Facial Keypoints Detection*

Yue Wang[†] and Yang Song[‡]
Stanford University

***Abstract*—In this project, we are given a list of 96×96-pixel 8-bit graylevel images with their corresponding $(\mathbf{x}, \mathbf{y})$ coordinates of 15 facial keypoints. We first adopt hold-out cross validation to randomly split the data set into a training set and a test set, so that we can develop our algorithm on the training set and assess its performance on the test set. Our algorithm first performs histogram stretching to enhance the image contrast by stretching the range of pixel intensity of each training image. Then, in order for noise reduction, we apply principal components analysis on the stretched images to obtain the eigenfaces. Using the resultant eigenfaces, we implement the mean patch searching algorithm with correlation scoring and mutual information scoring to predict the left- and right-eye centers for any query test facial images.**

I. Introduction

FACE recognition is one of the most significant branches in computer vision research. It aims to determine the locations and sizes of human faces on digital images, by detecting and extracting faces from the other surrounding objects, such as buildings, plants and other backgrounds. To develop a sophisticated face recognition algorithm, the most fundamental but by far the most important task is facial keypoints detection, that is, to find out the locations of specific keypoints on face images, which include left eyes, right eyes, noses, mouths and so forth. In this article, we explore this appealing yet challenging topic in depth.

The challege of facial keypoints detection is that the facial features may vary greatly from one image to another due to the difference of individuals' generic appearances. Besides, the facial features may also be significantly affected by other physical factors such as position, viewing angle, illumination condition, contrast, and even the psychological factors such as the emotion of individuals. Our objective is to develop a relatively accurate and efficient algorithm to automatically recognize where the facial keypoints are located on digital images.

A. Data

The idea and data set of this project come from the Kaggle online competition [10]. Our data set consists of a list of 7049 two-dimensional 8-bit graylevel training images with their corresponding (x, y) coordinates of the 15 facial keypoints as listed in Table I. (See Figure 1, which shows two sample images marked with those 15 keypoints.) Each image is represented by a 96×96 pixel matrix, whose entries are integers from 0 through $2^8 - 1 = 255$, characterizing the intensity of each of the $96 \times 96 = 9216$ pixels. Therefore, the given training set is a huge matrix of size 7049×31, where each row corresponds to one image, the first 30 columns give the (x, y) values of the 15 facial keypoints, and each entry in the last column is a long list of 9216 numbers representing the pixel matrix of each image melted row by row. We also note that in some examples, some of the target keypoint positions are misssing (encoded as missing entries in the `csv` file, i.e., with nothing between two commas).

TABLE I: The 15 facial keypoints

Left eye center	Right eye center
Left eye inner corner	Right eye inner corner
Left eye outer corner	Right eye outer corner
Left eyebrow inner end	Right eyebrow inner end
Left eyebrow outer end	Right eyebrow outer end
Mouth left corner	Mouth right corner
Mouth center top lip	Mouth center bottom lip
Nose tip	

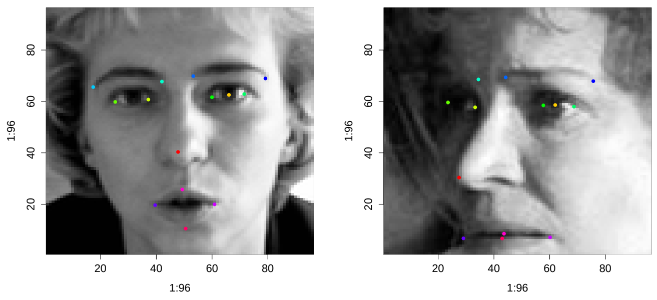

Fig. 1: Two sample images marked with the 15 keypoints as listed in Table I.

B. Goal

This is a supervised learning problem, where we are given the above-mentioned relatively large data set of 7049 training examples, consisting of their image pixel matrices and their corresponding true positions of the 15 facial keypoints. We would like to develop some efficient hypotheses that can provide relatively accurate prediction on positions of the facial keypoints for other query images. Due to time limitation, we would only concentrate on the positions of left and right eye centers. More specifically, our goal is that our resultant hypotheses are able to predict `left_eye_center` and `right_eye_center` given any query facial image represented by a 96×96 pixel matrix as input.

*Advisors: Prof. Andrew Ng and CS229 Course Staff Team.
[†]Email: `yuew@stanford.edu`
[‡]Email: `sy0716@stanford.edu`

II. Methodology

A. Preprocessing

Recall that we are given a set of 7049 examples (denoted as S). In order to evaluate the performance of our learning algorithm, we adopt *hold-out cross validation* and randomly split S into S_{train} and S_{test}, which contain 80% (≈ 5639 examples) and 20% (≈ 1410 examples) of the data, respectively. We will train our learning algorithm on S_{train}, and then test the resultant hypotheses on S_{test} by calculating the root mean sqaured error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{1}$$

where y_i is the true value and $\hat{y}_i$ is the predicted value. This RMSE on the hold-out test set is expected to give a reasonably good estimate of the generalization error of our learning algorithm, and our objective is to minimize this RMSE.

B. Histogram Stretching

As mentioned in the beginning, the images may be corrupted by random variations in contrast, illumination and other imaging conditions. [2] suggests that we deal with these problems in the early stages of vision processing. In this subsection, we resolve the problem of poor contrast; in this next subsection, we tackle the other noises.

Mathematically, poor contrast means the pixel intensity values of a certain image lie within only a small range. This is fairly common in real-life images, including our data set. In this subsection, we introduce *histogram stretching*, a simple but effective image enhancement technique that attempts to improve the contrast in an image by stretching the range of its pixel intensity to span a desired range of values, which is often the full range of pixel intensity allowed. For our 8-bit graylevel images, the full range is from $[0, 255]$.

Given an image (represented by its pixel matrix), the histogram stretching algorithm first scans the entries to find the minimum and maximum pixel intensity values currently present in the image, denoted as a and b, respectively. Thus, the current pixel intensity range is $[a, b]$. Let $[l, u]$ be the desired range that we would like to span (for our 8-bit graylevel images, $l = 0$ and $u = 255$). Then, each pixel intensity value p in the original range $[a, b]$ is mapped to a new value p' in the desired range $[l, u]$ in a such way that the proportion of p' in $[l, u]$ is the same as the proportion of p in $[a, b]$. That is,

$$\frac{p-a}{b-a} = \frac{p'-l}{u-l},$$

implying that

$$\begin{aligned} p' &= \left(\frac{u-l}{b-a}\right)(p-a) + l \\ &= \frac{u-l}{b-a}p + \frac{bl-au}{b-a}. \end{aligned} \tag{2}$$

The advantage of histogram stretching is that it enhances the image contrast without distorting the relative graylevel intensity values too significantly. However, selecting the current range limits a and b as the minimum and maximum pixel intensity values is very sensitive to outlying pixels if any, and this could lead to very unrepresentative scaling (see [7]). As remedy, we adopt a more robust version of histogram stretching (Eq. (2)), which selects a and b as the 5th and 95th percentile, respectively, in the histogram of the original pixel intensities. This prevents ouliers from affecting the scaling too much.

Now, back to our facial keypoints detection problem, we perform this *modified* histogram stretching algorithm on each of the 5639 training images and the 1410 test images with the desired range $[l, u] = [0, 255]$, yielding a total of 7049 *stretched images* with much better contrast. For each training and test stretched image, its true facial keypoints positions are unchanged.

Figure 2 shows three typical pairs of images before and after performing the modified histogram stretching algorithm. We can see that the stretched images have much better contrast effect.

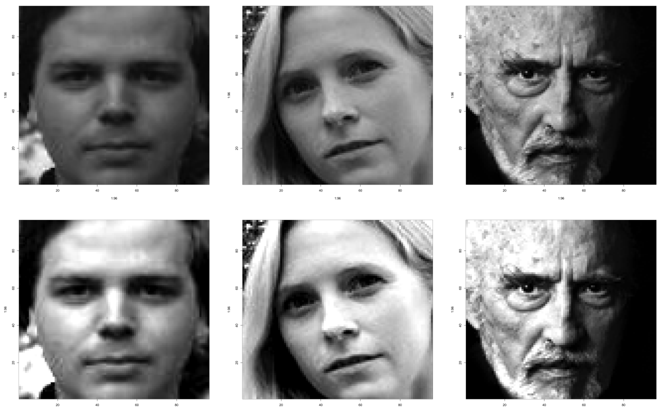

Fig. 2: Upper: Three original images. Lower: Three corresponding stretched images after performing the modified histogram stretching algorithm.

C. Principal Components Analysis

Aside from contrast, the images may also be affected by the other external factors such as illumination and imaging conditions. In order to reduce these noises, to reduce the complexity of the hypothesis class to help avoid overfitting, as well as to enhance computational efficiency, we employ one more unsupervised data exploration tool – *principal components analysis* (PCA) – before performing the supervised learning algorithm.

Since we are interested in finding principal components that can retain the systematic variations between faces while reducing the noises, we first write the design matrix $\mathbf{X}$ as

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ X_1 & X_2 & \cdots & X_{5639} \\ | & | & & | \end{bmatrix},$$

where each $X_i \in \mathbb{R}^{9216\times 1}$ ($\forall i = 1, \ldots, 5639$) corresponds to the i^{th} stretched training image pixel matrix melted row by row into a column vector. Performing PCA on $\mathbf{X}$ gives the principal components (called *eigenfaces* in computer vision; see [5])

$$\begin{aligned}
\mathrm{PC}_1 &= \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{5639,1}X_{5639}\\
\mathrm{PC}_2 &= \phi_{12}X_1 + \phi_{22}X_2 + \cdots + \phi_{5639,2}X_{5639}\\
&\cdots\cdots,
\end{aligned}$$

where $\phi_1 = (\phi_{11}\ \phi_{21}\ \ldots\ \phi_{5639,1})^T \in \mathbb{R}^{5639\times 1}$ with $\|\phi_1\|_2^2 = 1$, $\phi_2 = (\phi_{12}\ \phi_{22}\ \ldots\ \phi_{5639,2})^T \in \mathbb{R}^{5639\times 1}$ with $\|\phi_2\|_2^2 = 1$, …, are the loading vectors of PC_1, PC_2, …, respectively. The theory of PCA states that the loading vectors ϕ_1, ϕ_2, … of the principal components PC_1, PC_2, … are just the ordered sequence of the normalized eigenvectors of the matrix $\Sigma = \mathbf{X}^T\mathbf{X}$, and the variances of the PCs are the corresponding ordered eigenvalues.

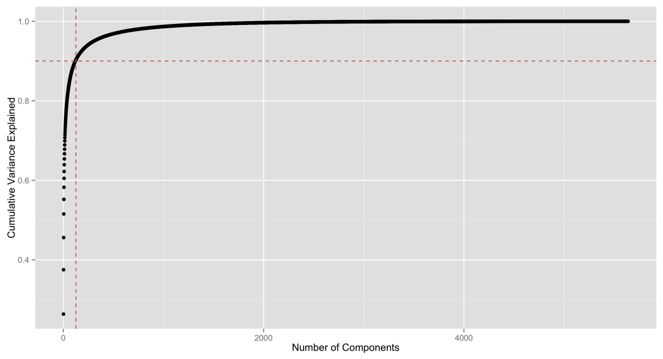

Fig. 3: Scree plot, depicting the cumulative proportion of variance explained (CPVE) by each principal component.

Figure 3 is the resultant scree plot, which provides a common way for us to decide on the number of principal components required to explain a sizable amount of variation in the data. One typically eyeballs the scree plot and looks for an *elbow* point at which the cumulative proportion of variance explained (CPVE) by the subsequent principal component significantly slows down increasing. By inspecting Figure 3, we find that the elbow point occurs at 125 principal components, which can explain more than 90% of the total variance in the training data. Therefore, we expect that the first 125 principal component stretched images (i.e., eigenfaces[1]) PC_1, …, PC_{125} can approximately represent the original 5639 stretched images X_1, …, X_{5639} with noises further reduced, and **will use them as our training images in supervised learning later**. For each principal component stretched image PC_j ($j = 1, \ldots, 125$), we obtain its "true" facial keypoints positions by computing the weighted average of the corresponding true positions in the original (stretched or not, since their true keypoints positions are the same) images X_i ($i = 1, \ldots, 5639$) with weight equal to ϕ_{ij}^2 (recall that $\|\phi_j\|_2^2 = \sum_{i=1}^{5639}\phi_{ij}^2 = 1$).

Figure 4 shows the first 6 eigenfaces extracted from the 5639 training stretched images.

[1]More precisely, they should be called *eigen-stretched-faces*. But we will be consistent with literature and just call them eigenfaces.

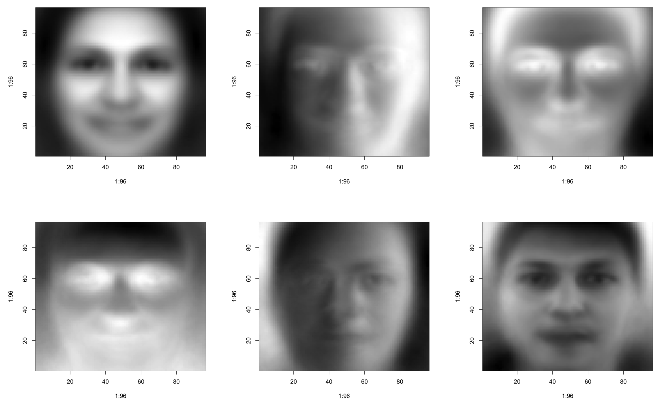

Fig. 4: The first 6 eigenfaces; row by row, PC_1 through PC_6.

D. Mean Patch Searching[2]

Finally, in this subsection, we perform supervised learning. We adopt a relatively simple but effective algorithm for facial keypoints detection, called *mean patch searching*. We take the facial keypoint `left_eye_center` as an example to illustrate this algorithm.

Given a set of n training images (in our problem, $n = 125$ eigenfaces after performing modified histogram stretching and PCA) with their true (x, y) coordinates of `left_eye_center`. From each of them we extract `patch_size` number of pixels in each direction around the true keypoint (x, y), yielding a set of n patches[3] of size $(2 * \texttt{patch\_size} + 1) \times (2 * \texttt{patch\_size} + 1)$. We then compute the *mean patch* as the entry-wise mean of the n patches obtained above. Mathematically, let $P_i \in \mathbb{R}^{(2*\texttt{patch\_size}+1)\times(2*\texttt{patch\_size}+1)}$ be the pixel matrix of the i^{th} patch ($i = 1, \ldots, n$), then the pixel matrix of the mean patch $\bar{P}$ is given by

$$\bar{P} = \frac{1}{n}\sum_{i=1}^{n} P_i.$$

This mean patch is expected to be a typical representation of the respective facial keypoints (i.e., `left_eye_center` in our illustrative example). Moreover, we can obtain the mean `left_eye_center` position (denoted as $(\texttt{mean\_left\_eye\_center\_x}, \texttt{mean\_left\_eye\_center\_y})$) across the n training images by computing the average of the n coordinates of $(\texttt{left\_eye\_center\_x}, \texttt{left\_eye\_center\_y})$.

Now, given a new test facial image (represented by a 96×96 pixel matrix), we search in a certain region of *candidate pixels* around $(\texttt{mean\_left\_eye\_center\_x}, \texttt{mean\_left\_eye\_center\_y})$ for our prediction on its `left_eye_center` position. A relatively simple but effective searching scheme is *square searching*. Denoting `search_size` (a tuning parameter) as the maximum search step we move in both directions starting from $(\texttt{mean\_left\_eye\_center\_x}, \texttt{mean\_left\_eye\_center\_y})$,

[2]This algorithm is inspired by [10], and we implement it with some modifications.

[3]If any patch is out of boundary, we just discard it. So more precisely, the number of patches is less than or equal to n. But for the purose of illustration here, we will assume we have obtained n such patches.

we essentially search in a square region of width $(2 * \texttt{search\_size} + 1)$ centered at that point. Around each of those candidate pixels is a *candidate patch* of size $(2 * \texttt{patch\_size} + 1) \times (2 * \texttt{patch\_size} + 1)$. We investigate those surrounding candidate patches to see which one has the highest relation with the mean patch $\bar{P}$ found above.

To measure the relation of two patches, we first melt each of the two pixel matrices row by row into a column vector and then calculate their relational score. We use two scorings: (i) correlation; (ii) mutual information (MI).

Finally, we select the candidate pixel whose surrounding candidate patch produces the highest score.

Note that `search_size` is a tuning parameter. This means we can adjust `search_size` in order for our model to produce the smallest RMSE (Eq. (1)) on the hold-out test set S_{test}.

III. Results and Discussion

Table II lists the training and hold-out test errors for predicting `left_eye_center` using histogram stretching plus PCA plus mean patch searching with correlation scoring and MI scoring, where the uning parameter `search_size` changing from 0 to 5. We complie them into the training and test error curves, as shown in Figure 5.

TABLE II: Training and test error rates for predicting `left_eye_center`.

`search_size`	Scoring	**Training error**	**Test error**
0	Correlation	3.4084	3.4282
	MI	3.4084	3.4282
1	Correlation	3.2029	3.2322
	MI	3.1827	3.2114
2	Correlation	3.1469	3.1763
	MI	3.0933	3.1187
3	Correlation	3.1307	3.2258
	MI	3.1067	3.1912
4	Correlation	3.1231	3.2792
	MI	3.0756	3.2427
5	Correlation	3.1199	3.4106
	MI	3.0747	3.3933

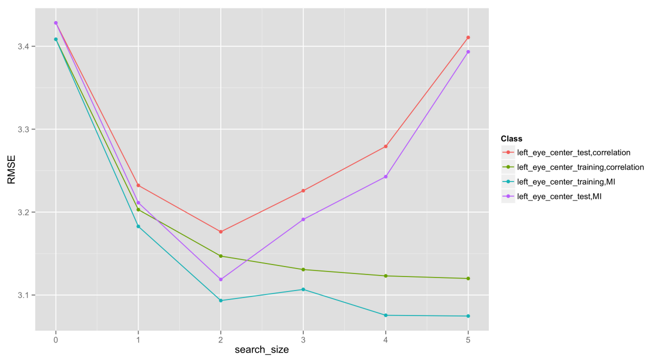

Fig. 5: Training and test error curves for predicting `left_eye_center`.

In parallel, we obtain the training and hold-out test errors for predicting `right_eye_center` in Table III, and depict them into Figure 6.

TABLE III: Training and test error rates for predicting `right_eye_center`.

`search_size`	Scoring	**Training error**	**Test error**
0	Correlation	3.0108	3.0544
	MI	3.0108	3.0544
1	Correlation	2.7842	2.8188
	MI	2.8586	2.8781
2	Correlation	2.7294	2.7485
	MI	2.8401	2.9327
3	Correlation	2.5823	2.6291
	MI	2.8513	3.0634
4	Correlation	2.5836	2.9329
	MI	2.8064	3.2946
5	Correlation	2.5788	3.0356
	MI	2.7956	3.6539

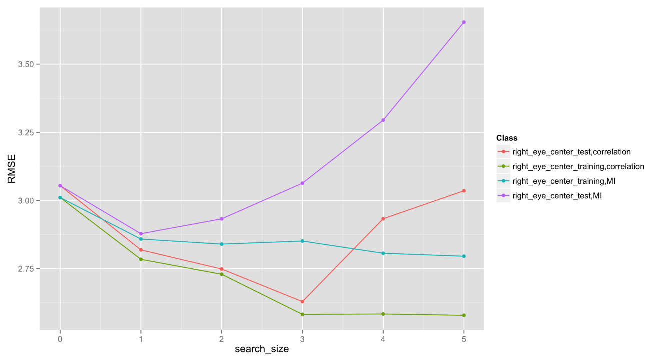

Fig. 6: Training and test error curves for predicting `right_eye_center`.

Several points are worth to be mentioned.

- For `left_eye_center` prediction, both of the scorings achieve the minimum hold-out test error rate at `search_size` = 2: 3.1187 for mutual information scoing and 3.1763 for correlation scoring. Moreover, mutual information scoring always outperforms correlation scoring in terms of both training error and test error. Therefore, for predicting `left_eye_center`, we would like to adopt histogram stretching plus PCA plus mean patch searching with `search_size` = 2 and mutual information scoring as our final hypothesis.
- For `right_eye_center` prediction, mutual information scoring attains its minimum hold-out test error rate at `search_size` = 1 (2.8781) whereas correlation scoring does at `search_size` = 3 (2.6291). Moreover, correlation scoring always outperforms mutual information scoring in terms of both training error and test error. Therefore, for predicting `right_eye_center`, we would like to adopt histogram stretching plus PCA plus mean patch searching with `search_size` = 3 and correlation scoring as our final hypothesis.
- Note that as `search_size` increases, the model flexibility (i.e., model complexity) increases. In both Figures 5 and 6, we observe a striking phenomenon that as model flexibility increases, the four training error curves monotonically decrease, while the four test error curves

all exhibit U-shape. This is a fundamental property in machine learning that holds regardless of the particular data set at hand and regardless of the learning algorithm being used. When the test MSE deviates vastly from the corresponding training MSE, the model is overfitting the data. This is the case when we choose a fairly large `search_size`, since the resultant model would be working too hard to find patterns in the training data and may be picking up some patterns that are just caused by random chance rather than by true properties of the unknown underlying mechnism.

- Another interesting observation is that for each of `left_eye_center` and `right_eye_center`, the hold-out test error rates for different scorings converge to each other at `search_size` $= 0$. This is also true for the corresponding training error rates. The reason is that at `search_size` $= 0$, we are just predicting each facial keypoint as the mean of the corresponding keypoint positions of the training stretched images, and thus the scoring does not matter.

IV. Conclusion

We first summarize our whole algorithm in the following flowchart (Figure 7).

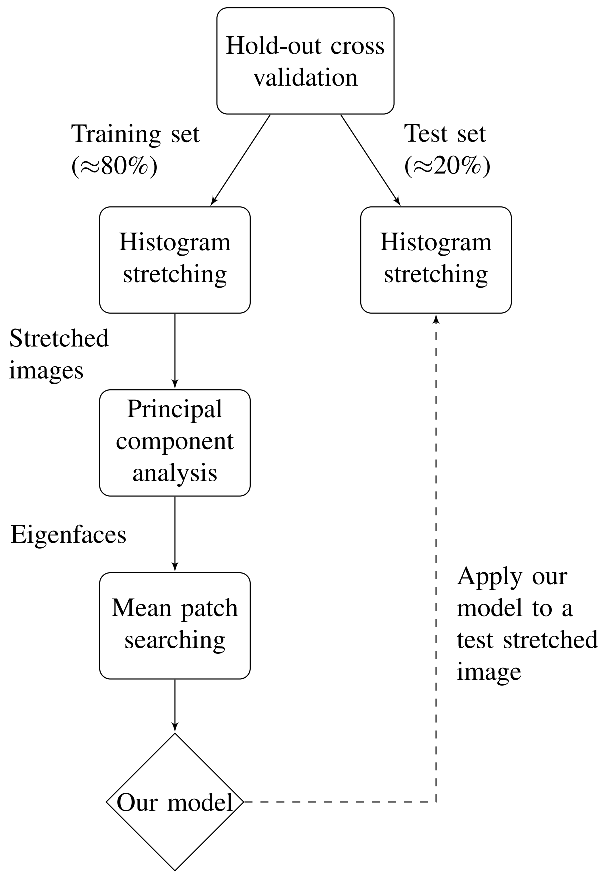

Fig. 7: Flowchart of our algorithm for facial keypoints detection.

As discussed in the previous section, we conclude our final hypotheses for predicting the locations of `left_eye_center` and `right_eye_center` as follow.

- `left_eye_center`:
  Histogram stretching → PCA → Mean patch searching with `search_size`$=2$ and MI scoring.
- `right_eye_center`:
  Histogram stretching → PCA → Mean patch searching with `search_size`$=3$ and correlation scoring.

The minimum hold-out test MSE for predicting `left_eye_center` is 3.1187, and the minimum hold-out test MSE for predicting `right_eye_center` is 2.6291. Thus, our best average test MSE for predicting both facial keypoints is $\sqrt{\frac{(3.1187)^2+(2.6291)^2}{2}} = 2.8843$. This result would make us rank 7 out of the 36 submissions in the outstanding public leaderboard of Kaggle.

V. Future Work

Due to time limitation, this term project has not exhaustively tried many different methods in the arsenal of machine learning. If this project could be continued in the future, in order to compensate the unequally distributed illuminations and other imaging details, we could perform several nonlinear image enhancement and correction approaches such as homomorphic filtering to normalize the brightness across an image and increases contrast. In addition, we could implement template matching to the image to search possible location of eye pair.. Moreover, we will also try locally linear embedding and other algorithms.

Acknowledgment

We would like to express our sincere gratitude to Prof. Andrew Ng and the CS229 Course Staff Team, who have given us insightful guidance and inspiring advice throughout this project.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman (2011). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition.
[2] R. Jain, R. Kasturi, and B.G. Schunck (1995). *Machine Vision*. McGraw-Hill, Inc.
[3] G. James, D. Witten, T. Hastie, and R. Tibshirani (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.
[4] A. Ng (2014). Lecture Notes for CS229 - Machine Learning. Stanford University.
[5] L. Sirovich and M. Kirby (1987). Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, vol. 4(3), pp. 519-524.
[6] Q. Wang and J. Yang (2006). Eye Location and Eye State Detection in Facial Images with Unconstrained Background. *Journal of Information and Computing Science*, vol. 1(5), pp. 284-289.
[7] Contrast Stretching.
`http : //homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm`.
[8] Eigenfaces for Dummies.
`http : //jmcspot.com/Eigenface/Default`.
[9] Eigenface Tutorial.
`http : //www.pages.drexel.edu/ sis26/Eigenface%20Tutorial.htm`.
[10] Kaggle Online Competetion: Facial Keypoints Detection.
`https : //www.kaggle.com/c/facial-keypoints-detection`.
[11] Reconstructing PCA Matrix.
`http : //www.r-bloggers.com/reconstructing-principal-component-analysis`