Real-time Multi-face Tracking with Labels based on Convolutional Neural Networks

by

Xile Li

uOttawa

School of Electrical Engineering and Computer Science
Faculty of Engineering, University of Ottawa
Ottawa, Canada

Abstract

This thesis presents a real-time multi-face tracking system, which is able to track multiple faces for live videos, broadcast, real-time conference recording, etc. The real-time output is one of the most significant advantages. Our proposed tracking system is comprised of three parts: face detection, feature extraction and tracking. We deploy a three-layer Convolutional Neural Network (CNN) to detect a face, a one-layer CNN to extract the features of a detected face and a shallow network for face tracking based on the extracted feature maps of the face.

The performance of our multi-face tracking system enables the tracker to run in real-time without any on-line training. This algorithm does not need to change any parameters according to different input video conditions, and the runtime cost will not be affected significantly by an the increase in the number of faces being tracked. In addition, our proposed tracker can overcome most of the generally difficult tracking conditions which include video containing a camera cut, face occlusion, false positive face detection, false negative face detection, e.g. due to faces at the image boundary or faces shown in profile. We use two commonly used metrics to evaluate the performance of our multi-face tracking system demonstrating that our system achieves accurate results. Our multi-face tracker achieves an average runtime cost around 0.035s with GPU acceleration and this runtime cost is close to stable even if the number of tracked faces increases. All the evaluation results and comparisons are tested with four commonly used video data sets.

Declaration

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Ottawa's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink.

Acknowledgements

This research was partly funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Ross Video, Canada

I would like to express my sincerest appreciation to my supervisor, Prof. Jochen Lang, for his guidance, support, patience and understanding for my whole graduate study period. His encouragement and precious help for my research make me persevere in finishing my master research which I could not believe I was able to complete my research when I have just started my work. I am so honored and proud of working with such a responsible and excellent supervisor, I was able to get huge amount of benefits from what he taught me not only for my current research journey but also towards my further career and life. I am very grateful for everything he has helped and the time we worked together.

I would also like to thank my parents and my sister, who unconditionally support and love me. I also thank all the members in my bible study, who encourage me all the time. Thanks to my pastor Dr. Fuller and my friend Chuan for editing help. I am so lucky to receive the help from all of you.

I appreciate the experience working at in Discovery Lab, VIVA Lab, and the valuable advice from my friends Muye, Yang, Dongfeng, etc.

Last but not least, I would thank the God's blessing and love.

Glossary of terms

- Non-Maximum Suppression (NMS)

  In object detection, NMS is used to transform a smooth response map that triggers many imprecise object window hypotheses in, ideally, a single bounding-box for each detected object.

- Principal Component Analysis (PCA)

  A statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables [22].

- Intersection over Union (IoU)

  The Jaccard index is also known as Intersection over Union and is related to the Jaccard similarity coefficient. Computing the Intersection of Union is as simple as dividing the area of overlap between the bounding boxes by the area of union. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.

- Intersection over Maximum (IoM)

  Intersection over Maximum (IoM), it is similar to IoU to calculate the similarity between two sample sets, and is defined as the size of the intersection divided by the size of the max one of the sample sets.

- Support Vector Machines (SVMs)

  SVMs are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories,

an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier [17].

- Facial landmarks

  Facial landmarks are a set of key points on human face images. These points are defined by their real coordinates (x,y). Facial landmark detection is an example of structured output problem which aims at predicting a geometric shape induced by an input face image. It plays an important role in face recognition and analysis [7].

- Face tracklet (tracker output)

  A face tracklet is a section of a 2D trajectory of faces over time in the video.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation of the problem

With the rapid development in Computer Vision, many techniques have been researched and developed; moreover some of these techniques have already been adopted and deployed in production in different industries. A multi-face tracker is a widely needed technique in many situations, such as recording conference meetings, colloquium recording, live broadcast controller, etc. There is a very high demand for controlling the camera automatically by tracking the main speakers in many products. Our research in this thesis is motivated by the real world demand for a real-time multi-face tracker.

Even the most up-to-date multi-face tracking algorithms [77, 19, 43, 78, 64, 84, 66]

are limited to not running in real time, low tracking accuracy, using part of testing video frame to train the network (which is not able to achieve real-time output), requiring pre-tunning the tracking parameters when applying on different types of videos, etc. Until now, a more accurate multi-face tracking algorithm introduced by Wu et al. [77] executes in two steps which cannot run at the same time. The first step is collecting all the faces in the video, and the second step is linking similar faces. Although this method achieves a high tracking accuracy, the algorithm itself is not applicable to live video or real-time broadcast. Another multi-face tracker presented by Le et al. [43] is limited by using part of the testing video to train the multi-face tracker to acquire accurate tracking results. This tracker's highly accurate results are relying on pre-training the frames in the video which will be tested afterwards. As of today, the tracker by Le et al. [43] is fast at a speed of 6 - 7 fps on average. However, it is still far below the standard of running multi-face tracking in real time. Runtime stability is another evaluation criteria for multi-face trackers. There is a tracker published by Shi et al. [66] that can achieve a speed of around 50ms per frame. However, this speed is only for tracking one face in the video, and the tracker exceeds 150ms per frame for tracking 4 people at the same time. This tracker's [66] speed, according to the figure shown in the paper, increases almost linearly with respect to the number of faces in the video. The diagram Shi et al. presents in the paper [66] shows that the runtime cost is around 300ms per frame for tracking 9 faces in the video.

The main challenges for multi-face trackers are camera cuts, face occlusions, false positive detections and false negative detections. These situations result in the tracker linking the faces with incorrect labels, which directly reduce the accuracy. Unfortunately wrongly matched labels cannot be easily corrected afterwards. Tracking a person's profile rather than a frontal view of a face is another challenge. Results of up-to-date multi-face trackers show that the number of tracked profile faces is small. This means that the tracker requires the face to be directly facing towards the camera, and it will fail as the face turns moderately away from the camera. The profile tracking accuracy is directly influenced by the face detector results.

We have developed a real-time multi-face tracker based on Convolutional Neural

Networks (CNNs). There are three parts comprising of our multi-face tracker: face detection, feature extraction and the tracker. In our multi-face tracking system, we adopt the Multi-task Cascaded Convolutional Networks (MTCNN) face detector [83], which contributes to our multi-face tracker performance after good training. In order to ensure our tracker is able to run in real time, we use a Convolutional Neural Network (CNN) to extract all the detected face feature maps. After acquiring the feature maps once at the beginning of the tracking step, it is possible to design a very shallow neural network to decide if two face patches match. For tracking, in order to make sure the runtime cost is less than 40ms per frame, based on the extracted feature maps, we design three very shallow neural networks which contain one layer, two layers and three layers, respectively. Three layers are the maximum depth of the neural network that we have considered while making sure our system runs in real time. Our tracker is nearly stable in runtime cost for tracking up to at least 10 people. Typically, this stability will only be influenced by tracking a very large number of people (e.g., 39 in the example in Chapter 4) at the same time.

Based on the standard evaluation matrics [32, 9], our multi-face tracker yields very competitive results compared with other recent multi-face tracking algorithms.

1.2 Thesis statement

Convolutional Neural Networks are an effective method to extract object features and to classify objects. In our thesis research, we use CNNs to detect human faces, extract faces' feature maps and track the moving faces using the feature maps. In order to simplify our multi-face tracking system process, we separate the extraction process of the features for the tracking from the actual tracking. Hence, our proposed multi-face tracking method can run in real time with a high accuracy. We present three tracker combinations with three different network designs. The average runtime cost for commonly used test video [55] are 0.03480s, 0.03541s and 0.03530s for each frame respectively. The average results of Multiple Object Tracker Accuracy (MOTA) based on the test videos [55] are 98.69%, 98.72% and 98.69% for these three combinations. The runtime cost is stable when the

number of people being tracked varies slightly. Our proposed method does not require pre-training on the test video. As a result, our multi-face tracker can be applied to live broadcasting, real-time conference recording or live videos tracking.

1.3 Objectives

The objectives of our thesis include:

- Design and implementation of a real-time multi-face tracker. Since accurate published multi-face trackers hardly achieve running in real time, this is a multi-face tracking combining accuracy and running efficiency.
- A new method is proposed for multi-face tracking that matches faces based on a shallow neural network. The shallow network design reduces runtime cost with features extracted in advance by a separate CNN, and as the detector is also real-time enables system to run in real time.
- Training of the neural network tracking classifier responsible for face matching is using feature maps instead of cropped images. The use of these separately calculated feature maps avoids repeating the feature map extraction step and speeds up the training process.

We used C++ and Convolutional Architecture for Fast Feature Embedding(Caffe) [36] library for the CNNs and the OpenCV library [33] for image processing implementation.

1.4 Thesis outline

The thesis is organized as follows:

- Chapter 2 presents background and related work. The analysis of current multi-face trackers is given from different aspects: runtime efficiency, evaluation and restriction of each up-to-date multi-face tracking algorithm. The evaluation methods are introduced in this chapter.

- Chapter 3 describes the methodology and the design of our multi-face tracker in detail.

- Chapter 4 provides the results of our multi-face tracker using two commonly used but different evaluation metrics. The comparisons with other multi-face tracking algorithms are shown in this chapter according to the evaluation metrics as well as examples from our tracking results.

- Chapter 5 concludes with the advantages and limitation of this research. Future work is also discussed.

Chapter 2

Related Work

Our work is a real-time multi-face tracking system based on CNNs, which is therefore directly related to CNN development in computer vision, feature extraction methods, face detection and face tracking. In this chapter, we will provide an overview of these topics. Section 2.1 briefly introduces CNNs; in Section 2.2, we review several feature extraction methods; as face detection and face tracking are both essentially classification problems, we also discuss several different classifiers in Section 2.3. Some popular face detection methods and current face tracking developments are presented in Section 2.4 and Section 2.5, respectively. Finally, in Section 2.6, we introduce commonly used evaluation criteria for multi-face tracking tasks. A summary of this chapter is given in Section 2.7.

2.1 Convolutional Neural Networks

Neural networks have been studied since the middle of the last century [71] and has experienced a surge in development at the end of the last century. Neural networks are a widely used method to implement deep learning.

A neural network is a hierarchical model inspired by the structure of the brain. A network consists of a series of nodes connected according to certain rules [44]. A simple feed-forward neural network can consist of only three layers [23], these are: the input layer, the hidden layer, and the output layer, it is to be noted that, the external input and the output are not directly related. The nodes between two adjacent layers are connected by directed edges. Each of these edges corresponds to a weighted value.

The two most obvious characteristics of neural networks are the introduction of non-linear activation functions and the use of a nested design between net layers. This allows it to represent a highly non-linear (relative to input) function. Thus neural networks have a strong modeling capability for complex data patterns [44]. In 2006, Hinton et al. [30] published *"Reducing the Dimensionality of Data with Neural Networks"* in the journal *Science*. This work provided an effective solution for deep network learning which involved using an unsupervised learning process to pre-train the network layer by layer.

Neural networks have developed into many types. A commonly used neural network is the CNN. The most significant feature of a CNN is the introduction of the convolutional operation and weight sharing. Given the characteristics, a CNN has a good performance on problems where the input is not numbers but images. The convolutional operation replaces the full connection of the feed-forward network [6] with local connections, and the weight of the connections between the different layers is shared [6]. When applying a convolution kernel onto an image, the convolution kernel looks as though it is an observation window at the detection time which gradually moves from the upper left corner of the image to the bottom right corner. Each position corresponds to an output node (each pixel point on the image corresponds to an input node). Specifically, the weights of different output nodes corresponding to the same input nodes are the same [44]. That is weight sharing.

In the field of image processing, CNNs have been used very successfully on the ImageNet data set [76]. The ImageNet data set contains more than ten million images which are classified into one thousand categories by hand-annotation. The annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has become a competition of object detection and object recognition. Compared to traditional image processing algorithms, one of the advantages of CNNs is avoiding the design of hand-crafted features according to the researchers. Using a CNN, one can directly enter the original image [15] and extract the object feature maps automatically.

CNNs were introduced by Y. LeCun [44], a well-known machine learning researcher at the end of the last century. He initially applied CNNs to the recognition of handwritten numbers. CNN's wide applications in the field of computer vision became apparent after the success of AlexNet in the general image classification task in 2012 [14].

A classic Convolutional Neural Network design is shown in Figure 2.1.

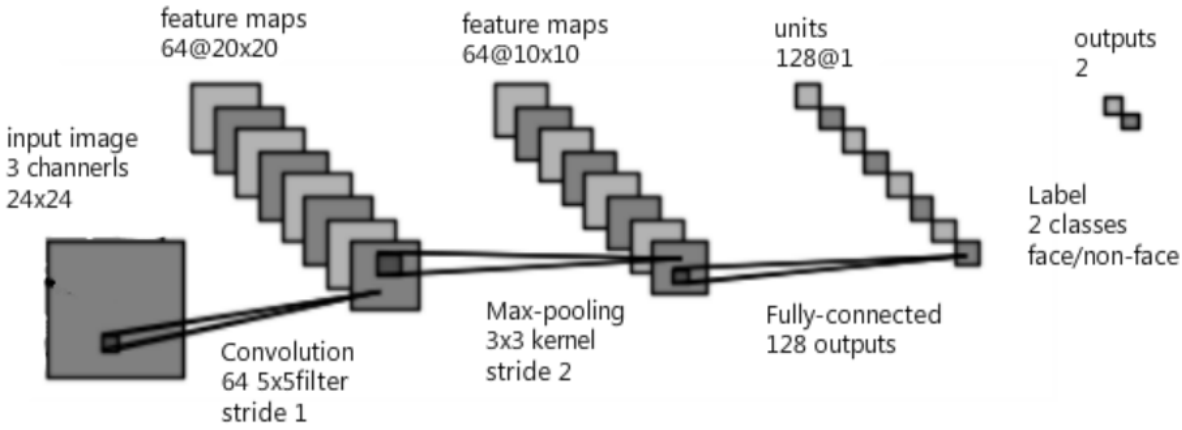

Figure 2.1: Convolutional Neural Network architecture [46].

A typical Convolutional Neural Network has the following major layers [44, 15]:

Data layer

First the data layer, which is also called the image processing layer. This layer is responsible for optional pre-processing and filtering. In face detection, the data layer will process the raw pixel values of the input images.

Convolution layer

The convolution layer is the sliding window calculation result of the input layer and each convolution kernel. Each parameter of the convolution kernel is equivalent to the weight parameters of a traditional neural network and connects to the corresponding local pixels. The sum of multiplying the parameters of the convolution kernel and the corresponding local pixel values (usually plus a bias parameter) is the result of the convolution layer. Figure 2.2 reveals this layer's working mechanism of a 3 x 3 kernel applied to a 4 x 4 image to gain a 2 x 2 output result. Each step calculation formula is shown in Formula 2.1, 2.2, 2.3 and 2.4.

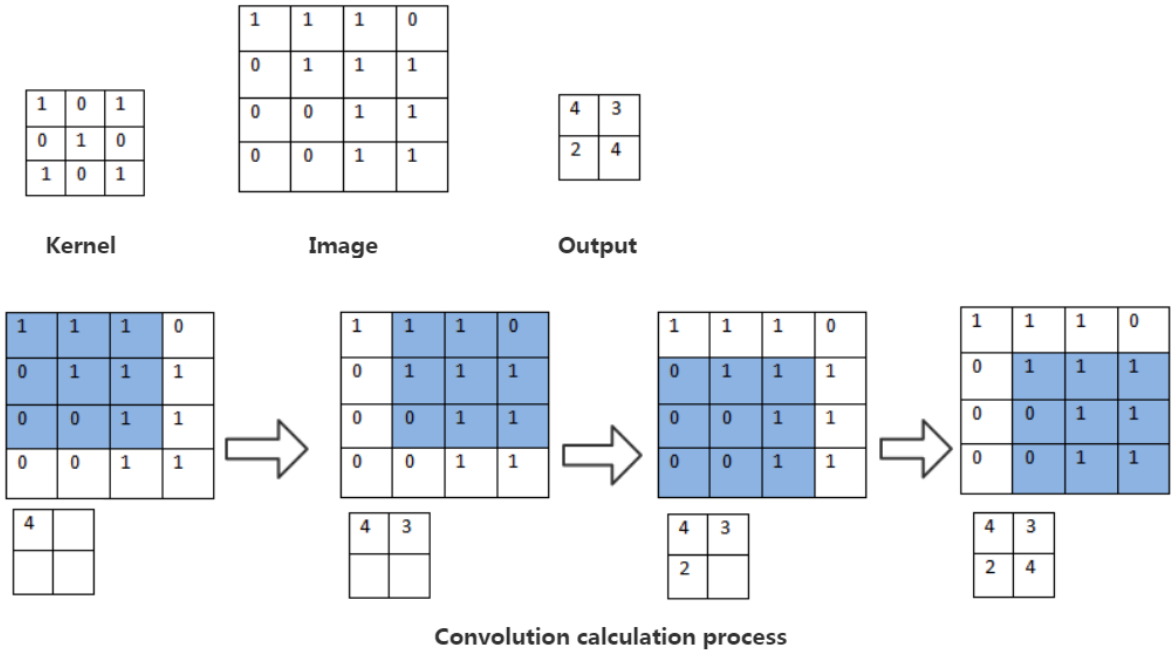

Figure 2.2: Convolution layer calculation process [42].

$$Outout(1,1) = 1*1+0*1+1*1+0*0+1*1+0*1+1*0+0*0+1*1 = 4. \quad (2.1)$$

$$Outout(1,2) = 1*1+0*1+1*0+0*1+1*1+0*1+1*0+0*1+1*1 = 3. \quad (2.2)$$

$$Outout(2,1) = 1*0+0*1+1*1+0*0+1*0+0*1+1*0+0*0+1*1 = 2. \quad (2.3)$$

$$Outout(1,1) = 1*1+0*1+1*1+0*0+1*1+0*1+1*0+0*1+1*1 = 4. \quad (2.4)$$

Pooling layer

After obtaining the features of the image through the convolution layer, in theory we can directly use these features to train the classifier. Though it is a big computational challenge, and prone to the so-called over-fitting phenomenon. Over-fitting typically happens when the error on the training set is driven to a small value, but when new data is presented to the network the error is large. The network has memorized the training examples, however, it has not yet learned to generalize to new situations. In order to further reduce the network training parameters and the degree of the over-fitting of the model, we will be sampling (Pooling) convolution layer outputs. Pooling/sampling is usually done in the following two ways:

1 Max-Pooling:

   Select the maximum value in the pooling window as the sample value; or

2 Mean-Pooling:

   Sum all the values in the pooling window and take the average as the sample value.

Max-Pooling is helpful to reduce the estimated value offsets resulting from convolutional layer difference error and keeps more image texture information compared with Mean-Pooling, so we use Max-Pooling in our system.

Activation layer

The commonly used activation functions are sigmoid, hyperbolic tangent and Rectified Linear Units (ReLU). In our work, we deploy ReLU for all the activation layers, because the lower computational cost of activation function in the back propagation gradient

derivation is lower with ReLU than with the sigmoid function [27]. The ReLU [35, 26] layer involves the ReLU function in the CNN implementation. This is used to add non-linear factors because the linear model's expression is not sufficient to calculate complicated features. The ReLU function is shown in Formula 2.5.

$$f(z) = max(z, 0) \tag{2.5}$$

Fully connected layer

The fully connected layer is also called inner product layer. That is equivalent to a classifier in the network, which calculates the class scores corresponding to output categories.

Dropout layer

Dropout layer was proposed by G. E. Hinton et al. [31] in 2012. When training a neural network model, if the training sample is limited, a Dropout layer can be added in order to prevent the model from over-fitting. During training, the dropout layer can stop half of the feature detector from working which can improve the generalization of the network capacity. However, in our system, we do not use Dropout layer.

2.2 Feature Extraction

Feature extraction can be regarded as the description of images in image processing applications [58]. The machine is only able to recognize a limited amount of the data, as well as process these data, the feature extraction is a process converting images to feature vectors, which can then be further analyzed by a machine.

Before deep learning technologies have been widely used in image processing (and face detection), empirical feature extraction methods were typically used.

2.2.1 Empirical features for face detection

Haar-like features

Haar-like Features [72] reveal light and dark relations among local areas for distinguishing face and non-face, as shown in Figure 2.3. For example, an eye area would be darker than the cheek area, which can be shown by the Haar feature. However, because it is a complex calculation of local area pixel gray value sums to obtain the Haar feature, Viola and Jones [73] introduced the integral plot to speed up Haar feature extraction in the so-called VJ detector. The integral plot is the same size as the input image but stores the gray value sums at the current point of all the integers within the rectangle consisting of the upper left corner point and the current point in the image, instead of storing each point's gray value.

There are two advantages of the integral plot, one is that it reduces the calculation of local area pixel gray value sum into only four steps, these four steps are not affected by the area size. Another one is that integral plot avoids repeated summation on the same pixel point. The integral plot significantly increases the speed of Haar feature extraction, and makes fast detection possible.

Haar-like feature improvement

Although Haar features can reveal and describe some facial features, considering the complexity of face detection under varied conditions, these Haar feature models are still too simple. So, there are various improvements of feature extraction based on Haar [49, 60]:

1 Circular black-white division mode replaced vertical direction mode and horizontal direction mode, as shown in Figure 2.3.

2 Rotated Haar feature that rotates part of the extracted Haar feature 45 degree clockwise or counter clockwise.

3 Separated Haar feature, that calculates one Haar feature from multiple separated black and white areas, instead of certain black and white areas from the same

rectangle.

4 Combined Haar feature that combines different Haar features or get the Haar features into a binary coding.

5 Weighted multiple channel Haar feature that uses different colors and shapes instead of only black and white.

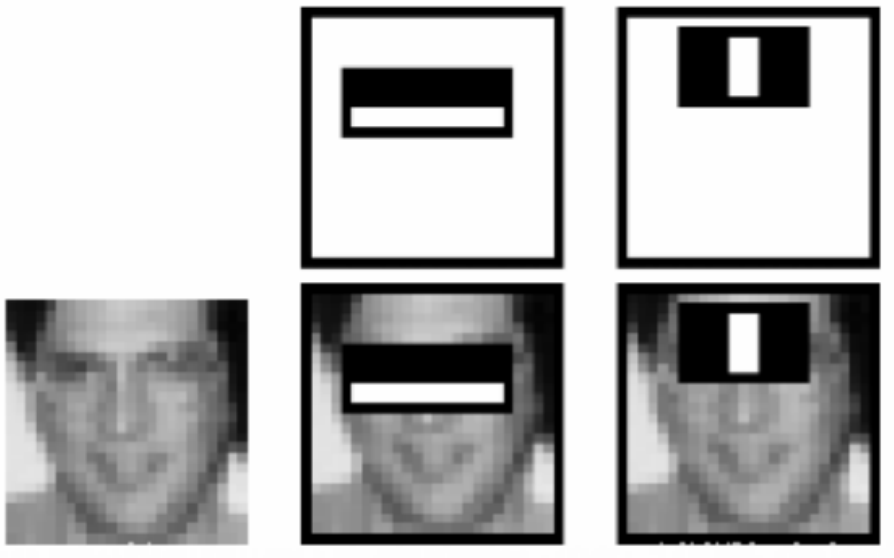

Figure 2.3: (©2017 IEEE) Haar feature representation [72]

Sparse granularity feature

The Haar-like features are linear combination of local area pixel values. Sparse granularity feature is a combination with uncertain parameters, which allows the user to adjust the Haar features as needs. Huang et al. [32] describes a Sparse Granularity Feature that linearly combines features of different sizes and locations. This is also known as a linear feature whose parameters can be learned and modified according to training samples, which is similar to a classifier parameter learning process.

LBP

In 2006, Ahonen et al. [4] introduced Local binary patterns (LBP). The LBP is a binary feature that calculates directly from pixel gray values. LBP features are faster to calculate than Haar features, however, Haar features can better distinguish face windows from non-face windows [37].

SIFT

The Scale Invariant Feature Transform (SIFT) [56] is a well-known algorithm in computer vision. It is mainly used to detect local features in images by looking at the information such as extreme points, positions, scale and rotation of these feature points in the image. Its application range includes object identification, robot map perception and navigation, image stitching, 3D model acquisition, gesture recognition, image tracking and action comparison. The description and detection of local image features can help to identify objects, and SIFT features are independent of size and rotation of the image, but based on some of the local appearance interest points on the object. It has a high tolerance for light, noise and direction changes.

SURF

In 2006, Herbert et al. [5] released the speeded up robust features (SURF) method. SURF is a similar feature method to Haar feature. SURF is more complex than Haar features and requires more execution time, but it can express image properties better with smaller numbers of features.

HOG

In 2005, Navneet et al. [18] presented a feature method called Histogram of Oriented Gradients (HOG). HOG features are also a type of feature extraction based on gradients. HOG features calculate gradients of different directions in a local area, and represents this area using a histogram of gradients. One HOG feature sample is shown in Figure 2.4.

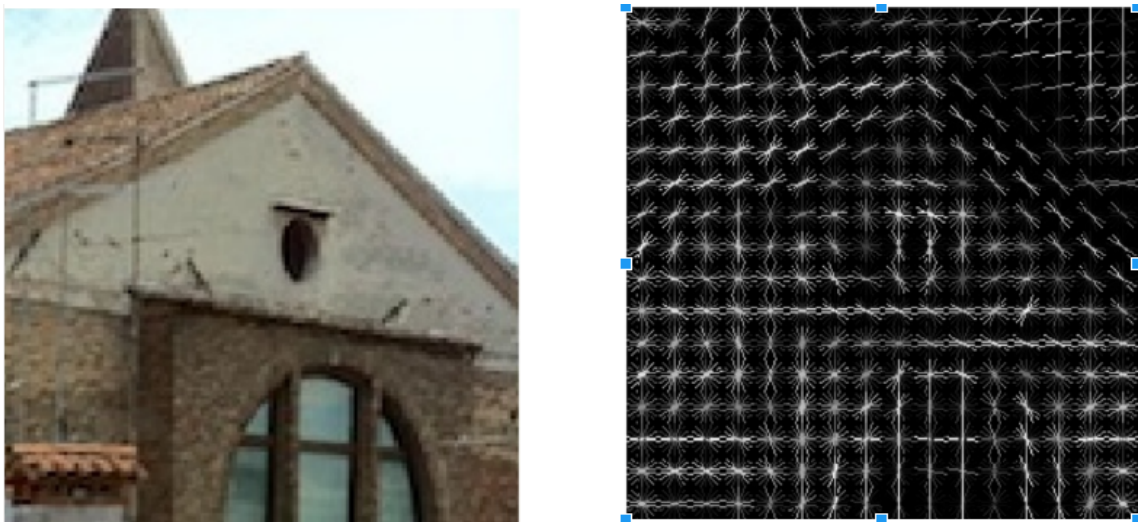

Figure 2.4: HOG feature representation [18]

In the past decades, many feature extraction methods were introduced. We just mentioned some representative ones above as examples. These features are all hand-crafted features, which could reflect the design ideas of the creators.

2.2.2 Deep features

Feature extraction is important area of research in computer vision but it also has an important role in the field of machine learning. In fact, deep learning became a hot topic at the beginning of 2006 [29, 8, 62], when an unsupervised learning procedure was introduced which could produce layers of feature detectors without requiring labeled data [45]. This development accelerates the study of feature extraction.

In our work, both face detection and face tracking rely on the features extracted by a deep learning method. In Chapter 3, we introduce more details about deep feature extraction.

2.3 Classifiers

A classifier performs a series of mathematical calculations based on the input in form of a feature vector. The categories will be the outputs of a classifier. Each category is usually

mapped to a numerical code, which is called the tag of this category. For example, in the face detection process, the face category could be coded as 1, and non-face category 0. A classifier is a function transferring feature vectors to category tags.

If the feature vector classifier is function f(x), then there is

$$x = (x_1, x_2, x_3, ..., x_n), \tag{2.6}$$

$$f(x) = \begin{cases} 1, \sum_{i=1}^{n} w_i x_i > threshold \\ 0, otherwise. \end{cases} \tag{2.7}$$

Here, the w is the weights of this classifier and the threshold decides the classifier output by the feature vector x. However, this is a very simple classification equation. Thresholding is the simplest way to decide the class. How to select parameters, such as the threshold of face confidence score in our face detection, is an essential part of creating a classifier.

In our face detection, classifiers are used to judge whether an image window belongs to the category face or not. At the beginning of the learning process, we set an initial value for the parameters of the classifiers, and then let the classifiers, according to the training data, continue to adjust the values of their own parameters to narrow the gap of actual classification result and known ground truth. Finally, when the classifiers have reached the pre-set target, or the classifiers have no way to continue to adjust, then the learning process stops. After that we can examine the accuracy of the classifier in the test set, which is the basis measurement we judge the classifier according to. A similar process is used for the classifiers in our face tracking.

2.4 Face detector

2.4.1 Viola-Jones face detector

A technical breakthrough in face detection happened in 2001, Viola and Jones [73] designed a fast and accurate face detector, which was approximately 100 times faster than any other face detection technology at that time. This face detector is often called Viola-Jones face detector, or VJ detector.

There are three main factors in making the VJ detector successful: fast feature calculation (integral plot) [73], an efficient classifier learning method (AdaBoost) [73] and an effective classification method (cascade structure design). The VJ face detector uses Haar features to describe each candidate window.

Applying a face detector directly to a complete image can result in a lengthy face detection process. Therefore, the VJ detector adopts a cascade structure which can quickly exclude most of the windows according to coarse detection at first, and carefully classifies the remaining windows to reduce the amount of time required.

The VJ detector combines multiple classifiers together, whose complexity and time cost are increasing with the order of application. For a given window, in the VJ detector, it will first go through the simpler classifiers of VJ detector structure, and then go further to the more complex classifiers if it is classified as face window by the simpler classifiers. This process does not stop until the window is excluded by some classifier or judged as a face window. This design reduces the number of windows with each classifier and excludes most non-face windows at the very beginning to reduce the runtime cost. Adjusting classifier complexity according to classifying difficulty level is efficient. The VJ detector, which uses the integral plot, AdaBoost and cascade methods, impacts face detection strategies in general beyond the specific VJ detector. The face detector we employ in our multi-face tracking system also refers to the cascade structure to improve running time efficiency and accuracy.

The VJ face detector is widely used in many face trackers. The integral plot, AdaBoost, the cascade structure and the Haar features to help improve the accuracy and

running time, however, compared with face detectors employing deep learning techniques, the VJ face detector is not competitive in terms of accuracy [34]. We do not use the VJ detector in our face tracking system.

2.4.2 R-CNN detector

Region-Based Convolutional Neural Networks, R-CNN [25] is an object detection method based on CNN by Girshick et al. It abandoned the sliding window paradigm (using detection window to scan input images), but adopted the selective search method to select some of the candidate windows. With an acceptable recall rate of the detection target, the number of candidate windows can be controlled within a certain number (which depends on the input). In fact all the windows are still checked once, but most of them are constantly being excluded. Compared with the VJ detector, the use of the candidate window generation method is based on the features of the image. Using the candidate window result, the detector can estimate where the object may be and how many objects there are in the input image.

The method used to generate candidate windows in R-CNN is called selective search. Selective search is a typical candidate window generation method, which uses the idea of image segmentation. Based on a variety of color features, the image is divided into multiple small pieces. Then the different pieces are merged from bottom to top. In this process, each merger of the pieces corresponds to a candidate window, and finally results in the selection of the most likely window containing the object as a candidate window [25].

Another advantage of R-CNN is that it not relies on hand-crafted design as the VJ detector, but that it uses the CNN to automatically learn features. Compared with the Haar feature process which uses hand-crafted features, a CNN is used for transformations without specifying the details of feature extraction. It is using training data to replace a human design. The disadvantage of this method is that the interpretation of learned features is usually poor. Another disadvantage is the dependency on training data sets.

The R-CNN detector has good performance by involving deep features compared to the VJ detector. However, because of the large computational complexity of the CNN,

the running speed is typically slower than the VJ detector. It can usually be applied to images in most applications, but it is not fast enough yet to apply it to processing videos. In summary, with regards to our real-time tracking system, we did not select the R-CNN detector because of our real-time processing requirement.

2.4.3 Faster R-CNN detector

The so-called Faster R-CNN detection algorithm [63] was developed by Shaoqing et al. Faster R-CNN has some improvements compared to the R-CNN detector. As its name suggests, faster R-CNN obviously achieves an improvement in speed.

There are three steps to accelerate the detection process.

1 The strategy of the integral plot is similar to the VJ face detector, however, this integral plot is calculated from the entire input image. In the R-CNN detection process, when there is an overlap between the two windows, the overlap feature is calculated twice. Thus, in Faster R-CNN detection process, the proposal regions are reflected to the last convolutional layer of CNN to calculate the feature maps. Then, for each candidate window, the feature map corresponding to the whole picture is extracted only once thus avoiding repeated calculations [63].

2 Faster R-CNN utilizes a matrix decomposition technique called SVD, which is applied in the fully connected layer of Faster R-CNN. Therefore, the computational complexity is reduced [63].

3 Faster R-CNN uses the CNN layers to generate candidate windows while it also utilizes the same CNN layers that generate the windows to share the convolution layers with other CNN layers used for the classifying and border regression. Since the same convolution feature maps can be used repeatedly in these steps, thus the amount of computation is greatly reduced [63].

Although, Faster RCNN detector has a lower computational demand than R-CNN, it is still hard to run in real time to apply in detection tasks in real-time videos. As a result, it does not meet our tracking system requirements.

2.4.4 Cascade CNN face detector

In 2015, Li et al. [46] presented the Cascade CNN face detector, which can be considered as a combination of traditional face detection technology and deep networks. This face detector contains a number of classifiers which are organized in a cascade. However, the difference is that the Cascade CNN face detector uses neural networks as classifiers for each level rather than using the AdaBoost method combining a number of weak classifiers into a strong classifier. Additionally, there is no longer a separate feature extraction process, feature extraction and classification are done in the CNNs.

In order to avoid high computational overhead, the first two CNN layers are very simple and the size of the input images are controlled to be very small. As the step size of the sliding window is set to 4 pixels, the number of candidate windows is reduced. The last CNN layer is more complex and is able to get accurate feature and classification results. The computational overhead of extracting features and classification for each window are controlled by the network design, so that the computation is made reasonably fast.

The whole cascade structure consists of three main parts:

1 The first CNN layer contains only one convolution layer and one fully connected layer for coarse classification results with 12 x 12 input images [46];

2 The second CNN layer increases the size of the input image to 24 x 24 in order to increase the ability of the classification process to distinguish between face and non-face windows. Although it still only contains a convolution layer and a full connection layer, the convolution layer has more convolution kernels, and the fully connected layer has more nodes [46]. Meanwhile, the same input images will be scaled into 12 x 12 and fed to the first CNN net, then the second CNN layer will deal with all the results from the second CNN layer and the first CNN layer together in a fully-connected layer to improve the accuracy; and

3 The third CNN layer also uses a similar approach to increase the size of the input image to 48 x 48. The layer is designed to be more complex. This more complex

layer uses two convolution layers and a fully connected layer to classify more precisely [46]. When running the third CNN layer, the second layer also process the same input images and all the results from these two CNN layers will be processed in the last fully-connected layer of the third CNN layer.

Aside from the three CNN layers, an additional calibration layer follows each CNN in order to modify the detection result positions more accurately. After getting the candidate windows from each CNN layer, the calibration layer will create 45 windows based on each output of the CNN layers. The calibration layer next recalculates the confidence value of each window. All the windows with a confidence higher than the threshold will be selected to calculate the average position as the detection result [46]. The process will be

$$(s, x, y) = \left(\frac{1}{Z}\right) \sum_{n=1}^{N} (s_n, x_n, y_n) I(c_n > t), \tag{2.8}$$

$$Z = \sum_{n=1}^{N} I(c_n > t), \tag{2.9}$$

$$I(c_n > t) = \begin{cases} 1, \text{ if } c_n > t. \\ 0, \text{ otherwise.} \end{cases} \tag{2.10}$$

Here s is the the scale element, x, y are the coordinates of the window's upper left corner position, t is the threshold and c_n is the confidence score. In the Face Detection Data Base (FDDB) data set, a commonly used benchmark for face detection algorithms, Cascade CNN reaches a detection rate of 85%. At the same time, for an image with a size of 640 x 480, under the condition that the detectable face size is 80 x 80, this detector is able to keep a processing speed close to 10fps on the CPU [46]. This running speed marks a drastic improvement compared with other face detectors considering that the Cascade CNN detector achieves a high detection accuracy at the same time. For our multi-face tracking system, we first employed the cascade CNN face detector for the detection step. The six CNN layers of the cascade CNN face detector appear to be somewhat redundant as the follow-up Multi-Task Cascaded Neural Network (MTCNN)

uses a simpler architecture with better results. Although, in our system, we select the MTCNN face detector, Cascade CNN face detector is an important design and successful implementation of a face detector with deep learning techniques.

2.4.5 MTCNN face detector

Several well performing face detectors based on neural networks have appeared recently [46, 63, 25, 20]. As our work is focusing on real time video face tracking, a fast detector to correctly initialize the location of each face is very important. In our work, we use a fast and effective face detector called Multi-task Cascaded Nerual Networks (MTCNN) detector.

In October 2016, Zhang et al. [83] introduced a new face detection algorithm using MTCNN. The detector is inspired by the Cascade CNN detector to identify the face location. However, instead of only using CNN layers to calculate the confidence of each face window, MTCNN detector also uses the CNN layers to calculate a bounding box regression and facial landmark (refer to glossary) localization at the same time, and hence the name "Multi-task" CNN. For each video frame, the MTCNN face detector only runs 3 CNN layer, which reduces the runtime cost.

Bounding box regression

As previously noted, the Cascade CNN detector is using another calibration layer after each CNN layer to get the offset of each face position. The MTCNN detector only uses the CNN layer to get two more types of results, which are the bounding box of the face in a regression [83] (an offset of each face window) and five point facial landmark localization (five points are two pupils, two corners of mouth and the tip of the nose). Bounding box regression determines the distance the input face window needs to move and in which direction, instead of feeding more windows into the calibration layers to re-calculate all the candidate windows confidence again. The MTCNN detector's bounding box regression gives the position offsets of the output to the input face window position to get the final output position. It significantly reduces runtime cost.

Facial landmark localization

Besides adding bounding box regression as an output, another output of the CNN layer is facial landmark localization [83]. It means, five facial landmarks will also be detected. Calculating facial landmark localization actually contributes to the face detection accuracy.

Cascade structure

There will be positive influence among relative tasks. The main task in face detection is locating features and predicting the location of key points, such as the center of the eyes, the tip of the nose, edges of the mouth and etc. Based on this idea, in 2014, the Joint Cascade [13] was introduced. This is known as an alternative cascading face detection classifier with regulators which are required to predict feature positions. Then face detection and locating features can be completed at the same time. This technique improves accuracy as different faces have different feature point locations.

Given that the MTCNN face detector can run faster than the detectors we have discussed above with good accuracy [83], we employ the MTCNN detector in our multi-face tracking system. In Chapter 3 we will introduce more implementation details of the MTCNN face detector.

2.5 Face tracking

With the development of image processing, face tracking emerged as one of the standard image processing tasks. In 1998, Liang et al. [48] presented a way of combining skin-color segmentation and geometric analysis to track a face at a speed of 15fps. A similar color segmentation method was also used for other face tracking algorithms [80, 67]. In 2015, Ranftl et al. [61] published a real-time face tracker based on optical flow. Liu et al. [50] presented a face tracking system based on Local Binary Patterns. In addition, in order to meet industrial needs, more face trackers are designed for controlling the user interface [11, 10, 16] (e.g. in a game) or for a camera [57]. Shakhnarovich et al. [65] demonstrated real time face tracking with a gender classification feature. Many image

processing methods were used for face tracking, such as Kalman filters [59], histograms analysis [59], mean-shift and wavelet networks [40].

2.5.1 Face tracking with feature extraction

After the feature extraction methods in increasing number was released, it is compared with image analysis method to extract facial feature, more attempts were made to extract facial features have been done in facial tracking.

Yang et al. [79] presented a face tracking method based on a radial basis function (RBF) network. For training the network, they used the Olivetti Research Laboratory (ORL), Cambridge, U.K., data set. This data set contains the faces of 40 persons from ten different angles in front of the same background. Although it is a well established data set for training, unfortunately all the images are only grey scale. Yang et al. [79] show that, the face tracker can track multiple people and verify the same person's face from two continuous frames. However, the experimental results in the paper show that the faces that are being tracked are very clear and the background is not complex. Moreover, the number of people is limited to three. In addition, they only show the accuracy tested on a grayscale data set. Hence, this face tracking system was still quite limited due to the above facts.

Minyoung et al. [39] introduced a face tracker based on face recognition with face feature extraction. They use the Honda/UCSD Video data set released by the University of California San Diego. This data set contains many different pictures of faces. Each picture only contains one person's face with high resolution. For each person, there are five pictures taken from different angles, but in front of the same background and with the same illumination. The data set are all grayscale, and this face tracker is only for tracking a single face in gray scale videos. Hence, it has a number of limitations for our intended application.

2.5.2 Runtime efficiency

There are many real-time face trackers [80, 67, 11, 10, 16, 40, 74, 67, 21] that are limited to tracking only one face at a time. There are many algorithms that are adopted by these face trackers, such as skin color segmentation [67], Kalman filter, Gabor wavelet networks [21], etc. Among these methods, the Kalman filter is one of most popular algorithms used in many trackers for predicting the object position in the next frame.

One tracker [28] in the 2014 VOT challenge used a CNN to achieve general object tracking. This object tracker used a CNN to track the object and made CNN tracking run in real time for the first time. However, this tracker is only working for single object tracking with manual target initialization. This CNN uses the position information and object features to track the object. Similarly, in our tracking process, the tracking network is also designed to calculate the similarity of two matching patches, referring to the position information together. In comparison, the method of this thesis is working in real time for multi-face tracking with automatic initialization of the face position.

2.5.3 Multi-face tracking

Multi-face tracking is more complicated than single face tracking for several reasons. The algorithms involve more calculation and additional decision making criteria. There are several algorithms [52, 51, 68, 3] for implementation of multi-face tracking, however, for there is no access to the evaluation video they use, we cannot do comparison with them. In 2017, Wu et al. [77] presented a multi-face tracker based on a Coupled Hidden Markov Random Field (CHMRF) algorithm. However, Wu uses a detector to locate all the face positions in the video first and then links the faces with the same label which are assumed to belong to the same person. Therefore, this method is only able to track in video that has been previously recorded. The algorithm itself runs in real time but it cannot apply to live videos. Although the algorithm limitation is obvious, the performance evaluation results show that this multi-face tracker has been better than most of the trackers in both accuracy or the number of label switches for a person.

Roth et al. [19] implemented a face tracker based on the Hungarian algorithm and

the Discrete Cosine Transform (DCT). This face tracker can run without pre-processing the entire video file in advance. However, because of the high computational complexity of the algorithm, the average runtime cost is about 2s per frame on an AMD Phenom X4970 quad core processor and improvements are still needed to run in real time. In addition, the accuracy of this algorithm is not as competitive as the algorithm Wu et al. [77]. A comparison is shown in Chapter 4.

Le et al. [43] introduce a method for multi-face tracking, which comprises of color histograms and feature extraction based on SURF, as well as the position information to follow the faces moving in the video. However, it is not a real-time tracker, the runtime is 6-7 fps. This tracker's performance is average in tracking accuracy as shown in Chapter 4, based on results with two evaluation metrics. The results show very few label switches for tracked people. However, it usually produces a larger number of tracklets (refer to glossary) than other trackers. Face tracklet is the 2D trajectory of faces over time in the video. This means that the tracker has a hard time to deal with camera cuts, face occlusion, etc. and it shows this tracker's limitations.

There is another method [78] which implements multi-face tracking based on a hidden Markov random field model but this tracker cannot run in real time. According to the results with the evaluation metrics used in this thesis, this multi-face tracker is not very competitive on accuracy nor on runtime cost.

2.5.4 Restrictions

Because of the challenges of the task and in designing the algorithms, many restrictions are presented in the above state-of-the-art multi-face trackers. The method by Le et al. [43] needs to modify some parameters of the tracker to adjust to videos containing different scenes and the tracker performance will partly depend on these parameter settings.

In comparison with the face trackers presented above, our multi-face tracker does not have the restrictions listed above. The multi-face tracking system described in this paper is a real-time tracker without on-line training based on the test video. This algorithm does not need to change any parameters according to the different input video conditions,

and the runtime cost will not be affected significantly with an increase in the number of faces.

2.6 Performance evaluation criteria

Almost all the face trackers released before 2000 did not mention the runtime performance of the face tracker. Some mention good classification results, but they do not present the data set used for testing. Even now, it is not easy to evaluate performance of a face tracker as there is no standard benchmark. However, after Li et al [47] released a series of the evaluation definitions (i.e. Recall, Precision, GT and etc) based on their multi-target tracking, authors of other multi-face tracking papers [43, 19, 64, 77, 78] present their experiment results according to the evaluation metrics of Li et al. There is another paper [9] that proposed an evaluation method involving a parameter MOTA, which is called CLEAR MOT, can be regarded as a comprehensive evaluation value for a multi-face tracker. The CLEAR MOT evaluation method is also often used by multi-face tracking algorithms [43, 64, 77, 78].

In our multi-face tracking evaluation, we will be using both of these methods to test the performance of our real-time multi-face tracking system.

To the best of our knowledge, there is no specific benchmark for evaluating face tracker performance, but most recently released multi-face tracking algorithms are tested on the videos of Frontal and Turning data sets provided by [55] and the TV program The Big Bang Theory season 1 episode 1. So, combining the methods of the evaluation metrics of Li et al. [47], CLEAR MOT [9] and these commonly used videos, different multi-face trackers can be compared.

2.7 Summary

As an increasing number of face detectors begin to adopt deep networks, the accuracy of face detection begins to increase significantly. In 2014, the academic community in Face Detection Data Set and Benchmark (FDDB) [34] used the JointCascade face detector

and obtained the best result of the detection accuracy of 84%. By 2015, this record was broken by Faceness-Net [81] with an accuracy close to 91%. Today the best detection record has reached 92.5% with more than one approach obtaining a rate of more than 90%. These results are obtained by face detectors based on deep networks.

The VJ Face Detector has inspired and influenced many subsequent works because its introduction of the integral plot and the cascade structure. These are still used in a variety of detectors in different forms today. Traditional face detection technologies excel in speed, while their accuracy cannot compete with the methods based on deep networks.

Deep networks-based detectors are currently able to achieve very high detection accuracy, and their versatility is very strong, but the cost of these benefits is also very high in terms of calculation complexity. The Cascade CNN detector can be considered as a representative of a combination of traditional technology and deep networks. It contains a number of classifiers; these classifiers are organized using a cascade structure. The MTCNN detector is an enhanced version of the Cascade CNN with a better accuracy. It yields a better performance in both speed, and accuracy. Therfore, we have incorporated this competitive face detector in our tracking system.

There are many algorithms to solve the one face tracking problem. However, far fewer algorithms have been released for multi-face tracking. Usually, people solve the multi-face tracking problem by face clustering. In addition, many multi-face tracking systems do not consider the runtime cost. As a result many multi-face tracking algorithms face limitations when running on live video. Because multi-face tracking is more complicated than single face tracking, the algorithms are usually designed in more complicated ways such that the computation complexity increases significantly. The algorithm in this thesis is designed for tracking multiple faces in live video, so its design is more motivated by runtime efficiency than other algorithms. According to the comparison results which are presented in Chapter 4, our tracker can run in real-time and it also yields better performance in accuracy than other multi-face trackers in the comparisons of this thesis.

Chapter 3

Multi-face Detection and Tracking System

In this chapter, we introduce a real-time multi-face tracking system. As presented in Chapter 2, many papers [80, 67, 11, 10, 16, 40] describe single face tracking systems which can run in real time but are limited to only a single face. There are some other papers [53, 12] that describe multi-face tracking without labels, which is confusing for distinguishing the face from the corresponding tracking window. There are some algorithms [43, 19, 64, 77, 78] that implement multi-face tracking with labels or different colors of the tracking windows. However, all of these algorithms cannot run in real time. They either need a long time for computation, or can only detect faces in videos that are pre-recorded, and

then link the same face to give the tracking results.

With the development of deep learning, face recognition has become very accurate. However, until now, face recognition [69] can only be implemented in deep neural networks, which run with a large time cost. Traditional face recognition algorithms [2, 4] with empirical feature extraction methods usually have a relatively high error rate which results in low accuracy. Therefore, face trackers using face recognition for matching the same face usually have the disadvantages of either a large runtime cost, or an unacceptable accuracy.

However, inspired by the implementation of face recognition based on CNNs, we build up a very shallow network for matching the detected faces from the current frame with the ones of last frame. Usually, a deeper neural network is able to determine a more precise result, but deeper networks cost more time for calculation and this results in higher runtime cost. Because of the depth limitation of the shallow neural network, we extract the face feature maps as the network input to do the matching. In this way, we reduce the tracking network computation amount and improve the result accuracy. In this thesis, we present a real-time face tracking system comprised of face detection, feature extraction and tracking. The tracker contains a shallow neural network for matching two faces from the last frame and the current frame. This multi-face tracker can run in real time and has a high accuracy.

In this chapter, we first present the overview of our proposed method in Section 3.1. Then, the face detection, which is the initialization of our system, will be introduced in Section 3.2. We describe the face feature map extraction and face tracker in Section 3.3 and 3.4, respectively. Section 3.5 introduces the cosine similarity method as a comparison and each component's contribution to this system. In addition, in Section 3.6, we present our considerations for face label retention. Section 3.7 shows the tracking network training preparation. Finally, there is a summary in Section 3.8.

3.1 Overview of the proposed method

The architecture of our multi-face tracking system is shown in Figure 3.1. There are three blocks constituting the whole system, these are face detection, feature extraction and tracker.

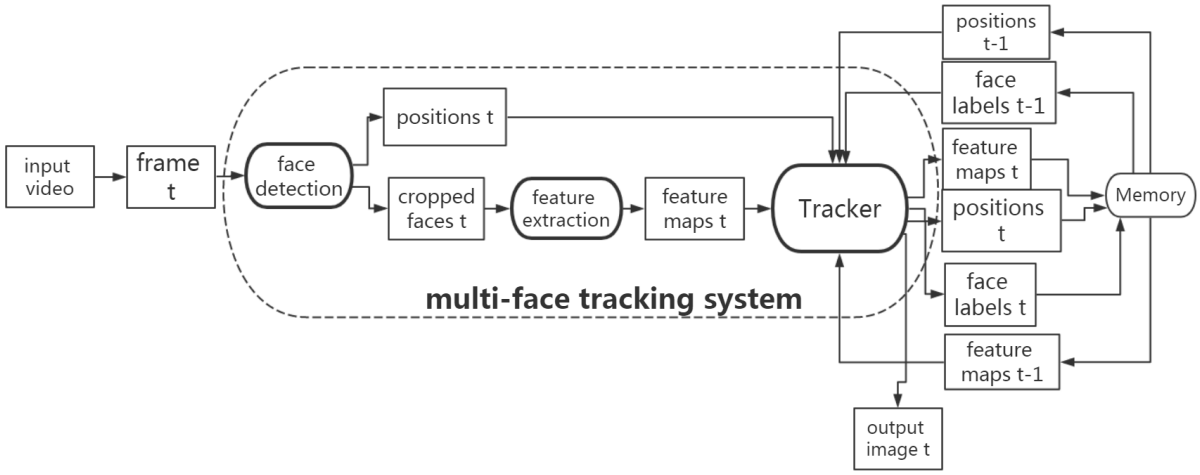

Figure 3.1: Architecture Overview.

In this multi-face detection and tracking system, the face detection is the first step to initialize the whole system by locating the face positions. For each face, the outputs of the face detection are classification result with scores, bounding box regression and locations of face landmarks. If it is the first frame of input video, the labels are put on each detected face in the order of the face detection results. In Figure 3.1, frame t stands for the last frame and frame t-1 stands for currently tracking frame. Face detection is done on each single frame.

Then we crop each face image in the original input image to pass through the feature extraction block to get the feature map of each face. According to the feature map of each detected face, the tracker will compare the current feature map of a face with the feature maps of the faces from the last frame for matching. If the current face gets a good match with a face from the last frame, it will inherit the same face label. Otherwise, it is regarded as a new face and will be assigned a new face label. Figure 3.2 presents the

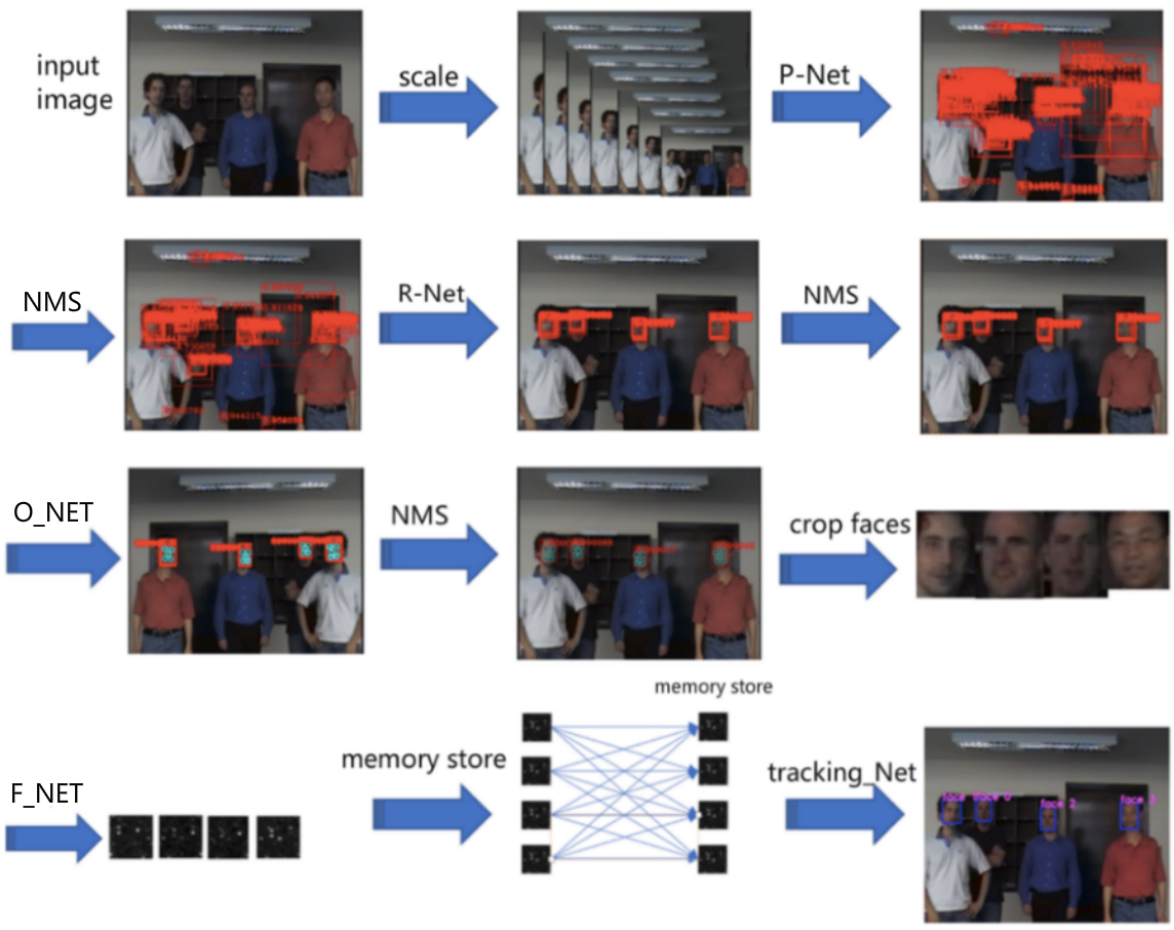

Figure 3.2: Face tracking system framework with outputs of each step. The original input image will be scaled by 0.5 until the image size is equal to or smaller than the minimum detection face size. P_Net, R_Net, O_Net and NMS are introduced in Section 3.2 F_Net and tracking_Net are introduced in Section 3.3 and 3.4, respectively. Image is from data set [55].

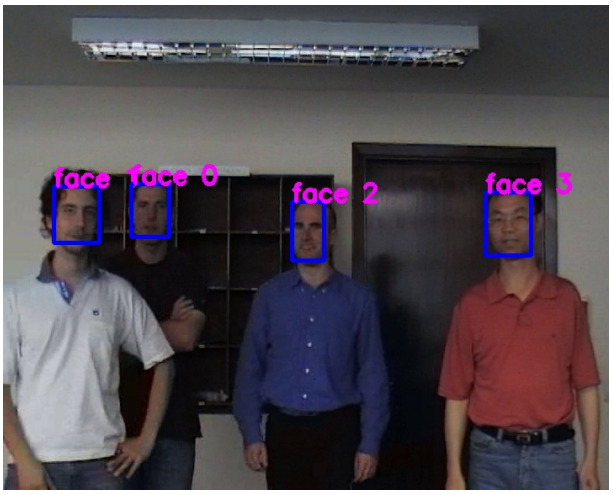

Figure 3.3: A sample of current frame t output. Image is from data set [55].

outputs of each step in the implementation procedure.

The input of this multi-face tracking system is the video, which can be a pre-recorded video or the live stream which needs tracking results in real time. Because we have designed three alternative networks (Network 1, Network 2 and Network 3) for the face matching task in the tracker block, and our implementation provides all three networks for comparison. Our system only uses one network to obtain the matching result. We make the decision which network to use at the beginning of the program. Then the system uses only that selected network to do face matching in the tracker block for the whole tracking process.

For each single frame, the output of this system is the output image, which will later constitute one frame of the output video. The output image is the input frame marked with the detected face locations and labels. Figure 3.3 shows a sample of the output image, a result on our testing video.

We will describe each function step by step in Section 3.2, 3.3 and 3.4, respectively.

3.2 Face detection

In our multi-face tracking system, we deploy the multi-task CNN face detector (MTCNN) [83] for face detection. The MTCNN face detector adopts the cascade structure to com-

bine 3 CNN layers to process the input image. These 3 CNN layers are shown in Figure 3.4. This process is similar to the cascade CNN [46]. However, instead of only using a CNN layer to calculate the confidence value of each face window, the MTCNN detector also gives bounding box regression (x0, y0, x1, y1) and facial landmark localization as another two outputs at the same time. This design reduces the computational runtime cost and improves the face detection accuracy.

The processing pipeline of this face detector is:

1 The input image will be built into an image pyramid [1]. The size of each level of this image pyramid is from the original input image size to the minimum face size (pre-set parameter). The level number is depending on both the size of input image and minimum detection face size. All the scaled images will be the inputs of the face detector and will be first fed into the P_Net, one by one.

2 The P_Net scans the whole scaled image densely to get the candidate windows with a size of 12 x 12. All of the 12 x 12 images are the input images of the P_Net. The input image is 12 x 12 x 3 (the height is 12 pixel, the width is 12 pixel and it has 3 channels of R, G, B).

   The CNN structure of the P_Net is shown in Figure 3.4. Each picture presents the output of each layer in the P_Net. The labels above each data block describe the previous layer's output and the next layer's input. For example, 10@10x10 means this layer's output is 10 images with a size of 10 x 10.

   The labels underneath the arrows describe each layer's information. For example, convolution 10 3x3 filters stride 1, means this layer is a convolution layer, and the layer output number is 10. This convolution layer contains kernels with a size of 3 x 3. The stride size of this convolution layer is 1 when sliding the convolution kernel over the input data. Stride is the distance in pixels which we slide the filter with at a time. In our neural networks, because we do not need to keep the input image size as the same as the output size, as well as our kernal size is small, we do not use zero-padding to make up around the border in the convolutional layer when we training the CNNs.

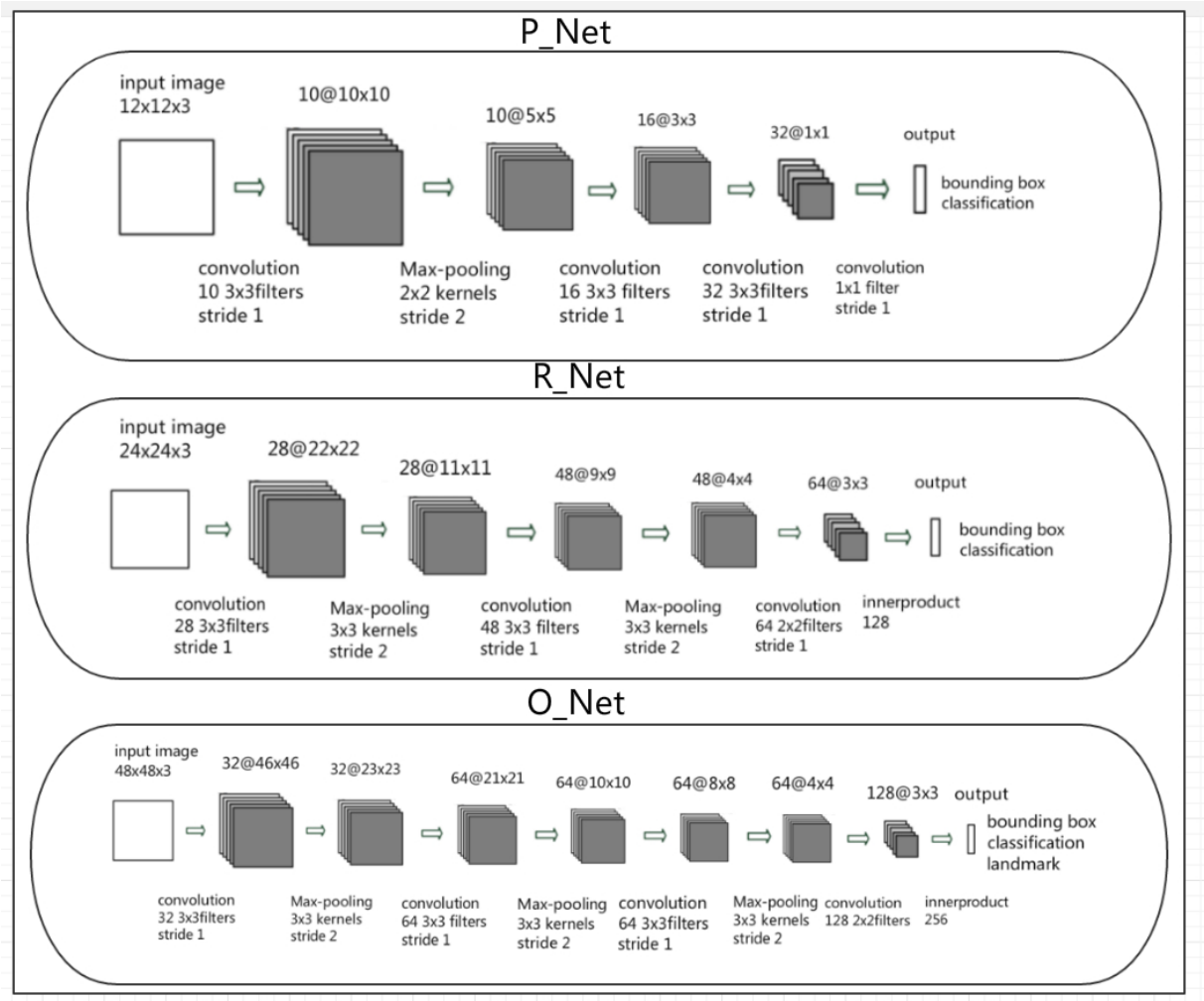

Figure 3.4: Face detection architecture.

10@10x10 means the output of this layer is 10 dimensional with a size of 10 pixels by 10 pixels; Convolution means convolution layer; 10 3x3 filters means this convolution layer has 10 outputs and the convolutional kernel size is 3 x 3; Stride 1 means this convolution layer's mapping stride is 1. The details of each layer function process are discussed in Chapter 2.

It is worth mentioning that we do not draw the activation layer in this structure diagram, as the activation layer does not change the image size. All the activation layers adopt a ReLU layer in this CNN. The complete structure of the P_Net is convolution layer, ReLU layer, pooling layer, convolution layer, ReLU layer, convolution layer, ReLU layer and convolution layer. All the parameter setting details of each layer are shown in the Figure 3.4, P_Net.

The output of the P_Net are classification results and bounding box regression. The classification results give each input image two scores, one stands for the detected face confidence, and another one stands for the non-face confidence. The bounding box regression contains 4 float numbers (x0, y0, x1, y1) as introduced in the Chapter 2.

In the P_Net, after calculating the confidence score for each window, most of the candidate windows whose confidence is lower than the pre-set threshold will be rejected.

3 The R_Net will crop out the remaining candidate windows in the original image and resize them into 24 x 24 as the input images. Again, the R_Net will reject most of the remaining candidate windows with a confidence score lower than the pre-set parameter (here we use 0.92).

The CNN structure of the R_Net is shown in Figure 3.4. The complete structure of R_Net is convolution layer, reLU layer, pooling layer, convolution layer, reLU layer, pooling layer, convolution layer, reLU layer, convolution layer, reLU layer, innerproduct layer. All the parameter setting details of each layer are shown in the Figure 3.4.

The outputs of this R_Net are classification result and bounding box regression. The classification results show the confidence and non-face probability.

All the remaining candidate windows will be fed into the O_Net afterwards.

4 The O_Net crops out images in the original image according to the left candidate windows (output from the last step and windows with higher confidence than

threshold), and then scales the cropped images into a size of 48 x 48 as inputs for this layer.

The CNN structure of the O_Net is shown in Figure 3.4. The complete structure of the O_Net is convolution layer, ReLU layer, pooling layer, convolution layer, ReLU layer, pooling layer, convolution layer, ReLU layer, pooling layer, convolution layer, ReLU layer, convolution layer, dropout layer, ReLU layer, inner product layer. All the parameter setting details of each layer are shown in the Figure 3.4.

The output of this O_Net is classification results, bounding box regression and facial landmarks. The classification results contains the confidence score. All the candidate windows with the confidence score lower than the threshold will be eliminated in this layer. All the candidate windows remaining after O_Net will be output as the bounding boxes of the detected faces.

After each CNN layer (P_Net, R_Net and O_Net), all the candidate windows are put into a step called non-maximum suppression (NMS). NMS will eliminate the candidate windows which have a large overlap where the IoM is bigger than 0.5. If two candidate windows share a very large overlap, it means these two windows are locating the same face, then the NMS will delete the one with the lower confidence score. Therefore during the face detection process, NMS will be called three times in total, as shown in Figure 3.2.

This face detector has multiple outputs, but we only use the face detection result in our multi-face tracking system.

3.3 Feature map extraction

The face feature map extraction is preparing the inputs for the tracker block. In order to avoid the bounding box regression offset of output feature map, we build up a CNN layer, F_Net, to achieve feature map extraction. The F_Net has a similar CNN structure as the O_Net but the output is only the feature map which is different from the multiple outputs of the O_Net. We crop the detected face of current frames and feed them into

the F_Net, whose structure is shown in the Figure 3.5. The output of this network is a 256 x 1 feature map vector and the input is each detected face image one by one.

The complete structure of F_Net is convolution layer, reLU layer, pooling layer, convolution layer, reLU layer, pooling layer, convolution layer, reLU layer, pooling layer, convolution layer, reLU layer, convolution layer, dropout layer, reLU layer, innerproduct layer.

The F_Net is very similar to but slightly different from the O_Net:

1 The input of F_Net are the detected face windows which are scaled into 4 x 48.

2 The F_Net has only one output when we run it in the test. After passing the last inner product layer with 256 kernels, the output is a 256 x 1 vector, which is the feature map of the input face image.

3 Training the F_Net is similar to training a face detection CNN layer. We initialize the F_Net with the weights from the O_Net which has the same layers. The output of training process is the confidence value. We also use the face and non-face images with "1" and "0" labels to train F_Net. However, there is no face landmarks and bounding box regression as outputs in F_Net.

This step transforms the tracked faces into features. Instead of putting the feature extraction process in the tracking CNN, we finish the feature extraction in advance in the F_Net. In this way, for each detected face, the feature extraction process will only be called once, rather than repeating it as part of the tracking network. It reduces the system's runtime cost and also allows us to maximize the simplification of the tracking network in the tracker block.

As we use similar CNN frameworks to get different results, the training process and training data set are similar for the F_Net to the O_Net but simplified to a degree such that there is no face landmark and bounding box regression. The implementation details are discussed in Section 4.1. When we prepare the training data, there is no bounding box regression and face landmarks but only face and non-face samples and corresponding labels.

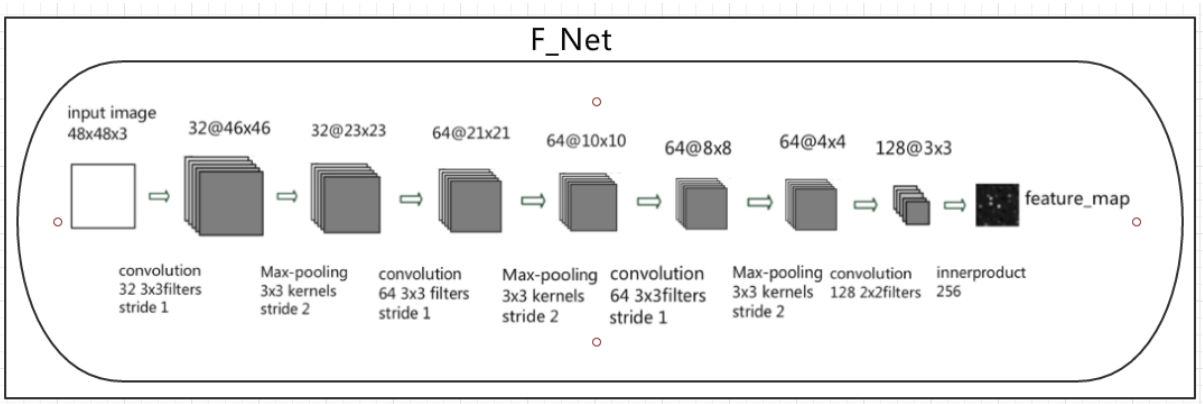

Figure 3.5: Feature Extraction Architecture.

3.4 Tracker

In our multi-face tracking system, the tracker block works on matching the faces of the current frame t with the ones from the previous frame t-1. The tracker will assign a label for each face according to the matching results.

Overview of tracker

The processing pipeline of the tracker block is shown in Figure 3.6. The tracker needs the inputs from the last frame t-1 which are stored in memory: the positions of each face, the feature maps and face labels. It also needs the input from the current frame t: the detected face positions and feature maps of the detected faces.

The tracker block contains four steps for implementation:

1 Position check

The tracker will use face positions from the last frame t-1 and the current frame t to filter out unlikely matches because of image distance.

2 Reformatting of feature maps

After the position check, all of the remaining face feature maps will be put into a square image-like format. Each 256 x 1 feature map vector will be formatted into

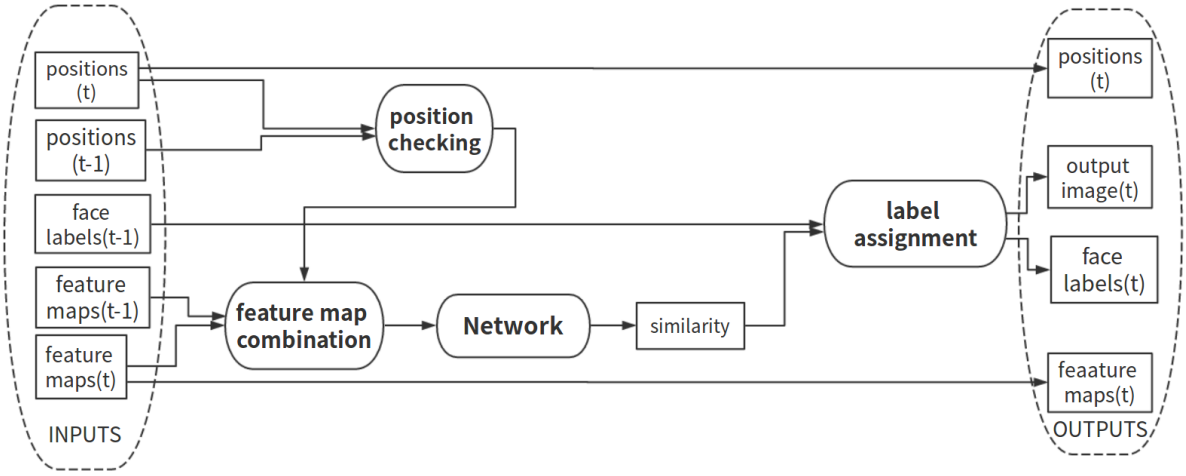

Figure 3.6: Tracker architecture.

a 16 x 16 one-channel grayscale image with the original real values. Two feature maps which are going to be compared need to be first put into a 16 x 16 two-channel image, i.e., two feature maps are put into one image's different channels. This image is the input to the tracking network.

3 Network

The network is to calculate two face similarity values. The input of this network is the 16 x 16 two-channel image. The output of this network is the similarity score of two faces.

On the condition that the system is able to run in real time, we design three networks. These three networks have the same function but different accuracy.

4 Label assignment

The faces in the current frame will inherit the label of the face who has the highest similarity value calculated with the faces of the current frame.

The outputs of the tracker block are the face positions, feature maps, face label and the current frame image with labeled faces.

Position check and reformatting of feature maps

In the tracker block, the position check step is to search for potential areas in last frame t-1 where the face in current frame t originates from. We use the face position information as a reference to filter out the unlikely matches and help the tracker to select the candidate face label to possibly inherit.

In the position check, we use the two face rectangles to calculate the Intersection-over-Union (IoU) result. If the IoU result is higher than the pre-set threshold of 0.5, which we refer to the setting from [34]. (We used the IoU with threshold of 0.2, 0.3, 0.35, 0.4, 0.45, 0.55, 0.6 and 0.65, however, on the test video, the threshold of 0.5 performs the best. The thresholds smaller than 0.5 increase identification switches, however, values bigger than 0.5 excludes even correct faces). Then we keep the face label as the candidate label to select from, or the face label is rejected in this step. The tracker will repeat this step for all the currently detected faces.

The second step in the tracker block is the feature map reformatting. In order to simplify the process of both training and testing, we combine two face feature maps into a 2-channel image for comparison. The output from the feature extraction network F_Net is 256 x 1 feature map vector. We first transfer a 256 x 1 vector into one 16 x 16 one-channel grayscale image. Then we combine two grayscale images into one two-channel floating point image as the input of tracking network.

We use a two-channel image to contain the whole input information to be compared in the tracking network instead of inputting two 256 x 1 vectors into the tracking network. Because if there are two inputs (two 256 x 1 vectors corresponding to each feature map) to be fed into the tracking network, there would have to be at least two more layers in the tracking network to process and combine these two inputs which will increase the network depth and adversely influence the design of our very shallow tracking network.

Network

In order to ensure runtime efficiency, we design the tracking network in the simplest way. Compared with harder tasks such as object recognition and face identification,

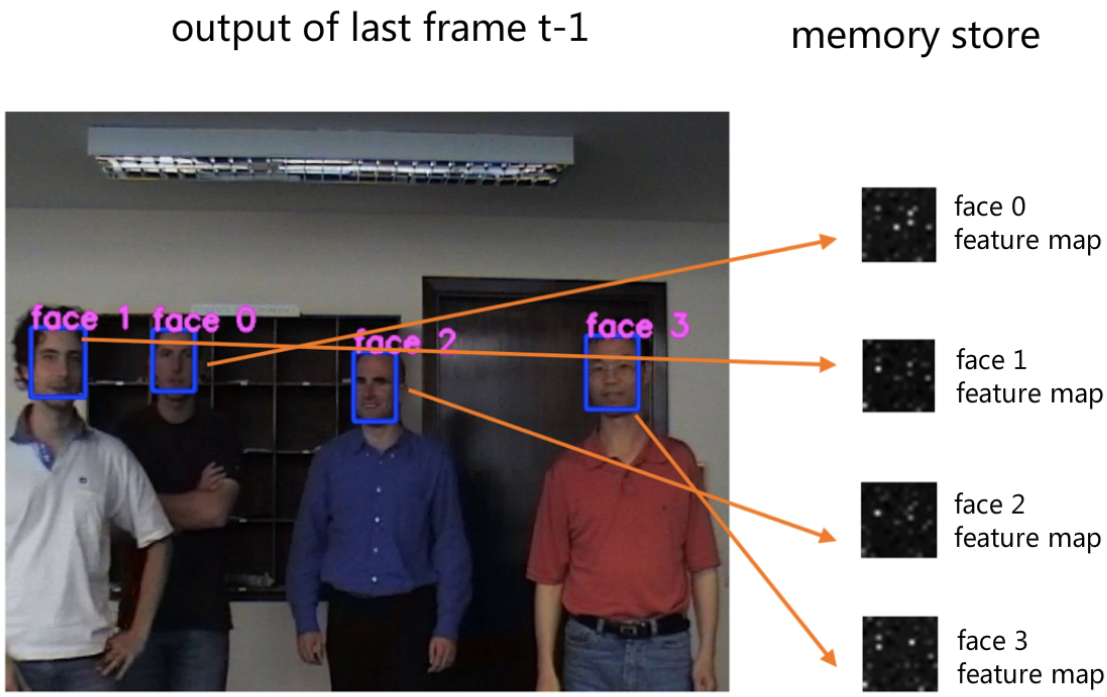

Figure 3.7: Information needed from last frame. Image is from data set [55].

our tracker only needs to calculate the similarity of two feature maps. The tracking network does not need to calculate face features, which enables a very simple design of the tracking networks. We start from the simplest way to build up the tracking network to keep the real-time runtime cost low, and then increase the number of layers gradually.

The input of the tracking network is a two-channel image, where one channel contains the face of the current frame t, and another one is a face from the last frame t-1 for comparison. The output of the network is the similarity value. The similarity value expresses how likely these two feature maps are from the same face or not.

We design three networks: Network 1, Network 2 and Network 3, which have the same function but different layers. The result in Chapter 4 shows that the 2-layer network works better than the 3-layer network and therefore, we do not design deeper networks for our tracking. Another reason is the deeper network will have an adverse influence on runtime cost of our tracking system and hence we stop at a 3-layer network. In Chapter 4, we will use the common evaluation metrics to compare the results of these three different tracking networks and their time performance. Figure 3.8 shows the design of the three networks: Network 1, Network 2 and Network 3.

1 Network 1:

   As the input image is a 16 x 16 image with 2 channels, we select a convolution layer with one 16 x 16 kernel as this network's only layer;

2 Network 2:

   In Network 2, we use one convolution layer and one inner product layer to calculate the similarity value. The first convolution layer contains ten 7 x 7 kernels and the second convolution layer contains one 10 x 10 kernel;

3 Network 3:

   In Network 3, we use one convolution layer, one pooling layer and one inner product layer. The first convolution layer contains ten 7 x 7 kernels, the pooling layer contains a 2 x 2 kernel and the second convolution layer contains one 10 x 10 kernel.

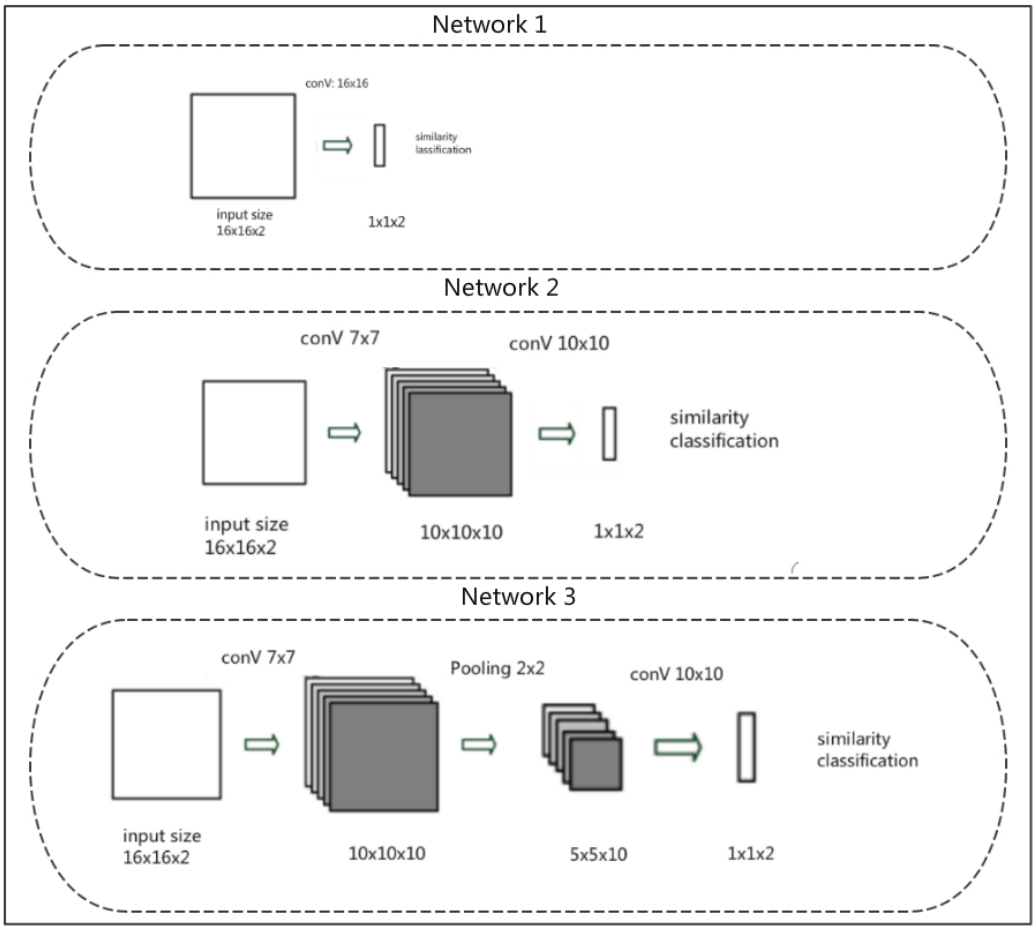

Figure 3.8: Tracking network structure.

The tracking network is a two-category clssifier. All of these three network inputs are a two-channel image which is the output from the feature map reformatting step. The output of each of these three networks are a 2 x 1 vector, where one value is the similarity score which stands for the probability of these two faces belonging to the same person and the other value is the probability of the two faces not being the same. When we run this multi-face tracking system, we only use one of the networks.

Label assignment

Label assignment is a step of the face tracking to decide and assign the face label to the detected faces in the current frame t according to the result of the position check and the result of the tracking classifier. For the label assignment step, the information we need from the last frame are the location of each face, the feature map of each face and the labels of the corresponding face. This is shown in Figure 3.7.

The implementation process is listed below:

1 For the first frame of the input video, there is no tracking process. All the faces detected by the face detector are labelled according to the detection output order. These assigned labels are the original face labels.

2 From the second frame, the IoU result will be calculated by each detected face in current frame t and each face from the last frame t-1. If the IoU result is bigger than the pre-set threshold of 0.5. We select the same threshold Jain and Miller used in [34]. The faces from the last frame t-1 that pass the threshold will be stored in the candidate vector.

3 If the candidate vector is not empty then we will calculate a similarity value for all the faces in the vector with the current target face. The current target face will inherit the label of the one with the highest similarity value.

4 If the candidate vector is empty, then the system will put the currently processed face into the vector of unmatched faces.

5 After finish the label finding process for all the faces in the current frame t , the system will check whether all the labels from the last frame have been matched. If there are some labels left, all the labels will be put into the vector of empty positions in order from the smallest to the biggest number. In the vector of empty positions, we count how many times a face has not been matched.

6 If the vector of non-matched faces is not empty, the system will check whether there is a label in the vector of empty positions that has been counted more than 3 times. If it has, the face in the vector of non-matched faces will inherit this label, otherwise the system will create a new label with a number plus one of the currently largest label number. If a face reappears, which has been lost for more than 3 frames, is likely that it can be matched. However, it will receive a previous face label and not a new face label as the system simply re-uses ID labels.

7 Each detected face in current frame t has a face label that will be stored into the memory with its corresponding face position and face feature.

3.5 Evaluation of tracking steps

As there are two steps to filter out the faces which are not belonging to the same face label in our multi-face tracking system, position check and tracking classifier, we compare each component's contribution to this tracking system here.

We run this multi-face tracking system with only the position check in the tracker block without a tracking classifier to evaluate the contribution of the position check. We also run this system only with the tracking classifier without the position check to see the results of the tracking classifier alone. We also only run the cosine similarity to compare results to see each component's contribution to this whole tracking system.

Cosine similarity

We also use the cosine similarity to calculate these two face feature map similarity, the cosine similarity function is

$$similarity = cos(\theta) = \frac{\sum_{i=1}^{n}(A_i B_i)}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}}, \tag{3.1}$$

The cosine similarity achieves the same function as the three tracking networks. The cosine similarity can therefore replace the tracking network in our multi-face tracking system. In Chapter 4, we further compare the results of the multi-face tracking system using cosine similarity method with the tracking networks.

We employ the commonly used CLEAR MOT evaluation method introduced by Bernardin and Stiefelhagen [9] to evaluate the tracking result with each single component. CLEAR MOT defines various evaluation scores including MOTA, which could give a comprehensive score of tracking performance. There are four more parameters in this evaluation metrics, FP_t (the number of false positives), $MISS_t$ (the number of false negatives), MR (the number of miss rate) and IDS_t (the number of identity switches). More details about this evaluation metrics is given in Chapter 4.

Method	MOTA	FP_t	$MISS_t$	MR	IDS_t
Position check	**88.77%**	1	92	2.16%	**386**
Network 1	60.51%	1	92	2.16%	1592
Network 2	64.26%	1	92	2.16%	1432
Network 3	64.85%	1	92	2.16%	1499
Cosine similarity	63.72%	1	92	2.16%	1455

Table 3.1: Each component evaluation result on video "Frontal"

Position check, Network 1, Network 2, Network 3 and cosine similarity comparison based on the evaluation method MOTA [9] and the test video "Frontal" [55]

From the CLEAR MOT evaluation, as shown in Table 3.1 and 3.2, we can see that the combination of the position check and face detection can achieve the highest score, while these three networks and the cosine similarity cannot work as well without the position

Method	MOTA	FP_t	$MISS_t$	MR	IDS_t
Position check	**86.85%**	2	3	0.11%	**363**
Network 1	51.16%	2	3	0.11%	1362
Network 2	53.20%	2	3	0.11%	1305
Network 3	51.73%	2	3	0.11%	1346
Cosine similarity	52.41%	2	3	0.11%	1327

Table 3.2: Each component evaluation result on video "Turning"

Position check, Network 1, Network 2, Network 3 and cosine similarity comparison based on the evaluation method MOTA [9] and the test video "Turning" [55]

check as shown in Table 4.11. However, the tracking results have been much improved by combining the position check and a tracking classifier. The evaluation results are presented and discussed in Chapter 4.

3.6 Face label retention

Figure 3.9: Example without face label retention

Example output of The Big Bang Theory Season 1 Episode 1 without face label retention. Two different characters inherit the same face label in this example. These examples show several camera cuts where the scene changes. This result is obtained by our multi-face tracker with Network 2.

Figure 3.10: Example with face label retention

Improved example output of The Big Bang Theory Season 1 Episode 1 with adding the face label retention. The improvement results show that our multi-face tracker overcomes the camera cut challenges in a number of cases. This result is obtained by our multi-face tracker with Network 2.

Improvement of face label retention

In the implementation process of label assignment, because we make the face label retention for 3 frames, which is shown in Section 3.4, Label assignment part, step 6, the tracking results get a big improvement. We call this improvement step as Face Label Retention. Figure 3.9 shows results without face label retention and Figure 3.10 shows the improved results. The comparison of Figure 3.11 also shows the improvement by adding the step of face label retention. These two examples are showing tracking challenges including camera cuts and face occlusions. Camera cuts are a challenge for many multi-face trackers. The result with the improvement step shows our multi-face tracker can overcome some of the camera cuts to track the same person who disappears in between some frames during the whole video stream. Our results with face label retention show that an occluded face can continuously be tracked with the same label across the occlusion.

There are some other challenging situations for face tracking task. For example, in the testing videos, the face detector might fail to detect one person's face in one frame, while there are detection results for the same person's face in the previous frame and in

Figure 3.11: Example comparison of face label retention

An output video fragment of "Frontal" testing video [55]. The left video frame cluster shows the output results without face label retention. And the right frame cluster shows the results with face label retention. This comparison shows that our multi-face tracker can overcome some of the face occlusion situations. This result is obtained by our multi-face tracker with Network 2.

the upcoming frames. This is a false negative, where face detector mistakes can result in tracker failure. According to the test and observation, with the improvement of face label retention, our tracker can often re-track the same face with the same label.

Also, the improvement step helps with situations where the face detection produces a false positive result. False positives happen often in the face tracking process, where the face detector detects other non-face objects (such as hands or a patch of video background) as a face. False positive face detection result might disorder the face label assignment process. In the tracking process of our multi-face tracker, the error rate is reduced after adding the improvement of face label retention. The system has a higher error tolerance rate.

In summary, face label retention allows the feature maps remains across the frames, which increases our face tracker's robustness in situations containing camera cuts, face occlusion, false positive face detection results, false negative face detection results, e.g. due to faces at the image boundary or faces shown in profile.

Face label retention comparison

We select 3 frames as the maximum number of frames in face label retention in our tracking system but another number of frames could have been used. In Table 3.3, we present a comparison of face label retention with 2 frames, 5 frames and 10 frames. In order to control only a single variable, we use the Network 2 with position check as the tracking classifier in this experiment. We also use the method MOTA [9] to evaluate the tracking results with different face label retention.

From the evaluation parameter (identification switch) IDS_t, we can see that the label switch IDS_t is lowest when we set the face label retention to 3 framesin this video. In our multi-face tracking system, we use 3 frames for face label retention. The evaluation results shown in Chapter 4 are all with 3 frames as face label retention.

Method	MOTA	FP_t	$MISS_t$	MR	IDS_t
0 frame retention	97.70%	1	92	2.18%	16
2 frame retention	97.77%	1	92	2.18%	4
3 frame retention	97.78%	1	92	2.18%	**2**
5 frame retention	97.76%	1	92	2.18%	7
10 frame retention	97.76%	1	92	2.18%	8

Table 3.3: Different face label retention evaluations

Different face label retention comparison based on evaluation parameter MOTA [9] and the test video "Frontal" [55]. The multi-face tracking system runs with the Network 2.

3.7 Training of tracking classifier

This section discusses the training of the tracking classifier networks. The training data are feature maps instead of images as common for CNNs in computer vision and hence there are some preparation steps necessary.

Normalization

One problem is that the feature map values are very small after we extract the map from the fully connected layer of the F-Net, e.g., in a range from -0.000812 to 0.0000453. While, there are also some values that are bigger than 1. In order to feed these extracted feature maps into the tracking network, we normalize them into the range from 0 to 1. Then we store them into a one channel image. We store the feature map image in the format of CV_32FC1 image (storage type is float and one channel grayscale image).

Training preparation

After the normalization step, the feature map data needs to be prepared for training the tracking network, as well as for testing when the system is trained. In our implementation process we use the library Caffe for all the network training and calculation. Caffe can only read the data in LEVELDB, LMDB and HDF5 formats. For training, we have to store all the image data into one of these formats above.

In order to reduce the loss of the data in these feature map images, we select Bitmap Image File (BMP) storage format to export all the images to hard disk. Before transfer the feature map images into BMP, in order to reduce the loss because of the small numbers, we normalize all the image pixel value from [0, 1] to [0, 255] with floating point style. In this way all the floating values in the BMP images are between [0, 255]. Then we merge all the image pairs into a two-channel image, which are the data we use for training the tracking networks. Before we merge the image pairs into one two-channel image, we normalize the floating point image pixel from [0, 255] to [0, 1]. We check each pixel value with the original feature map output value to make sure that the feature maps are the same after two normalization steps and storage in BMP format. Then we transfer all the two-channel images into HDF5 format to train the models based on the CAFFE library.

3.8 Summary

In this chapter we have presented our framework of a face detection and tracking system. One significant advantage of this face tracker is that the computation complexity is low and it runs in real time. Our face tracker is also designed to deal with incorrect face detection, faces that leave and reappear after in several frames, or faces which are covered by other objects for several frames. We will show the runtime efficiency results in Chapter 4.

In Chapter 4, we will show the details of the implementation process, which includes the training data sets of both, face detector and tracking networks, face detection accuracy tests, multi-face tracking evaluation criterion, tracking network visualization, the runtime of the whole system with different numbers of faces being tracked, comparison results of three different tracking networks and accuracy comparison with other published face tracker.

Chapter 4

Experiments and Evaluations

In this chapter, we present the details of the implementation of our work. In Section 4.1, we discuss the face detection implementation which includes the training and evaluation data set for detection, pre-pr ocessing, training procedure and evaluation. In Section 4.2, we introduce the tracking network training data set and feature map visualization. Section 4.3 defines the multi-face tracker evaluation criterion and a commonly used testing data set. Section 4.4 presents the evaluation results of our multi-face tracker with different selections of the tracking network. A comparison of our multi-face tracking system with other face trackers in terms of accuracy, restrictions and runtime cost is given. Then we conclude with a discussion in Section 4.5.

4.1 Face detection implementation

Since the face detector we deploy in our system works on both face detection and face landmarks, we need a data set containing ground truth annotations for faces including 5 landmarks. We use to WIDER FACE data set [82] to train this face detector. For training, there are some important steps. The training result is tested on the FDDB face detection benchmark [34], and the evaluation results are shown for continuous and discrete scores. Some example results from the FDDB data set are also presented in this section.

Training data set

WIDER FACE [82] contains 32,203 images and has labels for 393,703 faces with a high degree of variability in scale, pose and occlusion as depicted in the sample images. The WIDER FACE data set is organized based on 61 event classes. All the images of WIDER FACE are stored into 61 folders thus making it a well-organized data set. The various faces in the data set are acquired in the "wild".

Pre-processing

In order to get an accurate training result, more training datas are needed. There are several steps for data augmentation (increase the amount of training data) and pre-processing.

1 Positive sample generation

   In training neural networks such as for face detection, the standard positive samples are the annotated faces of the data sets. According to the face position coordinates listed in the data set ground truth document, we are able to crop all of the valid faces as positive training data, as well as the basic face image materials for positive augmentation.

2 Negative sample generation

The negative training data for the neural network are those cropped areas whose maximum IoU with the ground truth are less than 0.5, we select the same threshold Jain and Miller used in their paper [34]. We crop examples of non-face windows. The non-face windows either show no face or they show a partial face. Those regions with some overlap with the ground truth are more likely to be hard negatives [24]. To some degree, these hard negative samples have contributed to increase the face detection accuracy, specifically when there are boundary issues [75], i.e., how to classify the window with half face. It tends to give more false alarms during the training process but results in better boundary decisions during testing.

3 Balancing positive and negative sample ratio

During training, an imbalanced ratio of positive and negative samples will result in over-fitting and other adverse influence despite a large amount of training data. We will fix the ratio of negative samples to positive samples to 3:1 as Wan et al. [75], i.e., the number of negative samples is three times the number of positive samples in each training data set.

4 We use several data augmentation methods: vertical flip, color distortion, random rotation.

Vertical flip is to turn over the image left-right symmetrically. It doubles the number of training images.

There are many ways to implement color distortion, we use a sepia filter to change the image to achieve data augmentation. A sepia filter holds a kernel to map the image and change the image color. The equation and kernel we use are shown in Equations 4.1 and 4.2.

$$dst(I) = kernel \cdot src(I), \tag{4.1}$$

where src is the input array, dst is the output array of the same size and depth as

src and the kernel is:

$$\begin{bmatrix} 0.272 & 0.534 & 0.131 \\ 0.349 & 0.686 & 0.168 \\ 0.393 & 0.769 & 0.189 \end{bmatrix} \tag{4.2}$$

Random 2D rotation follows the calculation of Equation 4.3 and 4.4.

$$\begin{bmatrix} a & b & (1-a) \cdot center_x - b \cdot center_y \\ -b & a & b \cdot center_x + (1-a) \cdot center_y \end{bmatrix} \tag{4.3}$$

where:

$$a = scale \cdot \cos\alpha, a = scale \cdot \sin\alpha. \tag{4.4}$$

where center is the center of the rotation in the source image, angle is the rotation angle in degrees and scale is the isotropic scale factor. We use 90 degrees and -90 degrees in our thesis.

5 Zero-centering

After zero-centering, applying randomly initialized filters makes the output fields balanced around zero. The ReLU layer then will get rid of negative values because negative values indicate that the network layer is not responding well to the inputs.

6 Normalization

Another important data pre-processing step before training neural networks is data normalization. Data normalization is to map data into a limited range, e.g., [0, 1] or [-1, 1], or a narrower range. There are several benefits to data normalization: the input data units may not be the same and a data range may be particularly large, resulting in a slow convergence of the neural network and a significantly longer time for training. Data normalization helps to reduce computation time; as the range of the activation function of the neural network output layer is limited, it is necessary to map the target data of the network training to the range of the

activation function; if the activation function is an S-shape function, whose value range is limited to [0, 1], the output of training data must be normalized to the range [0, 1].

7 We extract the training images that the trained detector cannot classify correctly to re-train the detector. This step helps to correct the detector's classification mistakes and improves the training result. After the images extracted, we mix them with some images from the data set that we have not yet used for training into a new training data to train the network again. This step could also be called fine-tune. During the training process, the batch size for P_Net, R_Net, O_Net and F_Net are 128, 128, 64 and 64, respectively, which are decided by the computer property (e.g. memory limits of the GPU).

Testing on FDDB benchmark

FDDB is a face detection data set and benchmark [34], which contains a total of 5,171 faces. All of which are annotated with an elliptical shape in 2,845 images (with a size of no more than 0.25 megapixels), with a wide range of detection difficulties including occlusions, difficult poses, low resolution and out-of-focus faces.

FDDB is a popular benchmark for current face detection evaluation as the authors used a standardized algorithm for automatic ROC curves construction [38] according to the detection results. This algorithm has two ways to evaluate detection results. One is by a discrete score, and the second is by a continuous score. In the ROC curve, for the discrete scores, the detection is considered to be positive if the Intersection-over-Union (IoU) ratio of detection and annotation areas exceeds 0.5, while the continuous score is the average IoU and reflects the quality of the face localization. Figure 4.1 shows an example where the MTCNN face detector misses a face in the in FDDB benchmark and Figure 4.2 shows an example in which the detector successfully finds a face that is not marked in FDDB benchmark. Figure 4.3 and Figure 4.4 show our face detection evaluation curves for the discrete and continuous scores, respectively. Discrete and continuous scores are all gotten based on the FDDB benchmark test data set. Discrete and continuous scores

are defined as Formula 4.5 and 4.6.

$$Discretescore(DS) : y_i = \delta(IoU(d_i, v_i) > 0.5), \tag{4.5}$$

$$Continuousscore(CS) : y_i = IoU(d_i, v_i), \tag{4.6}$$

Where d_i and v_i denote the i th detection and the corresponding matching node in the matching. IoU is shown in glossy of terms.

As a comparison, we put results of the original MTCNN detector (as reported by Zhang et al. in [83]) in each diagram. Even though our training has been successful, we do not train this detector quite as well as the original authors do. Both, our discrete and continuous scores (FDDB benchmark evaluation curves) are lower than the author's results. Our discrete score is about 92%, while the continuous score is about 69%. However, the authors' results are 95% for discrete score, and 71% for continuous score. In these curves, the x axis is the false positives, which corresponds to all the face detections. The y axis stands for the true positive rate, which means the ratio of correctly detected faces over all the detected faces.

From the recall curves (both discrete and concrete scores) and some samples of our face detection results, we can see that the MTCNN face detector is a very accurate face detector after good training which is going to contribute to our face tracker performance.

4.2 Tracker implementation

Our multi-face tracker is implemented based on neural networks. The tracking classifier's task is to determine whether the feature maps for a face given from the last frame and the current frame belong to the same person or not. For training, we use the Boston head tracking data set [41].

Boston head tracking data set

This data set contains more than 100 videos, where each one only contains one person. Each video has 198 frames including the person's head moving along the x-axis, the

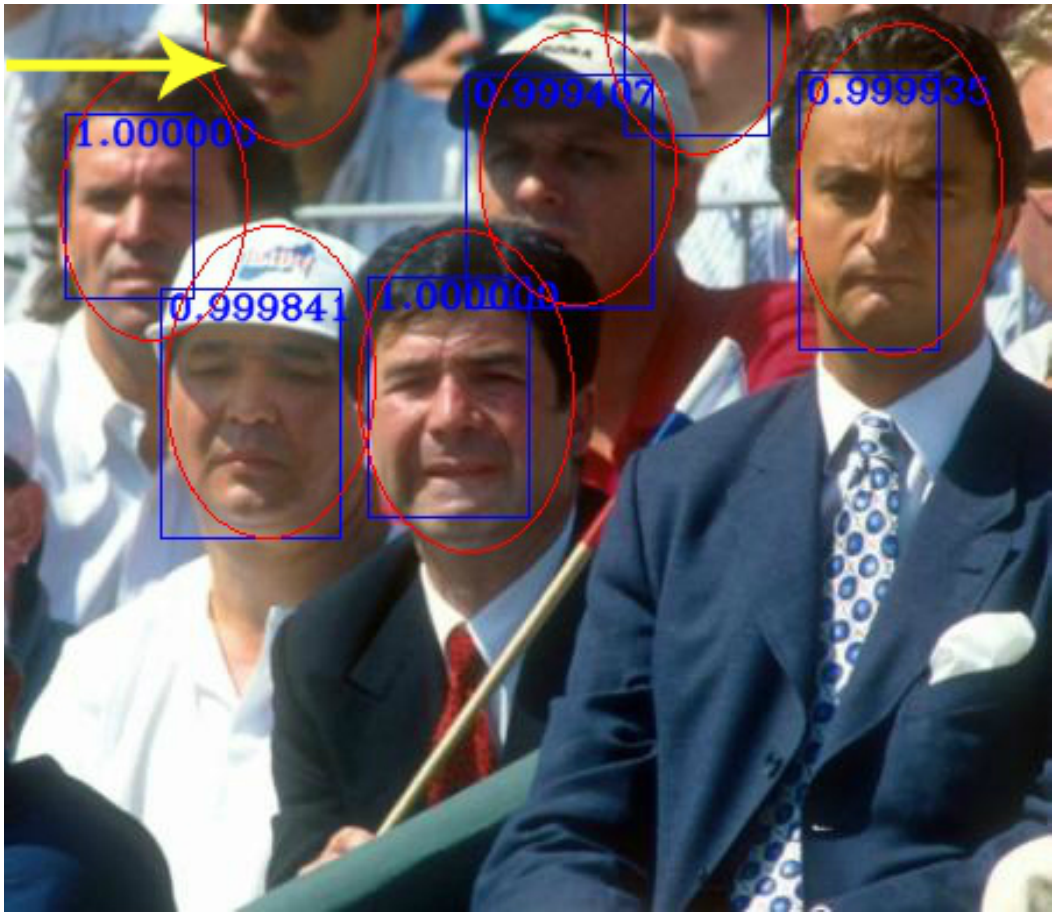

Figure 4.1: Face detection sample 1

An example where there is a face our implementation of the MTCNN detector fails to detect. The red ellipses on each face are the ground truth provided by the FDDB data set [34], the blue rectangles are the detected results of our detector and the number of each rectangle is the confidence score of the face classifier.

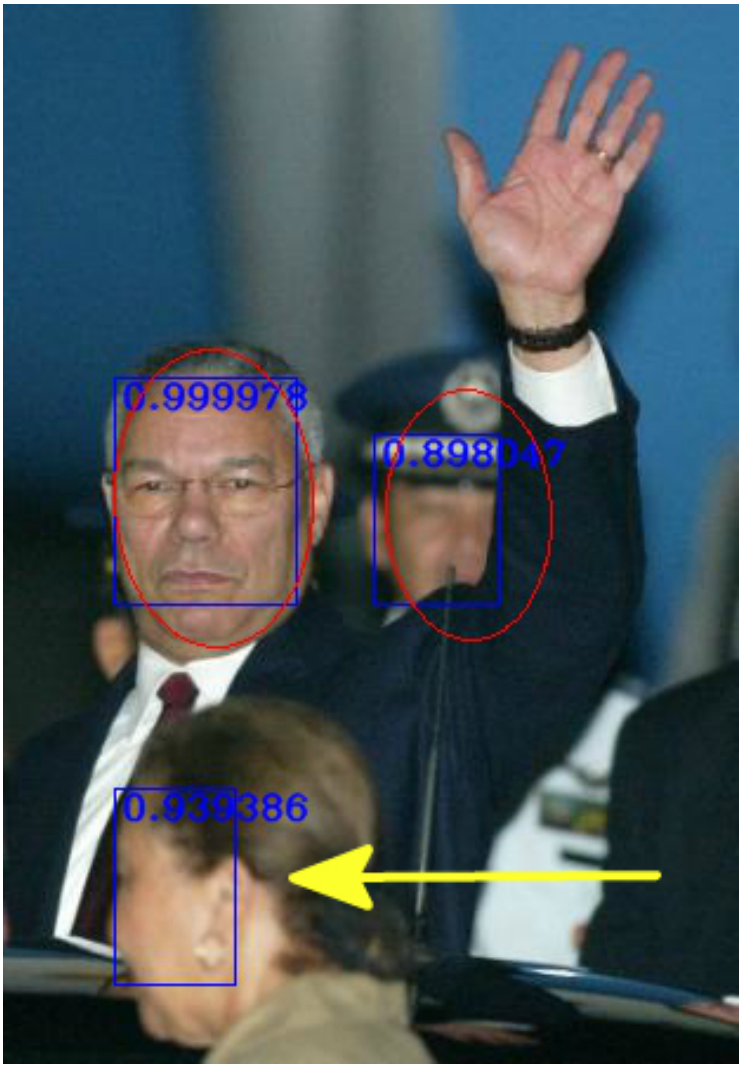

Figure 4.2: Face detection sample 2

An example where there is a face that our implementation of the MTCNN detector locates but the FDDB ground truth [34] does not include.

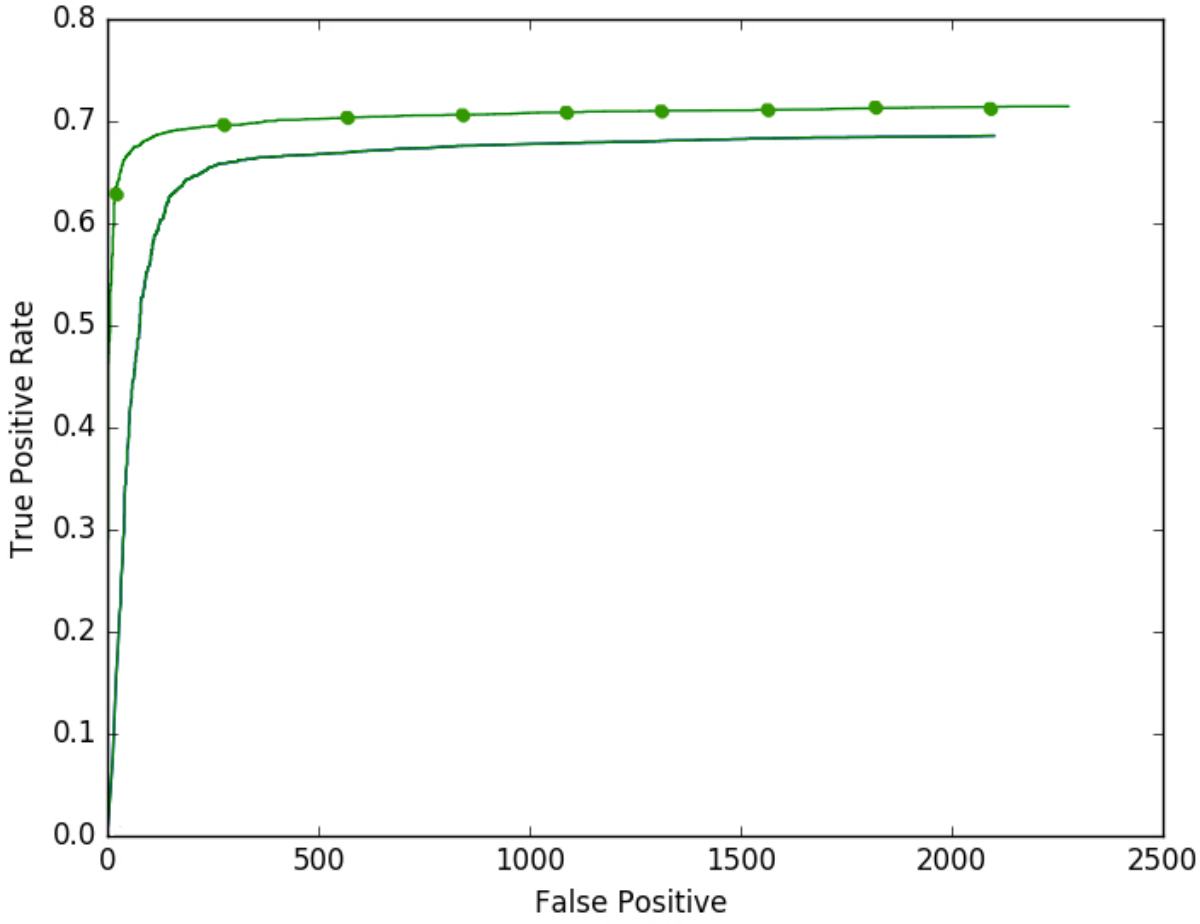

Figure 4.3: Face detection testing result (continuous score) on the FDDB benchmark . This shows the recall curve (continuous score). The line with nodes is the result by the authors of MTCNN [83] and the smooth line is our training result. The paper achieved a score of 0.714525, while ours is 0.685723, the numbers are the curves' end values, which is the curve maximum score. At the beginning of the x axis, the paper authors' curve slope is much bigger than ours, it exposes that authors' result has a higher probability to achieve a high accuracy even testing on several samples. However, our training result's probability is lower than theirs.

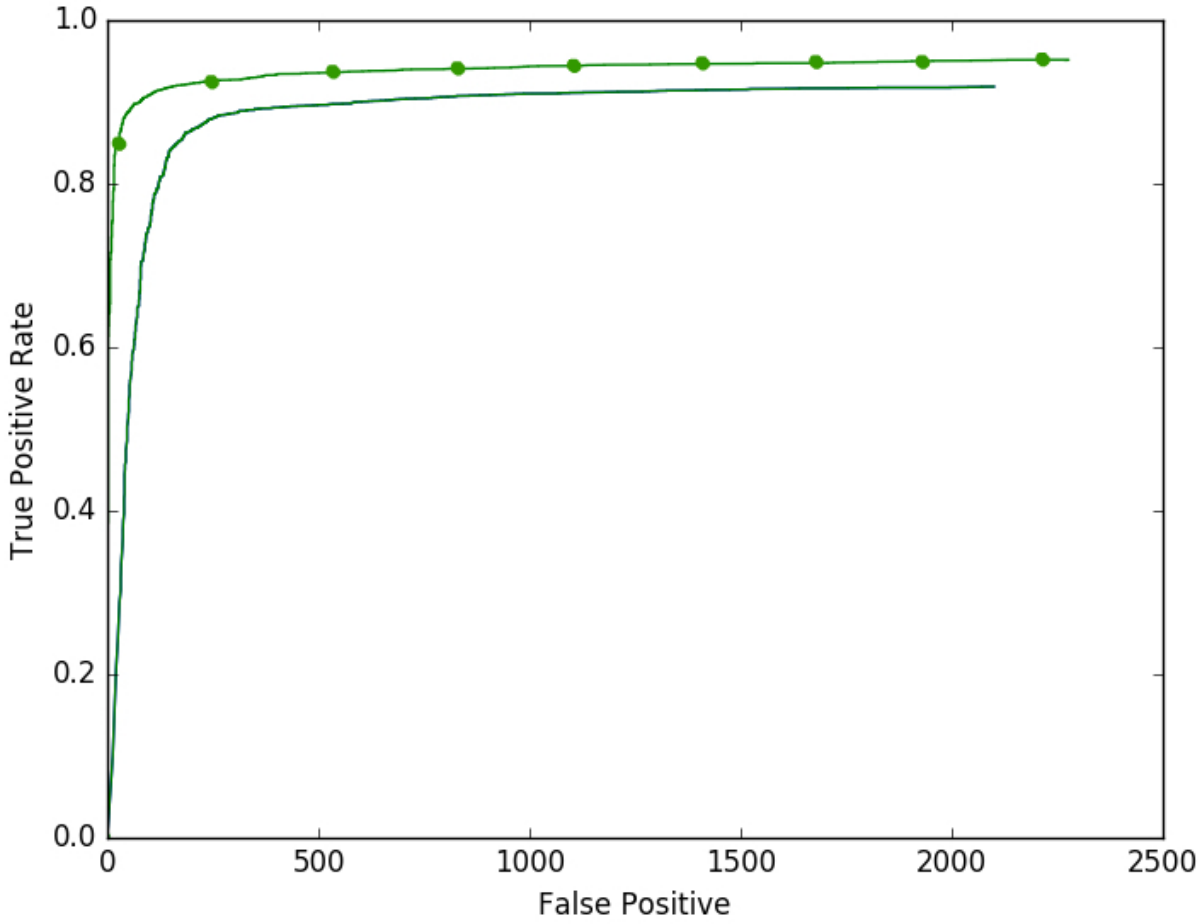

Figure 4.4: Face detection testing result (discrete score) on the FDDB benchmark. This shows the recall curve (discrete score). The line with nodes is the result by the authors of MTCNN [83] and the smooth line is our training result. The paper achieved a score of 0.951653, while ours is 0.918778.

y-axis and the z-axis. It also records the same person's head movement with changing illumination. This video data set provides enough training images for matching human faces with small changes between two neighboring frames. Sample examples of this data set are shown in Figure 4.5.

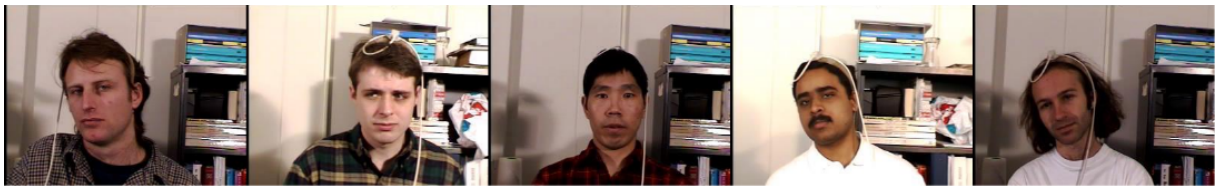

Figure 4.5: Example frames of the Boston head tracking data set [41].

Feature map visualization

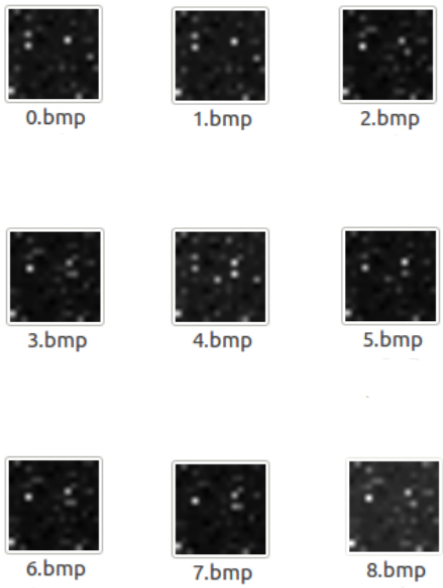

Figure 4.6: Visualization of feature maps corresponding to training faces from the Boston head tracking data set [41].

After cropping the faces from the video data set, we transfer the training data into feature maps, which is the raw data for training our tracking network. A visualization

of example feature maps is shown in Figure 4.6.

4.3 Evaluation criterion

For multi-face tracking evaluations, we select four videos which are used in many other evaluations of multi-face trackers as we discussed in Chapter 2: "Frontal", "Turning", "Fast" and "BBT0101". We use two popular evaluation methods for multi-face tracking in this section.

Frontal, Turning and Fast

This evaluation data set [55] consists of 3 videos recorded by a fixed camera. The first two videos "Frontal" and "Turning" involve four people moving around causing frequent face occlusions. In the "Frontal" video, these four people are shown mostly with their face frontal towards the camera, while, there are much more profile faces recorded on purpose in the video "Turning". In the video "Fast", there are only three people recorded with mainly very fast movements of frontal faces as well as profile faces.

The "Frontal", "Turning" and "Fast" videos contain 1277 frames, 1007 frames and 485 frames, respectively. Although it is not a big data set, it contains many complicated situations to test the performance of multi-face trackers. It is for now one of the most popular test data sets for evaluating multi-face tracking performance.

Figure 4.7 shows a frame sample of our multi-face tracker test results on the video "Frontal". We deploy the ground truth of Wu et al. [77] and the ground truth statistics of "Frontal" and "Turning"of this data set are presented in Table 4.1. We like to acknowledge the support of B. Wu in providing the ground truth annotations to us.

Video Ground Truth	Time(s)	Frames	Persons	Trajectories	Faces
Frontal	51	1277	4	43	4267
Turning	40	1007	4	50	2799

Table 4.1: Test video "Frontal" and "Turning" ground truth statistics used by Wu et al. [77]. A face tracklet is a section of a 2D trajectory of faces over time in the video.

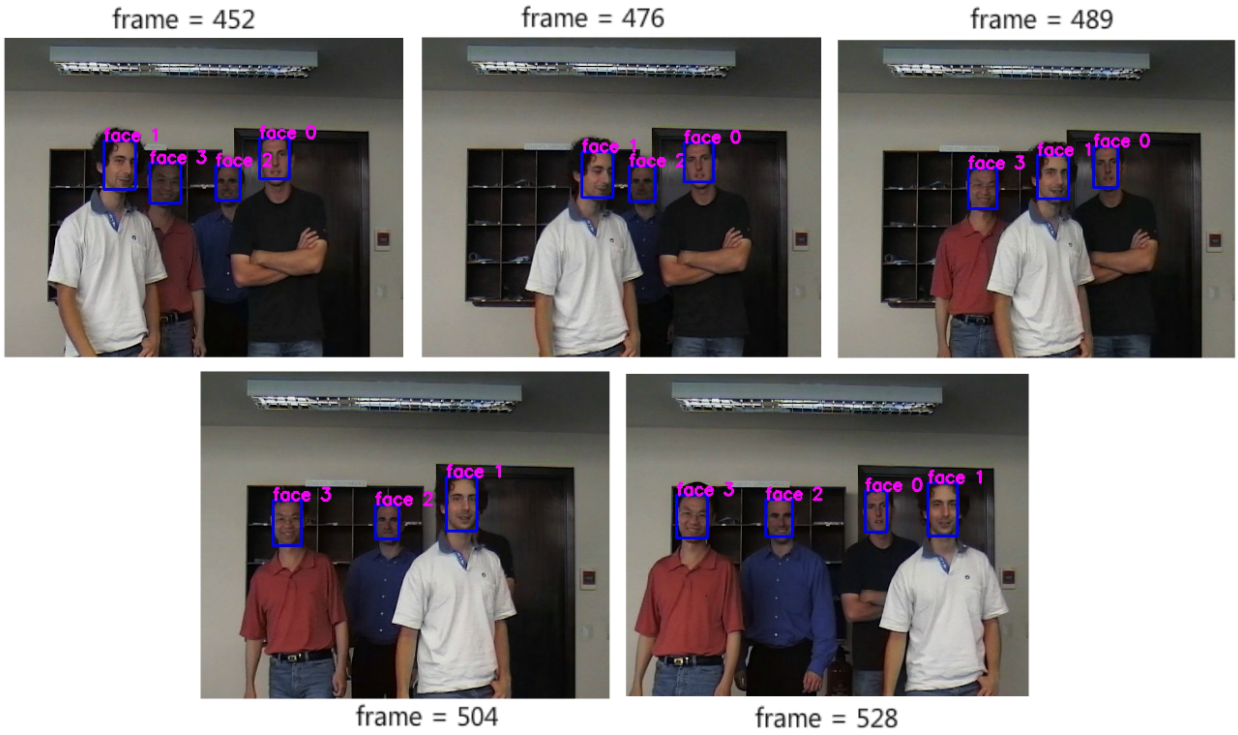

Figure 4.7: Output frames example

Some example frames from our multi-face tracker result on the video "Frontal" [55].

This result is obtained by our multi-face tracker with Network 2.

Because of the good performance of our face detector, we track more faces than the ground truth in the Table 4.1. After carefully check, the true positives of our detection results are shown in the Table 4.2. There are some faces missing in the ground truth used by Wu et al. [77], some examples are shown in the Figure 4.8. The green rectangles are the ground truth used by Wu et al. [77], and the blue rectangles are our results.

From the Table 4.2 and Figure 4.8, we can see the ground truth still has limitations since there are some faces missing. The video "Fast" is one of video of this data set [55], however, there is no ground truth for video "Fast".

Video "Ground Truth"	Time(s)	Frames	Persons	Faces
Frontal	51	1277	4	4175
Turning	40	1007	4	3245
Fast	19	485	3	1183

Table 4.2: Our true positive detection result statistic of test video "Frontal", "Turning" and "Fast".

The Big Bang Theory

Another very commonly used video for multi-face tracking evaluation is "The Big Bang Theory" season 1 episode 1 (BBT0101). This video is more challenging due to the fact that there are many camera cuts and scene switches in the video. We deploy the ground truth used by Tapaswi et al. in the paper [70]. We present the statistics of this video in Table 4.3. Because our system face detector could detect more faces than this ground truth [70], so we put our true positive face detection results in the Table 4.4 as a comparison. Because of the video versions are different, the frames are a little various. In this ground truth [70], there are some faces of crowds missing. But our systems tracks all of these clear faces in this show. In our system tracking process, we track 37 people in total appeared in this video. That results in we have more face in the Table 4.4.

In our true positive result statistic, we do not show the tracklets as the ground truth shows. Because we do not have clear definition of how to define a ground truth trajectory in a video. The frame numbers are a little various since the video is not provided with

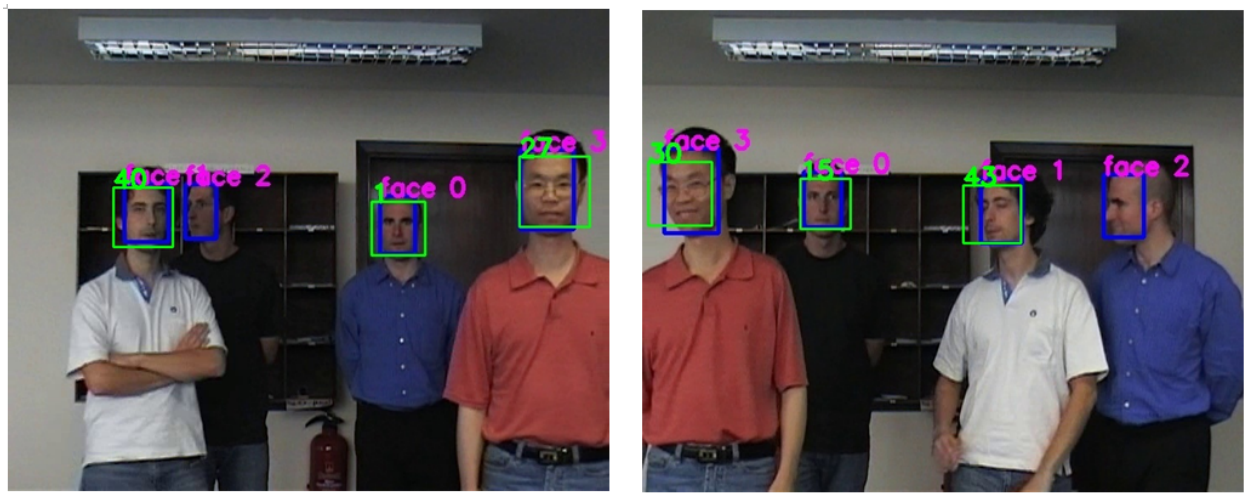

Figure 4.8: Our results compared with ground truth.
The green rectangle is the ground truth used by Wu et al. [77]. The blue rectangles are our tracking results. We can see there are some faces in this ground truth missing. This result is obtained from video "Turning" [55] by our multi-face tracker with Network 2.

Video Ground Truth	Time(s)	Frames	Persons	Trajectories	Faces
BBT0101	1373	32990	8	622	32641

Table 4.3: Test video "BBT0101" ground truth statistics made by Tapaswi et al. [70].

Video Ground Truth	Time(s)	Frames	Persons	Faces
BBT0101	1373	32977	37	47596

Table 4.4: Our true positive detection result statistic of veideo "BBT0101".

the ground truth for copyright issues.

Evaluation Metrics 1

Li at al. [47] defined an evaluation metrics for their multi-object tracking approach that has been widely adopted since. The definition of the evaluation metrics is shown in Table 4.5.

Evaluation parameter	Definition
Recall	Correctly matched objects total / ground truth objects (Frame-based).
Precision	Correctly matched objects total/ output objects (Frame-based).
GT	Number of ground truth trajectories.
MT%	Mostly tracked: Percentage of GT trajectories which are covered by tracker output for more than 80% in length.
ML%	Mostly lost: Percentage of GT trajectories which are covered by tracker output for less than 20% in length.
Frag	Fragments: The total number of times that a ground truth trajectory. is interrupted in the tracking result. The smaller the better.
PT%	Partially tracked: 1.0 - MT - ML.
IDS	ID switches: The total number of times that a tracked trajectory. changes its matched GT identity. The smaller the better.

Table 4.5: Evaluation metrics defined by Li et al. [47].

We calculate the number of fragments and ID switches following the rules introduced in the paper [47]. When we calculate the number of fragment, we check how many our tracklets covering one ground truth trajectory. The number of our covering tracklets minus one is the fragment number of this ground truth trajectory. The sum of fragment numbers of all the ground truth trajectories is the result of this video. When we calculate the ID switches, we check how many ground truth trajectories are covered by one of our tracker tracklets. The number of ground truth trajectories minus one is the number of ID switches of this tracklet. The sum is the IDS number of this video.

Evaluation Metrics 2

There is another popular evaluation criterion introduced by Bernardin and Stiefelhagen [9] for multiple object tracking. In their CLEAR MOT evaluation, one important parameter is called Multiple Object Tracker Accuracy (MOTA). MOTA is evaluating the ability of a tracker to consistently label faces over time. The definition is:

$$MOTA = 1 - \left(\frac{\sum_t (FP_t) + MISS_t + IDS_t}{\sum_t (GT_t)} \right). \tag{4.7}$$

Each parameter definition is shown in the Table 4.6. Please note that the parameter GT_t of Benardin and Stiefelhagen [9] is different from the GT of Li et al. [47] in Table 4.6. The GT_t defined by Bernardini and Stiefelhagen [9] stands for ground truth number of faces in the frame t. Each evaluation result shown in Section 4.4 will declare which evaluation metrics it is using.

Evaluation parameter	Definition
FP_t	Number of false positives at time t (the smaller the better)
$MISS_t$	Number of misses at time t(the smaller the better)
IDS_t	Identity switches at time t (the smaller the better)
FPR	False positive rates (the smaller the better)
MR	Miss rate (the smaller the better)
GT_t	Number of ground truth faces in the frame t

Table 4.6: Evaluation metrics definition provided by [9]

Trajectory and Tracklet Mapping Procedure

The metrics above require that the a mapping between groundtruth trajectory and tracker tracklets are found. As pointed out by Stiefelhagen and Bernardin [9], this is a non-trivial task if many tracklets and trajectories are involved and errors occur. Both of the metrics [47, 9] discussed above make use of the Hungarian algorithm to find an optimal assignment based on the distances between trajectories and tracklets. The mapping procedure of Bernardin and Stiefelhagen [9] has five steps:

1 Every mapping in the previous frame of tracklet and groundtruth is kept in the current frame if the distance between trajectory and tracklet is below a distance threshold in the current frame. (In the first frame no previous mapping exists and hence all mappings will have to be found in the following steps);

2 Minimize the total distance of the mappings between all remaining trajectories and tracklets (i.e., the ones not mapped in step 1) with optimal assignment solved by the Hungarian algorithm. If a new mapping is made which contradicts the mapping in the previous frame, then replace the previous mapping with the new one and count this as a mismatch (IDS);

3 Calculate each mapping distance after step 2;

4 Count the number of tracking results which are not matched as the false positive (FP), and ount the number of ground truth objects which are not matched as the misses (MISS);

5 Repeat from step 1 on next frame with the current set of mappings. Note, that the first frame of the video does not have IDS.

However,there is a pre-set threshold T used to measure the distance between ground truth and tracking results [9]. Bernardin and Stiefelhagen argue that the distance threshold between groundtruth trajectories and tracklets for a valid mapping is application dependent [9]. The comparison methods that we compare against do not state explicitly what threshold has been used. We can only speculate that they have followed Berbnardin and Stiefelhagen's recommendation of setting the distance threshold to 0 when using the metrics in 2D face tracking.

We feel that a given progress made face tracking a stricter threshold can be used. We use therefore IoU as it is a common practise in benchmarking face detection algorithm [34] and we do not seek an optimal assignment overall. We only count an individual assignment if it has the highest IoU of all possible assignments and expect a higher number of IDS, FP and number of missed faces. The steps of our mapping procedure are:

1 Check the overlap, if one rectangle from ground truth has an overlap with IoU bigger than 0.5 with one rectangle from tracking results, we put these two rectangles as a pair of the mapping on one frame;

2 Find all mappings in this frame;

3 If one rectangle from ground truth does not have an overlap IoU higher than 0.5 match from the tracking results, then there is one miss (MISS) counted.

4 If one rectangle from tracking results does not have an overlap IoU higher than 0.5 match in the ground truth, then there is one FP (false positive) counted.

5 Repeat from step 1 to get the whole mappings in this video;

6 Check how many ground truth trajectories are covered by one of our tracker tracklets. The number of ground truth trajectories minus one is the number of ID switches of this tracklet. The sum is the IDS number of this video.

Here we use the same threshold as used in the face detection evaluation benchmark FDDB [34]. Our mapping procedure is stricter. We use all the parameters we get from this mapping procedure and from the ground truth to calculate the MOTA value.

4.4 Evaluation results

In order to compare with other multi-face tracking results, we first present our tracking results based on the four evaluation videos and the two evaluation metrics discussed in Section 4.3. Then we will compare our runtime result with other trackers. Because each tracker uses different evaluation parameters, we will compare with each multi-tracker according to these parameters.

Our multi-face tracker evaluation results

We will start with the testing videos "Frontal and Turning" [55] and then the video "BBT0101" using the two evaluation matrics: [9, 47]. Our multi-face tracking results

with different tracking networks are shown in Table 4.7, 4.8, 4.9, 4.11, 4.12, 4.13 and 4.14. As a comparison, we also show the evaluation results with the cosine similarity method in Table 4.10 and Table 4.14. Because there is no ground truth for video "Fast", we put our our tracking results in the tables.

Video	Recall	Precision	PT	MT	Frag	IDS
"Frontal"	96.59%	99.98%	0	43	24	15
"Turning"	99.82%	100.00%	0	50	9	4
"Fast"	-	-	0	13	4	2
"BBT0101"	97.44%	96.98%	22	592	83	22

Table 4.7: Our tracker evaluation result with tracking Network 1 based on the evaluation metrics of [47].

Video	Recall	Precision	PT	MT	Frag	IDS
"Frontal"	96.59%	99.98%	0	43	23	13
"Turning"	99.82%	100.00%	0	50	8	4
"Fast"	-	-	0	13	4	2
"BBT0101"	97.44%	96.98%	23	593	78	22

Table 4.8: Our tracker evaluation result with tracking Network 2 based on the evaluation metrics of [47].

Video	Recall	Precision	PT	MT	Frag	IDS
"Frontal"	96.59%	99.98%	0	43	23	15
"Turning"	99.82%	100.00%	0	50	9	4
"Fast"	-	-	0	13	4	2
"BBT0101"	97.44%	96.98%	21	592	80	24

Table 4.9: Our tracker evaluation result with tracking Network 3 based on the evaluation metrics of [47].

Video	Recall	Precision	PT	MT	Frag	IDS
"Frontal"	96.59%	99.98%	0	43	23	15
"Turning"	99.82%	100.00%	0	50	9	4
"Fast"	-	-	0	13	4	2
"BBT0101"	97.44%	96.98%	22	593	82	24

Table 4.10: Our tracker evaluation result with cosine similarity based on the evaluation metrics of [47].

Video	MOTA	FP_t	$MISS_t$	FPR	MR	IDS_t
"Frontal"	97.70%	1	92	0	2.16%	15
"Turning"	99.68%	2	3	0.07%	0.11%	4
"Fast"	-	0	0	0	0	2
"BBT0101"	96.86%	143	861	0.44%	2.64%	22

Table 4.11: Our tracker evaluation result with tracking Network 1 based on the evaluation metrics of [9].

Video	MOTA	FP_t	$MISS_t$	FPR	MR	IDS_t
"Frontal"	97.75%	1	92	0	2.16%	13
"Turning"	99.68%	2	3	0.07%	0.11%	4
"Fast"	-	0	0	0	0	2
"BBT0101"	96.86%	143	861	0.44%	2.64%	22

Table 4.12: Our tracker evaluation result with tracking Network 2 based on the evaluation metrics of [9].

Video	MOTA	FP_t	$MISS_t$	FPR	MR	IDS_t
"Frontal"	97.70%	1	92	0	2.16%	15
"Turning"	99.68%	2	3	0.07%	0.11%	4
"Fast"	-	0	0	0	0	2
"BBT0101"	96.85%	143	861	0.44%	2.64%	24

Table 4.13: Our tracker evaluation result with tracking Network 3 based on the evaluation metrics of [9].

Video	MOTA	FP_t	$MISS_t$	FPR	MR	IDS_t
"Frontal"	97.70%	1	92	0	2.16%	15
"Turning"	99.68%	2	3	0.07%	0.11%	4
"Fast"	-	0	0	0	0	2
"BBT0101"	96.85%	143	861	0.44%	2.64%	24

Table 4.14: Our tracker evaluation result with cosine similarity based on the evaluation metrics of [9].

Comparison with other methods

We compare our multi-face tracker with six other up-to-date multi-face trackers [64, 77, 78, 66, 19, 84] according to the previously discussed evaluation metrics and videos. When compared with each algorithm, we just calculate the parameter values they present in the published papers.

Our thesis research presents a multi-face tracker with labels running in real time. We also compare our tracker's runtime cost with the runtime of these trackers. However, some trackers [77, 78] cannot run in real time and do not provide their runtime. Another multi-face tracker paper [84] does not mention the runtime cost. We can only compare with the ones [43, 19, 64] that provide runtime cost. In addition, we also compare the runtime cost with another multi-face tracker [66]. Because this tracker [66] presents runtime cost, but does not present the performance tested by standard evaluation methods, we only compare runtime cost for this tracker. The runtime cost comparison of our research with these four algorithms is shown in the Table 4.15.

Table 4.16 shows the evaluation comparison with the multi-face tracker of Roth et al. [64] and of Du and Chellappa [19] based on the evaluation method used by Bernardini and Stiefelhagen [9] on the video "BBT0101". We present our evaluation results with the three different tracking classifiers: Network 1, Network 2 and Network 3. Because these papers use different ground truth for evaluating, so the GT values are various. Roth et al. [64] mention that the ground truth they use only gives result per 5 frames, so the GT value is much smaller than ours. Du and Chellappa [19] do not mention about the

Method	Runtime cost(average)	Computing environment
Le et al. [43]	250-333ms/frame	Intel(R) Core(TM) i7-4930K CPU @ 3.40GHz
Du and Chellappa [19]	178.3ms/frame	eight-core 3.4 GHz CPU
Roth et al. [64]	1865.2ms/frame	AMD Phenom TM II X4 970 quad core processor
Shi et al. [66]	50-150ms/frame	-
ours (Network 1)	34.8ms/frame	Intel Xeon(R) CPU E5-1607 @ 3.10GHz GeForce GTX 1080
ours (Network 2)	35.41ms/frame	Intel Xeon(R) CPU E5-1607 @ 3.10GHz GeForce GTX 1080
ours (Network 3)	35.3ms/frame	Intel Xeon(R) CPU E5-1607 @ 3.10GHz GeForce GTX 1080
ours(Cosine similarity)	**30.50ms/frame**	Intel Xeon(R) CPU E5-1607 @ 3.10GHz GeForce GTX 1080

Table 4.15: Runtime cost comparison with algorithms in [43, 19, 64, 66].

ground truth data set they deploy. We deploy the groud truth mark each frame results.

From the Table 4.16 results, we can see that even we have higher MOTA results and lower FPR (false positive rate). The IDs is of Network 1 and Network 2 combination tracking system are smaller than these two algorithms [64, 19]. The FPR, Precision and Recall of our four methods are the same because we use the same detector. The FPR, Precision and Recall all are only related to the detection results.

Methods	GT	MOTA	Recall	Precision	FPR	IDS
Roth et al. [64]	1495	81.95%	-	87.29%	5.02%	33
Du and Chellappa [19]	2207	75.3%	81.7%	95.0%	8.11%	24
Ours(Network 1)	32641	96.86%	97.44%	96.98%	**0.44%**	**22**
Ours(Network 2)	32641	96.86%	97.44%	96.98%	**0.44%**	**22**
Ours(Network 3)	32641	96.85%	97.44%	96.98%	**0.44%**	24
Ours(Cosine similarity)	32641	96.85%	97.44%	96.98%	**0.44%**	24

Table 4.16: Comparison with algorithms in [64] and [19] on videos of "BBT0101" based on the evaluation method of Bernardini and Stiefelhagen [9].

Table 4.17 and Table 4.18 show the evaluation comparison with the tracking algorithms [43, 77, 64, 78] based on the evaluation method by Li et al. [47] on the video "Frontal" and "Turning". We present our evaluation results with the three different tracking networks: Network 1, Network 2 and Network 3.

From Table 4.17 and 4.18, we can see that our tracker can track all the ground truth tracklets with mostly tracking results. However, our trackers create higher number of fragments and ID swithes.

Table 4.19 shows the evaluation comparison with the tracking algorithms [77, 64, 78] based on the evaluation method used in paper [47] on the video "BBT0101". We present our evaluation results with the three different tracking classifiers: Network 1, Network 2 and Network 3. These results are obtained for the video "BBT0101". From this table, we can see that the MT (number of most tracked tracklets) of our results are much higher than other algorithms. Even though we have higher IDs. But our tracker's advantages are very obvious.

Methods	PT	MT	Frag	IDS
Le et al. [43]	23	6	16	**0**
Wu et al. [77]	20	5	**22**	3
Wu et al. [78]	15	5	25	10
Roth et al. [64]	11	4	24	13
Ours(Network 1)	0	**43**	24	15
Ours (Network 2)	0	**43**	23	13
Ours (Network 3)	0	**43**	23	15
Ours(cosine similarity)	0	**43**	23	15

Table 4.17: Evaluation comparison with algorithm in [43, 64, 77, 78]
Test video is "Frontal" based on the evaluation method used in [47].

Methods	PT	MT	Frag	IDS
Le et al. [43]	15	4	9	**0**
Wu et al. [77]	5	3	**6**	1
Wu et al. [78]	5	4	8	5
Roth et al. [64]	5	2	8	4
Ours(Network 1)	0	**50**	9	4
Ours (Network 2)	0	**50**	8	4
Ours (Network 3)	0	**50**	9	4
Ours(cosine similarity)	0	**50**	9	4

Table 4.18: Evaluation comparison with algorithm in [43, 64, 77, 78]
Test video is "Turning" based on the evaluation method used in [47].

Methods	PT	MT	Frag	IDS
Wu et al. [77]	75	69	80	7
Wu et al. [78]	79	68	83	**4**
Roth et al. [64]	72	68	81	10
Ours(Network 1)	22	592	89	26
Ours(Network 2)	23	**593**	**79**	24
Ours(Network 3)	21	592	85	25
Ours(Cosine similarity)	22	**593**	89	25

Table 4.19: Evaluation comparison with algorithms in [77, 78, 43] on video "BBT0101", and based on the evaluation method used in [47].

Figure 4.9: (©2017 IEEE) Comparison frames example 1
Example frames of the video "Frontal". The first line is the results produced by paper [19], and the results of the second line are obtained by our multi-face tracker with Network 2

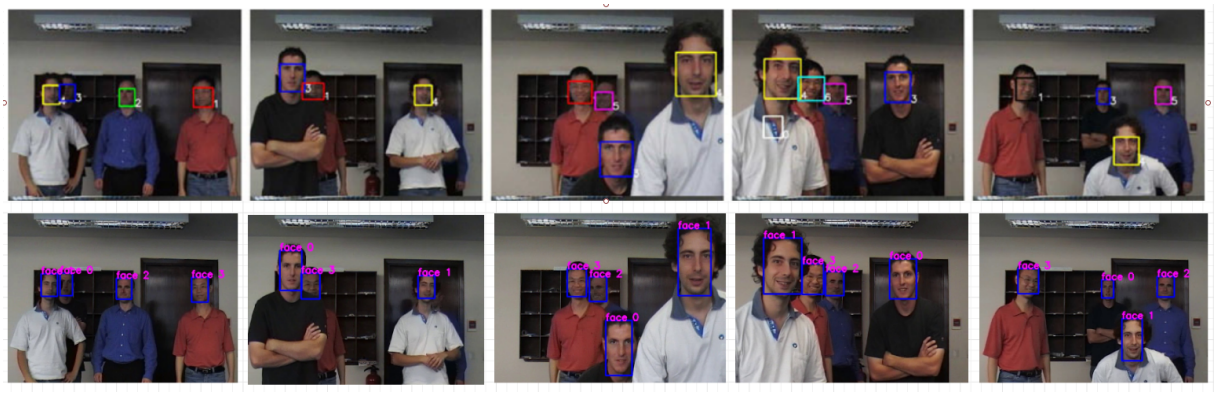

Figure 4.10: (©2017 IEEE) Comparison frames example 2
Example frames of the video "Turning". The first line shows the results produced by paper [19], and the second line shows the results by our multi-face tracker with Network 2

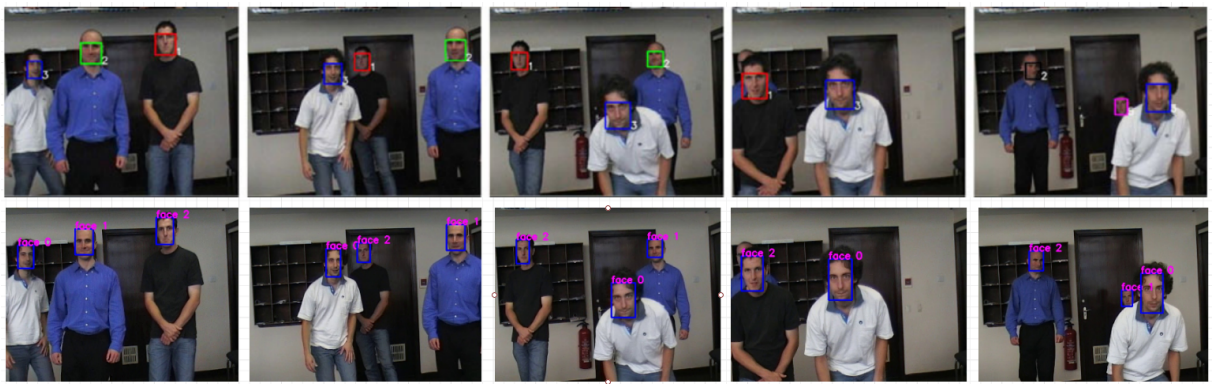

Figure 4.11: (©2017 IEEE) Comparison frames example 3
Example frames of the video "Fast". The first line are the results reproduced from paper [19], and the second line shows the result of our multi-face tracker with Network 2

We also compare sample frames with the results shown in [19]. From the sample frames in Figure 4.9, we can find that in the video "Frontal", our results are all correct. The first row of Figure 4.9 are the results from [19], and the second row is our tracking results. In our multi-face tracking results, the labels of the man in white, black, blue and red are "face 1", "face 2", "face 0" and "face 3" respectively. In the sample frames period, no Identification Switching (IDS) occurs. However, the results of [19] show identification switching (IDS) on the man in blue.

The example frames shown in Figure 4.10 are from the video "Turning". In our multi-face tracking results, the labels of the man in white, black, blue and red are "face 1", "face 0", "face 2" and "face 3", respectively. In these example frames, our tracking results are all correct, even though there are occlusions of faces. However, there is some identification switching (IDS) when there are face occlusions. The fourth picture of the first row shows a false positive, and the men in blue and red have switched their ID.

More example frames are shown in Figure 4.11. These frames are from the video "Fast". In our multi-face tracking results, the labels of the man in white, blue and black are "face 0", "face 1" and "face 2", respectively. The last frame face labels show that our tracker fails to track the men in blue and black. Similarly, the results of [19] show that the men in blue and black switched their original labels.

Runtime efficiency

Our multi-face tracker is designed for live video, runtime cost is an important element to consider in our implementation. Figure 4.12 shows our tracker runtime record on a test video. The average runtime cost for the multi-face tracker is 0.03480s, 0.03541s and 0.03530s per frame using Network 1, Network 2 and Network 3, respectively. The standard deviations of the runtime of these three versions are 0.005665s, 0.005115s and 0.006025s, respectively. This speed is sufficient for processing videos in real time. From the standard deviations, we can see the runtime of the tracking system with Network 2 is quite stable. As a comparison, we put the cosine similarity and only the face detector runtime results here. Cosine similarity is faster than our tracking networks. The numbers are shown in Table 4.20.

Tracking system type	Complete system runtime cost	Standard Deviation
Network 1	0.03480s	0.005665342
Network 2	0.03541s	0.00511463
Network 3	0.03530s	0.005940132
Cosine similarity	0.03050s	0.006024784
Face detection*	**0.02985s**	**0.004528475**

Table 4.20: Runtime cost and standard deviation comparison

The complete tracking system runtime cost and the standard deviation with three different combinations of detector and tracking classifier. As a comparison, Face detection* presents the runtime of only running face detector.

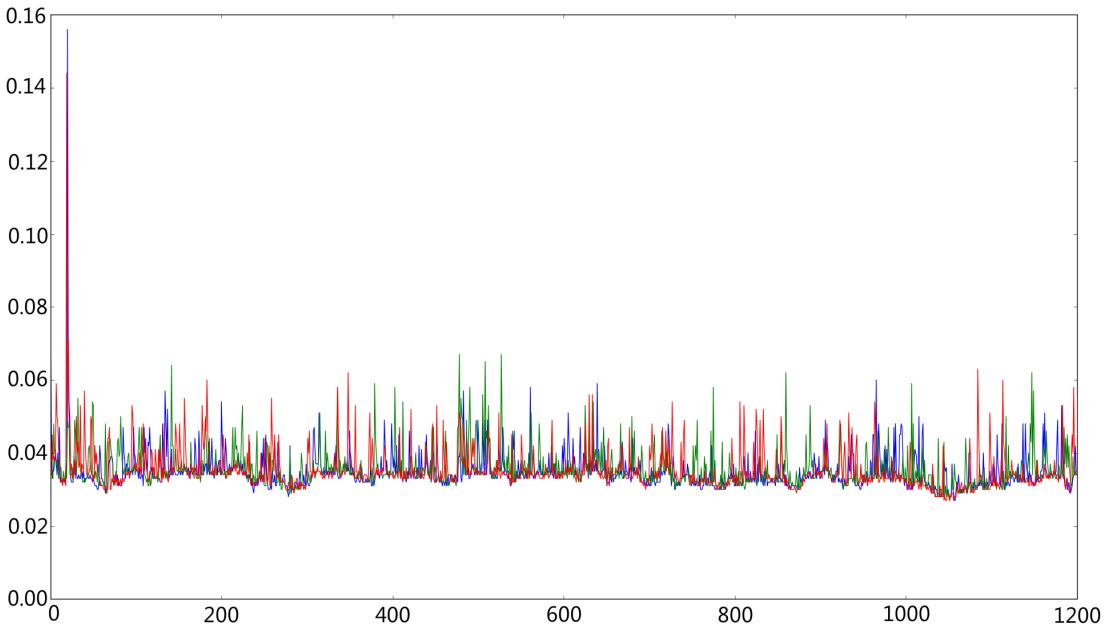

Figure 4.12: Runtime cost and standard deviation comparison

The runtime cost of our multi-face system with different tracking networks. The X axis is the frame number, while the Y axis is the time(s). The blue line is for Network 1, green line is for Network 2 and the red line stands for Network 3.

As previously introduced in Chapter 2, some multi-face tracker [66] runtime can be affected significantly by the number of faces in the video. A diagram in the paper by Shi et al. [66] shows that the runtime cost of their tracker increases linearly with respect to the number of faces being tracked. The runtime cost for tracking one face is around 50ms per frame, however, the runtime cost increases to around 300ms for tracking 9 faces.

In comparison, our multi-face tracker has a close to stable runtime cost for tracking a different number of faces. Figure 4.13 shows the correlation between our tracker's runtime cost and face number. In order to demonstrate the correlation between these two parameters, they are plotted on the same graph, where one is representing the number of faces detected in a frame, and the other is representing the runtime cost, both are plotted over the frame number. These two curves illustrate the changes in the number of faces have no significant effect on the time cost. However, if the face number in the video increases significantly, the runtime cost will also increase slightly. For example, in our experiment, if the face number goes from 2 to 39, the runtime cost will increase from approximately 0.028s to 0.078s. However, our multi-face tracker's runtime cost will not linearly increase with respect to the face number.

4.5 Discussion

In this chapter we have presented the results of different parts of our multi-face tracker and the comparison results with other multi-face trackers.

Face detector

In Section 4.1, it describes the detailed implementation of our face detector, the process of the pre-processing of training, and the training results. From the FDDB benchmark evaluation curve, we can see that we have trained a well-performing face detector. Specifically, this detector can detect a wide range of profile face, which directly contributes to our tracker's performance.

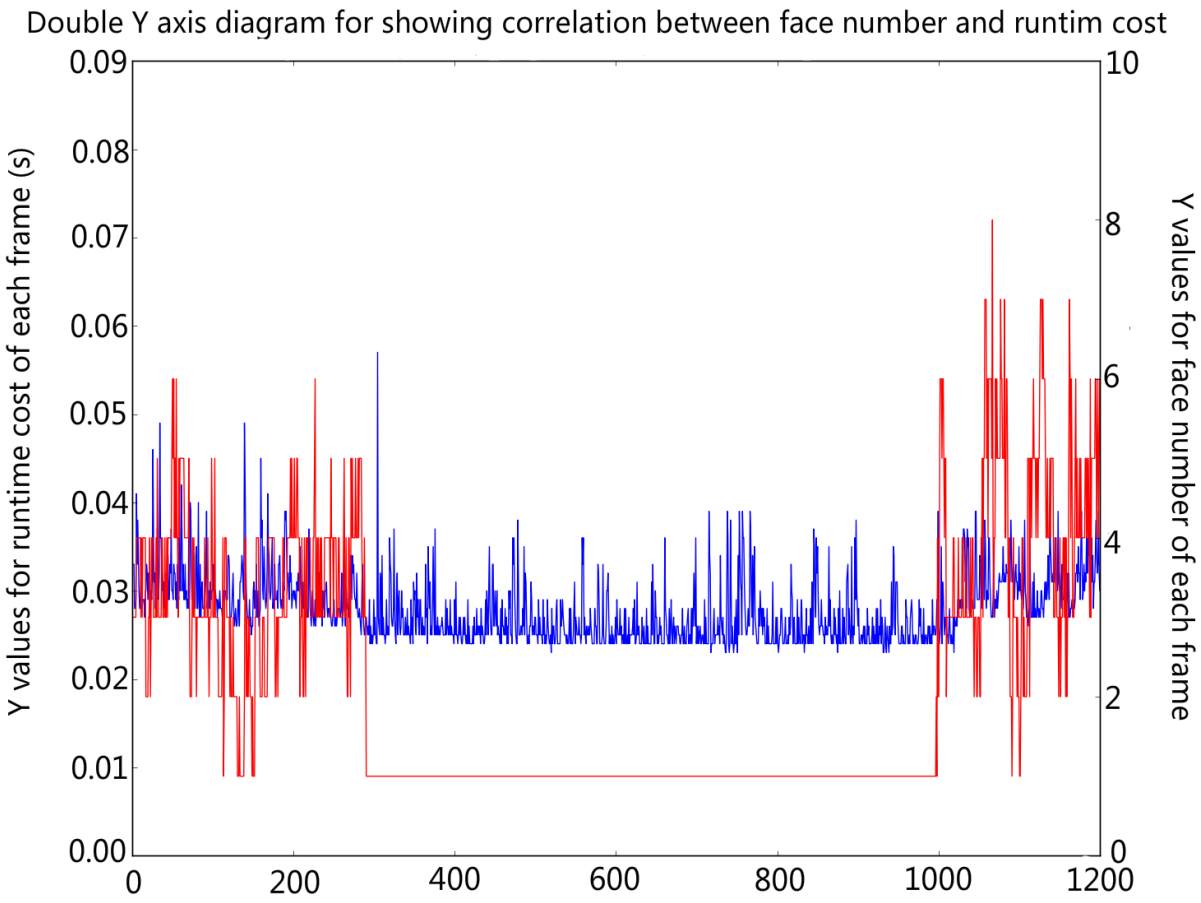

Figure 4.13: The correlation between number of faces and runtime cost.
The X-axis is the frame number, the Y-axis on the left is the runtime(s) and the Y-axis on the right is the face number in the frame and the blue line is the runtime cost curve and the red line is the face number curve. This result is obtained by our multi-face tracker with Network 2

Multi-face tracker

From the evaluation we can see that our tracker has similar tracking performance with these three different tracking classifiers (Network 1, Network 2 and Network 3). The tracker with Network 2 performs sightly better in both of the two evaluation methods [47, 9] that we have considered. The cosine similarity can also accomplish the tracking network's classification function. From the evaluation results, we can see that the cosine similarity accuracy is similar to Network 1, but not as good as Network 2. The runtime costs for these three trackers are also similar. Although the Network 3 is a 3-layer network, and hence has more layers than Network 1 and Network 2, the extra Pooling layer reduces the computational complexity and makes the tracker with Network 3 run faster than the one with Network 2. Although, even Network 3 is deeper, the accuracy does not increase compared with Network 2. From the standard deviation values, we can see that the tracker with Network 2 is more stable. The cosine similarity has smaller runtime cost than these these tracking networks, however, the standard deviation value of the cosine similarity is slightly higher than these three tracking networks.

The evaluation results show that our multi-face tracker has a high recall and precision. In the test video "Turning" which contains many profile faces, the evaluation result shows that our profile face tracking result is much better than other multi-face trackers. However, because of detecting more profile faces, there are more faces that need to be tracked in our tracking process. It also increases our tracking difficulties. So, if there is a more accurate ground truth, our tracker could get a better result compared with other algorithms.

The testing video data set [55] is recorded by a fixed camera, hence there will be no camera cuts in the video. In this case, our multi-face tracker does obtain a better result than these state-of-the-art multi-face trackers. However, another test video is the popular show "The Big Bang Theory" which contains a lot of scene switches. Some of these multi-face trackers are able to link the faces across the scenes. However, due to the limitation of our algorithm and system design, our multi-face tracker can run in real time but is not able to recognize the same face across different scenes even if the video has

recorded the same face. Instead it will treat the same face as a different face if a scene switch happens. In this test video, our tracker is more competitive than other trackers for MT parameters. The results show that our tracker can cover most of the tracklets in this vidoe with more than 80% of each tracklet length. Our multi-face tracker tracks all the people that appear in the whole video with a clear faces, which are 37 people in total and hence increases the tracking difficulty. This is another reason our tracker produces a larger number of IDs compared with other algorithms.

Comparison

The comparison with other multi-face trackers shows that our multi-face tracker performs better on the test videos based on both evaluation metrics. On the test video "Turning", our tracking result creates more faces than the ground truth. There are 2799 faces in the gruond truth of video "Turning", however, our results contain 3245 faces, where there are only 2 faces are false positive results. With the increase in faces, the tracklets are longer or more, which are challenges for multi-face trackers. It increases the tracking difficulty compared with the trackers which track fewer faces. But according to the ground truth, our result can cover all of the ground truth tracklets, which is much better than other algorithms we compare with.

The Big Bang Theory test video contains 277 scenes switches, which results in our multi-face tracker stop tracking at the switches. Our multi-face tracker can overcome some camera cut situations. However, when the scene totally changes, our multi-face tracker has to re-initialize the tracking labels. Due to the tracking network in our multi-face tracker only being able to match similar face windows and not being able to recognize the same face, our tracker fails when the scene changes, even if they are the same people in both the current scene and previous scene.

There are some examples where our multi-face tracker can overcome some camera cuts, and some other camera cuts where our multi-face tracker fails.

The Figure 4.14 shows a camera cut where two frames are recorded for the same scene. Frame 5969 shows the man's frontal face and the woman's profile face, however, the next several frames are recorded by another camera as shown in frame 6008. Although this

frame contains the same people, the face features change too much. It is very difficult for our tracking network to match the frontal face and profile face even they belong to the same person in the same scene. This is one of our multi-face tracker's limitations.

Figure 4.15 shows the situation where a scene switches. The frame 7435 and frame 7436 are presenting the same person, however, because the scene switching, the whole image changes too much. The person's face feature also changes a lot, which results in our multi-face tracker failing to track the same person using the same label.

Figure 4.16 shows another camera cut. After a camera cut, the camera continues to record the same two faces from a similar direction, which does not make the faces look very different. Our multi-face tracker can succeed to continuously track the face in this kind of camera cut.

Figure 4.14: Camera cut example 1

Our multi-face tracker fails in the shown camera cut. This result is obtained by our multi-face tracker with Network 2, on BBT0101.

However, according to the ground truth, our tracker still gains a better results compared with other algorithms.

Our multi-face tracker achieves a good performance evaluation result based on the test videos. However, it also fails on the video with many scene switches. As introduced in Chapter 1, our tracker is designed for live video, real-time conference recording and live broadcasting. The situations included in these applications would be similar to the ones

Figure 4.15: Scene switch example

Our multi-face tracker fails due to scene switching. This result is obtained by our multi-face tracker with Network 2, on BBT0101.

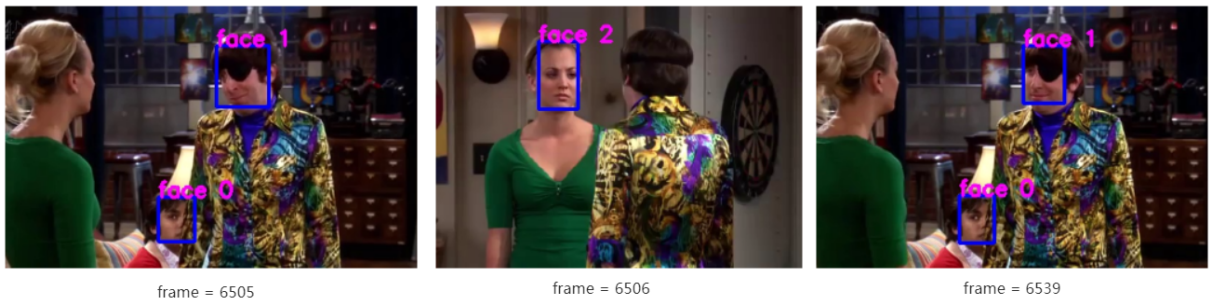

Figure 4.16: Camera cut example 2

Our multi-face tracker succeeds in the camera cut. This result is obtained by our multi-face tracker with Network 2, on BBT0101.

in the data set [55], where our multi-face tracker has achieved very high accuracy. Given the goals in our thesis, our multi-face tracker can meet the majority of the requirements.

Chapter 5

Conclusions

5.1 Conclusions and limitation

This thesis presents a real-time multi-face tracker based on convolutional neural networks. Although many multi-face trackers have been developed [43, 19, 64, 77, 78, 66], there is no such multi-face tracker that can run in real time. Our research is motivated by the real world industrial production needs, such as conference recording with name labels, live broadcasting, and some applications that requires the real-time location of the main speaker in order to control camera movements. These applications require real-time tracking outputs at the time when the input frames are processed.

Our multi-face tracker is designed under the condition of running in real time. After

the face detector generates the detection results, the CNN for extracting features of all the detected faces will help our system designed to run in real time. First, the feature extraction neural network generates all the face feature maps. The tracking network will use the feature maps to match these faces. Compared with the design such as deep neural networks applied to face recognition which use a deep network to process all the faces each time when recognition occurs, our design separates the feature extraction from the tracking process, which will greatly reduce the runtime cost. Since the deeper feature extraction network runs only once on each face, the shallow tracking network does not need to extract the tracking face feature each time when the network processes this face. In this design we can adopt different neural networks for face feature matching. Under the condition of running in real time, we design three alternative neural networks with different layers for each network.

From the evaluation results based on two different evaluation matrices [9, 47] and two testing data sets, our multi-face tracker achieves a better accuracy on videos recorded by one fixed camera compared with other multi-face trackers [64, 77, 78, 66, 19, 84]. The tracking results show that our multi-face tracker can track more profile faces than the other investigated trackers. The tracking results comparison with another multi-face tracker [84] clearly demonstrates that our multi-face tracker achieves a more accurate performance than this tracker [84].

There are some restrictions of other trackers [77, 78, 66, 66] that were previously introduced in Chapter 2, such as using part of testing video frames to train the tracker, the linear increase of runtime cost by increasing the number of tracking faces, and some trackers require pre-tuning the tracking parameters for different types of videos in order to perform better. Our multi-face tracker improves on these tracker in the above mentioned restrictions. The multi-face tracker presented in this thesis is a well-trained tracker, which does not need more training on the video that it is applied to. As the experimental results in Chapter 4 show, the runtime cost increases slightly when the number of tracked faces increases. The system speed is stable when tracking 10 people or less. Although the runtime cost will increase when the number increases to around 40, it is not on a linear scale. The runtime costs for our tracker tested on the video in dataset [55] are 0.03480s,

0.03541s and 0.03530s per frame, respectively. These three results are the total runtime cost for the tracking systems with the tracking network selection of Network 1, Network 2 and Network 3 respectively. The tracking accuracy is similar with these three selections. The runtime cost of the combination with Network 2 is slightly higher among these selections, and the multi-face tracker performance is also slightly better with Network 2.

This multi-face tracker we have developed has achieved most of the requirements regarding the research motivation. From the evaluation results based on the tests ran on video "The Big Bang Theory", our multi-face tracker shows the obvious disadvantages compared with other multi-face tracker in tracking faces at camera cuts when there are scene switches. Our algorithm design limits the memory of our tracker for storing IDs of face feature vectors to 3 frames. The large amount of camera cuts with scene switches in this video is difficult for our tracker. Since our tracker is not able to recognize the same face, across the scene switches, our tracker resets the face ID. For the new scene, new labels are created for each face, which interrupts the track even if the same person is shown in both, the current scene and the previous scene. If our tracker can overcome the interruptions from scene switching, the accuracy will increase even further. Although our face-tracker still has limitations and produces errors under certain situations, nevertheless in our evaluations, it is still the most competitive real-time multi-face tracker when applied to live video, live broadcasting and conference recording applications.

5.2 Future work

In future work, the use of a body detector to replace the face detector can implement a multi-human tracking system. Compared with a face, the whole body contains more information for the shallow tracking network to match, such as the clothes color, hair color and body features. Using the body as the tracking object, the tracklet may not as easily be interrupted when the tracked person turns around. With a human body detector, we do not consider situations such as how to track the profile of faces during tracking. The tracking accuracy should also increase without these difficulties to achieve overcoming camera cuts and scene switches. However, people tracking may not be applicable in

scenarios where the video is focused only on the head of the person, e.g., in broadcast recording of meetings or in conference calls.

Another future extension is applying re-identification [54]. Since re-identification increases runtime cost, it cannot be applied in every frame to this system now. However, if referring to the Re-identification result, the IDs and scene switches can be reduced greatly. There is still a gap between our face detection training result and MTCNN authors'. The more effective training methods may help to narrow the gap between our training results and MTCNN authors' results.

References

[1] Edward H Adelson et al. "Pyramid methods in image processing". In: *RCA engineer* 29.6 (1984), pp. 33–41.

[2] Yael Adini, Yael Moses, and Shimon Ullman. "Face recognition: The problem of compensating for changes in illumination direction". In: *IEEE Transactions on pattern analysis and machine intelligence* 19.7 (1997), pp. 721–732.

[3] Maedeh Aghaei, Mariella Dimiccoli, and Petia Radeva. "Multi-face tracking by extended bag-of-tracklets in egocentric photo-streams". In: *Computer Vision and Image Understanding* 149 (2016), pp. 146–156.

[4] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. "Face description with local binary patterns: Application to face recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 28.12 (2006), pp. 2037–2041.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *Computer vision–ECCV 2006* (2006), pp. 404–417.

[6] George Bebis and Michael Georgiopoulos. "Feed-forward neural networks". In: *IEEE Potentials (Volume: 13, Issue: 4, Oct.-Nov. 1994)* (2010), pp. 27–31.

[7] Soufiane Belharbi et al. "Input/Output Deep Architecture for Structured Output Problems". In: *CoRR abs/1504.07550* (2015).

[8] Yoshua Bengio et al. “Greedy layer-wise training of deep networks”. In: *Advances in neural information processing systems* 19 (2007), p. 153.

[9] Keni Bernardin and Rainer Stiefelhagen. “Evaluating multiple object tracking performance: the CLEAR MOT metrics”. In: *EURASIP Journal on Image and Video Processing* 2008.1 (2008), pp. 1–10.

[10] Gary R Bradski. “Computer vision face tracking for use in a perceptual user interface”. In: (1998).

[11] Gary R Bradski. “Real time face and object tracking as a component of a perceptual user interface”. In: *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*. IEEE. 1998, pp. 214–219.

[12] Jatin Chatrath et al. “Real time human face detection and tracking”. In: *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*. IEEE. 2014, pp. 705–710.

[13] Dong Chen et al. “Joint cascade face detection and alignment”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 109–122.

[14] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. “Multi-column deep neural networks for image classification”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 3642–3649.

[15] Dan C. Cireşan. et al. “Flexible, High Performance Convolutional Neural Networks for Image Classification”. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 1237–1242. ISBN: 978-1-57735-514-4.

[16] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. “Real-time tracking of non-rigid objects using mean shift”. In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. IEEE. 2000, pp. 142–149.

[17] C. Cortes and V Vapnik. “Support-vector networks”. In: *Machine Learning. 20 (3): 273–297.* (1995).

[18] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* Vol. 1. IEEE. 2005, pp. 886–893.

[19] Ming Du and Rama Chellappa. "Face association for videos using conditional random fields and max-margin markov networks". In: *IEEE transactions on pattern analysis and machine intelligence* 38.9 (2016), pp. 1762–1773.

[20] Sachin Sudhakar Farfade, Mohammad J Saberian, and Li-Jia Li. "Multi-view face detection using deep convolutional neural networks". In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval.* ACM. 2015, pp. 643–650.

[21] Rogério S Feris, Roberto M Cesar, and Volker Krüger. "Efficient real-time face tracking in wavelet subspace". In: *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on.* IEEE. 2001, pp. 113–118.

[22] Karl Pearson F.R.S. "LIII. On lines and planes of closest fit to systems of points in space". In: *Philosophical Magazine, 2:11, 559-572* (1901).

[23] Christophe Garcia and Manolis. Delakis. "A neural architecture for fast and robust face detection". In: *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on.* IEEE. 1992, pp. 379–385.

[24] Ross Girshick. "Fast R-CNN". In: *The IEEE International Conference on Computer Vision (ICCV).* Dec. 2015.

[25] Ross Girshick et al. "Region-based convolutional networks for accurate object detection and segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2016), pp. 142–158.

[26] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." In: *Aistats.* Vol. 15. 106. 2011, p. 275.

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[28] David Held, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks". In: *European Conference on Computer Vision*. Springer. 2016, pp. 749–765.

[29] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[30] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science*. American Association for the Advancement of Science. 2006, pp. 504–507.

[31] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[32] Chang Huang et al. "Learning sparse features in granular space for multi-view face detection". In: *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*. IEEE. 2006, pp. 401–406.

[33] Itseez Intel Corporation Willow Garage. *Open Source Computer Vision*. 1999. URL: `http://opencv.org/` (visited on 08/03/2017).

[34] Vidit Jain and Erik G Learned-Miller. "Fddb: A benchmark for face detection in unconstrained settings". In: *UMass Amherst Technical Report* (2010).

[35] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. "What is the best multi-stage architecture for object recognition?" In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 2146–2153.

[36] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the 22Nd ACM International Conference on Multimedia*. MM '14. Orlando, Florida, USA: ACM, 2014, pp. 675–678. ISBN: 978-1-4503-3063-3. DOI: `10.1145/2647868.2654889`. URL: `http://doi.acm.org/10.1145/2647868.2654889`.

[37] Kushsairy Kadir et al. “A comparative study between LBP and Haar-like features for Face Detection using OpenCV”. In: *Engineering Technology and Technopreneuship (ICE2T), 2014 4th International Conference on*. IEEE. 2014, pp. 335–339.

[38] Ilya Kalinovskii and Vladimir Spitsyn. “Compact convolutional neural network cascade for face detection”. In: *arXiv preprint arXiv:1508.01292* (2015).

[39] Minyoung Kim et al. “Face tracking and recognition with visual constraints in real-world videos”. In: *Computer Vision and Pattern Recognition, 2008*. IEEE. 2008, pp. 1–8.

[40] Volker Kruger, Alexander Happe, and Gerald Sommer. “Affine real-time face tracking using a wavelet network”. In: *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on*. IEEE. 1999, pp. 141–148.

[41] Marco La Cascia, Stan Sclaroff, and Vassilis Athitsos. “Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3D models”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.4 (2000), pp. 322–336.

[42] Yann Le Cun et al. “Handwritten digit recognition: Applications of neural network chips and automatic learning”. In: *IEEE Communications Magazine* 27.11 (1989), pp. 41–46.

[43] Nam Le et al. “Temporally subsampled detection for accurate and efficient face tracking and diarization”. In: *International Conference on Pattern Recognition. IEEE*. 2016.

[44] Yann LeCun and Yoshua Bengio. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural network*. 1997.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[46] Haoxiang Li et al. “A Convolutional Neural Network Cascade for Face Detection”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[47] Yuan Li, Chang Huang, and Ram Nevatia. “Learning to associate: Hybridboosted multi-target tracker for crowded scene”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 2953–2960.

[48] Yufeng Liang and Joseph Wilder. “Real-time face tracking”. In: *Photonics East (ISAM, VVDC, IEMB)*. International Society for Optics and Photonics. 1998, pp. 149–156.

[49] Rainer Lienhart and Jochen Maydt. “An extended set of haar-like features for rapid object detection”. In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 1. IEEE. 2002, pp. I–I.

[50] Leyuan Liu et al. “A low-cost real-time face tracking system for ITSs and SDASs”. In: *Software: Practice and Experience* (2016).

[51] Qiang Liu et al. “Camshift based real-time multiple faces match tracking”. In: *Intelligent Signal Processing and Communication Systems, 2007. ISPACS 2007. International Symposium on*. IEEE. 2007, pp. 726–729.

[52] Rong Liu et al. “Tracking and recognition of multiple faces at distances”. In: *International Conference on Biometrics*. Springer. 2007, pp. 513–522.

[53] Evangelos Loutas, Ioannis Pitas, and Christophoros Nikou. “Probabilistic multiple face detection and tracking using entropy measures”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 14.1 (2004), pp. 128–135.

[54] Bingpeng Ma, Yu Su, and Frédéric Jurie. “Bicov: a novel image representation for person re-identification and face verification”. In: *British Machive Vision Conference*. 2012, 11–pages.

[55] Emilio Maggio et al. “Particle PHD filtering for multi-target visual tracking”. In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 1. IEEE. 2007, pp. I–1101.

[56] Pauline C Ng and Steven Henikoff. "SIFT: Predicting amino acid changes that affect protein function". In: *Nucleic acids research* 31.13 (2003), pp. 3812–3814.

[57] Nuria Oliver, Alex Pentland, and François Bérard. "LAFTER: a real-time face and lips tracker with facial expression recognition". In: *Pattern recognition* 33.8 (2000), pp. 1369–1382.

[58] William K Pratt. "Image feature extraction". In: *Digital Image Processing: PIKS Scientific Inside, Fourth Edition* (1978), pp. 535–577.

[59] Richard J Qian, M Ibrahim Sezan, and Kristine E Matthews. "A robust real-time face tracking algorithm". In: *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. Vol. 1. IEEE. 1998, pp. 131–135.

[60] Geovany A Ramirez and Olac Fuentes. "Multi-pose face detection with asymmetric haar features". In: *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*. IEEE. 2008, pp. 1–6.

[61] Andreas Ranftl, Fernando Alonso-Fernandez, and Stefan Karlsson. "Face Tracking Using Optical Flow: Development of a Real-Time AdaBoost Cascade Face Tracker". In: *14th International Conference of the Biometrics Special Interest Group, BIOSIG, Darmstadt, Germany, 9-11 September, 2015*. 2015.

[62] Marc'Aurelio Ranzato et al. "Efficient learning of sparse representations with an energy-based model". In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. MIT Press. 2006, pp. 1137–1144.

[63] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[64] Markus Roth et al. "Robust multi-pose face tracking by multi-stage tracklet association". In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, pp. 1012–1016.

[65] Gregory Shakhnarovich, Paul A Viola, and Baback Moghaddam. "A unified learning framework for real time face detection and classification". In: *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*. IEEE. 2002, pp. 16–23.

[66] Xin Shi et al. "Real-Time Face Recognition Method Based on the Threshold Determination of the Positive Face Sequence". In: *Proceedings of the 22nd International Conference on Industrial Engineering and Engineering Management 2015*. Springer. 2016, pp. 125–136.

[67] Sascha Spors and Rudolf Rabenstein. "A real-time face tracker for color video". In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. Vol. 3. IEEE. 2001, pp. 1493–1496.

[68] Lukasz Stasiak and Andrzej Pacut. "Particle filters for multi-face detection and tracking with automatic clustering". In: *Imaging Systems and Techniques, 2007. IST'07. IEEE International Workshop on*. IEEE. 2007, pp. 1–6.

[69] Yi Sun et al. "Deepid3: Face recognition with very deep neural networks". In: *arXiv preprint arXiv:1502.00873* (2015).

[70] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. ""Knock! Knock! Who is it?" probabilistic person identification in TV-series". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2658–2665.

[71] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. "Original approach for the localisation of objects in images". In: *In Pattern Recognition 2002*. 2002, None.

[72] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–511.

[73] Paul Viola and Michael J Jones. "Robust real-time face detection". In: *International journal of computer vision* 57.2 (2004), pp. 137–154.

[74] Duc My Vo and Andreas Zell. “Real-time face recognition using local ternary patterns with collaborative representation-based classification for mobile robots”. In: *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 781–793.

[75] Shaohua Wan et al. “Bootstrapping Face Detection with Hard Negative Examples”. In: *arXiv preprint arXiv:1608.02236* (2016).

[76] Liang Wang and David Suter. “Imagenet classification with deep convolutional neural networks”. In: *In Advances in neural information processing systems*. 2012.

[77] Baoyuan Wu, Bao-Gang Hu, and Qiang Ji. “A Coupled Hidden Markov Random Field model for simultaneous face clustering and tracking in videos”. In: *Pattern Recognition* 64 (2017), pp. 361–373.

[78] Baoyuan Wu et al. “Simultaneous Clustering and Tracklet Linking for Multi-Face Tracking in Videos, IEEE Intl”. In: *Conf on Computer Vision (ICCV)*. 2013.

[79] Fan Yang and Michel Paindavoine. “Implementation of an RBF neural network on embedded systems: real-time face tracking and identity verification”. In: *IEEE Transactions on Neural Networks* 14.5 (2003), pp. 1162–1175.

[80] Jie Yang and Alex Waibel. “A real-time face tracker”. In: *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*. IEEE. 1996, pp. 142–147.

[81] Shuo Yang et al. “From facial parts responses to face detection: A deep learning approach”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3676–3684.

[82] Shuo Yang et al. “WIDER FACE: A Face Detection Benchmark”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[83] Kaipeng Zhang et al. “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks”. In: *IEEE Signal Processing Letters* 23.10 (Oct. 2016), pp. 1499–1503.

[84] Shun Zhang et al. “Tracking Persons-of-Interest via Adaptive Discriminative Features”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 415–433.