Project Report for EE7700

Name: Jing Chen, Shaoming Chen

Student ID: 89-507-3494, 89-295-9668

Face Tracking

1. Objective of the study

Given a video, this semester project aims at implementing algorithms to achieve the face tracking within the whole video. It definitely will help us get much deeper understanding about those face detection and object tracking algorithms.

2. Background work on the face tracking

There is substantial study based on the topic of face tracking. Basically, all the study can be simply divided into two categories. One is real time face detection, and the other is the combination of face detection and face tracking. Specifically, some face detection algorithms can get the faces in each frame timely, which can directly achieve the goal of face tracking; the other way to implement face tracking is to detect the face at the start frame first, and then adopt the object tracking algorithm to follow the face in the other frames. The second method of tracking face has been used in this project. In detail, viola jones face detection algorithm and camshift has been adopted to obtain the face and tracking the face, respectively.

3. Algorithms

3.1 Viola jones face detection algorithm

Generally, viola jones face detection algorithm has three critical steps, including feature extraction, boosting and multi-scale detection.

3.1.1 feature extraction

It is obvious that feature is very significant to any object detection algorithm. Basically, there are a lot of features, such as eyes, nose, the topology of eye and nose, can be used for face detection. In viola jones face detection, a very simple and straightforward feature has been used. Figure 1 shows four different feature in viola jones algorithm. Each feature can be obtained by subtracting white areas from the black areas. Here, the area means the summation of all the pixels' gray value within the rectangle. Aiming at calculating these features, a special representation named as integral image has been used. Specifically, integral image of a location (x, y) is the sum of the pixel values above and to the left of (x, y), inclusively. Figure 2 shows one fast way to compute the pixel sum within a rectangle. In Figure 2, the value of integral image at location '1' (v_1) is the sum of pixels in rectangle A; the value at location '2' (v_2) is the sum of pixels in

rectangle A and B; the value at location '3' (v_3) is the sum of pixels in rectangle A and C; the value at location '4' (v_4) is the sum of pixels in rectangle A, B, C and D. Based on this information, the sum of pixels in rectangle D can be easily got from $v_4 + v_1 - v_2 - v_3$. Therefore, the sum of pixels of any rectangle located at any position can be obtained following this principle very efficiently.

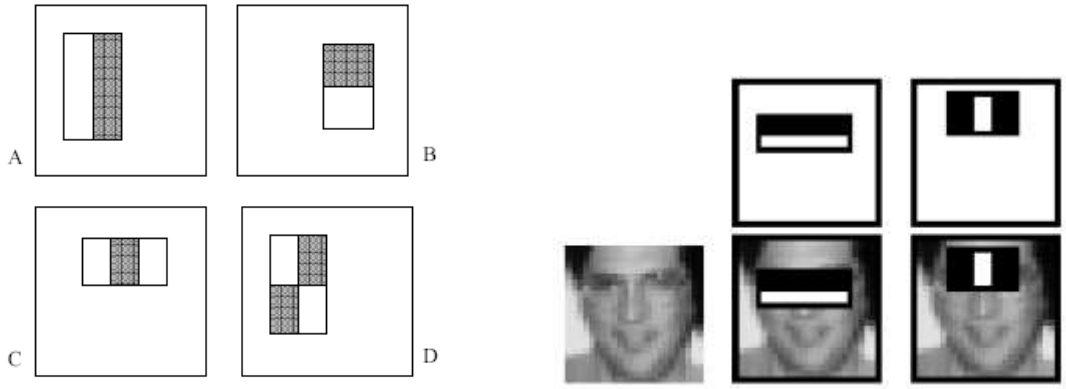

Figure 1: four basic features in viola jones algorithm

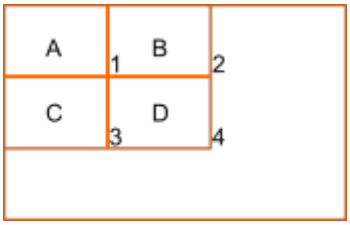

Figure 2: the calculation of pixel sum within a rectangle

3.1.2 boosting

The definition of boosting in viola jones face detection algorithm is the combination of several weak classifiers. This boosting idea makes the process of learning to be simple and efficient. Specifically, the boosting works as follows:

1. given a dataset, learn a single and simple classifier first and check where it makes errors;
2. reweight the dataset and give the data where it made errors a higher weight;
3. learn the second simple classifier based on the reweighted dataset;
4. combine the first and second classifier and reweight data where they make errors;
5. keep learning until we get T classifiers;
6. the final classifier will be the combination of all those T classifiers.

Figure 3 shows detail of the principle of boosting.

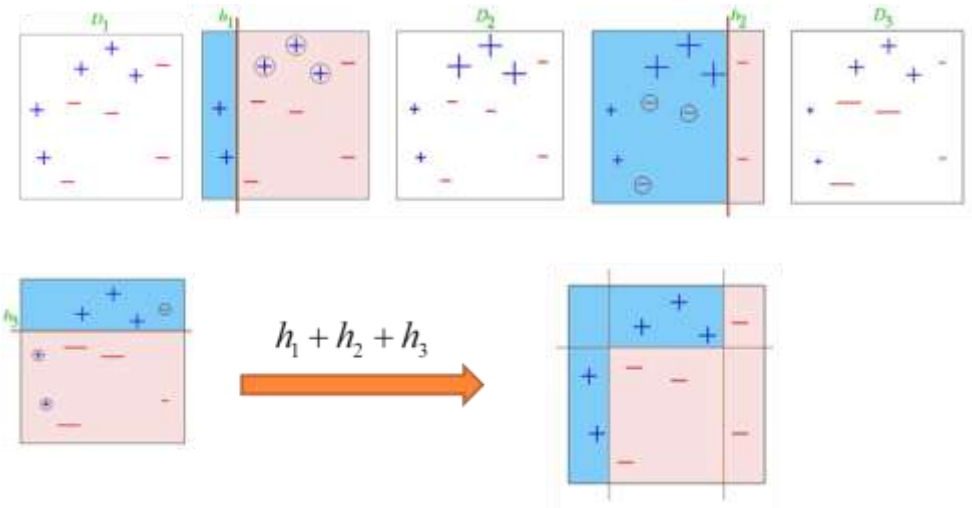

Figure 3: the process of boosting with 3 simple classifiers

3.1.3 Multi-scales detection algorithm

Another important step in viola jones face detection algorithm is multi-scale detection. It is obvious that we have no idea with the size of face in an image before doing face detection. Therefore, multi-scale detection should be adopted to guarantee that faces with any size can be detected. Features should be calculated at all different scales since the learning and testing are only based on the rectangle features. In this project, we used the scales by the factor of 1.2. In each iteration, the width and height of rectangle will increase to 1.2 times of previous one.

3.2 Camshift

CAMshift is called Continuously Adaptive Mean Shift based on the mean shift algorithm[2]. CAMshift uses the Hue channel to trace objects since by using the Hue channel based on HSV color model, objects with different colors can be recognized. Based on the color information, CAMshift tracks objects faster and consumes relatively little CPU resources. Lower computing resource requirement enable CAMshift to become a one of real-time face tracking algorithms

The procedure of CAMshift shown in the Figure 4 includes two important parts which are the histogram and the search of peak probability. The first step of Camshift is to obtain the histogram of tracked object. In the second step, the next frame will be converted into a map of skin color probability based on the histogram. In the third step, the peak probability center will be found based on zero moment and first moment. Finally, CAMshift will check whether it converges. If it is no, it will go to the third step;

otherwise, it gets the position of the tracked object in this frame and fetches the next frame to track the object continuously.

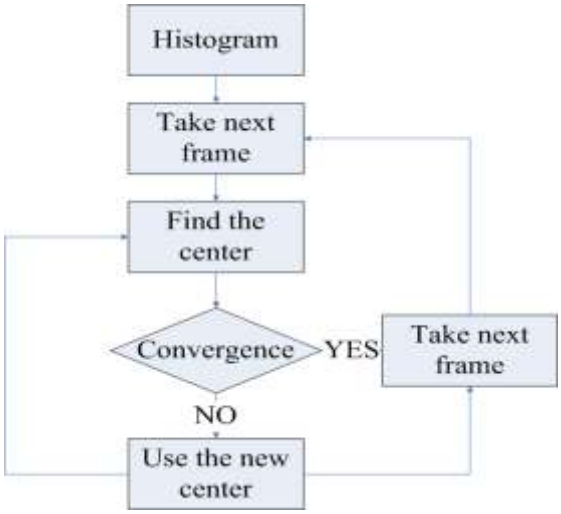

Figure 4 the procedure of CAMshift

3.2.1 Histogram

The histogram is the tracked object's color probability map. Because in the project we focus on the face tracking, the hue channel of the face shown in the Figure 6 is used to illustrate what is the histogram. In the first step, the whole area of tracked object is scanned and the map which record how many pixels have a certain hue value in the tracked object area is built. And then, CAMshift finds out the peak the number of pixel in a certain hue value and normalizes the map into skin color probability map or histogram shown in the Figure 5. The horizontal bar of the Figure 5 is hue value and the vertical bar is the skin probability. According to the histogram, the popular color on the face is revealed.

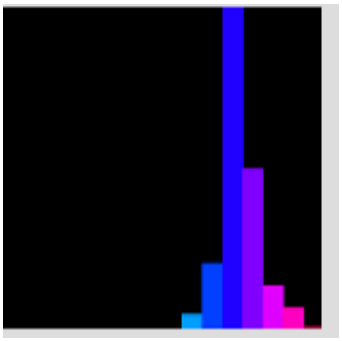

Figure 5 histogram

Figure 6 H channel of the face

3.2.2 Finding the center

Based on histogram, the skin probability of the next frame can be calculated in this way. In the first, the next frame is converted into H channel which represents the color information of the tracked object (from Figure 7 to Figure 8). And then looking up the histogram, we will get the skin probability for each pixel shown in the Figure 9 according to its hue value. The skin probability of the whole picture is obtained for the next step to find out the center of the face.

Figure7 the next frame

Figure 8 the hue channel of the next frame

Figure 9 the skin probability

Based on the skin probability of the next frame, the peak probability as the center of the face is estimated by the zero moment and first moment [2]. The zero moment (M_{00}) and first moment (M_{01} andM_{10}) are calculated by the equations shown below.

$$\boldsymbol{M_{00} = \sum_{x,y} I(x,y)} \quad \boldsymbol{M_{10} = \sum_{x,y} xI(x,y)} \quad \boldsymbol{M_{01} = \sum_{x,y} yI(x,y)}$$

In this way, the center of face can be estimated by the following equations.

$$\boldsymbol{x_c = \frac{M_{10}}{M_{00}}} \qquad \boldsymbol{y_c = \frac{M_{01}}{M_{00}}}$$

Finally, the convergence of the center of the face (x_c, y_c) is checked. If the center of the face is very close to the old center, the convergence is reached; otherwise, all the equations should be calculated again based on the new center.

4. Result

4.1 face detection

The result of face detection is shown in Figure 10 and 11. Those are the frames extracted from the video. Sometimes, face detection algorithm may get more than one result even there is only one face in the frame, such as Figure 12. In this case, the post-processing has been used. If the detector provides more than one rectangle, which indicates the position of the face, the distance of center points of these rectangles has been calculated. If this distance is smaller than a pre-set threshold, the average of these rectangles will be computed and set as the final position of the detected face.

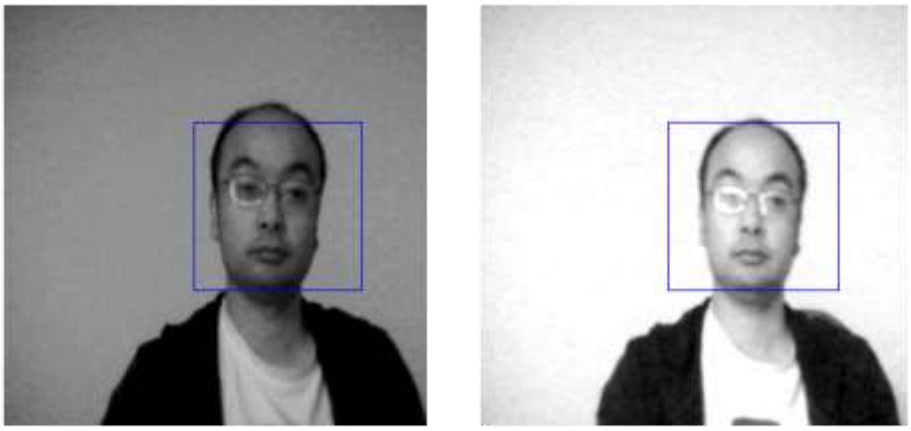

Figure 7: face detection

Figure 8: face detection

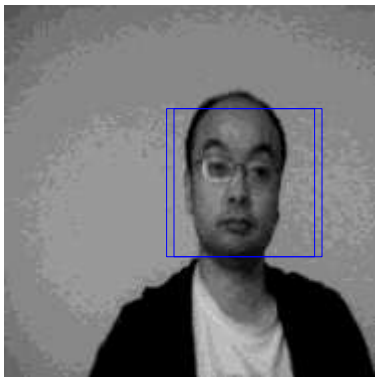

Figure 9: face detection with two detected faces

4.2 face tracking

The result is shown in Figure 13. The result is acceptable but not accuracy. Affected by non-uniform light condition, the result becomes worse since the light changes the color of the object. Moreover, the similar color background, hands and necks also lower the performance. The left image in Figure 13 is the version In the Matlab by using the essential CAMshift idea to track faces. The right image is the version in the OpenCV by using its library which supports CAMshift. The right result is better the left since the CAMshift in the OpenCV removes the noise of background according to objects' moving history. Tracking a face implemented in the OpenCV costs only around 0.5 second, compared with the result of face tracking based on optical flow shown in Figure 14 which costs around 2 second. With some optimizations, the real-time tracking face based on CAMshift is feasible.

Figure 10 face tracking result

Figure 11 face tracking by optical flow

5. Conclusion

In the project, we implement face tracking in the Matlab by using Viola jones face detection and CAMshift tracking. This method is verified and the limitations of the scheme are observed through testing and debugging our codes. And then, limited by Matlab performance, we shift to OpenCV to evaluate the speed of this face tracking scheme. Compared with other popular tracking algorithms such as optical flow, we found the Viola jones face detection and CAMshift tracking is more suitable for real-time face tracking since they requires less CPU resource and costs shorter time.

Individual responsibility:

Jing Chen is responsible for implementing the Viola jones face detection algorithm, and designing the final demonstration of face tracking with Matlab.

Shaoming Chen is responsible for implementing the CAMshift algorithm with Matlab, and designing the final demonstration of face tracking with OpenCV.

Reference

[1] P. Viola, M.J. Jones, Robust Real-Time Face Detection, International Journal of Computer Vision, Vol. 57, No. 2, May 2004, pp. 137-154

[2] G.R. Bradski, Computer video face tracking for use in a perceptual user interface, Intel Technology Journal, Q2 1998