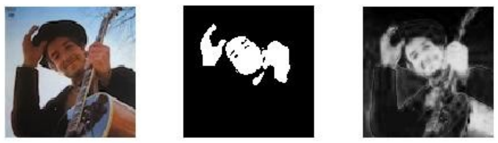
Machine Vision Coursework 1

Part I

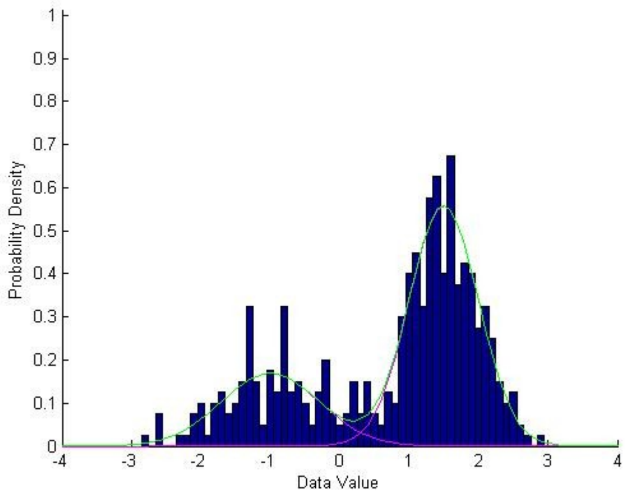
practicalMixGaussA

This figure shows the original image, the ground truth image and the predicted image that have the pixels of skin. We can see that using this one Gaussian model, the result is not so good because lots of non-skin pixels are mistakenly fitted to skin-pixel.

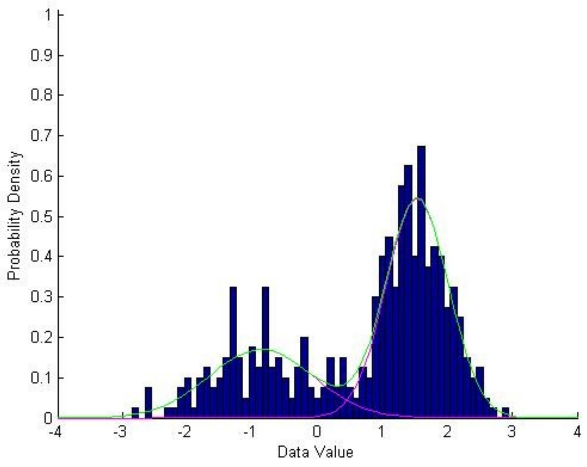


practicalMixGaussB

After iteration, the estimated data fit to the true distributions. Green line is the mix of two Gaussians distributions and each magenta line is one of the Gaussians distributions. We can see that with more iteration, the data fit better.

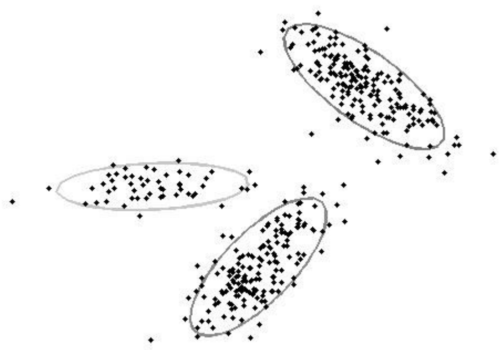


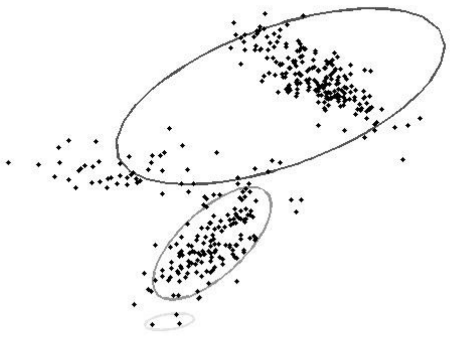
This figure shows the true Gaussians distributions of data. Green line is the mix of two Gaussians distributions and each magenta line is one of the Gaussians distributions.



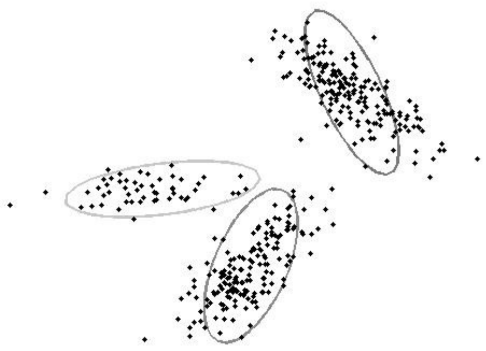
practicalMixGaussC

After certain iteration, this image shows how to fit a 3-dimensional mixtures of Gaussians model. However, because randomly choose Gaussian may cause some odd data, the fitting does not always correct (the second image shows the wrong fitting).





This is the true Gaussians data for comparing, we can see that there is a slightly difference between this image and the well estimated image. However, there will be a big difference if the fitting goes wrong.



TO DO (a)

For the input data, this part calculates its mean value and covariance value in a multidimensional way. The results will be used in further calculation of Gaussian probability.

TO DO (b)

For the input RGB pixel data, this part calculates its multivariate Gaussian distribution with certain mean and covariance value. The result is the likelihood.

TO DO (c)

With likelihoods as well as prior probabilities, the posterior probability can be calculated. The results can be converted to grayscale to represent the skin or non-skin pixel in an image.

TO DO (d)
According to the chosen h(with 30% probability equals to 1 and 70% probability equals to 2), select corresponding mean and covariance values to generate the Gaussian data. The data will be in two different Gaussian distributions.

TO DO (e)
This part calculates the log likelihood for the mixture of Gaussian model in a for-loop. Using the log is the calculation can be faster in cumulative sum, and to prevent likelihood to be zero due to the small number.

TO DO (f)
After turning covariance into upper triangular matrix, the data can be generated according to the certain mean value and covariance value, with the help of h. The data will be in three different Gaussian distributions (nGaussEst = 3).

TO DO (g)
Firstly calculate the Gaussian probability of the data with certain mean and covariance, then multiply the result by corresponding weight, so that to get the post hidden value. Then normalize the result to get posterior probability (responsibilities).

TO DO (h)
In iteration, update weighting parameters by dividing each Gaussian with total posterior probability.

TO DO (i)
In iteration, the weights from posterior probability will be used to update mean parameters, by calculating the average value.

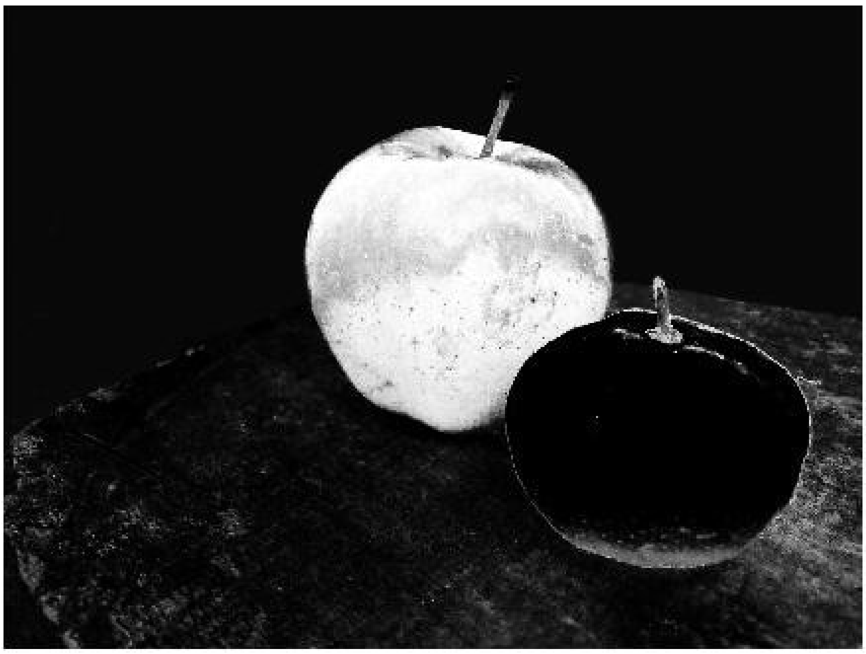
TO DO (j)
In iteration, the weights from posterior probability and the mean value will be used to update covariance values by calculating average value.

Part II

B) & C)
Firstly, using the mask images I try to get all the RGB data of apple pixels and non-apple pixels from all the three images. The data is written into two variables each of size 3 x N. Each column of these variables contains RGB values from one pixel. The code is written in 'a_GetTrainingData.m' and the training data is saved in 'Data_AppleNonApple.mat'.

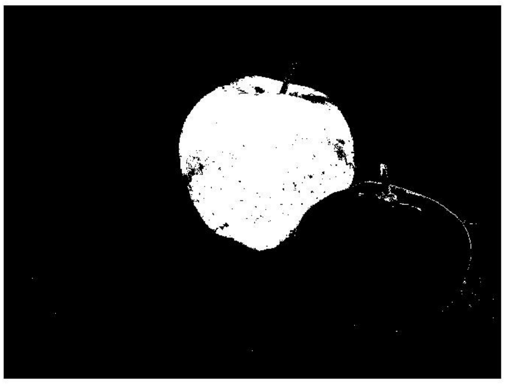
Afterwards we can use this data to train the machine and get the mean and covariance value. However the uint8 RGB values of each pixel need to be turn to the [0 1] range, so that to make sure the likelihood will not be too small to represent. (I tried the [0 255] range and the likelihood will always be zero due to the very small number). I write the 'fitMoGModel' function (fit the mixture of Gaussian model), which is similar to the practice C. And I define the number of estimate Gaussian as 3 and iteration number as 25 to get accurate training data. The code in written in 'b_Training.m', and the results are saved in 'RGB_Est(accurate).mat'.

Then we can validate whether the training data can help us to detect whether the pixel are likely to be apple or not. The priors are defined as 0.5 and 0.5. With the results from the training data, the posterior probability in each pixel of the test image can be calculated. The code is written in 'c_Validation.m' and the outcome image is as follow:

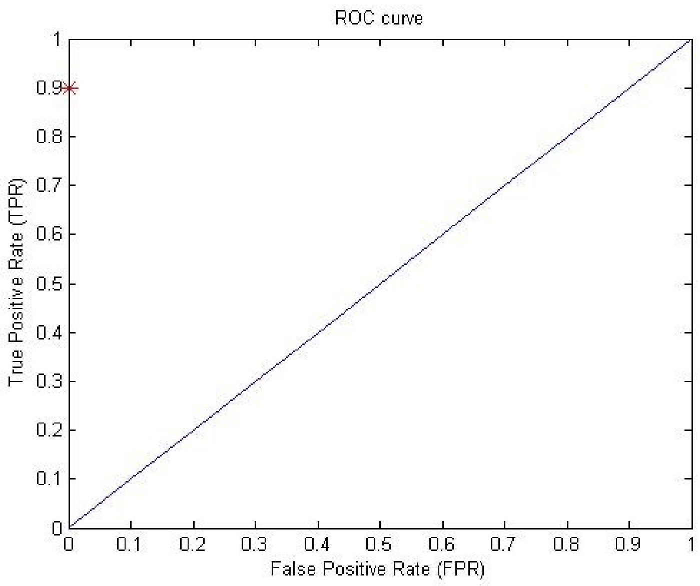


D)

I write a function('my_roc.m') by myself to get ROC data. After setting certain threshold (180) to get rid of some background pixels, the results can be compared.



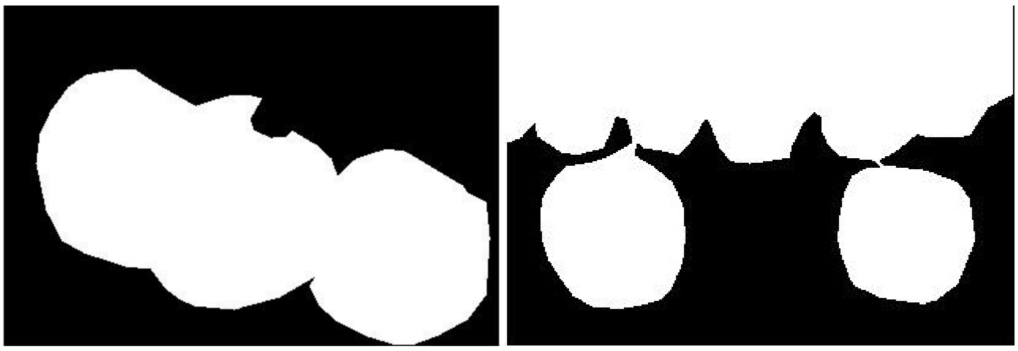
The results: TPR = 0.8999 and FPR = 0.0015. And the point for TPR and FPR (red-star) is plotted on the ROC diagram. It can be seen that the FPR is quite small, and the TPR is very close to 1. So the result is very good.



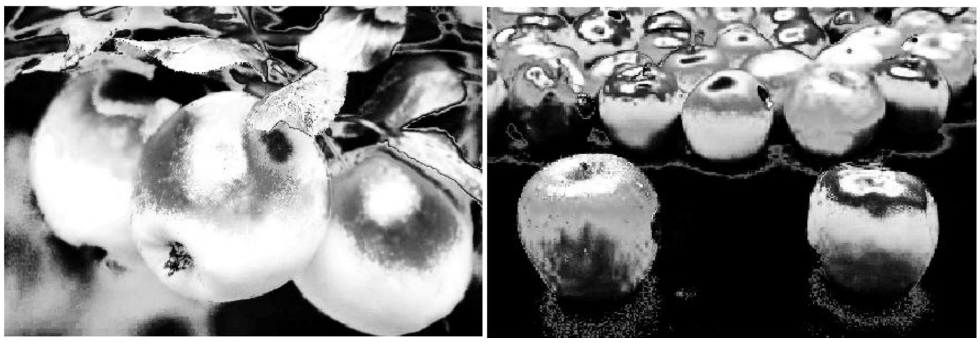
E)

I choose these two apple images for extra test-images:

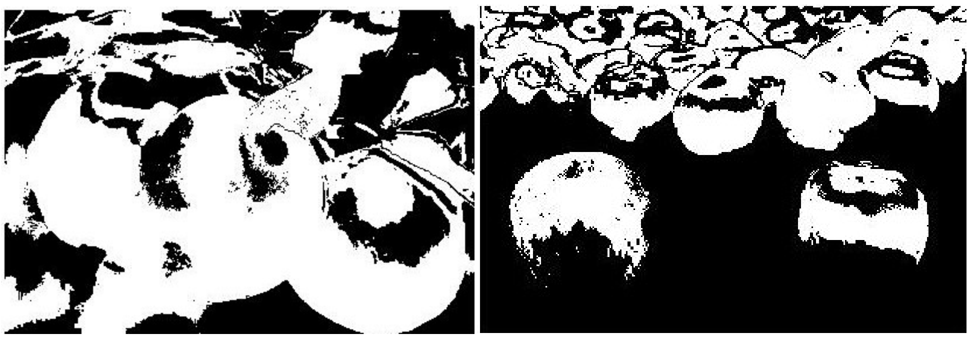
I used roipoly in Matlab to extract masks for the apples and saved the ground truth in 'extra_groundtruth1.mat'and 'extra_groundtruth2.mat'



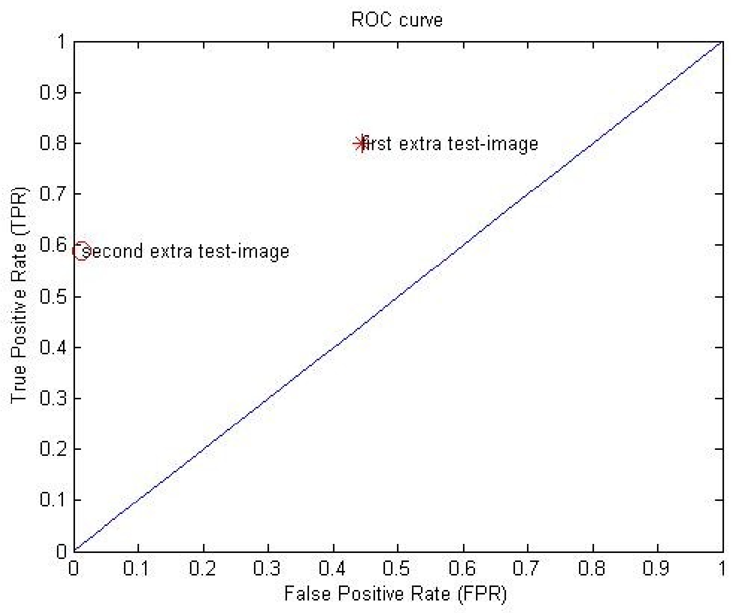
After calculating the posteriors for these two apple images, the results can be shown:



Using the threshold, the segmented images are as follow:



And subsequently their ROC can be analyzed. The ROC curve can be plotted as well. It can be seen that these points are not so close to the best position (top-left), and may vary when applying different images. So fitting mixture of Gaussians cannot always guarantee the best results.



For the first apple image: TPR = 0.7995; FPR = 0.4438
For the second apple image: TPR1 = 0.5896; FPR1 = 0.0138

F)

A training set is to train the data into certain model (the n-dimensional mixture of Gaussian model), without this set we cannot get the estimated likelihood or other training data for calculating the posterior probability.

Meanwhile, a validation set is to make sure our likelihood is correctly calculated, perhaps using some ground truth to verify our model.

Our data and model should be applied in a test set, so that to make use of the model to predict new data and solve real problem. Also a test set can evaluate the quality of our model (ROC or some other method).

Extra Credit

Manipulating the photographs' colors to improve the classification

What I think is to convert RGB pixels to the YCbCr color space, where Y dimension contains the luminance (grayscale) of the image, and Cb, Cr dimensions contain the chrominance of the image. In this way the mixture of Gaussians fitting can perform better in these color space rather than RGB space.

However, when I try the YCbCr color space in script and run to train the data, in the second iteration the log likelihood is always too small to represent. Therefore the results couldn’t be acquired.

Reference

http://en.wikipedia.org/wiki/Receiver_operating_characteristic
http://en.wikipedia.org/wiki/YCbCr