2nd Report

Face Detection Techniques And Algorithms

Supervised by:
Prof. Oussama El Issati
Prepared by:
Ait Said Noureddine
Ennouali Mohamed Amine

May 13, 2017

Contents

Chapter 1

Face detection techniques and algorithms

1.1 Introduction

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Object detection algorithms are used in face detection and face recognition. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, tracking a person in a video.

The concept is that every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found[1].

1.2 Face detection and tracking algorithms

Cascading is an ensemble learning based on the concatenation of several Classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Cascading is a multistage system.

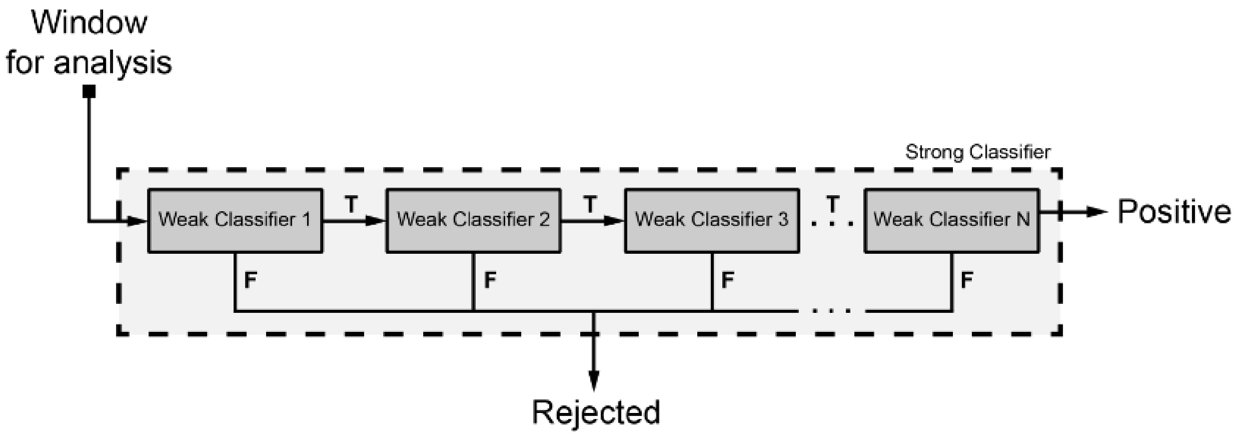
Cascading classifiers are trained with several hundred "*positive*" sample views of a particular object and arbitrary "*negative*" images of the same size.

After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location for the classifier. This process is most commonly used in image processing for object detection and tracking, primarily facial detection and recognition.

1.2.1 Cascade classification based algorithm: Viola and Jones

The first cascading classifier is the face detector of *Viola and Jones (2001).* The requirement was that the classifier be fast in order to be implemented on low CPU systems, such as cameras and phones[1].

Figure 1.1: Concept of cascade classification[5]



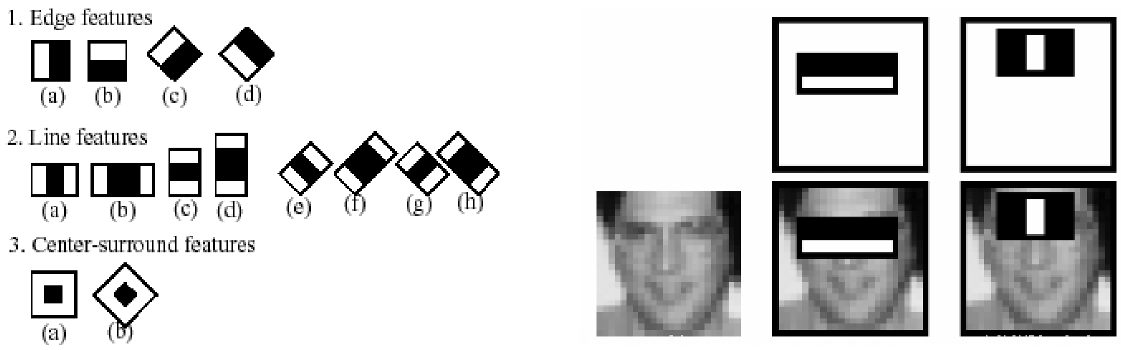
Characteristics of the algorithm

- **Robust:** the algorithm has a very high detection rate (true-positive rate) and very low false-positive rate[1].
- **Real Time:** At least 2 frames per second are processed thus making it a quick and an efficient algorithm. The algorithm comprises of four stages:
    1. Haar Features Selection
    2. Creating Integral Image
    3. Adaboost Training Algorithm
    4. Cascade Classifiers

HAAR Features

Haar-like features are digital image features used in object detection[2]. Haar features are similar to convolution kernels which are used to detect the presence of a feature in the given image.

Figure 1.2: Haar features applied to detect faces[4].



A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. [4]

There is a huge number of possible sizes and locations of each kernel. For example using a 24x24 window results over 160000 features. For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images to simplify calculation of sum of pixels[3].

Integral image

he algorithm introduces the concept of integral image to find the sum of all the pixels under a rectangle with just 4 corner values instead of summing up all the values.

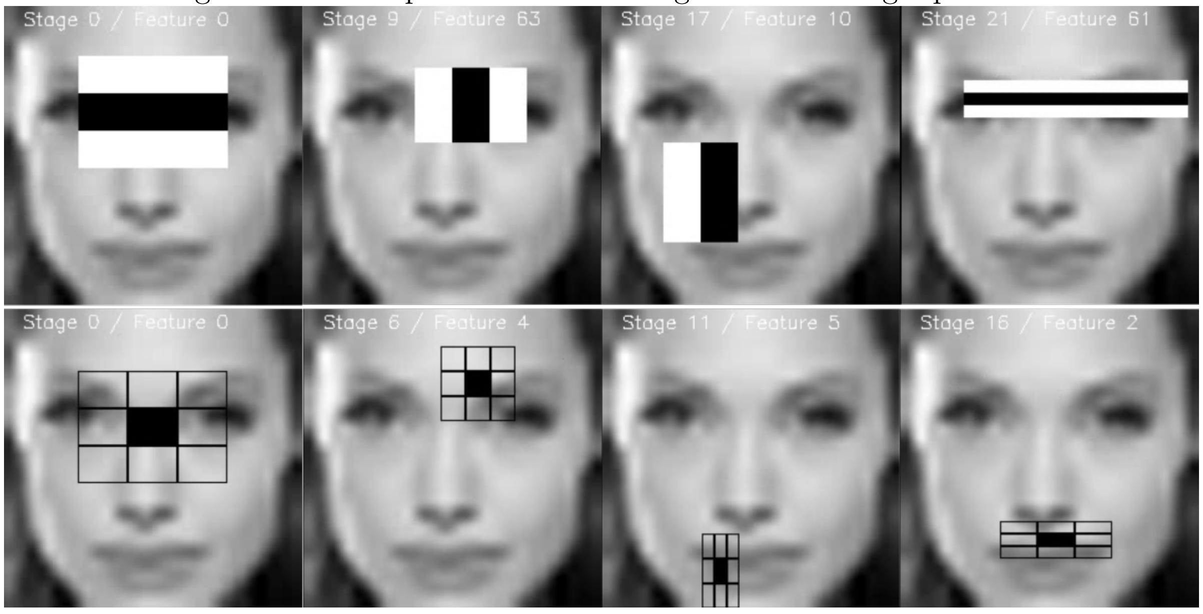
Adaboost

Since not all the features are relevant, we have to select only the best features using Adaboost. Which is a machine learning algorithm which helps in finding only the best features among all the 160000+ features. After these features are found, a weighted combination of all these features is used in evaluating and deciding if any given window (24x24) has a face or not. Each of the selected feature is considered to be included if they can at least perform better than random guessing. The little classifiers are called “weak classifiers”, Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

HAAR classifier generating

The process of generating a classifier includes two stages: the training and the detection stage. The detection stage using either HAAR or LBP based models. The library **OpenCV** comes with preprogramed applications that we can use to generate a classifier in *.xml* file, this file contains all the features that will be used to look for faces in future applications, OpenCV comes also with pre-generated *.xml* files ready to use.

Figure 1.3: The process of classifier generation using OpenCV.



Advantages

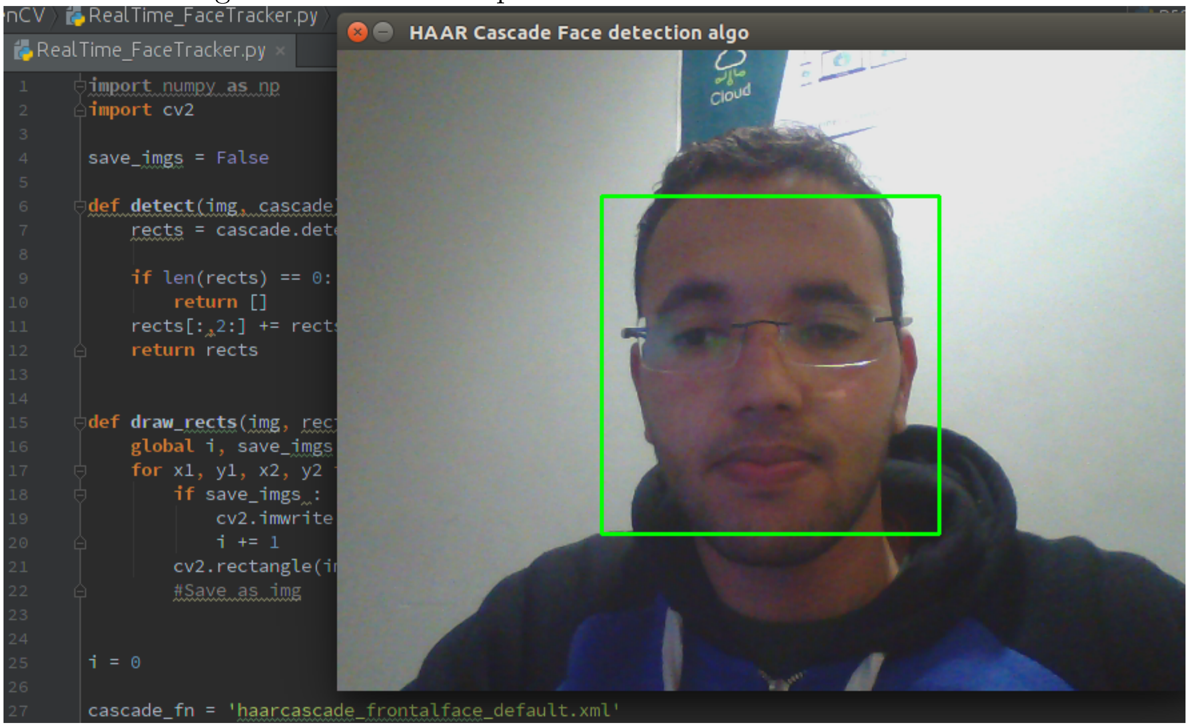
- Fast features are computed very quickly.
- The features are scaled instead of scaling the image.
- This is a generic detection scheme which can be used to detect other objects like hands, buildings, etc.

Disadvantages

- The detector is effective only in the case of frontal images of the face.
- If the face is turned 45 degrees, it fails to detect the face.
- It is sensitive to lighting conditions.
- Due to overlapping sub-windows, we might face the problem of multiple objects being detected as face.

Results

Figure 1.4: Default OpenCV HAAR cascade classifier test.



```
import numpy as np
import cv2

save_imgs = False

def detect(img, cascade
    rects = cascade.dete

    if len(rects) == 0:
        return []
    rects[:,2:] += rect
    return rects


def draw_rects(img, rec
    global i, save_imgs
    for x1, y1, x2, y2
        if save_imgs :
            cv2.imwrite
            i += 1
        cv2.rectangle(i
        #Save as img


i = 0

cascade_fn = 'haarcascade_frontalface_default.xml'
```

1.2.2 CAMshift Algorithm

CAMshift is a tracking algorithm, which is based on MeanShift algorithm, what CAMshift does meanShift in every single frame of a video, and record the results we got by MeanShift. CamShift algorithm includes these three parts[6] :

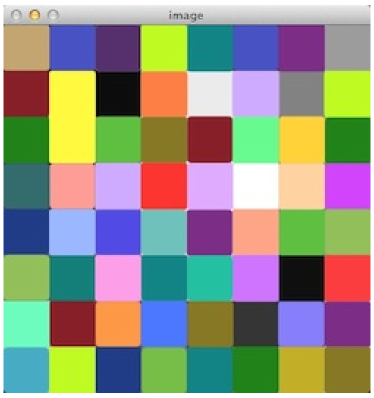
1. Back projection
2. Applying MeanShift
3. Tracking

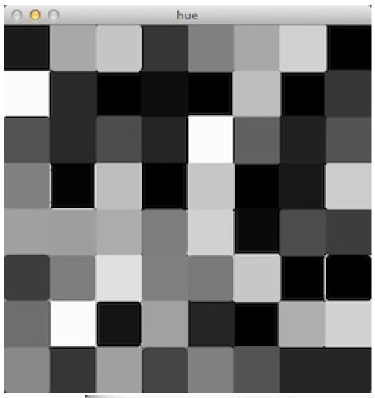
Back Projection

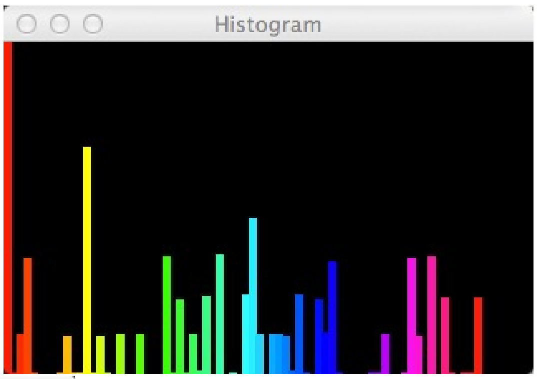
Back projection is a method using the histogram of an image to show up the probabilities of colors that may appear in each pixel. First we transform the picture space to HSV space which is a cylindric color base (or any space which include an H channel that represent the hue of each pixel, of course, value of hue is between 0 to 180). Secondly, we split the H channel out, as a single grayscale image, and get its histogram, and normalize it. Thirdly, we use "calcBackProject()" function to calculate the back projection of the image.

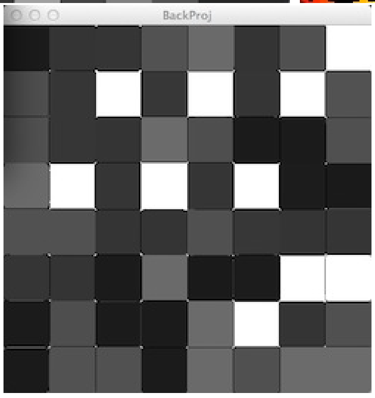
Example:

This is an example to explain how we get the back projection. We transform the picture into HSV space and the second image shows the hue channel. The third image shows its histogram. We calculate the weight of each color in the whole picture using histogram, and change the value of each pixel to the weight of its color in whole picture, the result of this step is shown in the last image.









Applying MeanShift

MeanShift is an algorithm which finding modes in a set of data samples representing an underlying probability density function, so the whole algorithm is:

1. Initialize the sphere, including the center and radius.
2. Calculate the current mass center.
3. Move the sphere's center to mass center
4. Rrepeat step b and c, until converge, that is, current mass center after calculate, is the same point with center of sphere.

Track

The last step is tracking, if we have a video, or frames captured by our web camera, what we need to do is just use meanShift algorithm to every single frame, and the initial window of each frame is just the output window of the prior frame.

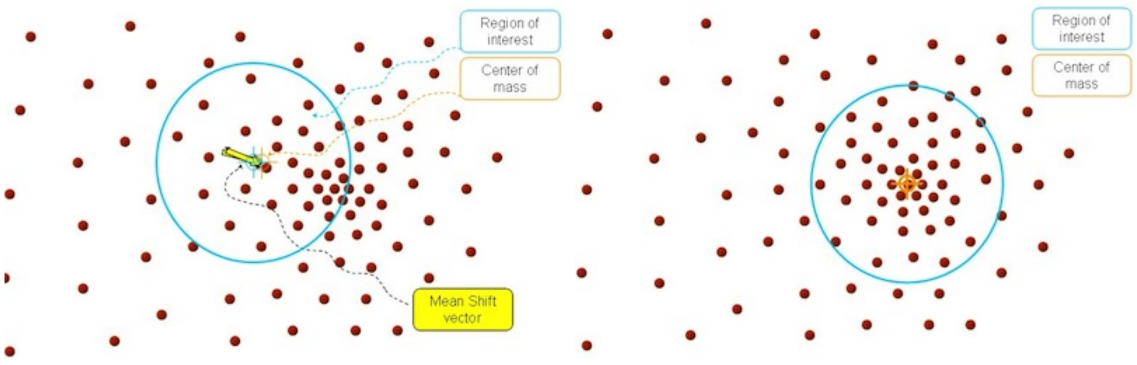


Figure 1.5: The meanshift method.

Bibliography

[1] *Wikipedia.*
`https://en.wikipedia.org/wiki/Object_detection`
`https://en.wikipedia.org/wiki/Computer_vision`

[2] P. Viola, M.J. Jones *"Robust Real-Time Face Detection, International Journal of Computer Vision"*, Vol. 57, No. 2, May 2004.

[3] *"Face Detection and Tracking: A Comparative Study Of Two Algorithms"* Sonia Mittal, Chirag Shivnani
`http://csjournals.com/IJCSC/PDF7-1/11.%20Sonia.pdf`

[4] *OpenCV Documentation Website*
`https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html`
`http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html`

[5] *"A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery"*
`http://epacis.net/jcis/PDF_JCIS/JCIS11-art.0101.pdf`

[6] *"Continuously Adaptive Shift"*
`http://eric-yuan.me/continuously-adaptive-shift/`