

2nd Report

Face Detection Techniques And Algorithms

Supervised by:

Prof. Oussama El Issati

Prepared by:

Ait Said Nouredine

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[?].

1.2 Face detection 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.

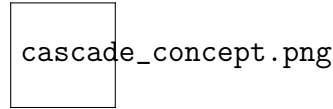
Cascading classifiers are trained with several hundred “*positive*” sample views of a particular object and arbitrary “*negative*” images of the same size.

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[?].

Figure 1.1: Concept of cascade classification[?]



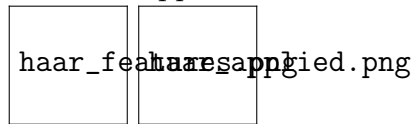
Characteristics of the algorithm

- **Robust:** the algorithm has a very high detection rate (true-positive rate) and very low false-positive rate¹.
- **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[?]. 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[?].



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. [?]

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[?].

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.



haar_stage_vis.png

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.



haar_test.png

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[?] :

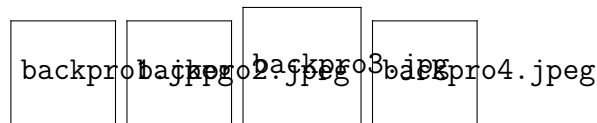
1. Back projection
2. Applying MeanShift
3. Tracking

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. Repeat step b and c, until converge, that is, current mass center after calculate, is the same point with center of sphere.

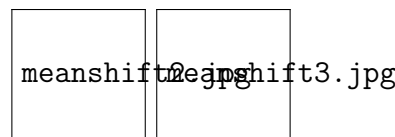


Figure 1.5: The meanshift method.

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.

1.3 Conclusion

As a conclusion, CAMshift and HAAR Cascade are two different methods used for detection ; the first one is used for tracking the specified object detected with the second method, so even if there is a permanent movement of the object that we want to detect ,with the collaboration of this two methods the detection will be succeeded.

Chapter 2

Face recognition techniques and algorithms

2.1 Introduction

We have developed a near-real-time computer system that can locate and track a subject's head, and then recognize the person by comparing characteristics of the face to those of known individuals. The technology of face recognition has become mature within these few years. System, using the face recognition, has become true in real life. In this paper, we will have a comparative study of three most recently methods for face recognition. One of the approach is eigenface, fisherfaces and other one is the Local Binary Patterns Histograms (LBPH). After the implementation of the above three methods, we learn the advantages and Disadvantages of each approach and the difficulties for the implementation.

2.2 Face recognition methods

2.2.1 The Eigenfaces method

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images.

Individual features

- eyes, nose, mouth, head outline
- position and size relationships

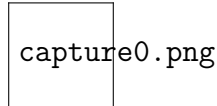
Disadvantages

- multiple views
- fragile and complex

The Eigenface approach

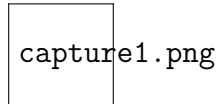
- * images are points in a vector space

Figure 2.1: The Eigenfaces method[?]



- * use PCA to reduce dimensionality
- * face space
- * compare projections onto face space to recognize faces

Figure 2.2: The Eigenfaces approach [?]



2.2.2 The Fisherfaces method

Fisherface Concept-Differing from the Eigenface concept, the fisherface method tries to maximize the ratio of the between-class scatter versus the within-class scatter . The result of this shapes the projections so that the distances between the classes are at a maximum, while the distances between samples of the same class are at a minimum. A possible disadvantage is if the between-class scatter is large, then the within-class scatter might also still be of a relatively large value.

conclusion

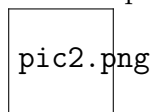
Eigenfaces and Fisherfaces take a somewhat holistic approach to face recognition. You treat your data as a vector somewhere in a high-dimensional image space. We all know high-dimensionality is bad, so a lower-dimensional subspace is identified, where (probably) useful information is preserved. The Eigenfaces approach maximizes the total scatter, which can lead to problems if the variance is generated by an external source, because components with a maximum variance over all classes aren't necessarily useful for classification. So to preserve some discriminative information we applied a Linear Discriminant Analysis and optimized as described in the Fisherfaces method. The Fisherfaces method worked great... at least for the constrained scenario we've assumed in our model. Now real life isn't perfect. You simply can't guarantee perfect light settings in your images or 10 different images of a person. So what if there's only one image for each person? Our covariance estimates for the subspace may be horribly wrong, so will the recognition. Remember the Eigenfaces method had a 96

The Eigenface approach

So in order to get good recognition rates you'll need at least 8(+1) images for each person and the Fisherfaces method doesn't really help here. The above experiment is a 10-fold cross validated result carried. So some research concentrated on extracting local features from images. The idea is to not look at the whole image as a high-dimensional vector, but describe only local features of an object. The features you extract this way will have a low-

dimensionality implicitly. But you'll soon observe the image representation we are given doesn't only suffer from illumination variations. Think of things like scale, translation or rotation in images - your local description has to be at least a bit robust against those things. the Local Binary Patterns methodology has its roots in 2D texture analysis. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. You'll end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels you'll end up with 2^8 possible combinations, called Local Binary Patterns or sometimes referred to as LBP codes.

Figure 2.3: LBPH principe[?]



2.2.3 Local Binary Patterns LBP

Local Binary Pattern used in face recognition was proposed by Ahonen . The method provide information about the shape and the texture. The original LBP operator labels the pixels of an image by thresholding the 33-neighbourhood of each pixel with the center value and consider the results as a binary number, and we make a histogram to describe the image. Here we use the extension to original operator called uniform pattern. A LBP is called uniform if it contains at most two bitwise transitions from 0 to 1. With this criteria, the number of bins of different patterns reduced from 256 to 59, 58 bins for different uniform patterns, and one bin for nonuniform patterns. A image is divided by $p \times p$ patches, each of them have a histogram of LBP. The histogram will be concatenated as a $p \times p \times 59$ vector feature descriptor. The parameters are the divided number p , and the input images pixel d . In the experiment, we find out if p is bigger, the result always be better. The image size 64×64 is most suitable for all cases. In the end, we decide $d = 64 \times 64$ and $p = 8$ for testing. Moreover, LBP reaches a highest accuracy.

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/>