A Survey of Recent Advances in Face Detection

Cha Zhang and Zhengyou Zhang

June 2010

Technical Report
MSR-TR-2010-66

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

Abstract

Face detection has been one of the most studied topics in the computer vision literature. In this technical report, we survey the recent advances in face detection for the past decade. The seminal Viola-Jones face detector is first reviewed. We then survey the various techniques according to how they extract features and what learning algorithms are adopted. It is our hope that by reviewing the many existing algorithms, we will see even better algorithms developed to solve this fundamental computer vision problem. [1]

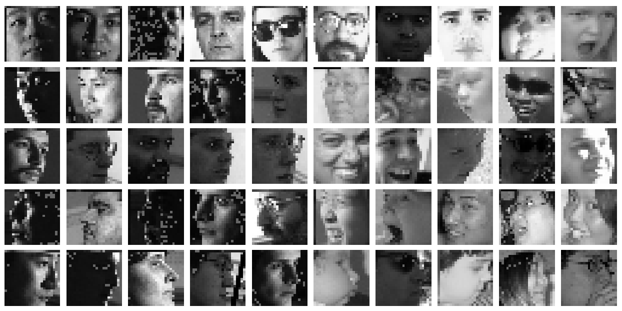
Figure 1. Examples of face images. Note the huge variations in pose, facial expression, lighting conditions, etc.

1. Introduction

With the rapid increase of computational powers and availability of modern sensing, analysis and rendering equipment and technologies, computers are becoming more and more intelligent. Many research projects and commercial products have demonstrated the capability for a computer to interact with human in a natural way by looking at people through cameras, listening to people through microphones, understanding these inputs, and reacting to people in a friendly manner.

One of the fundamental techniques that enables such natural human-computer interaction (HCI) is face detection. Face detection is the step stone to all facial analysis algorithms, including face alignment, face modeling, face relighting, face recognition, face verification/authentication, head pose tracking, facial expression tracking/recognition, gender/age recognition, and many many more. Only when computers can understand face well will they begin to truly understand people's thoughts and intentions.

Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face [112]. While this appears as a trivial task for human beings, it is a very challenging task for computers, and has been one of the top studied research topics in the past few decades. The difficulty associated with face detection can be attributed to many variations in scale, location, orientation (in-plane rotation), pose (out-of-plane rotation), facial expression, lighting conditions, occlusions, etc, as seen in Fig. 1.

There have been hundreds of reported approaches to face detection. Early Works (before year 2000) had been nicely surveyed in [112] and [30]. For instance, Yang et al. [112] grouped the various methods into four categories: knowledge-based methods, feature invariant approaches, template matching methods, and appearance-based methods. Knowledge-based methods use pre-defined rules to determine a face based on human knowledge; feature invariant approaches aim to find face structure features that are robust to pose and lighting variations; template matching methods use pre-stored face templates to judge if an image is a face; appearance-based methods learn face models from a set of representative training face images to perform detection. In general, appearance-based methods had been showing superior performance to the others, thanks to the rapid growing computation power and data storage.

The field of face detection has made significant progress in the past decade. In particular, the seminal work by Viola and Jones [92] has made face detection practically feasible in real world applications such as digital cameras and photo organization software. In this report, we present a brief survey on the latest development in face detection techniques since the publication of [112]. More attention will be given to boosting-based face detection schemes, which have evolved as the de-facto standard of face detection in real-world applications since [92].

The rest of the paper is organized as follows. Section 2 gives an overview of the Viola-Jones face detector, which also motivates many of the recent advances in face detection. Solutions to two key issues for face detection: what features to extract, and which learning algorithm to apply, will be surveyed in Section 3 (feature extraction), Section 4 (boosting learning algorithms) and Section 5 (other learning algorithms). Conclusions and future work are given in Section 6.

2. The Viola-Jones Face Detector

If one were asked to name a single face detection algorithm that has the most impact in the 2000's, it will most likely be the seminal work by Viola and Jones [92]. The Viola-Jones face detector contains three main ideas that make it possible to build a successful face detector that can run *in real time*: the integral image, classifier learning with AdaBoost, and the attentional cascade structure.

[1] This technical report is extracted from an early draft of the book "Boosting-Based Face Detection and Adaptation" by Cha Zhang and Zhengyou Zhang, Morgan & Claypool Publishers, 2010.

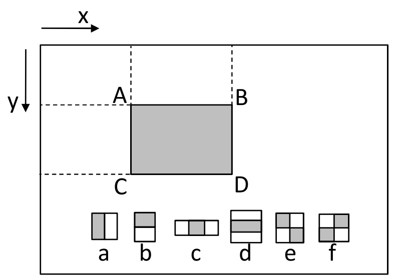

Figure 2. Illustration of the integral image and Haar-like rectangle features (a-f).

2.1. The Integral Image

Integral image, also known as a summed area table, is an algorithm for quickly and efficiently computing the sum of values in a rectangle subset of a grid. It was first introduced to the computer graphics field by Crow [12] for use in mipmaps. Viola and Jones applied the integral image for rapid computation of Haar-like features, as detailed below.

The integral image is constructed as follows:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y'), \tag{1}$$

where $ii(x,y)$ is the integral image at pixel location (x,y) and $i(x',y')$ is the original image. Using the integral image to compute the sum of any rectangular area is extremely efficient, as shown in Fig. 2. The sum of pixels in rectangle region $ABCD$ can be calculated as:

$$\sum_{(x,y)\in ABCD} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C), \tag{2}$$

which only requires four array references.

The integral image can be used to compute simple Haar-like rectangular features, as shown in Fig. 2 (a-f). The features are defined as the (weighted) intensity difference between two to four rectangles. For instance, in feature (a), the feature value is the difference in average pixel value in the gray and white rectangles. Since the rectangles share corners, the computation of two rectangle features (a and b) requires six array references, the three rectangle features (c and d) requires eight array references, and the four rectangle features (e and f) requires nine array references.

2.2. AdaBoost Learning

Boosting is a method of finding a highly accurate hypothesis by combining many “weak” hypotheses, each with moderate accuracy. For an introduction on boosting, we refer the readers to [59] and [19].

The AdaBoost (Adaptive Boosting) algorithm is generally considered as the first step towards more practical boosting algorithms [17, 18]. In this section, following [80] and [19], we briefly present a generalized version of AdaBoost algorithm, usually referred as *RealBoost*. It has been advocated in various works [46, 6, 101, 62] that RealBoost yields better performance than the original AdaBoost algorithm.

Consider a set of training examples as $\mathcal{S} = \{(x_i, z_i), i = 1, \cdots, N\}$, where x_i belongs to a domain or instance space $\mathcal{X}$, and z_i belongs to a finite label space $\mathcal{Z}$. In binary classification problems, $\mathcal{Z} = \{1, -1\}$, where $z_i = 1$ for positive examples and $z_i = -1$ for negative examples. AdaBoost produces an additive model $F^T(x) = \sum_{t=1}^T f_t(x)$ to predict the label of an input example x, where $F^T(x)$ is a real valued function in the form $F^T : \mathcal{X} \to \mathbb{R}$. The predicted label is $\hat{z}_i = \text{sign}(F^T(x_i))$, where $\text{sign}(\cdot)$ is the sign function. From the statistical view of boosting [19], AdaBoost algorithm fits an additive logistic regression model by using adaptive Newton updates for minimizing the expected exponential criterion:

$$L^T = \sum_{i=1}^N \exp\{-z_i F^T(x_i)\}. \tag{3}$$

The AdaBoost learning algorithm can be considered as to find the best additive base function $f_{t+1}(x)$ once $F^t(x)$ is given. For this purpose, we assume the base function pool $\{f(x)\}$ is in the form of confidence rated decision stumps. That is, a certain form of real feature value $h(x)$ is first extracted from x, $h : \mathcal{X} \to \mathbb{R}$. For instance, in the Viola-Jones face detector, $h(x)$ is the Haar-like features computed with integral image, as was shown in Fig. 2 (a-f). A decision threshold H divide the output of $h(x)$ into two subregions, u_1 and u_2, $u_1 \cup u_2 = \mathbb{R}$. The base function $f(x)$ is thus:

$$f(x) = c_j, \text{if } h(x) \in u_j, j = 1, 2, \tag{4}$$

which is often referred as the stump classifier. c_j is called the confidence. The optimal values of the confidence values can be derived as follows. For $j = 1, 2$ and $k = 1, -1$, let

$$W_{kj} = \sum_{i: z_i = k, f(x_i) \in u_j} \exp\{-kF^t(x_i)\}. \tag{5}$$

The target criterion can thus be written as:

$$L^{t+1} = \sum_{j=1}^2 \left[W_{+1j} e^{-c_j} + W_{-1j} e^{c_j} \right]. \tag{6}$$

Using standard calculus, we see L^{t+1} is minimized when

$$c_j = \frac{1}{2} \ln \left(\frac{W_{+1j}}{W_{-1j}} \right). \tag{7}$$

Plugging into (6), we have:

$$L^{t+1} = 2 \sum_{j=1}^2 \sqrt{W_{+1j} W_{-1j}}. \tag{8}$$

```
Input
  • Training examples $\mathcal{S} = \{(x_i, z_i), i = 1, \cdots, N\}$.
  • $T$ is the total number of weak classifiers to be trained.

Initialize
  • Initialize example score $F^0(x_i) = \frac{1}{2} \ln \left(\frac{N_+}{N_-}\right)$,
    where $N_+$ and $N_-$ are the number of positive and
    negative examples in the training data set.

Adaboost Learning
For $t = 1, \cdots, T$:
  1. For each Haar-like feature $h(x)$ in the pool, find the
     optimal threshold $H$ and confidence score $c_1$ and $c_2$
     to minimize the $Z$ score $L^t$ (8).
  2. Select the best feature with the minimum $L^t$.
  3. Update $F^t(x_i) = F^{t-1}(x_i) + f_t(x_i), i = 1, \cdots, N$,
  4. Update $W_{+1j}, W_{-1j}, j = 1, 2$.

Output Final classifier $F^T(x)$.
```

Figure 3. Adaboost learning pseudo code.

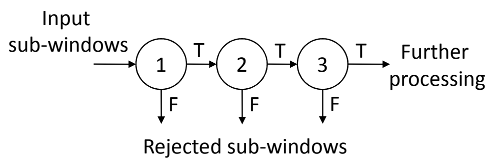

Figure 4. The attentional cascade.

Eq. (8) is referred as the Z score in [80]. In practice, at iteration $t+1$, for every Haar-like feature $h(x)$, we find the optimal threshold H and confidence score c_1 and c_2 in order to minimize the Z score L^{t+1}. A simple pseudo code of the AdaBoost algorithm is shown in Fig. 3.

2.3. The Attentional Cascade Structure

Attentional cascade is a critical component in the Viola-Jones detector. The key insight is that smaller, and thus more efficient, boosted classifiers can be built which reject most of the negative sub-windows while keeping almost all the positive examples. Consequently, majority of the sub-windows will be rejected in early stages of the detector, making the detection process extremely efficient.

The overall process of classifying a sub-window thus forms a degenerate decision tree, which was called a "cascade" in [92]. As shown in Fig. 4, the input sub-windows pass a series of nodes during detection. Each node will make a binary decision whether the window will be kept for the next round or rejected immediately. The number of weak classifiers in the nodes usually increases as the number of nodes a sub-window passes. For instance, in [92], the first five nodes contain 1, 10, 25, 25, 50 weak classifiers, respectively. This is intuitive, since each node is trying to reject a certain amount of negative windows while keeping all the positive examples, and the task becomes harder at late stages. Having fewer weak classifiers at early stages also improves the speed of the detector.

The cascade structure also has an impact on the training process. Face detection is a rare event detection task. Consequently, there are usually billions of negative examples needed in order to train a high performance face detector. To handle the huge amount of negative training examples, Viola and Jones [92] used a bootstrap process. That is, at each node, a threshold was manually chosen, and the partial classifier was used to scan the negative example set to find more unrejected negative examples for the training of the next node. Furthermore, each node is trained independently, as if the previous nodes does not exist. One argument behind such a process is to force the addition of some nonlinearity in the training process, which could improve the overall performance. However, recent works showed that it is actually beneficial not to completely separate the training process of different nodes, as will be discussed in Section 4.

In [92], the attentional cascade is constructed manually. That is, the number of weak classifiers and the decision threshold for early rejection at each node are both specified manually. This is a non-trivial task. If the decision thresholds were set too aggressively, the final detector will be very fast, but the overall detection rate may be hurt. On the other hand, if the decision thresholds were set very conservatively, most sub-windows will need to pass through many nodes, making the detector very slow. Combined with the limited computational resources available in early 2000's, it is no wonder that training a good face detector can take months of fine-tuning.

3. Feature Extraction

As mentioned earlier, thanks to the rapid expansion in storage and computation resources, appearance based methods have dominated the recent advances in face detection. The general practice is to collect a large set of face and non-face examples, and adopt certain machine learning algorithms to learn a face model to perform classification. There are two key issues in this process: what features to extract, and which learning algorithm to apply. In this section, we first review the recent advances in feature extraction.

The Haar-like rectangular features as in Fig. 2 (a-f) are very efficient to compute due to the integral image technique, and provide good performance for building frontal face detectors. In a number of follow-up works, researchers extended the straightforward features with more variations in the ways rectangle features are combined.

For instance, as shown in Fig. 5, Lienhart and Maydt[49] generalized the feature set of [92] by introducing 45 degree

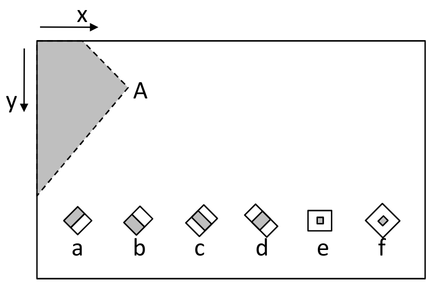

Figure 5. The rotated integral image/summed area table.

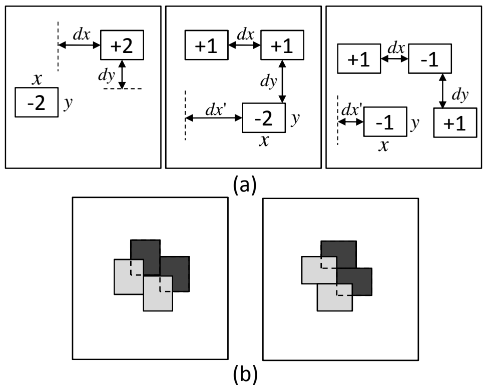

Figure 6. (a) Rectangular features with flexible sizes and distances introduced in [46]. (b) Diagonal filters in [38].

rotated rectangular features (a-d), and center-surround features (e-f). In order to compute the 45 degree rotated rectangular features, a new rotated summed area table was introduced as:

$$rii(x, y) = \sum_{x' \leq x, |y-y'| \leq x-x'} i(x', y'). \tag{9}$$

As seen in Fig. 5, $rii(A)$ is essentially the sum of pixel intensities in the shaded area. The rotated summed area table can be calculated with two passes over all pixels.

A number of researchers noted the limitation of the original Haar-like feature set in [92] for multi-view face detection, and proposed to extend the feature set by allowing more flexible combination of rectangular regions. For instance, in [46], three types of features were defined in the detection sub-window, as shown in Fig. 6 (a). The rectangles are of flexible sizes $x \times y$ and they are at certain distances of (dx, dy) apart. The authors argued that these features can be non-symmetrical to cater to non-symmetrical characteristics of non-frontal faces. Jones and Viola [38] also proposed a similar feature called diagonal filters, as shown in Fig. 6 (b). These diagonal filters can be computed with 16 array references to the integral image.

Jones et al. [39] further extended the Haar-like feature set to work on motion filtered images for video-based

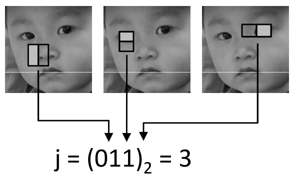

Figure 7. The joint Haar-like feature introduced in [62].

pedestrian detection. Let the previous and current video frames be i_{t-1} and i_t. Five motion filters are defined as:

$$\begin{aligned} \Delta &= |i_t - i_{t-1}| \\ U &= |i_t - i_{t-1} \uparrow| \\ L &= |i_t - i_{t-1} \leftarrow| \\ R &= |i_t - i_{t-1} \rightarrow| \\ D &= |i_t - i_{t-1} \downarrow| \end{aligned}$$

where $\{\uparrow, \leftarrow, \rightarrow, \downarrow\}$ are image shift operators. $i_t \uparrow$ is i_t shifted up by one pixel. In addition to the regular rectangular features (Fig. 2) on these additional motion filtered images, Jones et al. added single box rectangular sum features, and new features across two images. For instance:

$$f_i = r_i(\Delta) - r_i(S), \tag{10}$$

where $S \in \{U, L, R, D\}$ and $r_i(\cdot)$ is a single box rectangular sum within the detection window.

One must be careful that the construction of the motion filtered images $\{U, L, R, D\}$ is not scale invariant. That is, when detecting pedestrians at different scales, these filtered images need to be recomputed. This can be done by first constructing a pyramid of images for i_t at different scales and computing the filtered images at each level of the pyramid, as was done in [39].

Mita et al. [62] proposed joint Haar-like features, which is based on co-occurrence of multiple Haar-like features. The authors claimed that feature co-occurrence can better capture the characteristics of human faces, making it possible to construct a more powerful classifier. As shown in Fig. 7, the joint Haar-like feature uses a similar feature computation and thresholding scheme, however, only the binary outputs of the Haar-like features are concatenated into an index for 2^F possible combinations, where F is the number of combined features. To find distinctive feature co-occurrences with limited computational complexity, the suboptimal sequential forward selection scheme was used in [62]. The number F was also heuristically limited to avoid statistical unreliability.

To some degree, the above joint Haar-like features resemble a CART tree, which was explored in [8]. It was shown that CART tree based weak classifiers improved results across various boosting algorithms with a small loss

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.