A General Review of Facial Recognition Technology

1.Introduction

For us human, facial recognition (FR) is a trivial test. When you look at somebody you know, you input the his/her face to the brain by eyes, and then the brain analyzes the facial feature, searches for the information about him/her you know, and then tells you who he/she is. This article is not something about the biology, but about the FR technology in computer. Every single person can just know and remember a small number of people since our brains do not have enough memory to store the information of so many people. That is a reason why we would like to develop computer FR technology: it can store more data and access them quickly and accurately.

FR has wide application in daily life. For example, facial lock in cell phone can help you unlock the phone quickly without inputting the password and prevent strangers to unlock it; FR system in government and custom can help the staffs get your information and recognize that you are you; FR in CCTV camera can identify the fugitive's information and help the police find them sooner. All these tell us that FR can make our life more convenient. Besides, there is some great controversy on FR. Many people do not want to be monitored anytime and anywhere, and worry that FR would violate the privacy. This article is a review about the FR technology and therefore would not include the social problem of FR.

In computer, FR can be divided into two problem: Detection and Recognition. In the detection part, the computer should learn "what is face?", that is, the computer should detect the existence and location for the face from an input image. In the recognition part, the computer should learn "who is the face?", that is, after detection, the computer should match the face to a specific man in a given database. Although FR is an easy work for human, how can the computer learn the faces is a complex problem in the past 40 years. The main reason is that human face is not consistent. Different faces may have different expression, beard, hair color and skin color. Besides, for an image of a face of specific person, different lighting, different orientation, and partial occlusion will also make the face with different feature. This article would introduce some inspiring ideas of FR. Due to the limitation of time, the main part of the article is about the detection, and the recognition part would be shorter.

2.Face Detection

If for human there is still some difficulty on recognize the faces on the image, we can say that the detection process is too trivial since we are too familiar with the faces of our species. But as the mentions above, it is difficult for computer. When you take out a camera to take a photo for your friend, probably you would find a rectangle box around his/her face. That is the result of face

detection. Face detection is the necessary preparation process before face recognition. The face detection would firstly test whether there are some faces of a given image, then mark the position of the faces by a bounding box and pass it to the recognition model. The following I will introduce some idea of detection.

2.1 Some thinking about face detection myself

Based on my existing knowledge and thinking, I can conceive a way to train a face detection model in a special case where every given image should be in a particular size, and contains exactly one face. As we know, an image is consisting of hundreds of thousands of pixels arranged as a rectangle, that is, we can treat the image as a matrix of pixels (or RGB data). Therefore, the detection problem becomes a regression problem, where the input is such a matrix, and the output is the coordinates of the four vertices of the box that frames the face in the image. Then we can provide some training data (X,Y) (X be the pixel matrix and Y be a 2*4 matrix containing the four coordinates) to train the regression model. However, this method is not clever. Except the quite special prerequisites, the biggest problem is that the complexity of this model is so large that it would cost too much time and computation power to train this model Besides, for the calculation of such a giant size matrix, based on the computer knowledge, it would lost accuracy. For example, for an image with a width of 8 inches and a height of 6 inches, with a resolution of 50 PPI (pixels per inch), it is not so big and not so clear, but it contains 120000 pixels. That means the regression function in our model is equivalent to a 120000-to-4 function (because of the characteristic of rectangle, we can define a unique rectangle horizontal to the edge of the image by a pair of vertices coordinate on its diagonal, that is 4 values), or four 120000-to-1 functions, which is obviously hard to train.

To simplify the original image, we may be able to use the idea of mosaic. Also use the example of the 8 inches * 6 inches, 50 PPI image, We can regard every 10 * 10 pixels as a combination, and replace the RGB data of every pixel by the RGB data of upper left most pixel. After this transformation, the number of independent pixels of this image decreases to 1200, and the regression function of this model is equivalent to a 1200-to-4 function, which is simpler. However, the shortcoming of this idea is similarly awful. The accuracy of the detection would decrease as original image after the mosaic processing becomes more blurred. As we are testing the pixels of an image after mosaic, the output of the model, the position of the bounding box, would not be accurate for the origin image.

This model I came up with is too shallow and not practical. I found many interesting and inspiring method of face detection in the literature, and I will introduce two of them as follows.

2.2 Viola-Jones Algorithm [1]

Figure 2.2.1 Professor

Paul Viola and Michael J.Jones publish this algorithm in 2001. This algorithm can process a real-time, 15 frames per second detection on images with size 384*288 pixels on computers with 700 Hz Pentium III processor, which is very strong and advanced at that time. This algorithm is based on the feature of faces, so it can be easily understood by human's logic. The following I would like to make use of a photo of our respectful professor, Joseph Sung (Figure 2.2.1), as an example to conduct this algorithm.

Figure 2.2.2

The process of this method can be divided into two parts. First, we should choose a box on the image. Then we should test whether the box contains a face. I will describe the how to choose such box later, and recently suppose we luckily choose the box contained the face of Mr. Sung (Figure 2.2.2), how can the computer know that there is exactly a face in the box?

I should firstly introduce to Haar-Like Feature. As the method I came up with above, there are too many pixels on a simple small image. If we want to do some research based on the intensity (RGB value) of every single pixel, there will be a huge number of data and it will cost too much computation power, therefore it is impossible to practice. A Haar-Like feature can be described as the difference of intensity of two or more neighbor rectangles, which are both parts of the image. Here is an example of Harr-Like Feature (Figure 2.2.3). For each square with a determined size, called "Size 1", we choose two neighbor rectangles, denoted by "①" and "②" respectively, with both in a determined size and relatively determined position in the square. Then we define the rectangle contained exactly ① and ② as the "Position A", and "Feature 1" as the Haar-Like Feature of the square in "Position A". By the definition of Harr-Like Feature, we get that:

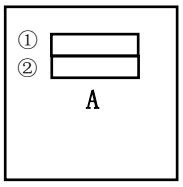


Figure 2.2.3

Feature 1= Σ (intensity of a pixel in ①) - Σ (intensity of a pixel in ②)

If Feature 1 > 0, we conclude that the intensity in ① is greater than it ②, or ① is brighter than ②. Similarly If Feature 1 < 0, we conclude that the intensity in ① is smaller than it ②, or ① is darker than ②. Then if we give a square of "Size 1", which is filled with gradient from black to white, as the Figure 2.2.4. The Feature 1 of this square is clearly less than 0, since the upper

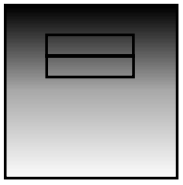


Figure 2.2.4

rectangle of “Position A” is obviously darker than the lower. That is how we define a Haar-Like Feature and how we describe the Harr-Like Feature of an arbitrary square with such determined size. The method to calculate Harr-Like Feature in Viola-Jones algorithm is make use of the integral image, which replaced calculating the sum of pixel’s intensity in an arbitrary rectangle by some calculation on it of four rectangle with the edges of parts of upper and left boundary of the whole image. This is an excellent method and can sufficiently save most of the computation, but it will not be covered in this essay.


Figure 2.2.5

Then the next question is, how is the Haar-Like Feature related to the feature on human’s face. Let us observe the figure 2.2.5. This is the chosen part of photos of Mr. Sung after decolorization and mosaic processing, to emphasize the intensity and remove unneeded details. We found that some parts look brighter than other parts. For example, the region of eyes looks darker than the region of upper cheek. This feature is common for all the image of human. Then we can test the Haar-Like feature of the corresponding position on the image (Figure 2.2.6). If the value of Haar-Like feature is over some threshold, we claim that the image is likely to be a face. Now two new questions appear.


Figure 2.2.6

The first question is that it seems too hasty to determine a face by simply one feature. Let us consider a Bayes’ equation:

$P(Face|Feature) = P(Feature|Face)*P(Face)/P(Feature)$

Obviously given a image with face, there must be a feature based on that face, so $P(Feature|Face)=1$. Also we have $P(Face)=0.5$. Therefore, $P(Face|Feature) =0.5/P(Feature)>0.5$. That means for a detector that test only this feature, the probability that it correctly claims there is a face is greater than 0.5, a little bit greater than toss a coin. We called it a weak classifier based on this feature, since it is not strong enough to claim exactly a face. An weak classifier based on a feature “i” can be quantify as follows:

$h_i = 1$ if Haar-Like feature of i > a threshold θ_i, i.e., the feature exists.

$h_i = -1$ otherwise.

Then we can determine many features and construct corresponding weak classifiers, and construct a weighted combination of them:

$h=\mathrm{sgn}(\Sigma a_i h_i)$,

which is strong enough to classify whether there is a face.

The second one is that how can we determine a feature that is helpful for our detection, and how can we choose the parameters we mentioned above, namely threshold θ_i and weight a_i? In the origin Viola-Jones algorithm, the base resolution of the detector is 24*24, therefore there are more than 160,000 features in totol, most of which are useless for the detection. Maybe we can point some features like what we did before, but that is surjective and unable

to quantify, also the position and parameters cannot be define. Machine learning can help us here. In the origin model generated by Viola and Jones, they make use of Adaboost training method, together with the training set of 4916 images with faces and 9500 images without face. Then every data we want can be reached after training.

Figure 2.2.7

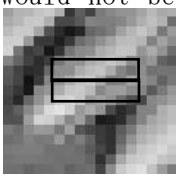


Figure 2.2.8

Viola-Jones algorithm use a cascade structure to reduce the computation. Rather than test by the whole model, an input image will be tested by part of features and determined whether it is possible to contained faces. If not, it would be eliminated and would not be tested by other features; if yes, it would be test by some other features, and then repeat this process for several times until the end of the model. The earlier steps in cascade would test some overall features of faces, and the later steps would test some detail features. For example, figure 2.2.7 is a picture of corns, and we do the decolorization and mosaic processing on it. Obviously, there is no face. But if we input this
image into the model without cascade, it will be tested whether it meets with all the features we defined. We want the model to know that such an image that obviously contains no face is not worthy to go through the complete testing, and it should be eliminated as early as possible. Therefore, some “typical” features (for example, the difference between region of eyes and the region of upper cheek) will be tested earlier (Figure 2.2.8). If an image does not fit with such typical features, it will be treated as “no face” and will not be enter the next step of cascade. Eliminating images without faces as early as possible can save the computation and improve efficiency.

This is the idea of detection of face on an image with determined size with Viola-Jones algorithm. However mostly the input image is not a square picture only with face. It is necessary to find some method to determine the positions of all the faces on the input image. The idea is brute-force search. We use a box of a determined size, starting from the upper left corner of the image, and process a detection with every small distance. After this round of detection, adjust the size of the box slightly and repeat the above operation. Record every position when a face is detected. This method indeed cost much computation, but it is effective not to miss faces with any magnitude and in every position of the image. (When it comes to the Figure 2.2.10, it is just a demonstration,

Figure 2.2.10

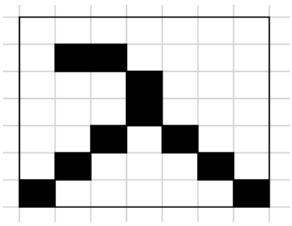
if you do use this box with this size (too big) and the distance between to detection is so large, all the result of these six detections is “no face”.)


Figure 2.2.11

The main shortcoming of Viola-Jones algorithm is that it can only detect the upright face, and it is not strong enough to detect the faces with different pose (like Figure 2.2.11) or being partially obscured. In some extension of Viola-Jones algorithm, one can input the face orientation annotations to specify the face angle of the image, but some new problems would come, and this would not be covered in this essay. Generally, Viola-Jones detection algorithm is a revolutionary creation, and it plays an important role on the field of face detection until today.

2.3 Face detection bases on Convolutional Neural Network (CNN) [2]

As we mentioned before, a picture will be treated as a matrix of pixels (RGB value), whose size is too hugh for training model. The convolutional neural network is a method to detect object (including human’s faces), which developed rapidly these ten years. The idea of CNN is generally as follows, and I will show it in details in the next few paragraph. Inputing an image to a CNN model, the size of the image will decrease but the feature of image will be reserved roughly. When the size of image is small enough, it can be used as a data to trained the model.

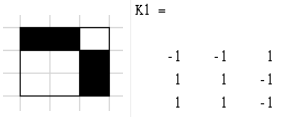


I will use the Figure 2.3.1 as an example. Observe this picture, it is black-and-white with a size 7*7. We may found that this picture looks like the symbol “λ”. This picture can be consider as a 7*7 matrix with every entry being “1” or “-1” (as Figure 2.3.2). Rather than consider the feature of the whole matrix, it will prepare some specific feature in small part (generally, 1*1, 3*3, 5*5, or 7*7), and it will slide from the upper-left to the lower-right of the origin matrix and test whether this specific feature are consistent to the feature of some parts of the matrix. This specific feature are defined to be the convolution kernel. In the right there are two 3*3 convolation kernels, and we called them K1 (Figure 2.3.3) and K2 (Figure 2.3.4) respectively. In reality, one can find that the total number of 3*3 kernels are 2^9= 512. The convolution of two matrixs in the same size can be regarded as the following function:

A =

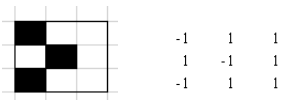
1	1	1	1	1	1	1
1	-1	-1	1	1	1	1
1	1	1	-1	1	1	1
1	1	1	-1	1	1	1
1	1	-1	1	-1	1	1
1	-1	1	1	1	-1	1
-1	1	1	1	1	1	-1

Figure 2.3.2


K1 =

-1	-1	1
1	1	-1
1	1	-1

Figure 2.3.3


K2 =

-1	1	1
1	-1	1
-1	1	1

Figure 2.3.4

f: R(m*m) * R(m*m) → R

f(A,B)= (Σ A(i,j)*B(i,j))/(m^2)

That is, the mean value of the product of (i,j)-th entry of A and (i,j)-th entry of B, 1<=i,j<=m.
For example, we calculate the convolution of K1 and the upper-left most 3*3 submatrix of A, then we get the result (sum(sum((A(1:3,1:3).*K1)))/(3*3))=0.11. Note that the 3*3 submatrix of A, with 2nd to 4th rows and 2nd to 4th columns are the same as K1, and therefore the convolution of these two matrixes is 1. We can conclude that the more similar the two matrixes are, the greater the convolution is. Therefore, we can slide the K1, K2 in A from upper-left to lower-right, calculate the convolution, and generate two 5*5 matrixes consist of the result of 1st-step convolution matrix of K1 and K2 respectively, called C11 and C21. The results are shown as the follows. We paint the two matrixes with different shades of grad, so it seems to be more straightforward (Figure 2.3.5 and Figure 2.3.6).

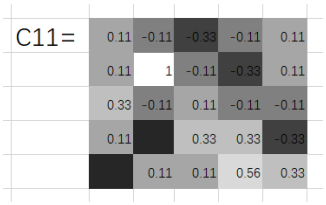


Figure 2.3.5

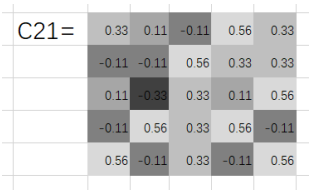


Figure 2.3.6

The former process is called the Convolutional Layer. The output of the convolutional layer is a matrix that contains the entries of “how the original image similar to the convolution kernel in parts”, and these entries can be treated as “neuron” of the network. Some properties of the origin image based on the kernel is reserved in the output matrix.

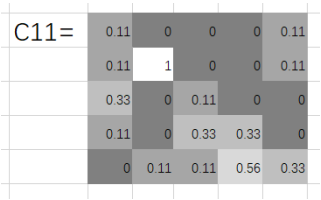


Figure 2.3.7

The following of the convolutional layer is called Rectified Linear Units Layer (ReLU Layer). For an input matrix in this layer, which is the output of the convolutional layer, (the most common processing is that) every negative entry would be replaced by 0. The C11 and C21 after ReLU layers will become as Figure 2.3.7 and Figure 2.3.8. That means it consider “how similar” rather than “how unsimilar”. A matrix with more 0 would be easier to compute.

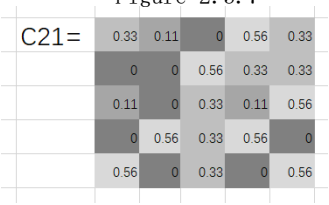


Figure 2.3.8

The following layer is called Pooling Layer. In this layer, mostly, we separate the input matrix into 2*2 submatrix from upper-left to lower-right, with no intersection. If on the edge there are no more 2*2 submatrix, we simply use the remaining part. The outputs of the pooling layer of C11 and C21, we called them P11 and P21, are matrixes with each entry be the maximum entries of the corresponding submatrix. That is, as the Figure 2.3.9.

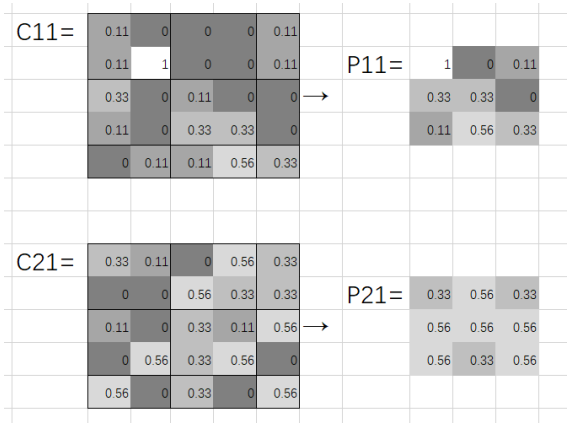


Figure 2.3.9

We call a Layer Stack as the combination of convolutional layer, a ReLU layer and a Pooling layer. We found that after a layer stack, the origin 7*7 image reduce its size to 3*3. For some image with greater size, we can apply the layer stack many times, where the output of a former stack becomes the input of a latter stack. Finally, we would have a series of small sizes matrixes corresponding to difference specific convolution kernel, then it would be easier to training the model.

Here come two questions. The first is that, the input images are usually continuously colored, rather than only with black and white, how could CNN deal with this? As we know that, there are three color channels in the computer, namely Red, Green, Blue (RGB). Every color is generated by difference shade of R, G and B. Therefore, a partial feature of a colorful image, or we say a convolutional kernel of the image, can be decomposed into the three channels. That is, for example, a convolution kernel with size 5*5*3, is three 5*5 convolution kernels of red channel, green channel, and blue channel of the original image. Then we can do something similar to the black and white image. The next question is that, we cannot point some features on our face base on the RGB value in some pixels, how can we find suitable convolution kernels for the model? This question is similar to find suitable Haar-Like features for the model base on Viola-Jones algorithm, and the answer is also similar: we let the computer find them itself. We can arbitrarily give some initial kernels and add a backprop function. Then we input a great number of training data, and the model would be trained and self-corrected, this is what we called “machine learning”. Therefore, the reason why CNN is so popular these years is, we just need to construct a origin model, (even if it may look ridiculous,) and input training data, and then we will get a fully trained model.

2.3.1 Multi-task Cascaded Convolutional Networks [3]

Here I would provide an interesting idea of facial detection model based on CNN, called Multi-task Cascaded Convolutional Networks (MTCNN), which is

published in 2016. Here I would use the Figure 2.3.1.1, which is a picture in the origin essay, to interpret the idea. For an input image, zoom it into different sizes to test the face with difference magnitude. The series of same image with difference size is called the Image Pyramid. Then for the testing model, as we mention in the part of Viola-Jones algorithm above, a cascade structure can eliminate some substandard images step by step and would not let them participate into the later testing, which significantly improves efficiency. MTCNN builds 3 stages of CNN testing as the cascade in the model. The first stage is called the Proposal Network (P-net), which is a fully convolutional network. The output of P-net is a lot of boxes which might contain faces. Then merge overlapped boxes with non-maximum suppression (NMS). The second stage is called the Refine Network (R-Net), which would eliminate most of boxes of the output of P-net that contained no face, and then use NMS again to concentrate the boxes. The third stage is called Output Network (O-Net), which is similar to R-Net. The output of O-net is a box contained face and five feature points on the face, namely two on the eyes, one on the nose and two on the corners of the mouth. All of the 3 nets is based on the CNN model. This method is very strong. It can test faces with difference condition, including difference pose and small part obscured, with the accuracy very close to 100% after sufficient training. In a CNN model, every test data with feedback is also used to train the model, so the accuracy will increase as it detects more times. Besides, the model is very fast. It can detect 16 frames per seconds by a CPU with 2.60GHz.

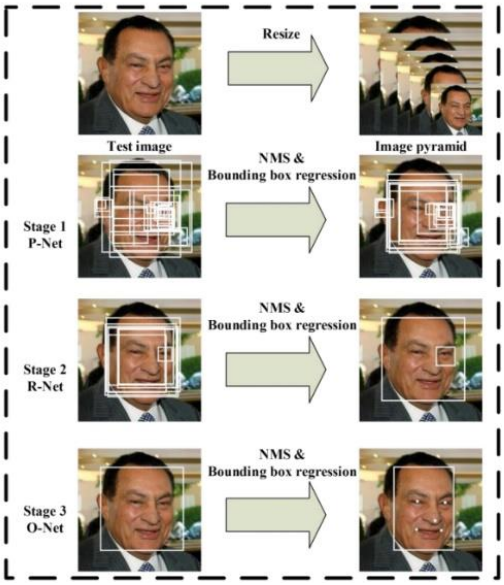


Figure 2.3.1.1

2.4 Summary in Face Detection

Here are two commonly used method for face detection. In reality, there are many interesting detection method, and they can be generally divided into two types [4]: feature-based, which analyze the special characteristic of human faces compared with other things, and an example is Viola-Jones algorithm; and image-based, which focus on learning the pixels in the image, and a example is MTCNN. Due to the time I cannot study more in this essay, but these are very worthy to research.

3.Face Recognition

We input an image with somebody's face, and after the face detection process, we get some boxes which contained face. Now we have a database, which contains the information of some people. The propose of face recognition system is to

extract and analyze the features of the detected faces and match them to a name in the database (if there are some). When the size of database is large, like 10,000 or more, face recognition becomes an impossible task for an individual since human cannot remember so many faces and their information. This is a reason why face recognition is so useful in big data analysis. Here I would give an interesting idea of face recognition model.

3.1 Eigenfaces [5]

Turk and Pentland published this method in 1991, and it has a great reference value until know. The basic idea of eigenfaces seeking a small set of typical feature images, called "eigenface", in one's face which distinguish to others best and encode them. As we mentioned many times, an image can be treated as a matrix of pixel with huge size. (The following data are the experiment data in the origin essay.) Suppose we now have 2500 images of face with the size 512*512. Then we can define an image space with the dimension 262144=512*512, where every image is a vector in this space. Our goal is to find a subspace with much lower dimension (256=16*16), which reserve the most typical features on the face. Then it is clear that we can apply the PCA method to reduce the dimension. That is where the name "eigenface" come from. It is interesting that in our process to calculate the PCA, we should add all face images up and divided by the cardinal number, and then we will get a "mean face", which could be said as the "almost most standard face for all human". The dimension is huge so it is computational expensive but calculating the PCA does not need to do something about the inverse of matrix so therefore it would not lose accuracy. After the model get trained, it will be very fast.

The next step is, how can we store the face information we already have in the model, and for a new face image, how can we match it to a stored information or output a "no matching"? Now a man, called Tom, provide us an image of him. This image is also a vector in the image space, and we can calculate its projection to the subspace of "typical feature". The new vector, called "v", is the point stands for "Tom" in this model. We can also set a ball with $||x-v||<\delta$ for some small δ as the confidence interval. We can do this many times until we store all the faces we know. Then we input a new image in this model and calculating the projection to the subspace of "typical feature", say "v'". If v' located in the confident ball of Tom, i.e., $||v'-v||<\delta$, we say that face is of Tom. If v' does not located in any confident ball we have, then that face is for someone outside the database.

The eigenface method is very easy to understand and strong at that time. The greatest limitation is that, its accuracy is affected a lot by conditions unrelated to facial features. For example, 20% variation of lighting would achieve a 100% accuracy, but 20% variation of orientation and size of face would leading a 94% and 74% accuracy, which is not acceptable. Besides, some

small obscure is not acceptable in testing. That means that if we want to accurately test our faces, we have to right facing the camera. This is acceptable for initiative recognition like in customs, but not practical for passive recognition like in CCTV.

3.2 Summary of Face Recognition

Eigenface is a very early face recognition system. During these 30 years, the development of face recognition has greatly improved. For example, DeepFace based on Alexnet in 2014 reach 97.35% accuracy for different condition of facial images, which is near the same as human. FaceNet based on GoogleNet-24 in 2015 reaches the accuracy of 99.63%.[6] Many new face recognition models can stably reach the accuracy more than 99%. There are a great number of interesting essays interpreted their models, and it is a great regret that there is no time for me to analyze this beautiful field more.

4.Conclusion

In this article, we know that FR can be divided into two parts, face detection and face recognition, and we introduce some interesting ideas to construct models to achieve these goals. In the past four decades, FR technology has advanced rapidly, and the recognition accuracy has been very high. Recently the FR technology is trying to improve the accuracy of detection and recognition faces in some extreme condition, like faces in burry pictures or faces with higher occlusion (mask or sunglasses). I believe as the knowledge of machine learning grown, the effect of FR would become greater and greater, and then it would bring more influence in our daily life.

5.Reference

[1] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, *57*(2), 137-154.

[2] Convolutional neural network. (2020). Retrieved 4 December 2020, from https://en.wikipedia.org/wiki/Convolutional_neural_network

[3] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499-1503.

[4] Hjelmås, E., & Low, B. K. (2001). Face detection: A survey. *Computer vision and image understanding*, *83*(3), 236-274.

[5] Turk, M. A., & Pentland, A. P. (1991, January). Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition* (pp. 586-587). IEEE Computer Society.

[6] Masi, I., Wu, Y., Hassner, T., & Natarajan, P. (2018, October). Deep face recognition: A survey. In *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)* (pp. 471-478). IEEE.