Designing and Testing an Anonymous Face Recognition System

Joris Diesvelt
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
j.j.diesvelt@student.utwente.nl

ABSTRACT
This paper contains a research in anonymous biometrics. It describes what basic components are needed to build an anonymous face recognition and different options will be discussed. For example, we will take a look at the Viola-Jones object detection method. The algorithms eigenfaces, fisherfaces and local binary patterns will also be discussed. The software library OpenCV is briefly introduced and when the preferred methods and tools are chosen, the software will be modeled and implemented. There is a step by step explanation of all the building blocks in this system. Eventually, the system will be tested with some video sequences from the ChokePoint database and the results will be interpreted.

Keywords
Biometrics, anonymous face recognition, OpenCV

1. INTRODUCTION
Since the uprising of digital security, company’s have put enormous effort into increasing the effectiveness of biometrics. Many practical applications are already used all over the world. Fingerprint readers and iris scanners probably sound familiar to the average reader. Another form of biometrics is face recognition. The concept of face recognition has existed for quite some time and there are already several commercial products available which claim to be able to detect and recognize faces with high accuracy. It is, for example, used as a security measure by authenticating users or in video surveillance. Although these applications are already quite advanced, there are still challenges to the improvement of these systems.

One of these applications is anonymous face recognition. It is basically the detection and recognition of people while no personal data of these people is present in the same system. This is most useful in public places where one would want to track the movements of subjects throughout an area and no prior data of these persons are present. If a computer can recognize people within this area, it can analyze their behavior and indicate whether the behavior is appropriate or out of place. Some examples of this are: "Is this person allowed to be in this area?", "How long are these people standing here, and is this normal?".

In this research, a basis of an anonymous recognition system will be built. This system will be able to detect people and try to recognize them in later occurrences. It won’t go as far as analyzing the behavior of these people, but will primarily focus on how well the system works. This is done by defining certain use cases in which the system will be tested. The results from this research could be used in further research which can then expand on further possibilities of this system.

The main goal of this research is to create and test an anonymous biometrics system. Because we cannot test all aspect of such a system, we constrain it by designing the system only for certain environments. These will be defined as the use cases in which the system is going to be tested. We can then conclude the main research question as follows:

- How well does an anonymous biometrics system, using state-of-the-art software, behave under certain use cases.

The use cases in this context are the environment and the setup of the system in this environment. For the purpose of this research, we choose an indoor location with a number of cameras connected to a client which runs the developed software. The most simple setup will be a single camera and the detection of a single person within its range. We can extend this setup by using more cameras and detecting multiple faces per camera. In both cases, the performance of the system will depend on the software.

2. METHODS & TOOLS
Before we can build face recognition software, we must first understand some logic behind it. Every face recognizer requires some building blocks that are based on algorithms. The basic components of our program will consist of a detector, a normalizer and a recognizer. In this section, we will explain these components and also try to find different solutions to achieve their functionality.

The first step of face recognition is the detection of faces inside an image. There are a number of ways to achieve this, but one of the most common and robust method is the Viola-Jones method [6]. This method makes use of the unique features present in objects. To make proper use of it however, the algorithm must be trained with datasets. These datasets can be created for any given object, including a persons head, eyes, or even ears and noses. Creating a dataset costs effort and time, but luckily many have already been created. Some of these, however, work better than other. Therefore, the Viola-Jones method is only as strong as the dataset it is given. The reason we want to
use the Viola-Jones method because it has a high detection rate which makes it possible to process more data than most other methods [2]. This is important for us, because face detection is the first step in our process and we can therefore not afford to lose much data in this process. If we cannot accurately detect faces, it will be very difficult to process them later on.

After a face has been successfully detected, the image will be normalized. There is a good reason why we cannot skip this step. Face recognition is based on the features that can be extracted from human faces. To provide good results, the features must be as homogeneous as possible. Using different techniques, this can increase the accuracy of the recognizer in many ways [4]. These techniques tend to alter the images by changing color, contrast, illumination etc. but they can also use deformation techniques, like rotation, scaling or translation. The purpose of this research is not to find the absolute best options for our program, therefore we use only a few basic techniques. These include image rotation, scaling, grayscaling, equalizing histograms. They will all be discussed in the section Software Design.

The recognition of faces from pictures is a very different topic. Now that we can create normalized images we have to compare them to each other to determine the likelihood of these pictures having the same person on them. Again, a number of algorithms have to be considered. A very basic algorithm is the eigenface method, which uses principal component analysis (PCA) [5]. This algorithm takes a database of images and creates a set of eigenfaces. In the most simple explanation, these eigenfaces are the average images of the faces in the database. A new image can then be compared to these eigenfaces by calculating the "distance" between them. Based on this distance, a probability, or confidence score, can be given to the comparison. The eigenface method is notable for its speed but unfortunately is very sensitive to lightning conditions and the pose of the face. A different technique called "fisherfaces" improves the results by tackling these problems [1].

Although these algorithms can work very well, they have a downside. When the database is updated with new faces, the algorithm must be trained again with the new set of images. Because we are creating an anonymous face recognition system, we constantly have to update the database with new information. Every time a new person is captured, the database has to be updated and the so does the algorithm. Therefore, another algorithm called Local Binary Patterns might be more useful to us. This algorithm detects patterns in images by dividing them into cells. It then compares the neighboring pixels to these cells and stores that information [3]. Two images can be compared by examining the differences in their patterns. The algorithm can be used to identify the closest image in a database to a given image, as well as providing the certainty, or likeness, that this is correct.

3. SOFTWARE DESIGN

Now that we understand our basic components for the face recognizer, it is time to model and implement the software. We find ways to use the previously discussed methods in our program and add more detail to their actual usage. But first, we have to get acquainted with some necessary tools.

3.1 Development environment

Building the entire system by ourselves would be very difficult, not to mention time consuming. This is beyond the scope of this research and we will therefore make use of a library called OpenCV (Open source Computer Vision). This library is maintained by a number of people and updated regularly. The purpose of this library, or API as some would call it, is to help programmers use difficult algorithms with more ease. As the name suggests, OpenCV focuses mainly on image processing and contains more than enough suitable software for this project.

The next piece of software we use is the JavaCV wrapper. This wrapper makes it possible to use OpenCV in the Java language, by creating callback functions that hook back to the native functions in OpenCV. JavaCV is also open source and is maintained by enthusiasts.

There are a few pros and cons to coding in the Java language. First of all, it is easier to write, compile and debug. Since this research is constrained by a number of weeks, time is of the essence and learning a new language can be very time consuming. Java is also platform-independent, which means the application can be transferred to almost any system. The downside of using Java is its inability to match the speed of languages like C++. But in our case, we do not have to worry about performance too much.

3.2 Abstract design

In the Methods & Tools section, we talked about the basic components that we need to create our anonymous recognition software. We can define an abstract model of the program around these basic components as can be seen in Figure 1. We start by capturing frames from a video input. This can be either a direct link to a camera or a video file. Both will be useful when testing and using the software. The frame will then be passed through the detector which returns the faces discovered in the image. It will also try to detect the eyes of these faces. The frame and the detected faces/eyes then go through the normalization, which returns a normalized image that can be used in the recognizer algorithm. This recognizer then compares the normalized image to all images acquired before. As a result, it gives a prediction of the image it looks like most, alongside a confidence score. At last, we must choose whether we agree on the prediction and register this person as a match, or determine that we encountered a new person.

3.3 Detection

Previously, we determined the Viola-Jones method the most suitable solution to face detection. OpenCV offers both an interface for this algorithm and the training data to recognize faces and eyes. To ensure that we get the best results out of this algorithm, we control the output by performing a few checks. For example, in Figure 2 we can see how the output from the face detection is used in the eye detection. The eye detector checks if the eyes found are within the boundary’s of the previously found
When at least one face has been found in the frame, we can start the normalization process. First we try to rotate the image to the normal "straight" position. This can be done by calculating the center point between the eyes and then the rotation between the center and one of the eyes (Figure 3). A transformation matrix is then created from this rotation and applied to the image. If the detector failed to detect eyes in this image, we have no choice but to skip this step.

After the rotation we cut the face out of the original frame. Thereafter, we convert the cut image to grayscale and equalize the histogram of the image. OpenCV has the necessary functions to do this almost automatically. Finally, we adjust the size of the image to a default rectangle. This ensures that all images in the database have the same dimensions.

If all previous steps are done correctly, we now end up with a series of images of subjects. We can now use the OpenCV’s FaceRecognizer library to compare our new faces. This library stores an internal image database that we can update with new images. It also supports multiple algorithms, including LBP.

When we give the face recognizer an image, he returns a prediction of the person he thinks looks most like the given image. He also returns the confidence score of this prediction. The lower this score, the more the given image looks like the predicted image (a score of zero means the image is exactly the same as the predicted image). With these two attributes, we can choose one of the following actions:

- If the confidence score is below a certain threshold, we assume the prediction is correct.
- If the confidence score is above a certain threshold we assume the prediction is wrong.

If the program is not confident enough to give an accurate prediction, we determine that there was not a match. This can have many causes. The input frame might have been blurred or otherwise of bad quality for example. The detection might also have gone wrong and of course, the person could simply not have been added to the system yet. This brings us to the last option:

- If the confidence score is above a certain threshold and we have a high quality picture, we assume the system encountered a new subject.

Naturally every person has to be added to system first before he/she can be recognized. This option allows us to do so. Note that we explicitly say that we need a high quality picture before update the face database. If lower quality images would be allowed, the system could become clouded with bad images and the accuracy of predictions will lower. We can determine whether a picture is of high quality or not by looking at the previous processes. If, for example, the detection process was able to find eyes in the picture, we can be fairly certain that we are dealing with a picture of a human. This also means that we were able to rotate the image to the correct position and thus improving the quality of the picture.

From the listed options we can see that, for the system to work, we need to determine certain thresholds. These thresholds differ largely depending on the environment the system is used in. Therefore we need to gather test data to determine the optimal thresholds for our use case.

4. VERIFICATION AND TESTING

To verify the workings of the program, test data needs to be acquired. Then the previously discussed thresholds can be adjusted to optimal levels. Once we know that the system is working as expected, a larger experiment can be conducted.

4.1 Adjusting thresholds with test data

For the initial test data, we ask 7 subjects to randomly move around an area. In this area, a camera records the movements of these subjects. This recording is later used as input for the software. From this simple test, a few things can be concluded. First, the detection of eyes is a lot harder than initially thought. The image resolution must be decent and the image cannot be blurry, otherwise eyes will not be captured. It is also difficult to detect to eyes of people wearing glasses.

Aside from this problem, it is also very hard to determine specific threshold for the recognizer. Since the face database grows over time, the recognizer has more images to compare to. As a consequence, the average confidence score lowers. To counterbalance this, the threshold should also be lowered. It is, however, not easy to decide how much and this is also outside the scope of this research. We will therefore decide on a constant values that provides the best results given the test data.
4.2 ChokePoint Experiment

Now that we have used the system in practice and improved it where needed, a larger experiment can be carried out. In this experiment, we will make use of a database with videos based on previous research called ChokePoint [7]. This dataset is specifically created for person identification in video surveillance. Every video is shot above a door which acts as a portal (hence the name ChokePoint). A variety of subjects then walk through the portal in an natural way. Each portal has different illumination conditions and has a different pose. This makes it challenging to use a face recognition system with multiple camera’s.

For this experiment, we choose six different videos. These videos are divided over two portals, P1 and P2. In these videos, 25 subjects are present. In each video, they walk along the camera at least once. The system takes these videos as input and analyzes them frame by frame. It will then output whenever a new person is added to the database, or when a person is recognized. To verify the output of the system we check it by hand. When a person is recognized as someone else, we count this as a false positive. However, when the output is correct, we count it as a correct result. When the experiment is complete, we can make conclusions about the accuracy of the system.

4.3 Results

Table 1 shows the results from the first video sequence. The first row shows the subject id, sorted by number. The second row shows the number of newly created items in the database. It can be observed that 14 subjects are added to the database. The third row displays the number of false positives and the fourth row the number of correct recognition. It is interesting to see that all subjects that are added to the database are correctly recognized throughout the video sequence. Three subjects, however, do not appear to be added but have some characteristics with other subjects. A final thing to observe is that 8 subjects are not recognized at all. These people could not be captured very clear on camera or they had certain features that prevented the system from recognizing them. These features included glasses and a long hairstyle.

Table 2 contains the same type of results. Again, the subjects id’s are shown, as well as the number of creations, false positives and correct recognitions in this video sequence. What is interesting to note is that, even though this is the last sequence, there are still new items in the database created by the recognizer. This is because the person is not recognized, but the image quality is good enough to assume it is a new person. This behavior is also observable in other sequences. These extra creations can be suppressed by increasing the threshold that allows creations, but this will in turn result in other “good” creations being suppressed.

Furthermore, we can see that there were no false positives in this sequence. Because many pictures where added to the database during the first five sequences, the chances of false positives have decreased. This is a desired effect when an anonymous face recognition system works in an area for a longer period of time. However, the duplicate creations could cause an increase in the chances of false positives because the database because clouded with reference material from many subjects. It would therefore be wise to delete old items from the database when there have not been used for a longer period of time.

Some subjects in this experiment where never registered or recognized. This can depend on both the quality of the video, or the detection rate of the recognizer. From the initial test phase, we could already see that some people are harder to detect than others. Apparently some people are almost invisible to the system.

5. CONCLUSIONS

In this research we tried to answer how well an anonymous biometrics system can perform. We discussed different methods of detection and recognizing faces. We chose the Viola-Jones method for face and eye detection because of its robustness. We considered the eigenfaces and fisherfaces for face recognition but chose the local binary pattern algorithm because of its ability to constantly update its database. We have seen which components an anonymous biometrics system needs and we modeled and implemented them step by step.

After creating the system and testing it, we can conclude that, as we can see from the test results, the system is far from perfect. It sometimes falsely registers a person double or detects a false positive. Some people are never even detected.

However, with the combined results from the initial test and the ChokePoint experiment, we can get a basic idea of the factors that influence the behavior of the face recognizer. We have seen, for example, that the threshold to the confidence score can make quite an impact. This knowledge can be used in further experiments improve the accuracy and dependability of an anonymous face recognition system.

6. REFERENCES


