FACE DETECTION AND SMILE DETECTION

[1]Yu-Hao Huang(黃昱豪), [2]Chiou-Shann Fuh (傅楸善)

[1]Dept. of Computer Science and Information Engineering, National Taiwan University
E-mail:r94013@csie.ntu.edu.tw
[2]Dept. of Computer Science and Information Engineering, National Taiwan University
E-mail:fuh@csie.ntu.edu.tw

ABSTRACT

Due to the rapid development of computer hardware design and software technology, the user demands of electric products are increasing gradually. Different from the traditional user interface, such as keyboard and mouse, some new human computer interactive system like the multi-touch technology of Apple iPhone and the touch screen support of Windows 7 are catching more and more attention. For medical treatment, there are some eye-gaze tracking systems developed for cerebral palsy and multiple sclerosis patients. In this paper, we propose a real-time, accurate, and robust smile detection system and compare our method with the smile shutter function of Sony DSC T300. We have better performance than Sony on slight smile.

1. INTRODUCTION

1.1. Motivation

From Year 2000, the rapid development of hardware technology and software environment make friendly and fancy user interface more and more possible. For example, for some severely injured patients who cannot type or use mouse, there are eye gaze tracking system, by which the user can control the mouse by simply looking at the word or picture shown on the monitor. In 2007, Sony has released its first consumer camera Cyber-shot DSC T200 with smile shutter function. The smile shutter function can detect at most three human faces in the scene and automatically takes a photograph if smile is detected. Many users have reported that Sony's smile shutter function is not accurate as expected, and we find that the Sony's smile shutter is only capable of detecting big smile but not able to detect slight smile. On the other hand, smile shutter would also be triggered if the user makes a grimace with teeth appearing. Therefore we propose a more accurate smile detection system on a common personal computer with a common webcam.

1.2. Related Work

The problem related to smile detection is facial expression recognition. There are many academic researches on facial expression recognition, such as [12] and [4], but there is not much research about smile detection. Sony's smile shutter algorithm and detection rate are not available. Sensing component company Omron [11] has recently released smile measurement software. It can automatically detect and identify faces of one or more people and assign each smile a factor from 0% to 100%. Omron uses 3D face mapping technology and claim its detection rate is more than 90%. But it is not available and we can not test how it performs. Therefore we would test our program with Sony DSC T300 and show that we have a better performance on detecting slight smile and lower false alarm rate on grimace expressions.
From section 2 to section 4 we would describe our algorithm on face detection and facial feature tracking. In section 5, we would run experiments on FGNET face database [3] and show results with 88.5% detection rate and 12.04% false alarm rate while Sony T300 performs 72.7% detection rate and 0.5% false alarm rate. Section 6 will compare with Sony smile shutter on some real case video sequence.

2. FACE DETECTION

2.1. Histogram Equalization

Histogram Equalization is a method for contrast enhancement. We could always take our pictures with under-exposure or over-exposure due to the uncontrolled environment lightness, which would make the details of the images difficult to recognize. Figure 1 is a gray image from Wikipedia [17] that shows a scene with pixel values very concentrated. Figure 2 is the result after histogram equalization.

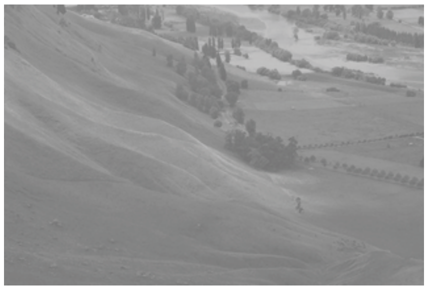
Figure 1: Before histogram equalization [17].

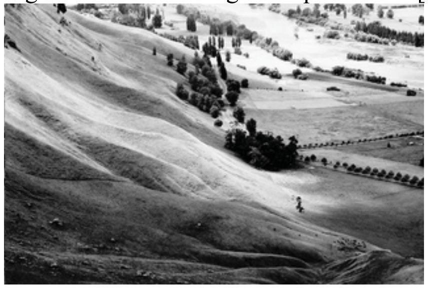
Figure 2: After histogram equalization [17].

2.2. AdaBoost Face Detection

To obtain real-time face detection, we use the method proposed by Viola and Jones [15]. There are three components inside the paper. The first is the concept of "Integral Image", which is a new representation of an image for people to calculate the features quickly. The second is the Adaboost algorithm introduced by Freund and Schapire [5] in 1997, which can extract the most important features from the others. The last component is 'cascaded' classifiers. We can eliminate the non-face regions in the first few stages. With this method, we can detect faces from 320 by 240 pixel images at 60 frames per second with Intel Pentium M. 740 1.73 GHz. We will briefly describe the three major components here.

2.2.1. Integral Image

Given an image *I*, we define an integral image *I'(x, y)* by

$$I'(x',y') = \sum_{x<x',y<y'} I(x,y)$$

The value of the integral image at location *(x, y)* is the summation over all the left and upper pixel values of the original image *I*.

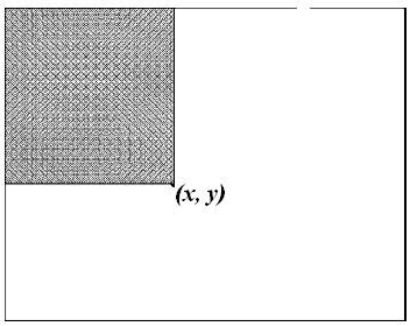

Figure 3: Integral image [15].

If we have the integral image, then we can define some rectangle features shown in Figure 4:

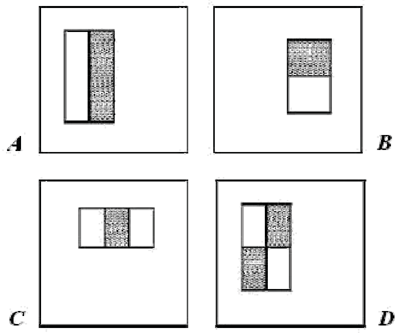

Figure 4: Rectangle feature [15].

The most commonly used features are two-rectangle feature, three-rectangle feature, and four-rectangle feature. The value of two-rectangle feature is the difference of the pixels sum over gray rectangle to the pixels sum over the white rectangle. These two regions have the same size and are horizontally or vertically adjacent as shown in blocks *A*, *B*. Block *C* is a three-rectangle feature whose value is also defined as the difference of pixels sum over the gray region to the pixels sum over the white regions. Block *D* is an example of the four-rectangle features. Since these features have different areas, it must be normalized after calculating the difference. After calculating the integral image in advance, it would be easy to obtain one rectangle region's pixels sum by one-plus operation and two-minus operations. For example, to calculate the sum of pixels within rectangle *D* in Figure 5, we can simply compute 4 + 1 – (2 + 3) value in the integral image.

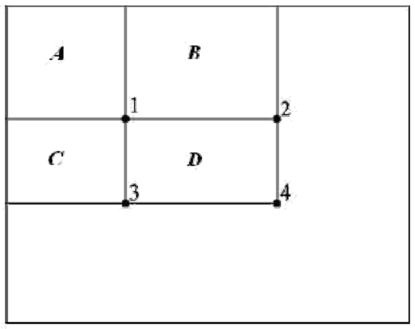

Figure 5: Rectangle sum [15].

2.2.2. AdaBoost

There will be a large number of rectangle features with different sizes. For example, for a 24 by 24 pixel image, there are 160,000 features. Adaboost is a machine-learning algorithm used to find the *T* best classifiers with minimum error. To obtain the *T* classifiers, we will repeat the following algorithm for *T* iterations:

```
• 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, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
    2. Select the best weak classifier with respect to the
       weighted error
           $\epsilon_t = \min_{f,p,\theta} \sum_i w_i \, | h(x_i, f, p, \theta) - y_i |$ .
       See Section 3.1 for a discussion of an efficient
       implementation.
    3. Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where $f_t$, $p_t$, and $\theta_t$
       are the minimizers of $\epsilon_t$.
    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:
      $C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$
  where $\alpha_t = \log \frac{1}{\beta_t}$
```

Figure 6: Boosting algorithm [15].

After running the boosting algorithm for a goal object, we have *T* weak classifiers with different weighting. Finally we have a stronger classifier *C(x)*.

2.2.3. Cascade Classifier

Since we have the *T* best object detection classifiers, we can tune our cascade classifier with user input: the detection rate and the false positive rate. The algorithm is shown below:

```
• User selects values for $f$, the maximum acceptable false
  positive rate per layer and $d$, the minimum acceptable detection
  rate per layer.
• User selects target overall false positive rate, $F_{target}$.
• $P$ = set of positive examples
• $N$ = set of negative examples
• $F_0 = 1.0$; $D_0 = 1.0$
• $i = 0$
• while $F_i > F_{target}$
  – $i \leftarrow i + 1$
  – $n_i = 0$; $F_i = F_{i-1}$
  – while $F_i > f \times F_{i-1}$
      ∗ $n_i \leftarrow n_i + 1$
      ∗ Use $P$ and $N$ to train a classifier with $n_i$ features using
        AdaBoost
      ∗ Evaluate current cascaded classifier on validation set to
        determine $F_i$ and $D_i$.
      ∗ Decrease threshold for the $i$th classifier until the current
        cascaded classifier has a detection rate of at least
        $d \times D_{i-1}$ (this also affects $F_i$)
  – $N \leftarrow \emptyset$
  – If $F_i > F_{target}$ then evaluate the current cascaded detector on
    the set of non-face images and put any false detections into
    the set N
```

Figure 7: Training algorithm for building cascade detector [15].

3. FACIAL FEATURE DETECTION AND TRACKING

3.1. Facial feature location

Although there are many features on human face, most of them are not very useful for facial expression representation. To obtain the facial features we need, we analyze these features from BIOID face database [6]. The database consists of 1521 gray-level images with resolution 384x286 pixels. There are 23 persons in the database and every image consists of a frontal view face from one of them. Besides, there are 20 manually marked feature points as shown in Figure 8.

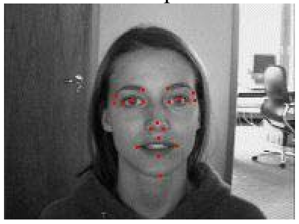

Figure 8: Face and marked facial features [6].

Here is the list of the feature points:

0 = right eye pupil
1 = left eye pupil
2 = right mouth corner
3 = left mouth corner
4 = outer end of right eye brow
5 = inner end of right eye brow
6 = inner end of left eye brow
7 = outer end of left eye brow

8 = right temple
9 = outer corner of right eye
10 = inner corner of right eye
11 = inner corner of left eye
12 = outer corner of left eye
13 = left temple
14 = tip of nose
15 = right nostril
16 = left nostril
17 = centre point on outer edge of upper lip
18 = centre point on outer edge of lower lip
19 = tip of chin

We first use Adaboost algorithm to detect the face region in the image with scale factor 1.05 to get as precise position as possible, and then normalize the face size and calculate the feature relative positions and their standard deviation.

Figure 9: Original image.

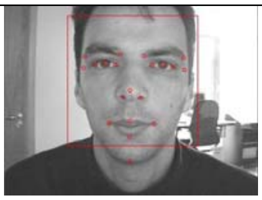
Figure 10: Image with face detection and features marked.

We detect 1467 faces from 1521 images with detection rate 96.45%, and we drop some false positive samples and finally get 1312 useful data. Figure 11 shows one result, in which the center of the feature rectangle is the mean of the feature position and the width and height correspond to four times *x* and *y* feature point standard deviation. Then we can find initial feature position fast.

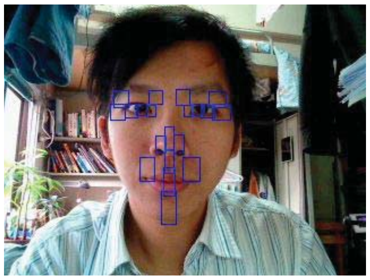
Figure 11: Face and initial feature position with blue rectangle.

Table 1 shows the first four feature points experiment results.

	(Pixel)	(Pixel)	(Pixel)	(Pixel)
Landmark Index	*X*	*Y*	*X* St Dev	*Y* St Dev
0: right eye pupil	30.70	37.98	1.64	1.95
1: left eye pupil	68.86	38.25	1.91	1.91
2: right mouth corner	34.70	78.29	2.49	4.10
3: left mouth corner	64.68	78.38	2.99	4.15

Table 1 Four facial feature locations and error mean with faces normalized to 100x100 pixels.

3.2. Optical flow

Optical flow is the pattern of motion of objects [18], which is usually used for motion detection and object segmentation. In our research, we use optical flow to find the displacement vector of feature points. Figure 12 shows the corresponding feature points in two images. Optical flow has three basic assumptions. The first assumption is brightness consistency, which means that the brightness of a small region remains the same. The second assumption is the spatial coherence, which means the neighbors of a feature point usually have similar motions as the feature. The third assumption is temporal persistence, which means that the motion of a feature point should change gradually over time.

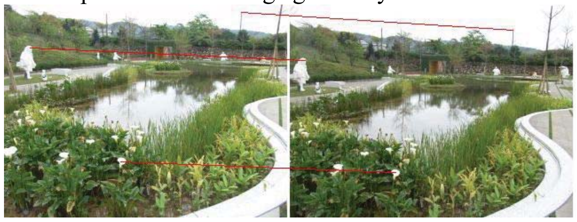
Figure 12: Feature point correspondence in two images.

Let $I(x, y, t)$ be the pixel value at location (x, y) at time t. From the assumptions, the pixel value would be $I(x + u, y + v, t + 1)$ with displacement (u, v) at time $t + 1$. Vector (u, v) is also called the optical flow of (x, y). Then we have $I(x, y, t) = I(x + u, y + v, t + 1)$. To find the best (u, v), we select a region around the pixel (for example, a window of size 10 x 10 pixels) and try to minimize the sum of the square error as below:

$$E(u,v) = \sum_R (I(x+u, y+v, t+1) - I(x,y,t))^2$$

We use Taylor series to expand the first order derivatives of $I(x + u, y + v, t + 1)$ as

$$I(x+u,y+v,t+1) = I(x,y,t) + I_x(x,y,t)u + I_y(x,y,t)v + I_t(x,y,t)$$

Replace the expansion in the original equation and we would have $E(u,v) = \sum_R (I_x u + I_y v + I_t)^2$.

Equation $I_x u + I_y v + I_t = 0$ is also called the optical flow constraint equation. To find the extreme value, the two equations below should be satisfied.

$$\frac{dE}{du} = \sum_R 2(I_x u + I_y v + I_t) I_x = 0$$

$$\frac{dE}{dv} = \sum_R 2(I_x u + I_y v + I_t) I_y = 0$$

Finally we have the linear equation:

$$\left[\sum_R I_x^2\right]u + \left[\sum_R I_x I_y\right]v = -\sum_R I_x I_t$$

$$\left[\sum_R I_x I_y\right]u + \left[\sum_R I_y^2\right]v = -\sum_R I_y I_t$$

By solving the linear equation, we can obtain optical flow vector (u, v) for (x, y). We use the concept of Lucas and Kanade [8] to iteratively solve the (u, v). It is similar to Newton's method.

1. Choose a (u, v) arbitrarily, and shift the (x, y) to ($x + u$, $y + v$) and calculate the relative I_x and I_y.
2. Solve the new (u', v') and update (u, v) to ($u + u'$, $v + v'$).
3. Repeat Step 1 until (u', v') converges.

To have fast feature point tracking, we build the pyramid images of the current and previous frames with four levels. At each level we search the corresponding point in a window size 10 by 10 pixels and stop the search to get into next level with accuracy of 0.01 pixels.

4. SMILE DETECTION SCHEME

We have proposed a fast and generally low misdetection and low false alarm video-based method of smile detector. We have 11.5% smile misdetection rate and 12.04% false alarm rate on the FGNET database. Our smile detect algorithm is as follows:

1. Detect the first human face in the first image frame and locate the twenty standard facial features position.
2. In every image frame, use optical flow to track the position of left mouth corner and right mouth corner with accuracy of 0.01 pixels and update the standard facial feature position by face tracking and detection.
3. If x direction distance between the tracked left mouth corner and right mouth corner is larger than the standard distance plus a threshold T_{smile}, then we claim a smile detected.
4. Repeat from Step 2 to Step 3.

In the smile detector application, we strongly consider that x direction distance between the right mouth corner and left mouth corner plays an important role in the human smile action. We do not consider y direction displacement. Since the user can have little up or down head rotation and that will falsely alarm our detector. How to decide our T_{smile} threshold? As shown in Table 1, we have mean distance 29.98 pixels between left mouth corner and right mouth corner and their standard deviation value 2.49 and 2.99 pixels. Let D_{mean} be 29.98 pixels and D_{std} be 2.49 + 2.99 = 5.48 pixels. In each frame, let D_x be x distance between two mouth corners. If D_x is greater than $D_{mean} + T_{smile}$, then it is a smile, otherwise, it is not. With large T_{smile}, we have high misdetection rate and low false alarm rate, and low misdetection rate and high false alarm rate with small T_{smile}. We run different T_{smile} in FGNET database and results are shown in Table 2. We use 0.55 D_{std} = 3.014 pixels as our standard T_{smile} to have 11.5% misdetection rate and 12.04% false alarm rate.

Threshold	Misdetection Rate	False Alarm Rate
0.4**Dstd*	6.66%	19.73%
0.5**Dstd*	9.25%	14.04%
0.55**Dstd*	11.50%	12.04%
0.6**Dstd*	13.01%	8.71%
0.7**Dstd*	18.82%	4.24%
0.8**Dstd*	25.71%	2.30%

Table 2 Misdetection rate and false alarm rate with different thresholds.

5. REAL-TIME SMILE DETECTION

It is important to note that the feature tracking will accumulate errors as time goes by and that would lead to misdetection or false alarm results. Since we do not want users to take an initial neutral photograph every few seconds, which would be annoying and unrealistic. Moreover, it is difficult to identify the timing to refine feature position. If the user is performing some facial expression when we refine the feature location, it would lead us to a wrong point to track. Here we propose a method to automatically refine for real-time usage. Section 5.1 would describe our algorithm and Section 5.2 would show some experiments.

5.1. Feature Refinement

From our very first image, we have user's face images with neutral facial expression. We would build user's mouth pattern grey image at that time. The mouth rectangle is surrounded by four feature points: right mouth corner, center point of upper lip, left mouth corner, center point of lower lip. Actually we would expand the rectangle wider and higher to one standard deviation in each direction. Figure 13 shows the user's face and Figure 14 shows the mouth pattern image. For each following image, we use normalized cross correlation (NCC) block matching method to calculate the best matching block to the pattern image around the new mouth region and calculate their cross correlation value. The NCC equation is:

$$C = \frac{\sum_{(x,y)\in R,(u,v)\in R'} (f(x,y)-\overline{f})(g(u,v)-\overline{g})}{\sqrt{\sum_{(x,y)\in R}(f(x,y)-\overline{f})^2}\sqrt{\sum_{(u,v)\in R'}(g(u,v)-\overline{g})^2}}$$

The equation shows the cross correlation between two blocks R and R'. If the correlation value is larger

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.