CS229 Project Final Report

Detection of a Single Hand Shape in the Foreground of Still Images

Toan Tran (dtoan@stanford.edu)

1. Introduction

This paper is about an image detection system that can detect the presence of human hands organized in a heart shape in still images. For example, people very often create heart shapes using their hand, such as the pictures in *Figure 1*. The input to the detector is a still image. The output of the detector consists of two pieces of data:

- A binary label indicating if there is a heart hand shape in the image (y = 1) or not (y = 0)
- The contour of the heart shape that identifies the area enclosed by the hands.

This project is part of a bigger project we are undertaking called Hand-Augmented Photography for Google Glass and other hands-free wearable cameras. Hands-free wearable cameras, such as Google Glass, allow the users to take pictures by voice command and thus leaving the users' hands free. The users can use their hands as an additional interface to the camera, for example:

- **Hand-as-Aperture**. While taking pictures, the users create a hand shape and put it in front of the device camera as an aperture to customize the picture frame and what part of the scene will be enclosed by the hand figure. The resulted image is cropped to the shape of the hand shape.
- **Hand-as-Zoom**. The users can also create a rectangular frame with their hands, and then enlarge or shrink the rectangular frame to indicate zoom out and zoom in, respectively. This is very useful as Google Glass camera does not have an optical zoom component.
- **Document Scanning**: The users can put their hands alongside the boundary of a document while taking its picture. Then, the document will be cropped out.
- **Hand-Gesture Language**: Google Glass can detect and understand the meaning of different hand shapes, and thus can assist the users in understanding the emotion and implication of the subject through their hand gestures.

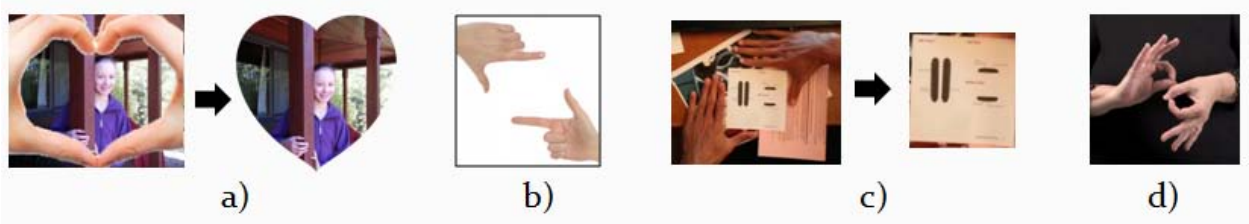

Figure 1. Images with hands organized in a heart shape. (a) Hand as aperture. (b) Hand as zoom. (c) Document scanning. (d) Hand-gesture language.

2. Training and Testing Data

Training Image:

- 3000 positive images: each contains a heart hand shape
- 3000 negative images: images that do not contain any heart hand shape

Figure 2. (First row) positive images. (Second row) negative images.

Testing Image:

- 2300 positive images
- 2300 negative images

3. Hand-Shape Detector using a Adaboost Cascade of Haar-Like Features

This algorithm adopts Viola-Jones object detection method [1]. The algorithm is based on Haar-like features, Adaboost algorithm for classifier learning, and a cascade of fast and simple classifiers to build an efficient strong classifier.

Haar-like Features:

A Haar-like feature is window that is divided into sub regions. *Figure 3*. shows Haar-like features used in this project.

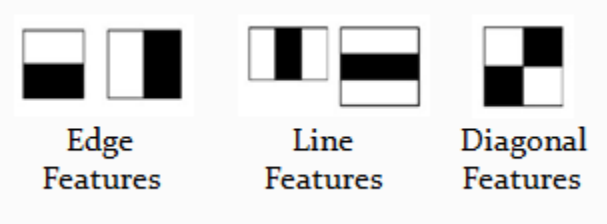

Figure 3. Haar-like features.

Haar-like features are used to detect features present in images. For example, edge Haar-like features detect edges in images as follows.

- The Haar-like feature window is scanned over the image.
- The value of the feature is computed by the intensity difference between the sum of pixels in the white area and the sum of pixels in the black area.

$$f = \sum pixels\ in\ white\ area - \sum pixels\ in\ black\ area$$

- An edge is detected if the f value is greater (or smaller) than some threshold θ_f.

A single-feature classifier *h(x)* for an image *x* is given by:

$$h_f(x) = \begin{cases} 1, & p_f f(x) > p_f \theta_f \\ 0, & otherwise \end{cases} \qquad \text{(eq. 1)}$$

where p_f is a binary indicating the direction of the inequality sign. Threshold θ_f is a parameter that will be learned from the training images.

Simple Haar-like features can be combined to produce more complex features, for example the following C-shape Haar-like feature in *Figure 4*.

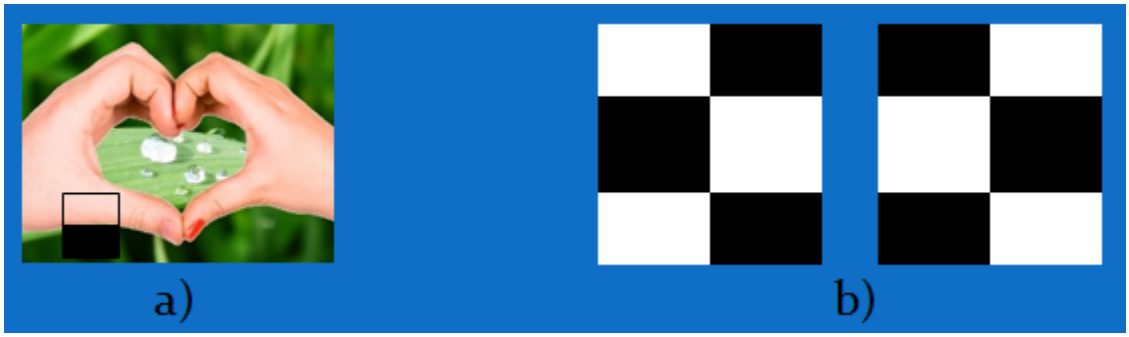

Figure 4. a) Find edges using edge Haar-like features. b) C-shape Haar-like features.

We can see that C-shape Haar-like features simulate the C-shape appearance of left hand and right hand when two hands are making the heart shape. The training phase will determine what Haar-like features to be used together with the positions and sizes of each Haar-like features that allows it to classify positive and negative images. More specifically, for each Haar-like feature and a 32x24 image, there are about 180.000 possibilities of position and size of that feature when applied on the image. The training process will choose only about 100-1000 most important features that yields minimum classification error rate. *Figure 5*. shows an example of 15 most important Haar-like features that were selected by the training process. We can see that the combination of these features characterizes some curves similar to the way the hands are organized to make the heart shape.

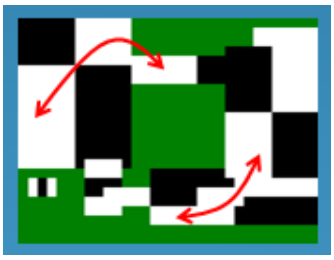

Figure 5. Fifteen Haar-like features are selected to form a classifier.

Adaboost Classifier Training Algorithm:

Adaboost will scan the training images many times, each time with a different size and position of Haar-like features, to find the best size and position of the feature and a threshold that best separate the positive and negative training images. Adaboost algorithm learns a classifier as follows [1].

- Start with uniform weights on training examples. Let $\{(x_i, y_i), ..., (x_n, y_n)\}$ denote training images where $y_i = 1$ or 0 is the label. The initial weight of a training image i is given by:
  - $w_i^1 = \frac{1}{2p}$, if $y_i = 1$, where p is the number of positive training images.
  - $w_i^1 = \frac{1}{2q}$, if $y_i = 0$, where q is the number of negative training images.
- For each iteration $t = 1,.., T$ (where T is the number of Haar-like features we want to incorporate into each classifier, for example $T = 15$ in *Figure 6*):
  - Normalize all the weights: $w_i^t = \frac{w_i^t}{\sum_{j=1}^{n} w_j^t}$
  - For each feature f (out of 180.000 possible features), train a classifier $h_f(x)$ (eq. 1). Then compute the classification error of this classifier as a weighted sum of incorrectly samples: $\epsilon^t = \sum_{j=1}^{n} w_j^t |h_f(i) - y_i|$
  - Choose the classifier $h^t(x)$ that has minimum classification error ϵ^t ($\epsilon^t > 0.5$)
  - Re-weight the training samples:

    - Reduce the weight of correctly classified samples: $w_i^{t+1} = w_i^t \frac{1-\epsilon^t}{\epsilon^t}$

    - Keep the weight of incorrectly classified samples unchanged.
- T best features are finally combined to create the classifier. The decision strength of each feature is weighted by the error rate it makes. If a feature makes less error, its decision (the label it assigns to a sample) is more likely to be the final label of the sample.

$$h(x) = \begin{cases} 1, & if \sum_{t=1}^{T} \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^{T} \alpha_t \\ 0, & otherwise \end{cases}$$

The re-weighting step reduces the weight of correctly samples while keeping the weight of incorrectly samples high. Let V denote the set of these incorrectly classified samples. In the next iteration, samples in V will contribute more to the error rate, forcing the Adaboost to select a feature that can classify correctly as many samples in V as possible.

Cascade Classifier:

Scanning a large number of best features, in the order of 100 or 1000, will take a long time. To quickly reject negative image windows as early as possible, the set of best features are divided into a number of cascade stages each of which contains 10-100 features. An image must pass all the stages to be classified positive. Failure at any of the stage will make the image rejected - classified negative. The cascade classifier is shown in *Figure 6*.

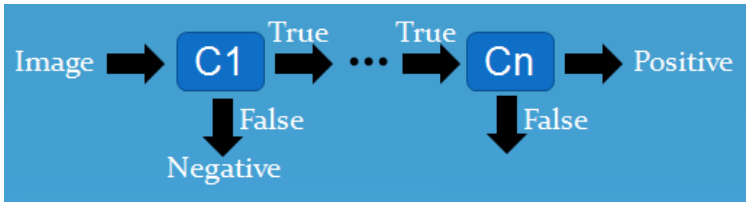

Figure 6. Cascade of classifiers

4. Hand Shape Contour Extraction

The hands in the foreground of an image characterize strong edges from the image background. A state-of-the-art contour detection can be used to extract the contour of the hand shape. This project uses the contour detector developed by Arbelaez *et al.* [2]. The hand shape contour extraction is as follows.

- Run the contour detector.
- Choose top 5 strongest contours from the set of all contours found by the detector.
- Remove opened and short contours.
- The biggest closed contour left is predicted the hand shape contour.

Figure 7. Original positive image and its hand shape contour

5. Results

The classifier is tested on 2300 positive images and 2300 negative images. The testing result is given in the following table.

- Testing Accuracy: 88%
- Precision: 83%
- Recall: 96%

	Actual Positive	Actual Negative
Predicted Positive	2215 (TP)	448 (FP)
Predicted Negative	85 (FN)	1852 (TN)

There are more revenues to improve the classifier's performance.

- **Clean training data**: some negative images contain objects that have heart shapes and/or C-curve shapes. We need to clean the negative images.
- **Add skin color**: skin color may be an effective additional indicator of hands because skin features a small range of color and texture [3].
- **Train by edges**: Since the hand shape is in the foreground of the image, its contour dominates most of the contours of other objects in the background. Hence, the positive training images should be converted into edges before being fed to the training phase.

6. Reference

[1] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," cvpr, vol. 1, pp.511, 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01) - Volume 1, 2001

[2] P. Arbelaez, M. Maire, Ch. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. X, 2011

[3] D.A. Forsythe and J. Ponce, "Computer Vision: A Modern Approach", Prentice Hall, 2 ed.