UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
CS543/ECE549: COMPUTER VISION

PROJECT PROGRESS REPORT

April 17, 2018

Chaohua Shang, Department of Electrical and Computer Engineering
Wenhua Lin, Department of Computer Science
Zhenye Na, Department of Industrial and Enterprise Systems Engineering

Statement

Update goal: according to the feedback from professor, we add one more goal to achieve for this project: face recognition.
We want to develop a project that can detect face from a picture and returns high precision face bounding box. After detecting faces, we want to implement a algorithm for face recognition. Given an image, we need to output if there is a match in the database and if there is a match we need to output the name for this person(identify this person). If time permits, we can implement face tracking algorithm for a video stream.

Collaboration strategy and member roles

- Member Roles
  - Viola Jones Face Detection: Zhenye Na Chaohua Shang
  - Face Recognition: Chaohua Shang, Wenhua Lin
  - Face Tracking: Zhenye Na, Wenhua Lin
- Collaboration strategy
  We uploaded our project to Github, so each member could update their implementation and review others' codes on time. For the report, we used Google doc where we can write together. We usually meet twice a week at library to discuss each member's progress, resolve the problem that we have, or arrange the new task for each member. Wechat is the tool for communication if any one need helps.

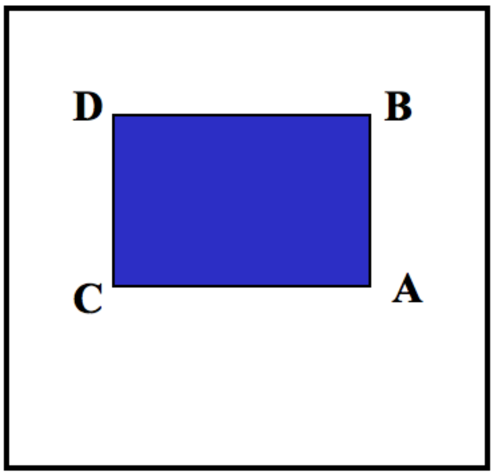
Proposed approach

1. **Viola Jones Face Detection**
   - **Integral image**

     The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel. The reason why we need integral image is that in the next step (Haar-Like Features), we will try to subtract two regeions to get the 'score' of the particular features in particular size.

It will reach high computational complexity. However with integral image, we can transform the polynomial problem to a $O(1)$ problem.

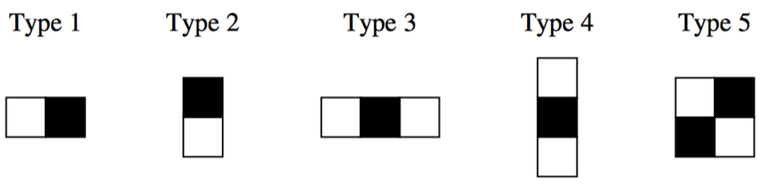
A brief explanation why we would like to use integral image is that suppose we try to sum over the blue region in the original image below.



If we use normal method we need compute four regions and then do some computation, we can get the result. It seems to do not require too much computational resources. However, imagine you have thousands of sample images and thousands of feature types and several feature size. This result in a $O(MNT)$ in computational complexity, which is really high. However, with integral images, what we need are just the four number in the integral image and do some simple (sum and subtraction) calculation.

- **Haar-Like Features**

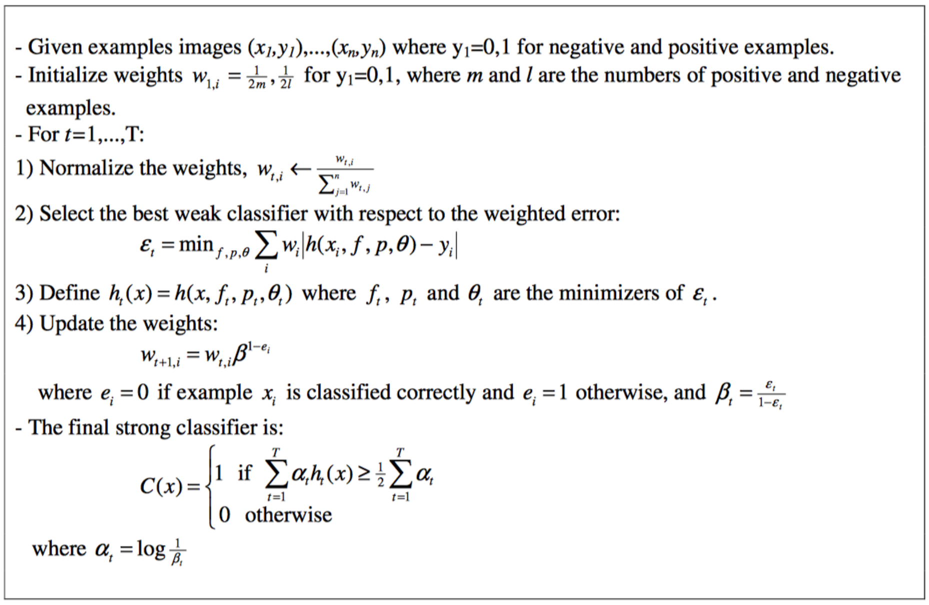
The Viola-Jones face detector analyzes a given sub-window using features consisting of two or more rectangles. The different types of features are shown in figure below.



Each feature results in a single value which is calculated by subtracting the sum of the white rectangle(s) from the sum of the black rectangle(s).
Viola-Jones have empirically found that a detector with a base resolution of 24×24 pixels gives satisfactory results. When allowing for all possible sizes and positions of the features in Figure 4 a total of approximately 160.000 different features can then be constructed. Thus, the amount of possible features vastly outnumbers the 576 pixels contained in the detector at base resolution. These features may seem overly simple to perform such an advanced task as face detection, but what the features lack in complexity they most certainly have in computational efficiency.

- **Adaboost**

As stated above there can be calculated approximately 160.000 feature values within a detector at base resolution. Among all these features some few are expected to give almost consistently high values when on top of a face. In order to find these features Viola-Jones use a modified version of the AdaBoost algorithm to do feature extraction. Below is the modified version of the AdaBoost algorithm.



- Given examples images $(x_1,y_1),\dots,(x_n,y_n)$ where y_1=0,1 for negative and positive examples.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for y_1=0,1, where m and l are the numbers of positive and negative examples.
- For t=1,...,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:

$$\varepsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$

3) Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t and θ_t are the minimizers of ε_t.
4) Update the weights:

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

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

- The final strong classifier is:

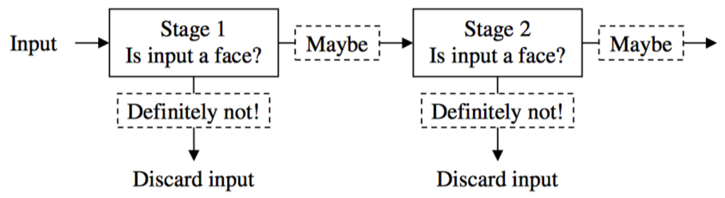
$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

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

We first initialize uniform weights for all positive and negative examples. Then we try to find the best feature which give the minimal error in every images and assign the feature which gives the smallest error to one of the weak classifiers in the strong classifier.

- **Cascaded classifier**

  In stead of finding faces, the algorithm should discard non-faces. This is another approach when we do face detection.



2. **Face Recognition**

- **Face detection**
  Use Viola-Jones face detector implemented in part 1 to output face image windows
- **Face alignment** [1]
  After we found the face image windows, we need to align the face so that we can extract features later.

Algorithm 3 Training of cascade and joint face detection and alignment.

```
Input: all training samples $\{\mathbf{x}_i\}$, class labels $\{y_i\}$
Input: ground truth shapes $\hat{\mathbf{S}}_i$ for positive samples, $y_i = 1$
Output: all weak learners $\{\mathcal{CR}_k^t\}$, classification thresholds $\{\theta_k^t\}$
set the initial face shapes $\mathbf{S}_i^0$ as random perturbations of the mean shapes in windows of $\mathbf{x}_i$
set all initial classification scores $f_i = 0$
for $t = 1$ to $T$ do
    for $k = 1$ to $K$ do
        for each training sample $i$ do
            compute its weight $w_i$ according to Eq. (6)
        end for
        select a point ($k$ mod $L$) for regression /*local learning in Section 4.1*/
        learn the structure of classification/regression tree $\mathcal{CR}_k^t$ as in Section 4.2
        for each tree leaf do
            set its classification score according to Eq. (7)
        end for
        for each training sample $i$ do
            update its classification score as $f_i = f_i + \mathcal{CR}_k^t(\mathbf{x}_i, \mathbf{S}_i^{t-1})$
        end for
        use all $\{f_i\}$ to set the bias $\theta_k^t$, according to a preset precision-recall condition
        remove samples whose $f_i < \theta_k^t$ from training set
        perform hard negative sample mining if negative samples are insufficient
    end for
    learn the shape increments of all leaves /* global learning in Section 4.1 */
    compute $\mathbf{S}_i^t$ for all samples according to Eq. (2) and (5)
end for
```

- **Feature extraction** [2][3]
  There are three approaches for feature extraction [4].
  - Holistic approach: eigenfaces, fisherfaces, SVM, NFL and independent-component analysis based on PCA
  - Feature-based approach: local feature like nose, and eyes are segmented and used as input data
  - Hybrid approach

  We properly choose the Fisherface algorithm used PCA for dimensionality reduction to find the vectors, since it has higher accuracy rate than eigenface algorithm.
- **Face verification and face identification**
  Compute the similarity and find the most similar one

3. **Face Tracking (maximum goal)**
   We can apply face detection for each frame, but this is computational expensive. We can implement a real-time face tracking using improved CamShift. The CamShift (Continuously adaptive mean SHIFT) is an improved MeanShift algorithm that adjusts the size of the search window on its own. When the user inputs the size and position of the initial search area, it repeatedly compares histograms to track the target object. The biggest difference from MeanShift is that by applying variable window size, CamShift easily tracks objects that change size. Below is the order that CamShift processes.

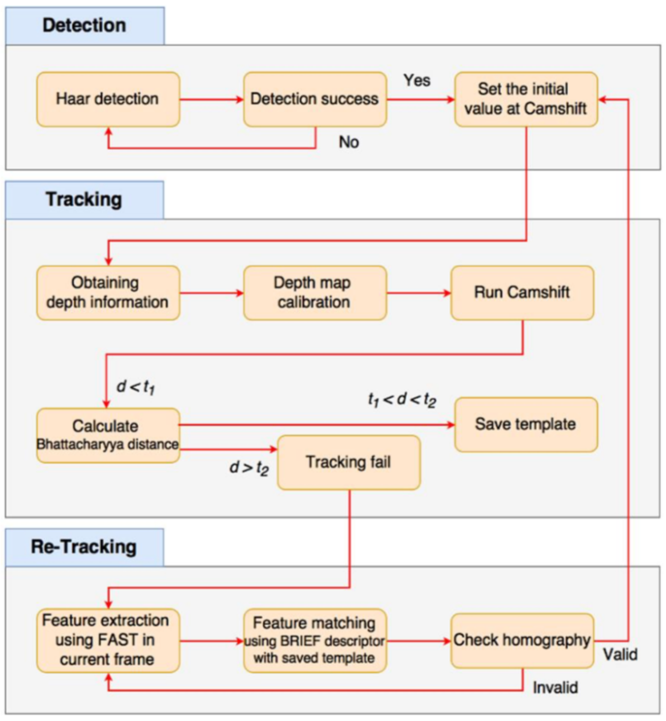
   - Set the size and initial position of the search window.
   - Calculate the probability distribution of color information and perform the MeanShift to find the center of the search window.
   - Reset the search window by using the center position and size of the color histogram.
   - Using the reset search window, repeatedly perform the MeanShift until the search window converges, or repeat Steps b-d as often as defined. The position and size of the search window is determined through the calculation of the zeroth, 1st, and 2nd moments of the color histogram within the window. The zeroth, 1st, and 2nd moment are obtained by Eqs. (1)-(3), respectively.

$$M_{00} = \sum_X \sum_y I(x,y) \quad (1)$$
$$M_{10} = \sum_X \sum_y xI(x,y),\ M_{01} = \sum_X \sum_y yI(x,y) \quad (2)$$
$$M_{20} = \sum_X \sum_y x^2I(x,y),\ M_{02} = \sum_X \sum_y y^2I(x,y) \quad (3)$$

Where I(x,y)represents the search window's pixel value at (x,y). Using the zeroth, 1st and 2nd moment, we can calculate the center position along with the following equation.

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

Below figure shows the diagram of this face tracking algorithm.


Detection
Haar detection
Detection success
Yes
No
Set the initial value at Camshift
Tracking
Obtaining depth information
Depth map calibration
Run Camshift
d < t1
t1 < d < t2
Calculate Bhattacharyya distance
Save template
d > t2
Tracking fail
Re-Tracking
Feature extraction using FAST in current frame
Feature matching using BRIEF descriptor with saved template
Check homography
Valid
Invalid

A detailed outline, pseudocode, or prose description. Be specific about how you plan to implement each step with references, pointers to external code, etc. One or more references are required at this stage. Below is part of references we need at this stage.

Face Recognition Algorithms
Real Time Simple Face-Tracking Algorithm based on Minimum Facial Features
Computer Vision Face Tracking For Use in a Perceptual User Interface

Initial results

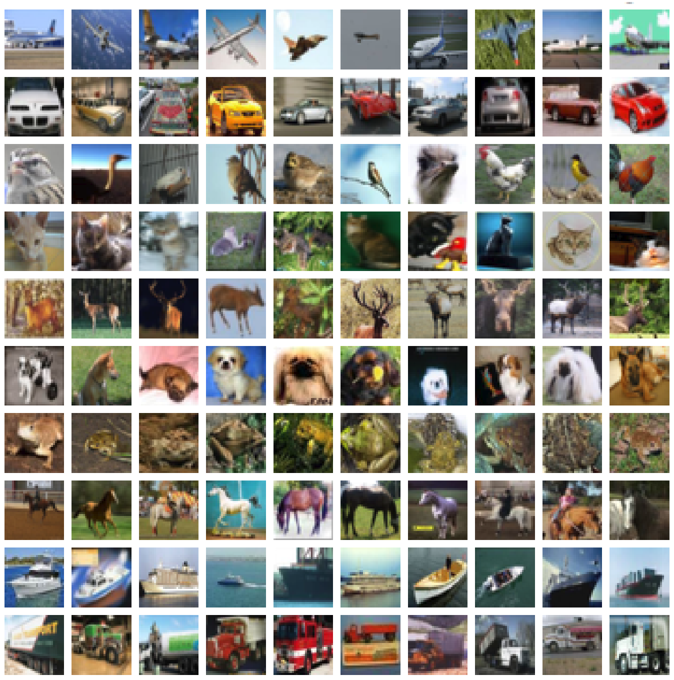
By now, we have completed most part of the first mission of this project - Viola Jones Face Detection. We initialize three approaches for data preprocessing (default, lab, hsv. see here for detailed information) and training/validation/test image process approach. We have already build a HaarFeature Class and figure out the skeleton of our modified AdaBoost Algorithm.

Data

To train the different stages of the cascaded classifier, Adaboost Algorithm needs to be fed with positive examples, which is images of faces. We are going to use LFW (Labeled Faces in the Wild) database, a database made and distributed by The University of Massachusetts as positive examples. This dataset contains more than 13,000 images of faces. For the training in cascade, negative examples are also required, so we use CIFAR10 as negative examples for Face Detection.

- For face detection, we use label 1 for images with human faces and label 0 for images with no face.
- For face recognition, we use label of the name of the person in the picture. There are 1680 of the people has more than two pictures in the dataset.

Type of image: jpg
URL: `http://vis-www.cs.umass.edu/lfw/`
Example:
LHS is LFW dataset (positive examples) and RHS is CIFAR10 dataset (negative examples)



Reference

1. Joint Cascade Face Detection and Alignment

2. Computer Vision Face Tracking For Use in a Perceptual User Interface

3. Face Recognition Using Principal Component Analysis Method

4. Facial feature extraction for face recognition: a review