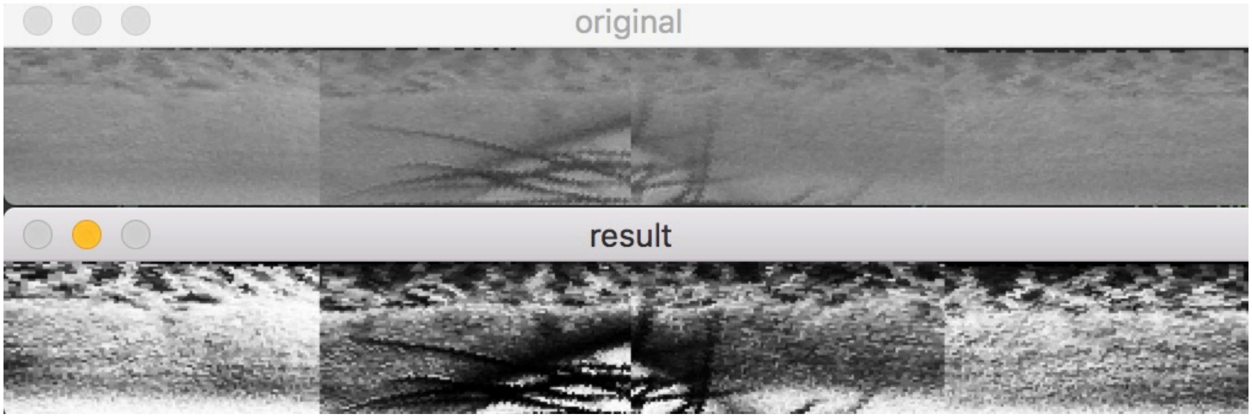
Explanation for each file:

IrisLocalization.py:
getimagecenter(): get the center of given image
getcenterregion(): get a 120 * 120 area square around the center
getimagecentroid(): get the centroid based on contour detection
IrisLocalization(): use the functions above to find the center of pupil and iris(small circle and big circle). Here I restricted two different areas to tell big and small circles.

IrisNormalization.py:
getcirclepointbyangle(): define the position of a given angle in a given circle
IrisNormalization(): use the big_circle(iris) and small_circle(pupil) defined in iris localization to transform the coordinate system. Here I ignored the part of points in upper 45 degree and lower 45 degree, where the eyelash and eye lid mostly appears. That is the reason why it looks like splited parts. Here I unfold the circle as a 64 X 512 picture.

ImageEnhancement.py:
get_item(): given x, y and a image, this function could be used to find the color value at point(x, y) in the image. Here x and y do not need to be integers.
getw():
bicubicinterpolation(): expand the given image(this would influence the speed a lot thus it is not finally used in enhancement)
ImageEnhancement(): Here I used cv2.equalizeHist() to enhance the image by increasing contrast.

Here comes the example result for this function.



FeatureExtraction.py:
getkernal(): Here I define this function to calculate the corresponding G according to the equations (3)
FeatureExtraction(): Here I selected two groups of theta (3, 1.5) and (4.5, 1.5) as described in Ma's paper, then use cv2.filter2D(spatial filter) to process the enhanced image and extracted features with loop. Then use loop to calculate the feature values for each 8 * 8 block to form a 1D feature vector.

IrisMatching.py:
IrisMatching(): In training set for each iris I use this function to define acceptable distance under 3 different kinds of definition for distance.
IrisMatchingDist(): Similar from the one above, calculate the distance for each test one.
In this file I calculated the distance for each test sample from training one and store them in a file for further use. Different from what mentioned in the document about hints for this project, I did not use LDA to reduce number of dimension until when I performance test in PerformanceEvaluation.py. I recommend not to run this file immediately as it save the training result in file and running this file takes a lot of time, even hours. An possible way to check how it runs is to change the name of file it may output to so that the data file will not be overwritten.

PerformanceEvaluation.py:
get_one_round_tmp(): Shuffle the training set and create a test set containing 108 iris classes.
get_one_round_te(): For each class in resulting test set, chose one sample from all available test samples of this class
calculate_one(): Use LDA-related function in package sklearn to reduce dimension and compute CRR, FMR and FNMR
calculateCRR(): I used the functions defined above to repeat the whole process 500 times and get the average for final answer. Here the default dimension used in LDA is 200, as mentioned in the hints. This could be changed.
calculateCOR(): use function roc_curve and auc from sklearn.metrics to plot ROC curves.

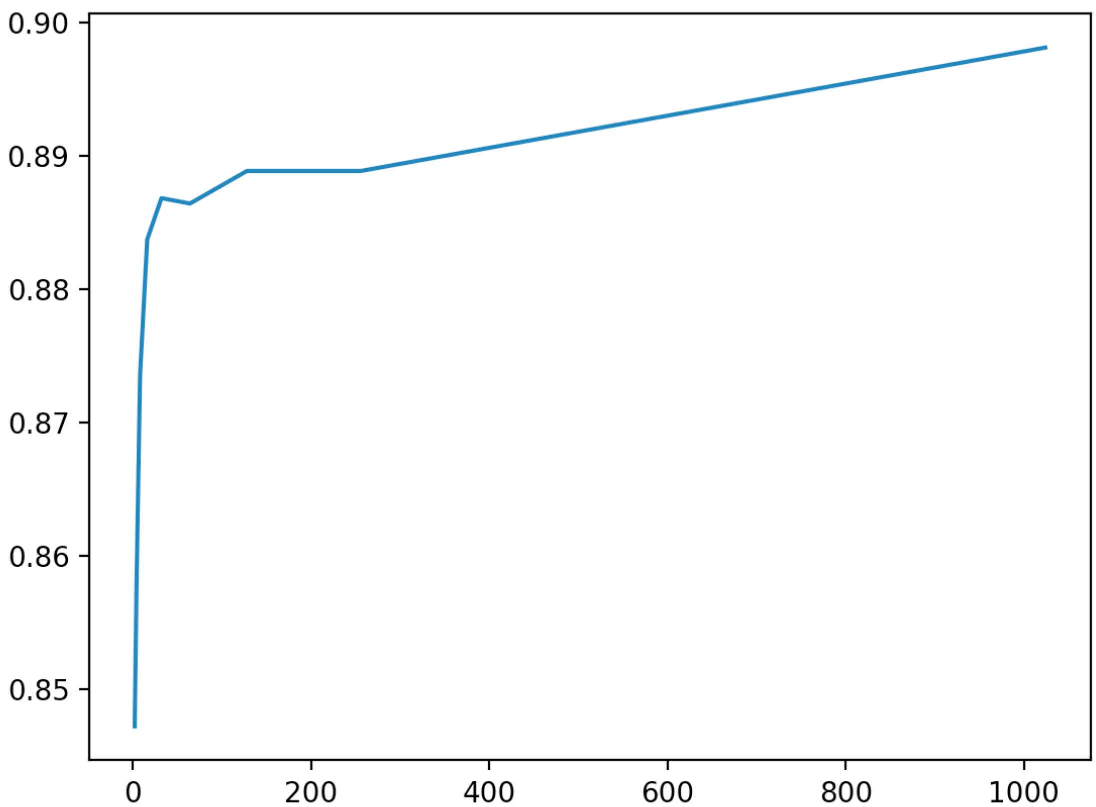
IrisRecognition.py:

According to the original document for this individual project, this is the main function using all files above.

Here comes the chart for recognition results using different similarity measure:

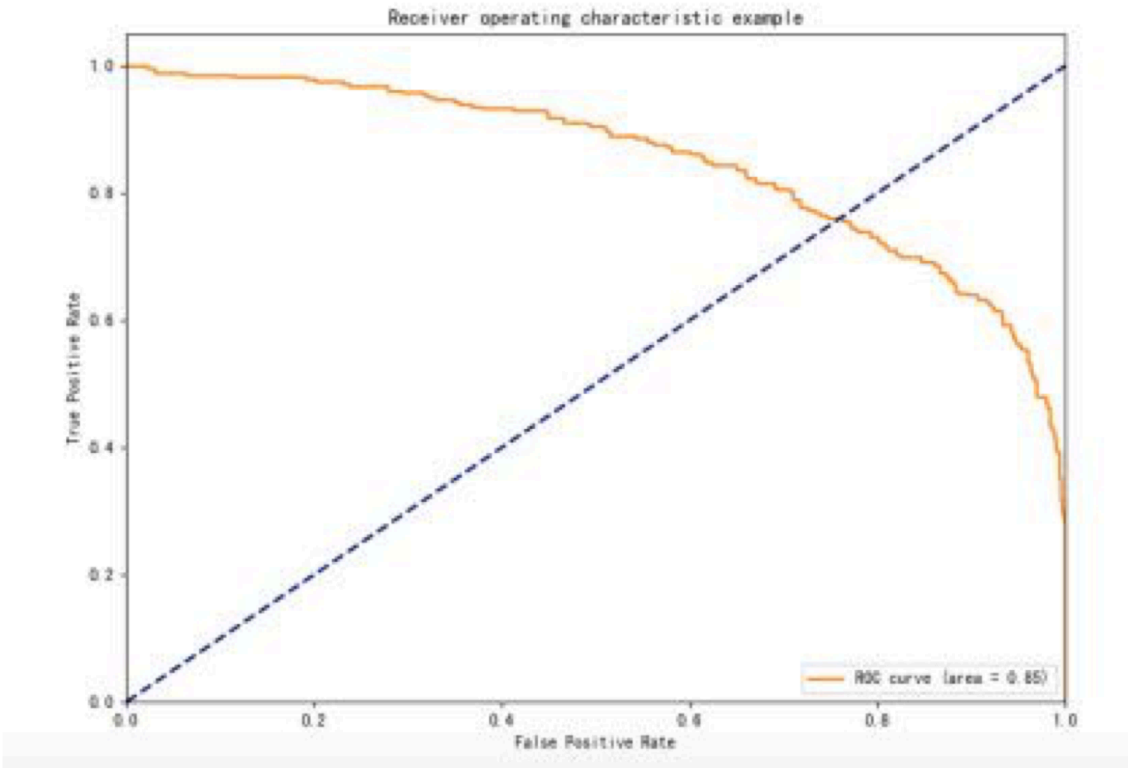
Similarity Measure	Correct recognition rate(%)	
	Original feature set	Reduced feature set
L1 distance measure	0.877962962963	0.881666666667
L2 distance measure	0.884197530864	0.891851851852
Cosine similarity measure	0.886481481481	0.892283950617

Here comes the accuracy plot for different dimension reduction.



The chart for threshold, fmr and fnmr could be derived as 3 lists. The number on ith position of first list is threshold, of second and third list are fmr and fnmr correspondingly.

Here comes the plot:



Something to improve:
In this experiment, I found that for many pictures, the position of the eyelashes is much larger than what is said in the paper. The treatment in the paper is to cut the generalized 64 X 512 picture into a 48 X 512 picture so as to eliminate the position of the eyelashes. However, this is far from enough in this experiment. I think that the texture features of the iris in a circle of the iris are similar. There is no need to pursue a complete circle of iris information here. Selecting the right and left sides that can be completely taken out would be better.