}{s} + \frac{1}{2}$
13: **end for**
14: $refine(D, 0.50, x \pm \frac{M^w}{4}, y \pm \frac{M^h}{4}, \frac{1}{s} \pm \frac{1}{4})$
15: $refine(D, 0.40, x \pm \frac{M^w}{8}, y \pm \frac{M^h}{8}, \frac{1}{s} \pm \frac{1}{8})$
16: $threshold(D, \tau)$
17: **return** D

4. Experiments and results

As a proof of concept, we have performed experiments to investigate the feasibility of the proposed idea. For this we have compared our proposed face detection method with the sliding-windows approach using an equivalent complex boosted classifier.

4.1. Experimental setup

The training samples were generated using the BANCA [1] English face dataset (6240 images) and the CALTECH-101 [4] background dataset (451 images). The BANCA dataset contains images taken in controlled and uncontrolled conditions of a single person in an office environment. We normalized these images to have the eyes hori-

zontally aligned and with 32 pixels distance between them. Then we collected roughly 6 billion sub-windows of 24×24 size using a very fine discretization of location and scale.

Each model was trained using 500,000 randomly selected samples. The training samples were generated to be evenly distributed over the output values: the class labels $\{-1, +1\}$ for classification and the Jaccard distance values $[0, 1]$ for regression. The classifier required one more restriction to overcome the uncertain area problem (see Section 2.2): the positive samples to overlap at least 90%, while the negative samples to overlap at most 10% respectively with the ground truth face location.

The same feature parametrization (see Section 3.1) was used for both models. The MBMCT features were generated with the cell size varying in the range $\{1 \dots 8\} \times \{1 \dots 8\}$. This generates roughly 7,000 features per sample. We used the second-order boosting procedure with 200 rounds to train both models (see Section 3.2). The only difference is in the choice of appropriate loss functions: the exponential $l_1(y, f) = exp(-yf)$ and the sum of exponentials $l_4(y, f) = exp(y-f)+exp(f-y)-2$ losses (see Table 1) were used for the classifier and the regressor respectively.

We have chosen the BioID dataset [8] as the test dataset because it contains face images captured with a setup close to the one used as the training dataset (BANCA), although significantly more challenging to detect. This dataset contains 1521 images of a single person taken in different office environments.

The ROC curves are built by varying the threshold τ (see Algorithms 1 and 3) and measuring the detection rate (DR) and the number of false alarms (FA). Multiple detections are integrated using non-maxima suppression. This is performed iteratively: first we chose the detection with the highest classification score or lowest Jaccard distance and second, we remove the other detections that overlap more than 60% with it.

4.2. Results

There are several aspects we investigate: the evolution of the training loss, the face detection performance and the speed of the proposed JScan method. Finally we provide some examples of the proposed detection process produced on the BioID dataset. We would like to point out that the aim of this paper is to assess the proposed method and not to produce the best possible results. Our goal is to compare a boosted classifier and a boosted regressor on the same datasets. It is clear that improved results can be obtained with more datasets, but it is out of the scope of this study.

Training loss

The training loss evolution provides an insight into how difficult a task is to solve for a particular model. We have plotted in Fig. 2 the logarithmic evolution of the training loss for the face classifier and the Jaccard distance-based regressor. It can be noticed that the loss decreases exponentially in the case of classification, but only linearly in the case of regression. This suggests that it is significantly harder to learn how far a sub-window is from the ground truth than classifying it as face or background. Still, a slowly increasing accurate regression output is produced as the number of boosting rounds increases.

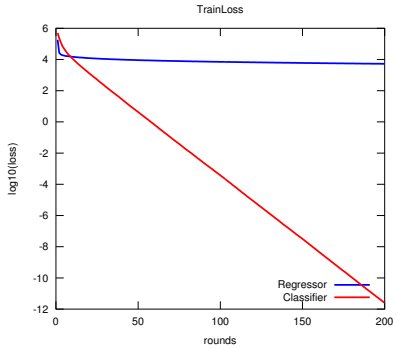


Figure 2. The logarithmic evolution of the training loss for the face classifier (red) and the Jaccard distance-based regressor (blue).

Face detection performance

We have evaluated the face detection performance of our method JScan and the baseline SScan on the BioID dataset. The sliding-windows approach depends on the search space parameters for location and scale. Hence, we have used two scenarios: the coarse search ($dx = 0.25$, $dy = 0.25$, $ds = 0.20$) and the fine search ($dx = 0.20$, $dy = 0.20$, $ds = 0.10$), which we denote as *SScan (coarse)* and *SScan (fine)* respectively.

The logarithmic ROC curves are plotted in Fig. 3. It can be noticed that the performance of the SScan method clearly depends on the search parametrization: the fine search significantly outperforms the coarse search. This is at the cost of a slower face detector, because the number of sub-windows increases rapidly with the search parameters.

The proposed JScan method performs significantly better than the baseline with a DR 5% larger for the same number of false alarms. This performance is maintained for various number of boosting rounds (see Fig. 4) with a noticeable larger improvement for small number of boosting rounds. This correlates with the evolution of the training loss (see Fig. 2) when the regressor learns faster for the first rounds, but significantly slower for large number of boosting rounds. This allows the classifier to close the gap in terms of performance.

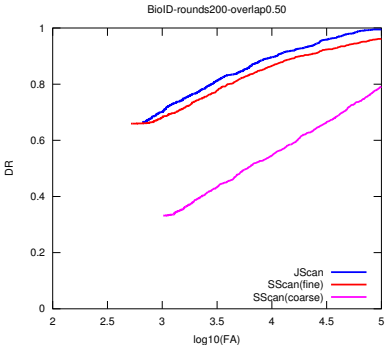
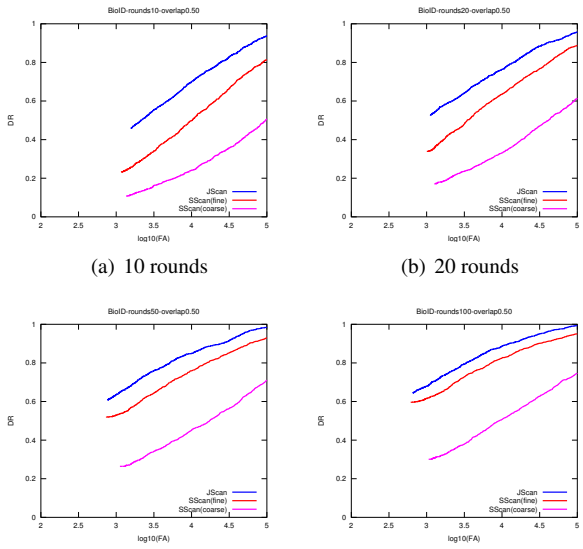


Figure 3. The logarithmic ROC curves for the BioID dataset using JScan (blue) and SScan with coarse (magenta) and fine (red) search parametrization. All models were trained using 200 boosting rounds.



(a) 10 rounds (b) 20 rounds

(c) 50 rounds (d) 100 rounds

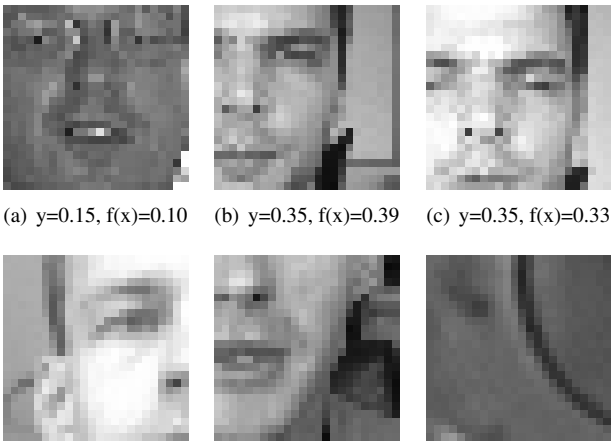
Figure 4. The logarithmic ROC curves for the BioID dataset using various number of boosting rounds. The results for JScan are plotted with blue, while for SScan with coarse and fine search parametrization with magenta and red, respectively.

Detection speed

We have also studied the number of processed sub-windows for each method (see Table 2) at various number of boosting rounds. It can be noticed that as the number of boosting rounds increases, the JScan method processes fewer sub-windows for both test datasets. This is because the Jaccard distance-based regressor becomes more reliable and fewer sub-windows need to be refined (see Algorithm 3). This contrasts with both SScan instances that process the same number of sub-windows. Hence, it is possible to achieve an even higher speed with a more accurate regressor. We conclude that our proposed method JScan is significantly faster than the baseline methods for a similar complexity of the model. Indeed, the classifier and the regressor contain the same number of parameters.

Boosting rounds	10	20	50	100	200
			BioID		
JScan	2,941	2,668	2,462	2,388	2,347
SScan (coarse)	8,485	8,485	8,485	8,485	8,485
SScan (fine)	21,055	21,055	21,055	21,055	21,055
Speed-up factor	**2.88**	**3.18**	**3.44**	**3.55**	**3.61**

Table 2. The number of processed sub-windows (in thousands) for the BioID dataset using various number of boosting rounds. The JScan speed-up factor is computed relative to SScan (coarse).



(a) y=0.15, f(x)=0.10 (b) y=0.35, f(x)=0.39 (c) y=0.35, f(x)=0.33

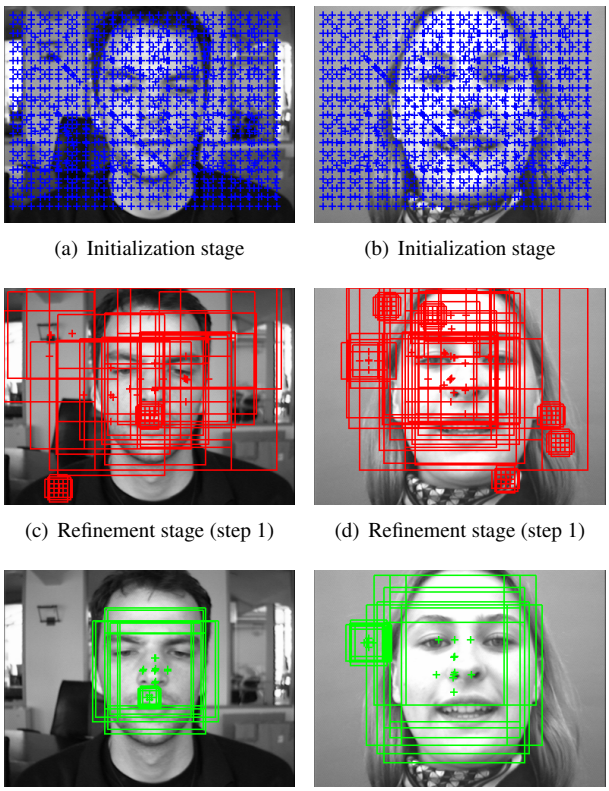
(d) y=0.62, f(x)=0.63 (e) y=0.64, f(x)=0.59 (f) y=0.98, f(x)=0.93

Figure 5. Examples of sub-windows (x) processed by the JScan method on the BioID dataset. The number on the left of the caption (y) is the Jaccard distance and the number on the right $f(x)$ is the estimated one.

Examples

Figure 5 presents some sub-windows processed by our proposed method on the BioID dataset. The number on the left of the caption (y) is the Jaccard distance and the number on the right $f(x)$ is the estimated one. These samples contain faces at different location displacements and scale variations. This makes the Jaccard distance modelling a more difficult task than face classification. For example the samples b, c, d and e (see Fig. 5) are excluded when training the face classifier because they are ambiguous. But the Jaccard distance model must cope with these difficult samples. This results in significant errors at testing, but still the predictions are accurate enough for successfully guiding the refinement of potential face detections (see Fig. 6).

As shown in Fig. 6, the proposed detection refinement stage concentrates the effort on the most promising locations (hopefully around the ground truth face locations). This is because the number of detections to refine decreases



(a) Initialization stage (b) Initialization stage

(c) Refinement stage (step 1) (d) Refinement stage (step 1)

(e) Refinement stage (step 2) (f) Refinement stage (step 2)

Figure 6. Illustration of the detection process with the JScan method for two images (left and right column respectively). On the first row we display the centres of the initialized detections, while on the second and third rows the refined detections in the first and the second step respectively.

at each step: the ones with a score smaller than 0.50 and 0.40 for the first and second step respectively. This corresponds to detections that are closer, in terms of the Jaccard distance, than 0.50 and 0.40 respectively from the ground truth.

5. Conclusions

In this paper we present a new face detection method. We train a model to learn the Jaccard distance between a sub-window and the ground truth location. For this we generalize the boosting algorithm for binary classification to optimize any smooth loss function. Then the binary classifiers and single output regressors can be trained with the same algorithm, the only difference being in the choice of appropriate loss function.

The experimental results have shown that our face detector processes significantly fewer sub-windows than the baseline sliding-windows approach using an equivalent complex classifier. The face detection performance is improved over the baseline on the BioID dataset a detection rate 5% higher for the same number of false alarms. These encouraging results show that the idea is feasible. We plan to perform experiments on other more challenging datasets to further assess its performance.

An interesting finding is that the number of sub-windows to process decreases with the number of boosting rounds. This suggests that a more accurate Jaccard distance regressor would improve the proposed detection method as to process fewer sub-windows which results in a faster face detector. To achieve this we envisage several future works: boosting more powerful weak learners, faster optimization in respect to the number of boosting rounds and adapting bootstrapping from classification to regression.

6. Acknowledgment

The authors would like to thank the FP7 European MOBIO project (IST-214324) and the Hasler Foundation (CONTEXT project) for their financial support.

References

[1] E. Bailly-Bailliére, S. Bengio, F. Bimbot, M. Hamouz, J. Kittler, J. Mariéthoz, J. Matas, K. Messer, V. Popovici, F. Porée, B. Ruiz, and J.-P. Thiran. The banca database and evaluation protocol. page 1057. 2003. 5

[2] D. Cristinacce and T. Cootes. Boosted regression active shape models. pages xx–yy, 2007. 1, 3

[3] M. Everingham and A. Zisserman. Regression and classification approaches to eye localization in face images. pages 441–448, 2006. 1, 3

[4] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. page 178, 2004. 5

[5] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2), 2000. 2

[6] B. Froba and A. Ernst. Face detection with the modified census transform. *Automatic Face and Gesture Recognition, IEEE International Conference on*, 0:91, 2004. 1, 3

[7] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901. 4

[8] O. Jesorsky, K. J. Kirchberg, and R. Frischholz. Robust face detection using the hausdorff distance. In *AVBPA '01: Proceedings of the Third International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 90–95, London, UK, 2001. Springer-Verlag. 6

[9] L. Mason, J. Baxter, P. L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 221–246. MIT Press, 2000. 2

[10] R. E. Schapire. The boosting approach to machine learning: An overview, 2002. 2

[11] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:854–869, 2007. 2

[12] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing System 14*, pages 1311–1318. MIT Press, 2001. 3

[13] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004. 1

[14] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li. Face detection based on multi-block lbp representation. In *ICB*, pages 11–18, 2007. 1, 3