STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Computer Vision
Spring 2015

I pledge my honor that I have abided by the Stevens Honor System

CS558 Final Project
Face Tracking

Yichen Zhai

Yunfeng Wei

Xuanyu Liu

Zhe Han

Submission date

May 11, 2015

In this project, we are going to track human faces in a video. We detect faces frame by frame using face color detection and Boosted Cascade of Simple Features, then we mark all the face areas in every frame. Finally combine all the individual frames to a new successive video in which the faces are marked by rectangle blocks.

1 Video Decomposition and Combination

We took a test video with a human face for about 8.5 seconds, which contains 257 frames. We used readFrame function to obtain each of the frames, and example is shown in Figure 1.

Figure 1. One frame from the video

We used the detection methods to find out and mark the face areas in this image, which will be described in later page. After a series of detection, we obtained the result image with marking blocks, which is shown in Figure 2.

Figure 2. Frame with detected face area

Then we used writeVideo function to combine all of these images to a new video, that is our result face tracking video.

2 Face Color Detection

Our feature detector, which is described in the 3rd part, can only deal with 12 by 12 gray image patches. If we extract any image square in any scale from one frame, there will be too much calculation to do, as a result, we use some face color detection method to find out the potential face areas first, then use feature detector to determine the exact face areas.

We exploited YCrCb color space instead of RGB to detect face areas, which has a better performance and simpler calculation dimensions. In this expression space, Y stands for brightness, Cb is the difference between blue component and a reference value, and Cr is the difference between red component and a reference value. The conversion from RGB to YCrCb is shown below.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112.000 \\ 112.000 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

Because the brightness component has no influence on the color of a human face, we only used Cb and Cr components for detection. As an experience, the statistical distribution of Cb and Cr are usually in these range.

$$77 \leq \mathrm{Cb} \leq 127 \ , \ 133 \leq \mathrm{Cr} \leq 173$$

We used these algorithms to deal with Figure 3, if the Cb and Cr value of a pixel is in the above range, this pixel is described in white, otherwise in black, which is, to binarize the the image. Then we obtained Figure 4.

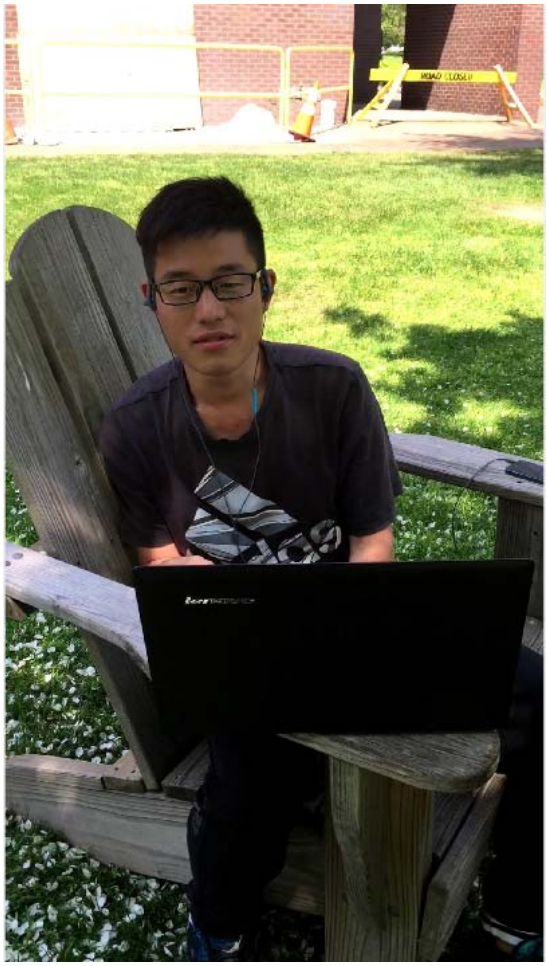

Figure 3. Original test image

Figure 4. Binarized image with face-like color areas

In Figure 4, the face area is divided by the glasses into two parts, so we employed morphological closing operation to enlarge all the white parts, so they could be connected to adjacent parts. This is the code in Matlab:

```
se=strel('square',20);
P=imclose(P,se);
```

We defined a square kernel of 20 by 20 pixels, and used this kernel for closing operation. The result is in Figure 5. As we can see, the two parts divided by glasses are connected together.

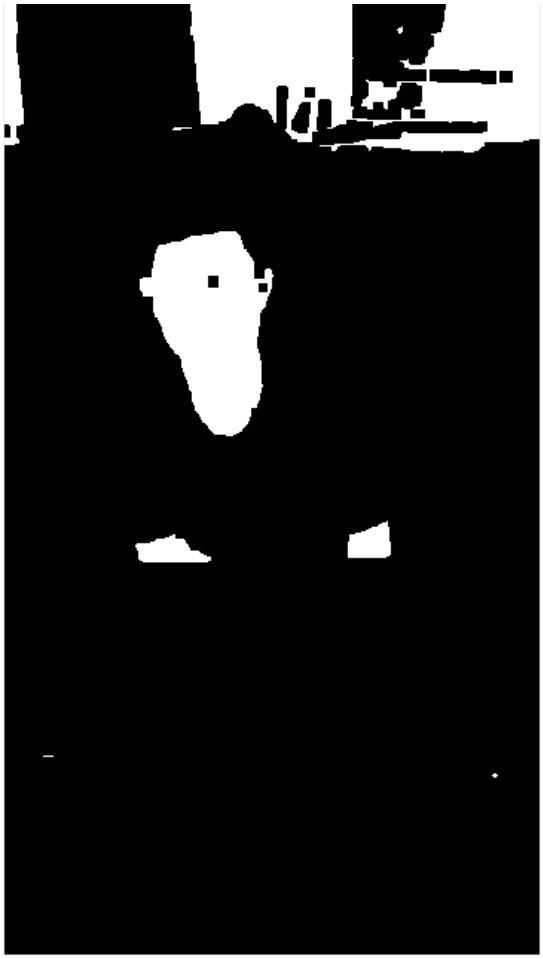

Figure 5. Closing Operation

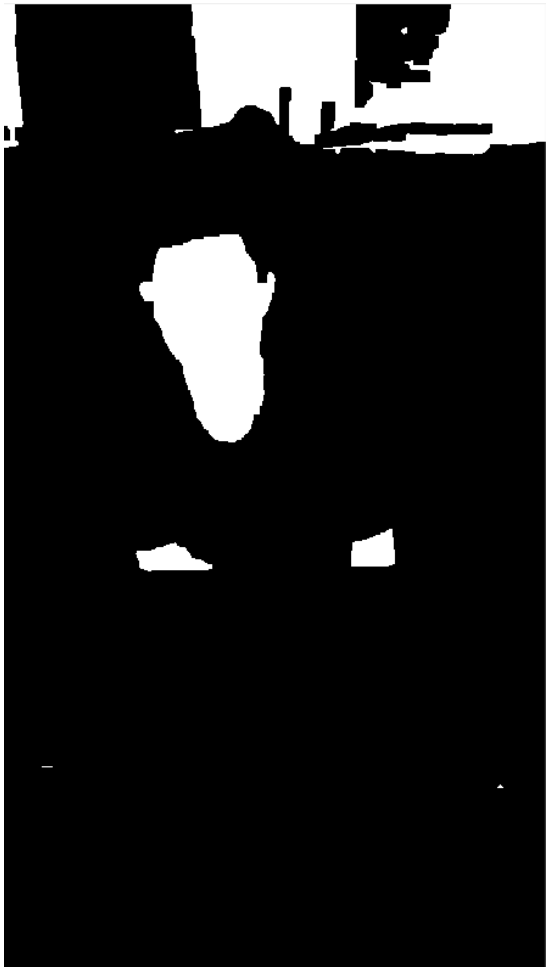

Figure 6. Filling holes

Then we used operation to fill in holes, which is , to fill in all the black parts surrounded by white areas with white. Result is in Figure 6. The code is:

```
P = imfill(P,'holes');
```

Because the whites parts was once enlarged, we need to shrink them to restore them to a more exact size and position. We used morphological opening operation to shrink them, and the kernel should be the same as the closing operation. This operation can also eliminate the white blobs.

```
P=imopen(P,se);
```

The result image is shown in Figure 7.

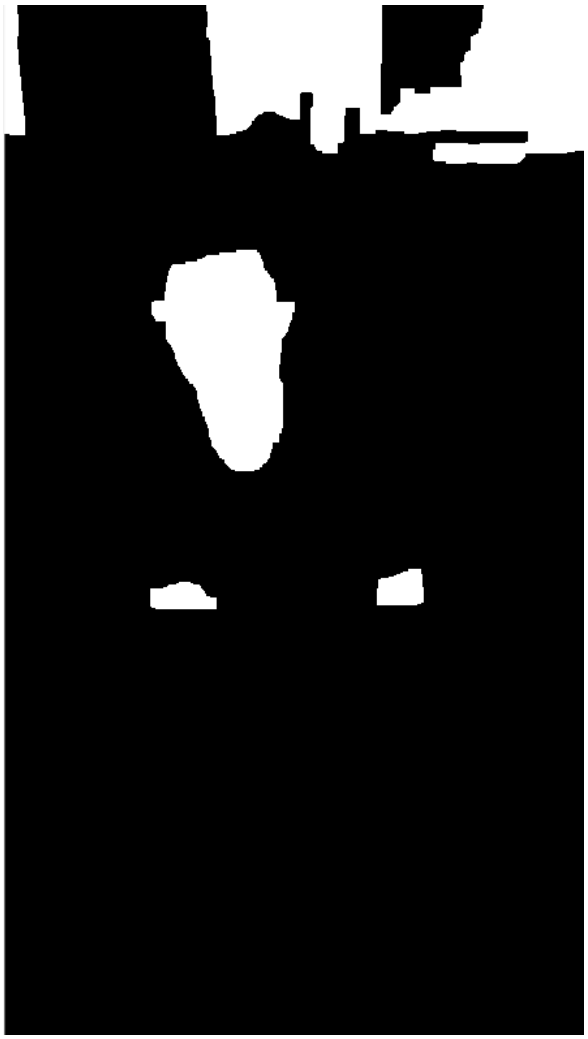

Figure 7. Opening operation

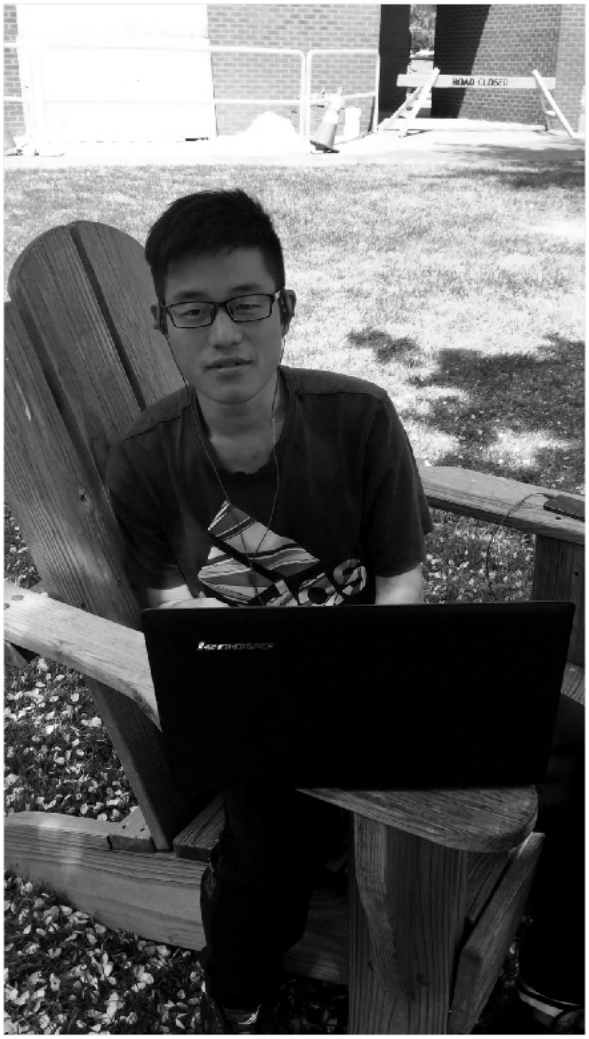

Figure 8. Gray scale image

Figure 8 is the gray scale image of the original one. Because all the following calculation and detection will be based on gray scale, we generate this image first.

We used the Matlab functions to label and calculate properties of all the connected components in Figure 7.And we obtained the area and bounding box of all the components. Then we find the four largest areas,

the four areas in gray scale image is shown in Figure 9.

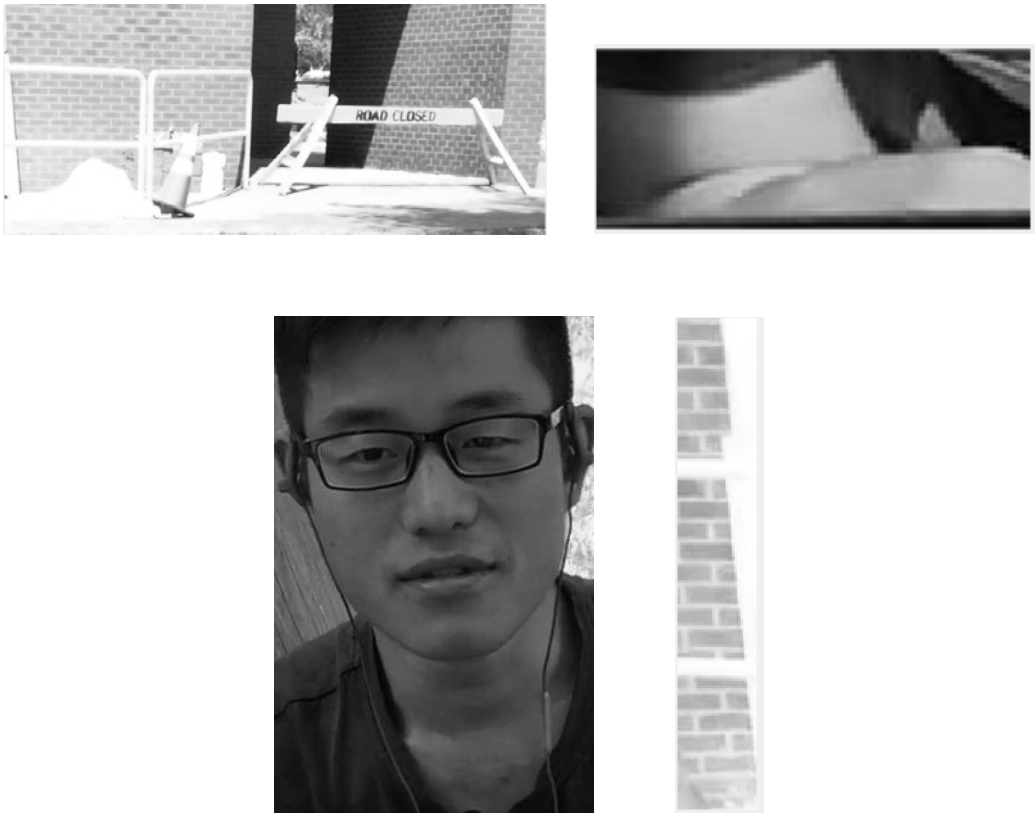

Figure 9. Detected four largest areas

Because the height to width ratio of face area is always in the range of 0.8 and 2.0, according to this criteria, we deleted all the areas out of this range, then we obtained Figure 10. In this case, there is only one result, but these could be more in other cases.

Figure 10. Ratio restrict area result

Then we copy the result in Figure 10 to the center of a black square area, because the feature detector can only deal with 12 by 12 square. This image contains a large part of neck, which should not be considered as face part, and usually neck part could not take more than 30% of the vertical line, so we crop the square part by 70%. These two images are in Figure 11.

Figure 11. Square potential face area and 70% crop

Then we resized the 70% cropped image to 12 by 12 pixels, which will be further determined by feature detector.

3 Boosted Cascade of Simple Features

In order to detect the face from a 12 by 12 pixels image, we can use a good detector introduced by Viola and Jones (2004). Their technique was the first to introduce the concept of boosting to the computer vision community. Boosting involves constructing a classifier h(x) as a sum of simple weak learners. Each weak learner h(x) looks like:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

In the formula, Pj is a parity indicating the direction of the inequality sign, θ is the threshold and f(x) is a feature function.

There are four kinds of features which can be used to detect faces, as the figure below.

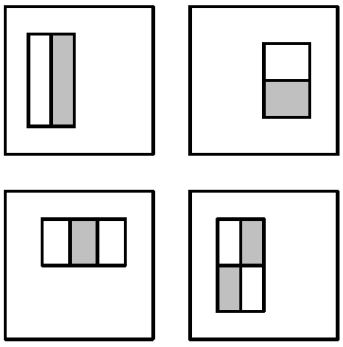

Figure 12. Four features

To find the strong classifier, we can add the weak classifiers together like the figure below to consist a strong classifier which can detect faces correctly.

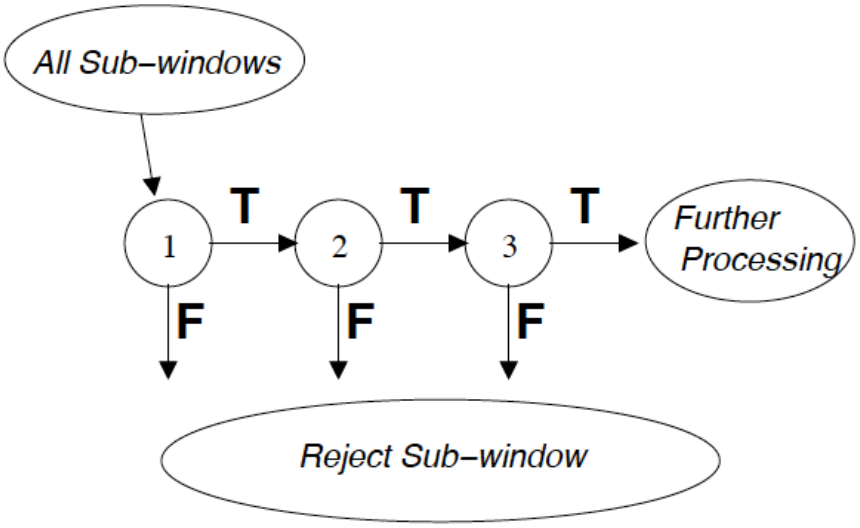

Figure 13. Components of weak classifiers

The Viola-Jones algorithm is briefly shown as the following figure. Following each step, we can finally construct a strong classifier which has a high rate to detect faces correctly.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:

  1. Normalize the weights,

  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

  so that w_t is a probability distribution.

  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:

  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Actually, we set the loop number to 3 so that we can get the best three weak classifier which can be combined to a strong classifier.

To training the strong classifier, we use the dataset from WUSTL which contains almost 4000 face images and 8000 non face images as the training set.

Finally, we get the three h(x) parameters as the figure shows.

bestHParam

3x8 double

	1	2	3	4	5	6	7	8
1	2	2	1	11	7	0.2324	1	0.2324
2	2	10	8	12	8	0.2647	1	0.2702
3	2	7	2	3	1	0.2760	1	0.2902

Column 1 indicates the kind of the feature detector, 2 means the detector is a left/right feature, Column 2 and Column 3 indicates the two row numbers of the feature region, Column 4 and Column 5 means the two column numbers of the feature region, Column 6 is the threshold of this weak classifier, Column 7 is the parity for the weak classifier and Column 8 is the error of the weak classifier.

We can find that the total error rate for this strong classifier is

Error rate = 0.2324 * 0.2702 * 0.2902 = 0.018223 = 1.8%

The corresponding alpha value for each weak classifier is indicated as follows,

1x3 double

	1	2	3
1	1.1947	0.9936	0.8945

This is the strong classifier constructed manually to be used for this face detection project.

Conclusion

After these detection, we successfully obtained all the frames with the face area marked, and we generated a moving video with the face area tracked, which has a high accurate.

Reference

[1] Zhang Chong, Research on face recognition algorithm based on skin color segmentation and matching, Science and Technology College of NCHU, 2013.5.

[2] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features.CVPR 2001.

[3] P. Viola and M. Jones. Robust real-time face detection.IJCV 57(2), 2004.