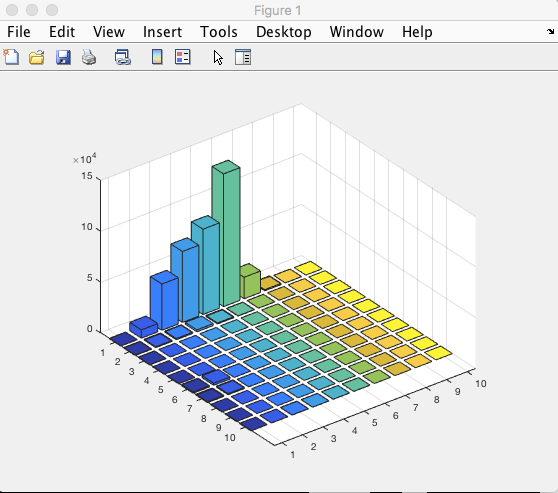
IIP CW report

There are three objectives of the CW of IIP, task1 for face region detection, task2 for filters implementation and task3 for vessel retina distraction. The following will report respectively on these three tasks and begin with task1.

***Task1: Implement a face detector using HSV color space or HUV space.***

Here I choose HSV space to detect the face region and for *face1.jpg*, I found a face picture from the internet to trained my algorithm to find proper HSV cluster set to detect the face region in *face1.jpg*. ***figure 1.1.1*** is the training picture and ***figure 1.1.2*** is the cluster set information generated by Matlab.

***figure 1.1.2***

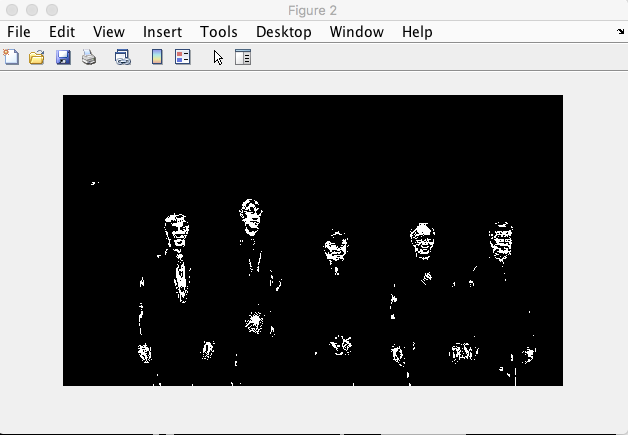
***figure 1.1.1***

When using this training picture, the best face-pixel cluster set I chose for later work is [0,0.1;0.4,0.6]. (match the format [H\_start, H\_end; S\_start, S\_end]).

Then, change the RGB colorful *face1.jpg* to gray picture first by using **rgb2gray()** function and using the cluster set above to process the binary image, ***figure1.1.3*** is the origin picture of face1.jpg, ***figure1.1.4*** is the result of this operation.



***figure 1.1.3***



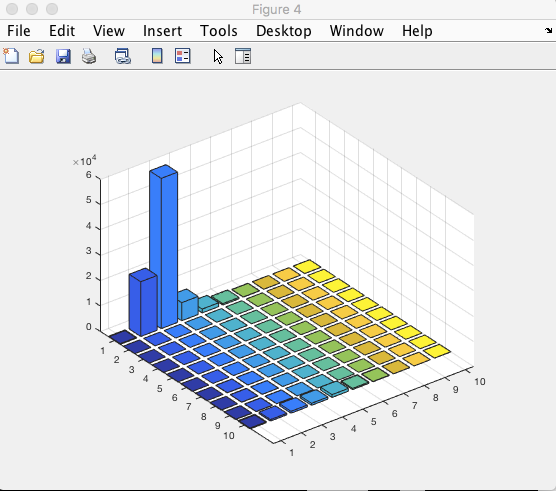
***figure 1.1.4***

While it looks that there is much noise on the binary image, so last step is to reduce the noise to highlight the face region, ***figure 1.1.5*** show the final detection result for face1.jpg.



***figure 1.1.5***

For face2.jpg, similar steps as face1.jpg, figure 1.2.1, training picture, figure 1.2.2, cluster set information generated by Matlab.

***figure 1.2.2***

***figure 1.2.1***

When using this training picture, the best face-pixel cluster set I chose for later work is [0,0.1;0.1,0.3]. (match the format [H\_start, H\_end; S\_start, S\_end]).

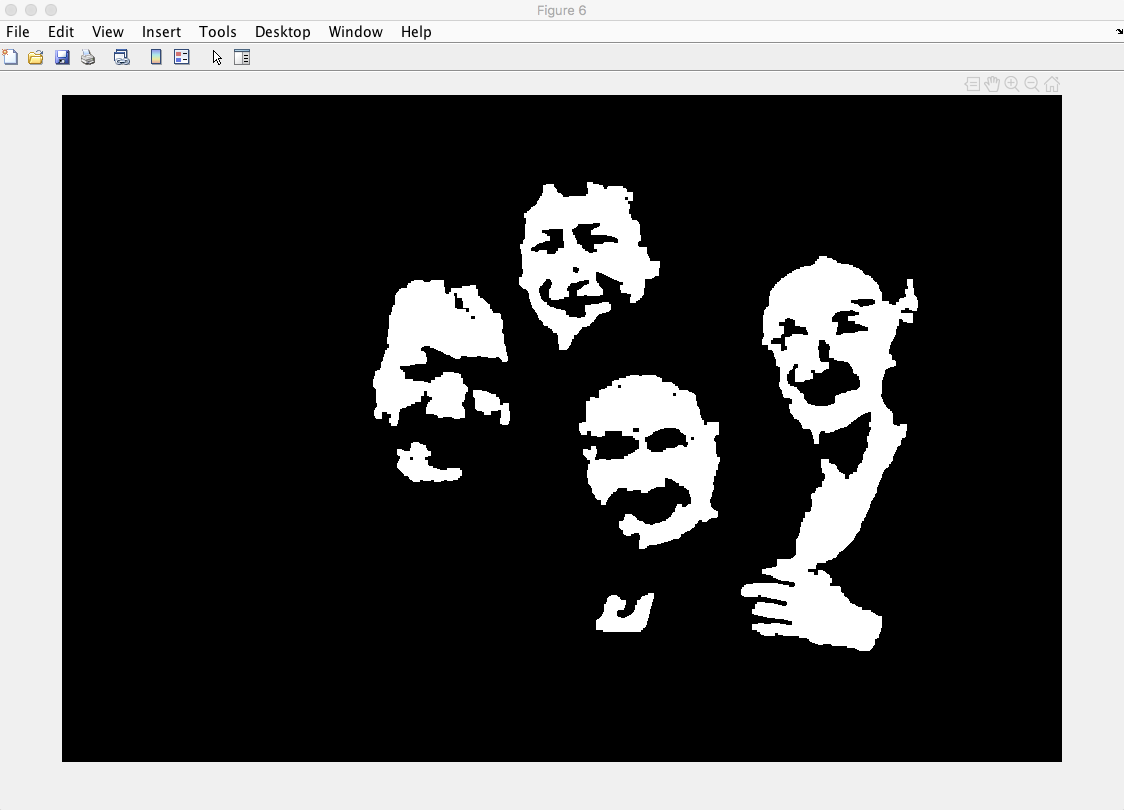
The same translation as operation on *face1.jpg*, and ***figure 1.2.3*** is the origin picture of *face2.jpg*, ***figure 1.2.4*** is the results of processing the picture using HSV cluster sets offered above, ***figure 1.2.5*** is the result for detecting the face region of *face2.jpg.*



***figure1.2.3***



***figure 1.2.4***



***figure 1.2.5***

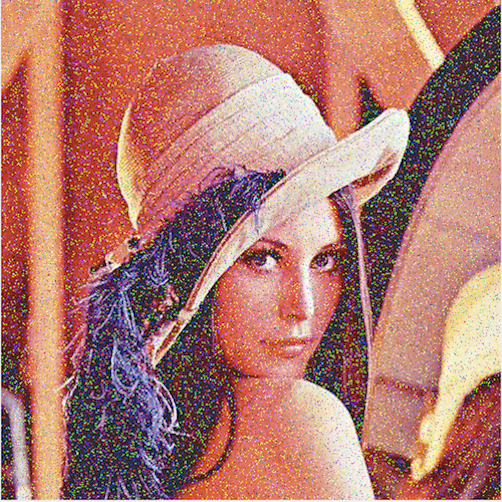
***Task2: Implement and compare spatial filters***

This task is to implement 5 filters for processing images discussed in the lecture, and they are 3 x 3 Mean Filter, 5 x 5 Gaussian Filter with sigma = 1, 3 x 3 Meidian Filter, 3 x 3 Anisotropic Filter with similarity function of (D-d)/D, 5 x 5 Bilateral Filter with space Gaussian sigma = 1 and range Gaussian sigma = 10.

When implemented these filters, next step is to add some noise to the image, firstly, add Gaussian noise with sigma = 20 on RGB channels of lena.jpg. Secondly, add another type of noise on the image, salt and pepper noise with noise rate of 10%.

The follow showed the results of after using different filters to process the image.

**Origin picture (figure 2.1), Gaussian noise picture (figure 2.2), Salt & Pepper noise picture (figure 2.3)**

***figure 2.1***

***figure 2.2***

***figure 2.3***

**Gaussian noise picture convoluted by 7 x 7 Gaussian filter with sigma = 2 (figure 2.4)**

****

***figure 2.4***

**Salt & Pepper picture convoluted 7 x 7 Gaussian filter with sigma = 2 (figure 2.5)**

****

***figure 2.5***

**Gaussian noise picture processed by the filter I implemented**

**1 Noise picture convoluted by 3 x 3 Mean filter (figure 2.6.1)**

****

***figure 2.6.1***

**2 Noise picture convoluted by 5 x 5 Gaussian filter with sigma = 1 (figure 2.6.2)**

****

***figure 2.6.2***

**3 Noise picture convoluted by 3 x 3 Meidian filter (figure 2.6.3)**

****

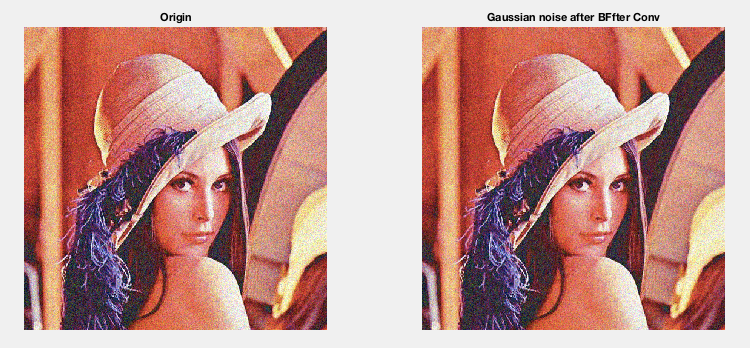
***figure 2.6.3***

**4 Noise picture convoluted by 3 x 3 Anisotropic filter (similarity fcn: (D-d)/d) (figure 2.6.4)**

****

***figure 2.6.4***

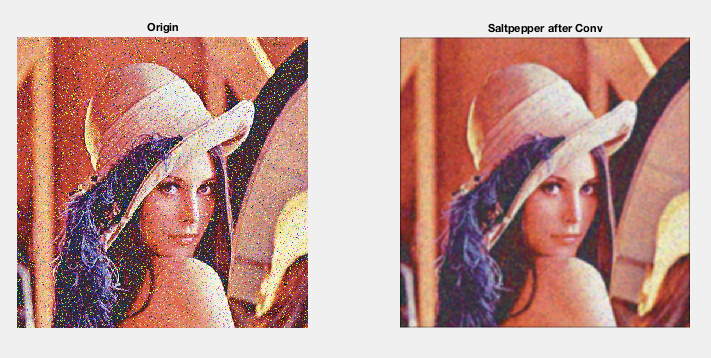
**5 Noise picture convoluted by 5 x 5 Bilateral filter with with space Gaussian sigma = 1 and range Gaussian sigma = 10. (figure 2.6.5)**

****

***figure 2.6.5***

**Salt & Pepper noise picture processed by the filter I implemented**

**1 Noise picture convoluted by 3 x 3 Mean filter (figure 2.7.1)**

****

***figure 2.7.1***