SMAI PROJECT REPORT

A.Aravind Reddy
A.Raja shekar Reddy

Face Detection

Introduction :

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process. There are many applications based on Face Detection such as Facial Recognition, Photography, Marketing etc.

In contrast to Face Recognition, it just focuses on detection of faces. Primary difference is that Face Detection Algorithms focuses on locating any face on an image whereas Face Recognition focuses on recognizing the person too.

Abstract :

The problem to be solved is detection of faces in an image. The target is to take an image as input and to detect whether there is any face or not. A human can do this easily, but computer needs precise instructions and constraints. Theoretically we trained the computer by giving large set of different facial images as input, modelled a classifier and when test samples are given, it decides whether there is any face or not.

Approach:

Initially we went through several research works as the first step to learn anything is to learn about the previous work. Then we came across the famous Viola Jones Algorithm.

We followed the same algorithm ,the first ever real-time face detection system. There are three ingredients working in concert to enable a fast and accurate detection: the integral image for feature computation, Adaboost for feature selection and an attentional cascade for efficient computational resource allocation.

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

- Robust – very high detection rate (true-positive rate) & very low false-positive rate always.
- Real time – For practical applications at least 2 frames per second must be processed.
- Face detection only (not recognition) - The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

The algorithm has four stages:

1. *Haar Feature Selection*
2. *Creating an Integral Image*
3. *Adaboost Training*
4. *Cascading Classifiers*

1. Haar Like Features – All human faces share some similar properties. These regularities may be matched using **Haar Features**. They can be simply viewed as kernels which we use for edge detection.

A few properties common to human faces:

- The eye region is darker than the upper-cheeks.
- The nose bridge region is brighter than the eyes.

Composition of properties forming matchable facial features:

- Location and size: eyes, mouth, bridge of nose
- Value: oriented gradients of pixel intensities

The four features matched by this algorithm are then sought in the image of a face (shown at left).

Rectangle features:

- Value = Σ (pixels in black area) - Σ (pixels in white area).
- Three types: two-, three-, four-rectangles, Viola & Jones used two-rectangle features.
- Each feature is related to a special location in the sub-window.

So how are we going to use these Haar features? Viola Jones Algorithm uses a 24x24 window as the base window size to start evaluate these features in any given image. It takes each feature and tries to apply it right from starting pixel. By changing the dimensions it tries to find values for all features. So likewise if we consider a 24x24 window, there will be 160,000+ features playing around. This is quite a bit to decide for an image of 1024x1024 size. So there is a large requirement of dimension reduction where **Adaboost** comes to rescue.

Before that let us first see what is an **Integral Image** and what purpose it serves us.

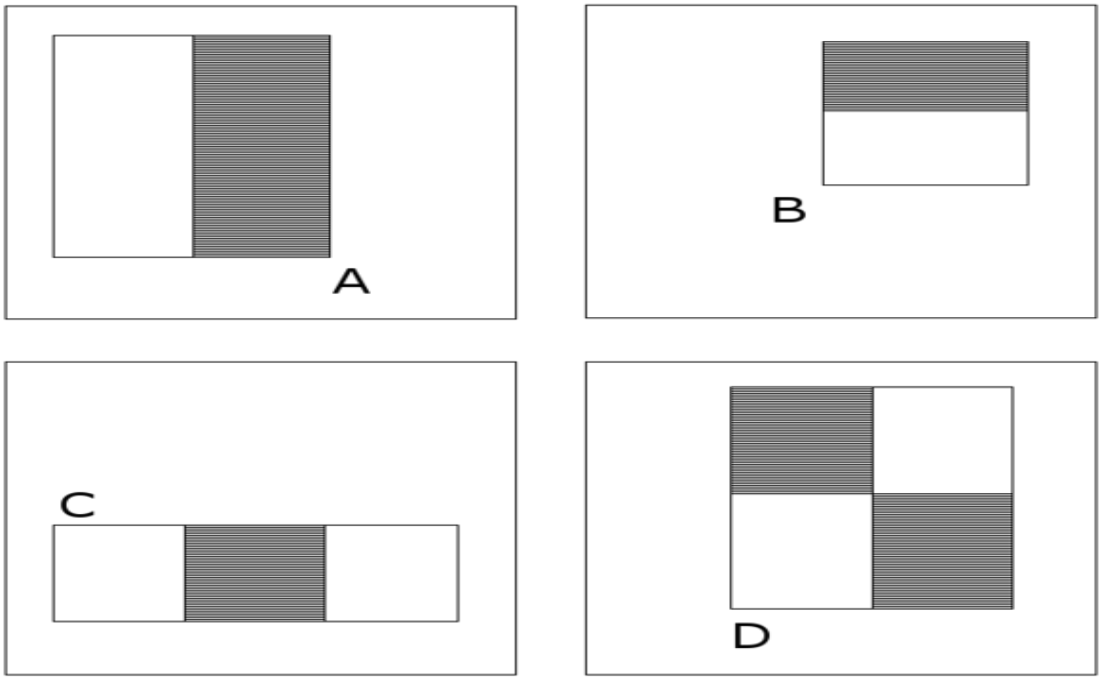


Fig1: Feature types used in Viola and Jones

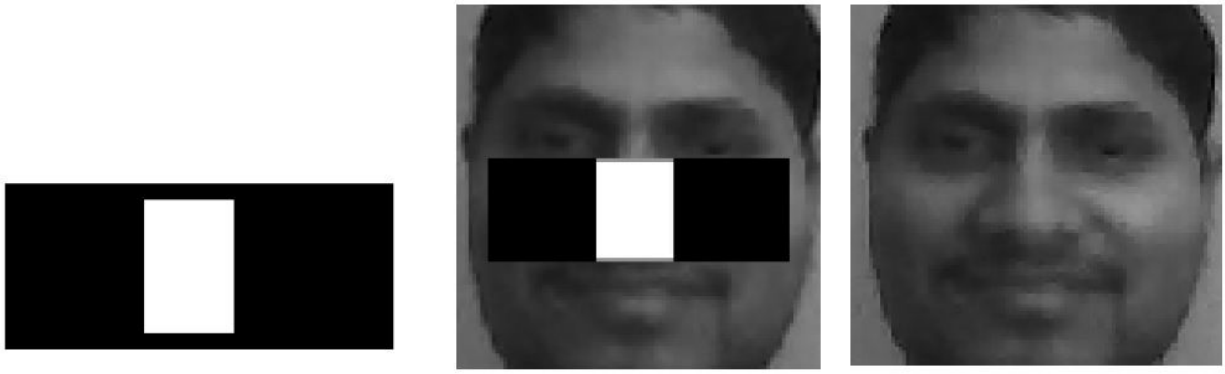


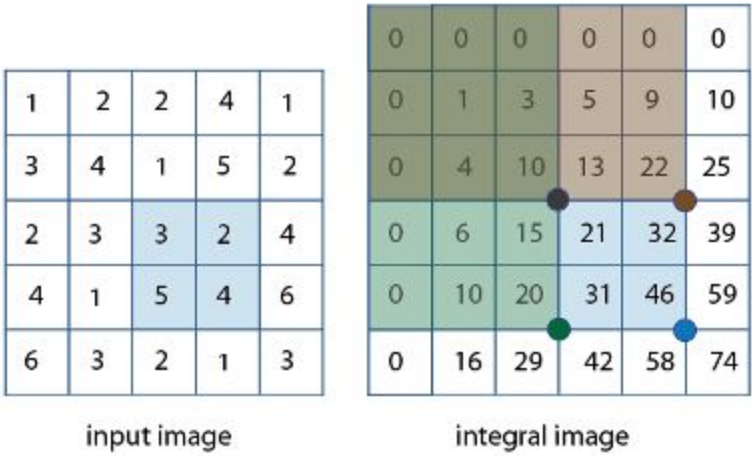
Fig:2 Haar Feature that looks similar to the bridge of the nose is applied onto the face

Fig:3 Haar Feature that looks similar to the eye region which is darker than the upper cheeks is applied onto a face

Fig:4 3rd and 4th kind of Haar Feature

2) *Integral Image* : An image representation, called the **integral image** evaluates rectangular features in *constant* time, which gives them a considerable speed advantage over more sophisticated alternative features. Because each feature rectangular area is always adjacent to at least one other rectangle, it follows that any two-rectangle feature can be computed in six array references, any three-rectangle feature in eight, and any four-rectangle feature in nine.

In an integral image the value at pixel (x, y) is the sum of pixels above and to the left of (x, y). So how does it help? It can be better explained using the below image.



This is the clear explanation for the formation of integral image from an input image. Now for suppose we need the sum of pixels in the blue region. Then we can simply do (10+46) - (20+22) instead of all additions. (here it may seem a bit awkward but in real time problems there will be many elements to add for the internal region)

3) *Adaboost* : As we already saw Adaboost is used for eliminating the redundant features. Now we know that all features are not relevant and some are irrelevant. We can remove the irrelevant features and continue with the relevant ones. Adaboost takes care of relevance and irrelevance of features. So basically Adaboost is a machine learning algorithm which helps in finding only the best features among all. After these features are found a weighted linear combination of all these features is used in evaluating and deciding any given window has a face or not. Each of the selected features are considered okay to be included if they can at least perform better than random guessing.

These features are also called weak classifiers. Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

Each weak classifier is a threshold function based on the feature f_j.

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

The threshold value θ_j and the polarity $s_j \in \pm 1$ are determined in the training, as well as the coefficients α_j. Here a simplified version of the learning algorithm is reported:

Input: Set of N positive and negative training images with their labels $(\mathbf{x}^i, y^i)$. If image i is a face $y^i = 1$, if not $y^i = -1$.

1. Initialization: assign a weight $w_1^i = \frac{1}{N}$ to each image i.
2. For each feature f_j with $j = 1, ..., M$
    1. Renormalize the weights such that they sum to one.
    2. Apply the feature to each image in the training set, then find the optimal threshold and polarity θ_j, s_j that minimizes the weighted classification error. That is

$$\theta_j, s_j = \arg\min_{\theta,s} \sum_{i=1}^{N} w_j^i \varepsilon_j^i \quad \text{where} \quad \varepsilon_j^i = \begin{cases} 0 & \text{if } y^i = h_j(\mathbf{x}^i, \theta_j, s_j) \\ 1 & \text{otherwise} \end{cases}$$

    3. Assign a weight α_j to h_j that is inversely proportional to the error rate. In this way best classifiers are considered more.
    4. The weights for the next iteration, i.e. w_{j+1}^i, are reduced for the images i that were correctly classified.
3. Set the final classifier to h(x).

4) *Cascading* :

A cascade of gradually more complex classifiers achieves even better detection rates.The evaluation of the strong classifiers generated by the learning process can be done quickly, but it isn't fast enough to run in real-time. For this reason, the strong classifiers are arranged in a cascade in order of complexity, where each successive classifier is trained only on those selected samples which pass through the preceding classifiers. If at any stage in the cascade a classifier rejects the sub-window under inspection, no further processing is performed and continue on searching the next sub-window. The cascade therefore has the form of a degenerate tree. In the case of faces, the first classifier in the cascade – called the attentional operator – uses only two features to achieve a false negative rate of approximately 0% and a false positive rate of 40%. The effect of this single classifier is to reduce by roughly half the number of times the entire cascade is evaluated.

In cascading, each stage consists of a strong classifier. So all the features are grouped into several stages where each stage has certain number of features.

The job of each stage is to determine whether a given sub-window is definitely not a face or may be a face. A given sub-window is immediately discarded as not a face if it fails in any of the stages.

Data Set Used :

Images of different facial expressions are taken from MIT CBCL Facerec Dataset for training and testing.

**********Training Set **********

Faces - 3240 images

Non Faces - 18 images

********** Test set **********

Faces - 1647 images

Non Faces - 18 images

Output :

Loading faces..

..done. 3240 faces loaded.

Loading non faces..

..done. 18 non faces loaded.

Creating haar like features..

..done.

162336 features created.

Calculating scores for features..

..done.

Selecting classifiers..

..done.==========]

Loading test faces..

..done. 1647 faces loaded.

Loading test non faces..

..done. 18 non faces loaded.

Validating selected classifiers..

..done. Result:

Faces: 1637/1647

non-Faces: 18/18

THANK YOU