COMP4102 Project
Face Detection
Qihang Peng
101013464

Abstract

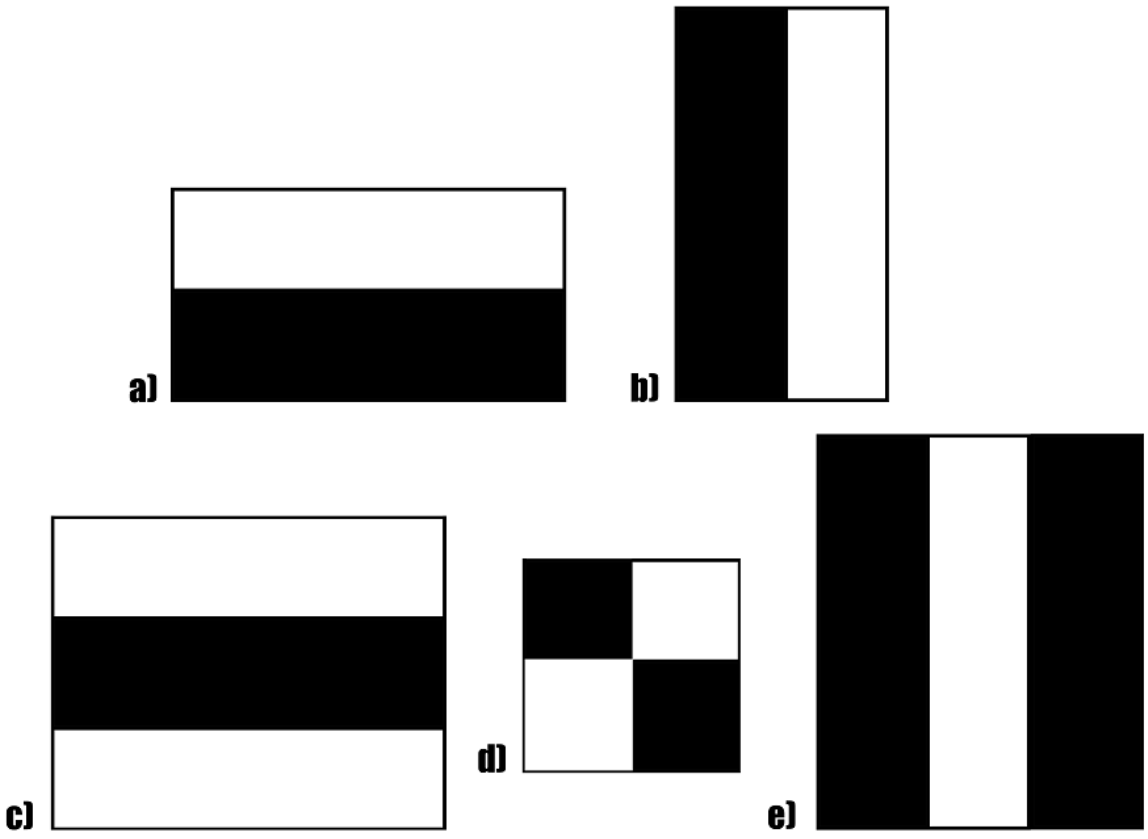
For this project, I attempted to implement a face detection(in image) program using python3, openCV and numpy.

Introduction

The initial plan was to do an emotion detection. After doing some research, I found that work can be divided into two part, face detection and emotion detection. Then I was facing a choice between two options since doing both is a bit too challenging for me. First option is to implement a face detection from scratch. Second is to use pre-trained cascade classifier to detect face, then implement emotion detection. I went with the prior. After some research I found that there are several ways to detect face. The most common one is to train a cascade classifier and use the trained classifier to detect face, but just passing command to openCV to train a classifier is not a good project. Therefore the method I used is to use Haar filters to locate features that matches human face structure.

Background

It was discussed in Viola-jones Face Detection Technique, also known as Haar cascade, an object can be detected using some filters show below(there are more shapes that can detect



round object and more). The idea was to locate some common features in human face like, eyes, nose, mouse, etc. to detect a face.

Methodology

Haar Filter

Firstly, some Haar filters that are useful in face detection are defined(a, b, c from image above). Then to use these defined filter, need to use the sum of pixel values of the white area subtract sum of the pixel of black area. The greater the result, the more the feature matches the structure of the filter. Next a threshold is needed to determine if the feature is a match or not.

Integral Image

Calculating the sums will be a lot of computation since the filters needs to scan through the whole image with different sizes. Therefore a technique called integral image is used. Integral image is not an image but an image-like(2d) list. Each value in the integral image represent the sum of the value from top left pixel to the corresponding pixel in the original grey scale image. To calculate the integral image we only need to scan through the grey scale image once. Then any sum needed for Haar filter can be calculated from integral image without looping.

Face Features

Haar filters can be used to detect certain feature on human faces. Filter (a) can be used to detect eyebrows. Filter (b) and (e) can both be used to detect noses. After observer some face images that are turned to black(0) and white(255) with threshold set to 110, it is found that depending on direction of light, the nose is shown differently in the black-white image. For example, if the light comes from one side of the face, the other side would be black therefore filter (e) cannot be used in this case.

Since filter (b) is a more general case of filter (e), we used filter (b) for noses. Also, in some cases, if the face is directly facing the light, there would be no clear horizontal edge, therefore, we also use filter (a) to detect the shade of bottom of the nose.

For mouse, we use filter (c) since it is quit clear that in most of the cases, mouse is just a thick horizontal line(with some curve).

Now we have chosen what filters we are using, next steps is where to put these filters. Since the exact location of each feature varies in different faces, we can only set the area we are looking for theses features. Given a face, eyebrows are located on top half of the face, so we detect one eyebrow in top left and one in top right of the scan window. For nose, it is located in middle so we divide the scan window to a 3 x 3 window and we try to find nose in the middle. Lastly, mouse is in bottom half of the window. Normally the mouse would be in the middle so we divide bottom half to 5 even parts and mouse should be in the middle 3 parts.

Now the location of the features are set, then we need to set the size of each filter. Since the image is scanned with different sizes of windows, the size of the feature is bond to the size of the window. For example, the width of a mouse is usually 3/5 of the face size, so we set the filter width to be 3/5 of the scan window width. Also, the sizes of the features shown in image is also effected by the brightness of the image, therefore the ratio of the each feature varies.

Once these are all set, we just need to scan the features in the corresponding areas in the window, if all the features are detected, then we say a face is found(there will be duplicates and false positives, these issues will be mentioned in limitation section). For noses, since we have 2 types of filters used, if any of the two filters detect a nose, we say we detected a nose. And if there are at least one of each feature, we say we detected a face.

Scanning

We can now find face in a scan window, then we need to scan the image with different window size. Since scanning the whole image with all possible window size is extremely computation expensive, we set the ratio of the scan window to be 4 x 5(width x height), and initial scan window size is 40 x 50 pixels. Then after each scan loop, the size of the window expand 10% until the window size exceed the size of image. To further reduce computation. The scan window moves 10% of window size. For example, if the window size is 40 x 50, each time after a scan, it moves right 40*0.1 = 4 pixel, after it finishes a line, move down 50*0.1 = 5 pixels. If the window size is 100 * 125, it moves 10 pixel horizontally each time.

If a face is found in a window(all required features are found), store the coordinates and sizes of the window. Then we apply non-max expression to reduce duplicates. The left sets of faces are the final results.

Result

Face are detected in some test cases. There are still duplicates and false positives.

Limitation

The face detection would be more accurate(reduce false negatives) if we can detect a face with all possible filter size in a scan window, but limited by computation time, the filter sizes are fixed(only change depend on size of scan window). There are false positives detected since the structure of features we are detecting are quit simple and common. We could add more detail to the structure, for example, a filter (e) for the space between the eyebrows. Also the sizes of features varies therefore there are false negatives.

Additional Attempt

I have also attempted to write a cascade classifier(thus some files related to training a classifier), but I have little knowledge of machine learning. Therefore the plan was aborted.