Real-time Face Detection and Recognition in Images and Videos

Abstract

Face detection and recognition in images and videos can be applied to many areas. And the development of hardware and algorithms make it possible to detect and recognize faces in images and videos in real-time. In this project, we implement one real-time face detection and recognition system. Every user will firstly input one short video and our system will train out one face feature for each user. Later, when some video or image is input, our system will detect faces in images or videos and label known name or "unknown" if not existed in database.

1 Introduction

Nowadays, face recognition has been widely used in all aspects especially when video cameras are extremely cheap today. And it is also a relatively complicated process, which can be summarized by 3 steps: face detection, face alignment and face feature extraction.

Face detection is an important area in computer vision and model detection. Histogram of Oriented Gradients[1] and latest CNN [2] are commonly used for face detection.

Face alignment plays an important role in pre-processing step. It is to locate a set of facial landmarks, such as eye corners, mouth corners, etc. With these landmarks, algorithms can better refine the face contour and feature locations.

Face feature extraction is to encode facial landmarks into feature vector. The goal of feature extraction is both robustness and distinctiveness between different faces.

This paper will introduce our face recognition system. Section 2 will introduce the whole pipeline. Section 3 and 4 will introduce algorithms used for face detection and face encoding. Section 5 will share how we get measurement from training videos. Section 6 will introduce our recognition process. Our experiment results are presented in Section 7 and section 8 will conclude our system and possible improvements in the future.

2 Pipeline

Pipeline of our system can be concluded as figure 1. Firstly, manager needs to input one short video for every target user to train. The result face measurements will be stored in face database. Then, when image or videos are input, our system will detect and encode the faces to compare with the database. If some face matches, the username will be labeled on the video or image. If no match, "unknown" will be labeled.

3 Face Detection

Face detection is to find all faces in the picture. There are many face detection methods. Here, we will implement and compare two face detection methods.

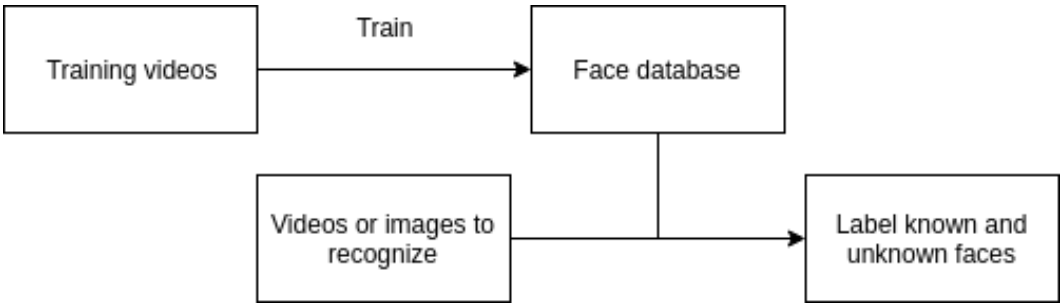


Figure 1: System Pipeline

3.1 Histogram of Oriented Gradients

Histogram of oriented gradients[1] is commonly used in face detection and has a reliable result. These methods usually use **HOG** to calculate features and trained **SVM**[3] to check if there are face features. Figure 2 shows the visualization of HOG feature of human face.

3.2 CNN

Convolutional neural network begins to be widely used in object detection and classification from 2012[4]. And [2] uses CNN for face detection and has a robust performance when the training sample size is large. Fortunately, there are many public human face datasets available on Internet.

[2] proposes three different network structures as in figure 3. $12-net$ is a very shallow classification CNN to quickly scan the images. $24-net$ is an intermediate binary classification CNN, which is more robust than $12-net$. Compared to $12-net$ and $24-net$, $48-net$ adopts the multi-resolution design and deeper network, which makes $48-net$ more powerful but slower.

3.3 Implementation and Comparison

Compared to HOG method, CNN face detection is more robust. However, if there is no GPU assisting, the detection will be much slower compared to HOG face detection. In order to meet different needs, we implement both methods.

In our system, we implement both HOG and CNN face detection methods based on **dlib** library [5]. It's one modern C++ toolkit for machine learning and provides easy to use python bindings. Fortunately, many users have shared their well-trained model, so we don't need to train the model from scratch.

Figure 4 show our comparison results of CNN and HOG detection. From the results, we can see that HOG and CNN can both detect face correctly when face is right against camera. However, when the angle is critical or there is some blocking, the CNN shows much better performance than HOG. Thus, in our system, the default option is CNN detection.

4 Face Alignment and Encoding

After detecting faces, we will deal with the different directions of faces. To wrap the image so that eyes and lips are always in the sample place, we will use one face landmark estimation algorithm [6]. The basic idea is that we will come up with 68 specific points that exist on every face as figure 5. And with the landmarks, we can wrap the image to center the face as figure 6.

With the centering face, we use FaceNet [7] to generate 128 measurements for face.

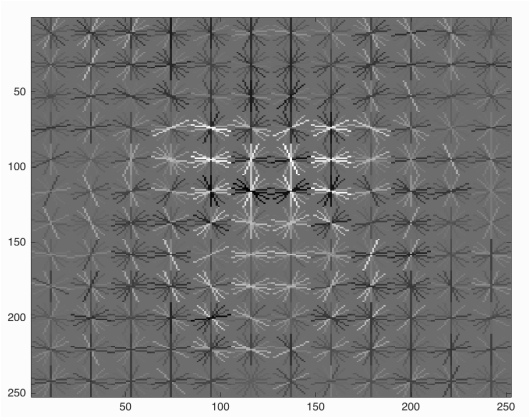


Figure 2: Face HOG Feature Visualization

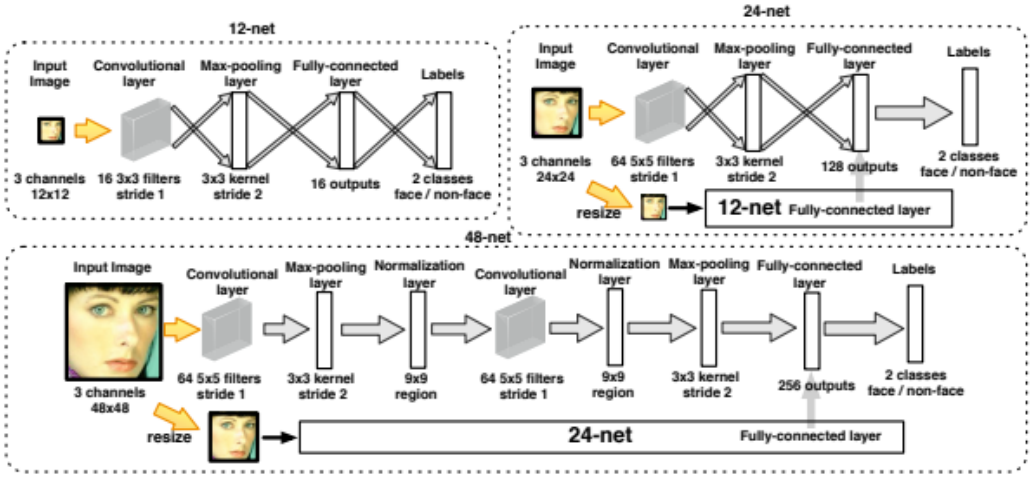


Figure 3: CNN structures of the 12-net, 24-net and 48-net

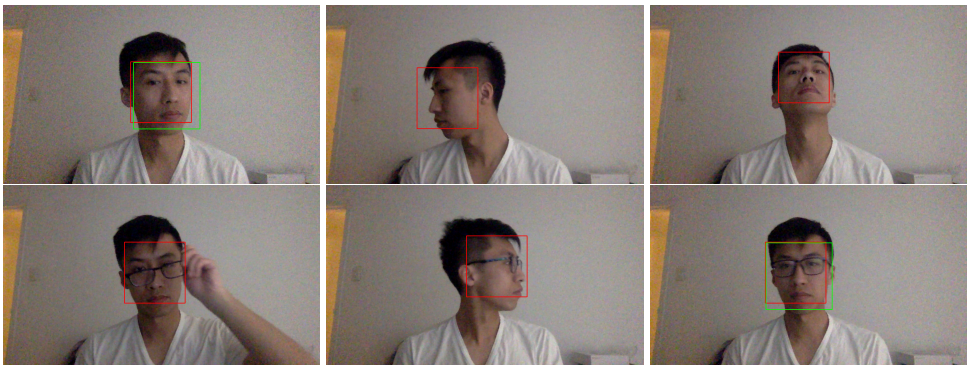


Figure 4: Face detection results. Red is detection of CNN, Green is detection of HOG

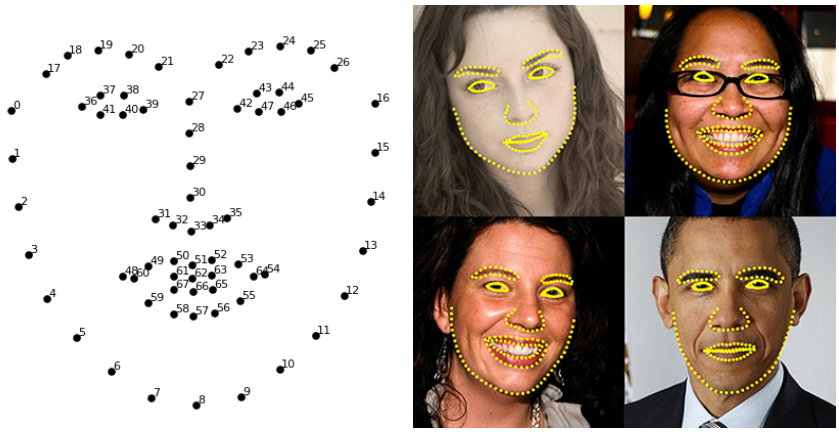


Figure 5: 68 Landmarks

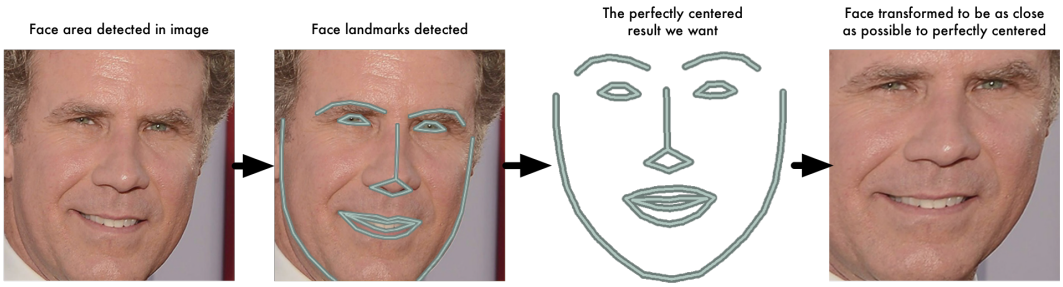


Figure 6: Face Alignment

5 Training Process

We use videos instead of images as the training source because large number of frames can be easily got from one short video. Given one video and corresponding username, our training process is as below:

1. Get enough frames for training from the video. Managers can adjust the training time and accuracy by changing the number of frames for train;
2. Use face detection algorithm to find faces in every frame. Drop all frames with no face or more than 1 face detected;
3. Get face measurements from each frame. Here, in order to be robust, we use RANSAC [8] algorithm to get rid of outliers. That is, during every iteration, we randomly pick one measurement and count other measurements whose distance is less than distance threshold. If the count is greater than ratio threshold (i.e. 90%), we calculate the mean of all these inlier measurements as the final result. Otherwise, we continue iteration;
4. Store the measurement and corresponding username in database.

6 Recognition Process

When recognizing faces in videos, videos are separated into frames to handle. Considering that movements are smooth in continuous frames, manager can set every how many frames (i.e. 3) system recognize once, and left frames (i.e. 2) frames are interpolated. Figure 7.

Since processing small images is quicker than large images, manager can set down-sampling times to accelerate. Recognition can be divided into these steps:

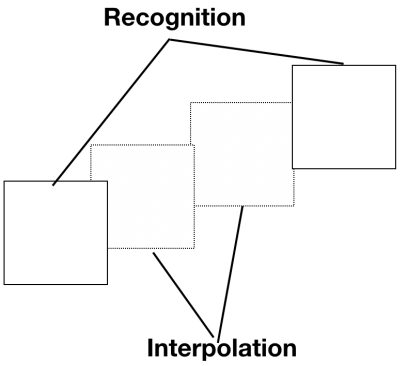


Figure 7: Face Location Interpolation

1. Read known face measurements from database;

2. Detect, align and encode all faces in the image or frame to get measurements;

3. Compare them with known measurements. If nearest distance is less than threshold, label the face with that name. Otherwise, label "unknown".

7 Experiment Result

Experiments were run on a computer with quadcore Intel Core i5-7500 with 16GB-RAM. GPU is GeForce GTX 1060 6GB. Figure 8 show our system results on image. From the results, we can see that our system can quickly detect faces out and recognize known and unknown faces. Also, the faces with different lighting and size can be recognized correctly, which shows the robustness.

For one video of 1280×720 and 30 frames per second, if every frame is processed with original size, every second video will cost 20 seconds to recognize. Thus, we down-sample every frame to half height and half width. Moreover, we recognize 1 frame every 4 frames and interpolate 3 other frames. The final time for 1 second video will cost 1.5 second, and accuracy can totally meet our requirements.

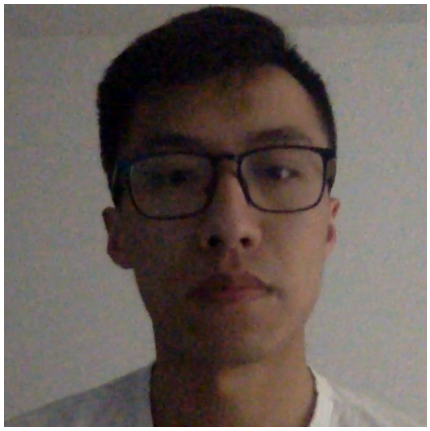
8 Future Improvement

There are still many possible improvements for this system.

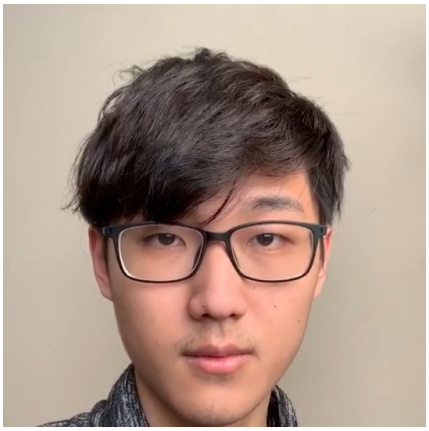
1. If the number of measurements in the database becomes large, we can use kd-tree[9] to search.

2. Face recognition in different frames are independent. We can utilize parallel computing to accelerate.

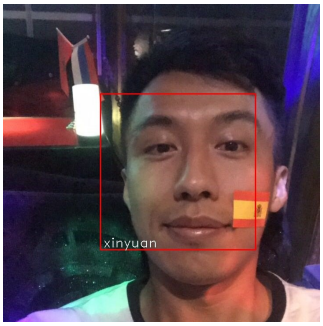
3. Directly interpolating between different frames may ba too coarse-grained. We can use more advanced algorithm like flow motion detection to gain higher accuracy.



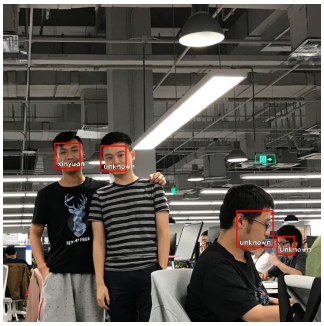
(a) Xinyuan Gui Training Video



(b) Chen Ling Training Video

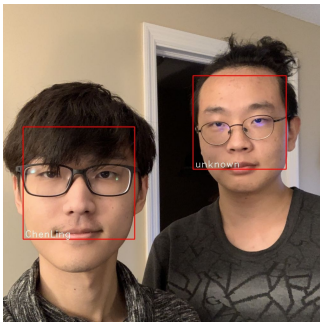


(c) 720 × 1280, cost 0.92s

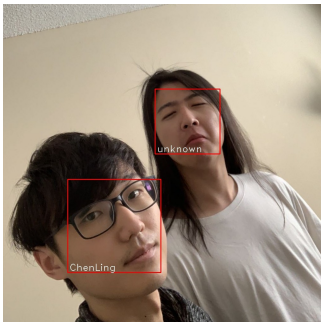


(d) 1440 × 1080, cost 0.90s

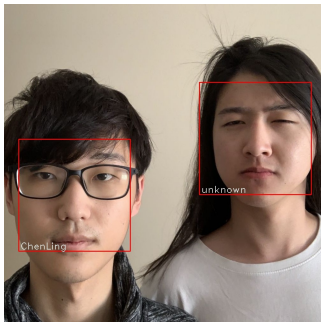
(e) 1440 × 1080, cost 0.82s



(f) 1440 × 1080, cost 0.47s



(g) 1080 × 1440, cost 0.46s



(h) 1440 × 1080, cost 0.46s

Figure 8: Experiment Result and Time Cost

References

[1] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[2] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5325–5334, 2015.

[3] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] dlib c library.

[6] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014.

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[8] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[9] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM Transactions on Graphics (TOG)*, volume 27, page 126. ACM, 2008.