Project 2

0410886 何哲佑

Introduction

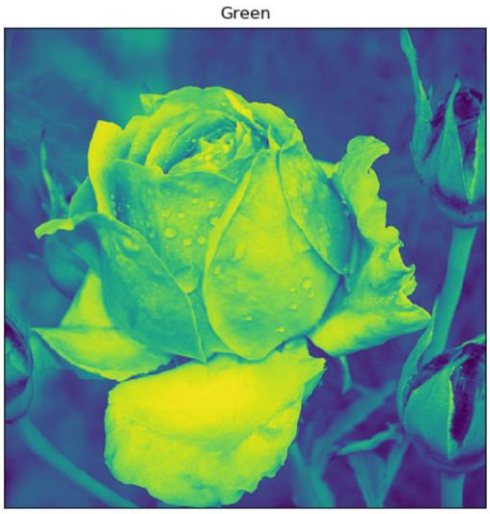
In this project we are applying our knowledge in image processing in order to do color image processing. We will first be splitting our image BGR in three different channels. On the second part we will start dealing with edge detection on images by applying two different algorithms, the Marr-Hildreth algorithm and the Hough algorithm. At the final part we will attempt to use a multiple global threshold to make a partition of the image in three segments.

Part 1

For starting we present our image to be processed:

Now we proceed to separate all the channels of our image. RGB in 3 different pictures:

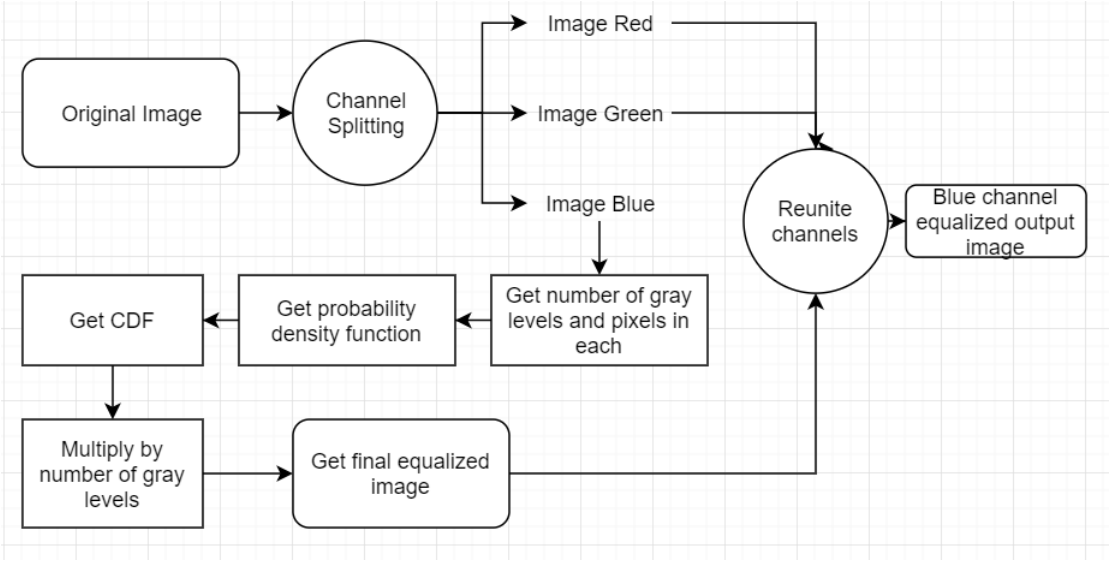
(Following page)



An image is an array with three different channels. For splitting our image we just take one of each of the three channels of the image and graph them seperately. Following this we are going to apply histogram equalization on the blue channel. After applying the equalization to this channel we are going to put the channels back together.

The result is the following →

Flow Diagram of part 1



Part 2

For this part we first are going to treat the Marr-Hildreth algorithm. When performing this edge detection algorithm, first, we need to get the Gaussian of our image. For this we are going to use a sigma = 1 and a 5x5 kernel. This is the result we get:

Original

Gaussian

Following the first step we apply the Laplacian filter to the image:

LoG

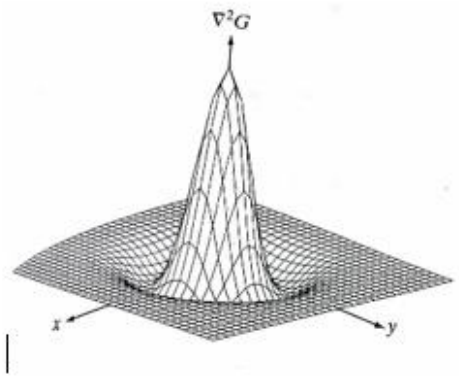
$$G(x,y) = e^{\frac{x^2+y^2}{2\sigma^2}}$$

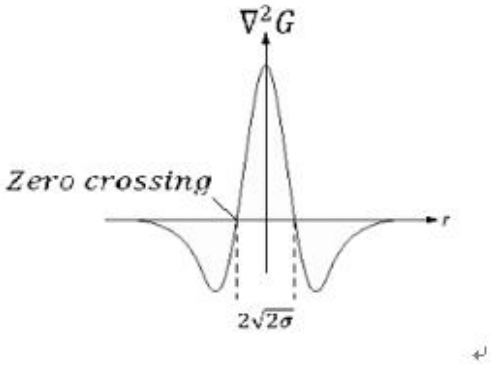
$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{\frac{x^2+y^2}{2\sigma^2}}$$

Now we use zero crossing detection technique with 2 different thresholds:

Zcr : 0% Zcr : 4%

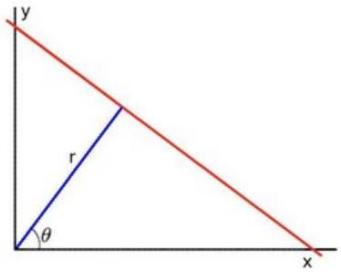
In zero crossing we check for all neighbors, the following conditions at each pixel must be meet, smim(i,j) = pixel(i,j):






```
if (smim(i,j)>0)
            if (smim(i,j+1)>=0 && smim(i,j-1)<0) || (smim(i,j+1)<0 &&
smim(i,j-1)>=0)
               zc(i,j)= smim(i,j+1);
           else if (smim(i+1,j)>=0 && smim(i-1,j)<0) || (smim(i+1,j)<0 &&
smim(i-1,j)>=0)
               zc(i,j)= smim(i,j+1);
           else if (smim(i+1,j+1)>=0 && smim(i-1,j-1)<0) || (smim(i+1,j+1)<0
&& smim(i-1,j-1)>=0)
               zc(i,j)= smim(i,j+1);
           else if (smim(i-1,j+1)>=0 && smim(i+1,j-1)<0) || (smim(i-1,j+1)<0
&& smim(i+1,j-1)>=0)
               zc(i,j)=smim(i,j+1);
```

For proceeding to the next part we are going to take our 4% zcr. In this case we are taking our output image in order to perform edge linking. The edge linking technique to be used is the the hough algorithm. First we explain the transformation between polar and cartesian coordinates.



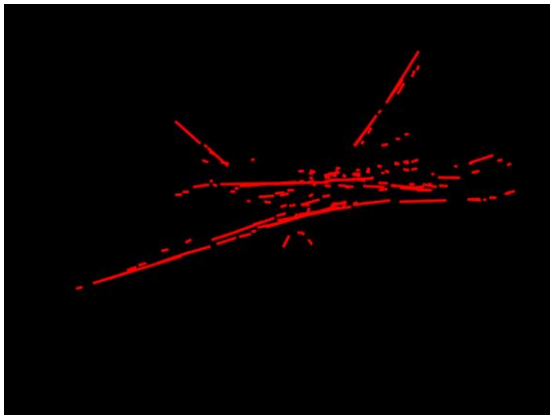
$$y = mx + y_0$$

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

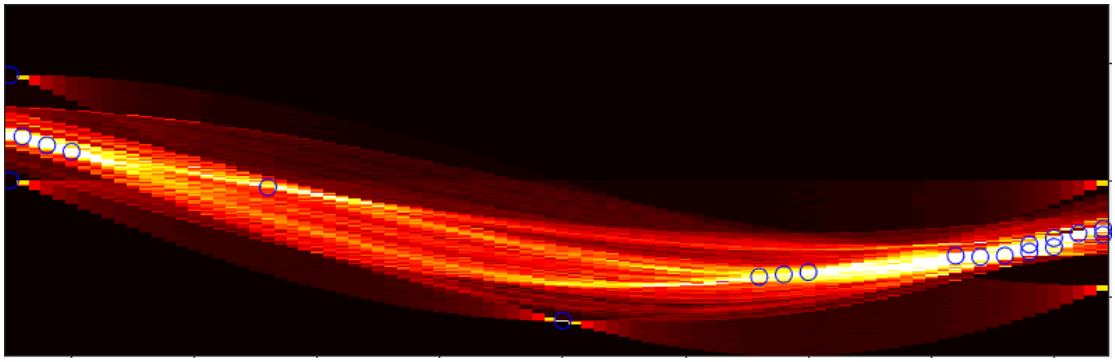
$$r = x\cos\theta + y\sin\theta$$

The pixels that were obatained from our Marr-Hildreth algorithm are to be converted into polar form. Their position is represented in terms of theta and r. The center of the image is used as reference. Edge pixels mapping to the same theta and r are assumed to define a line in the image. Then we connect this points and get the edge lines. These are the results :

b)



a) Parameter space (rho, theta)

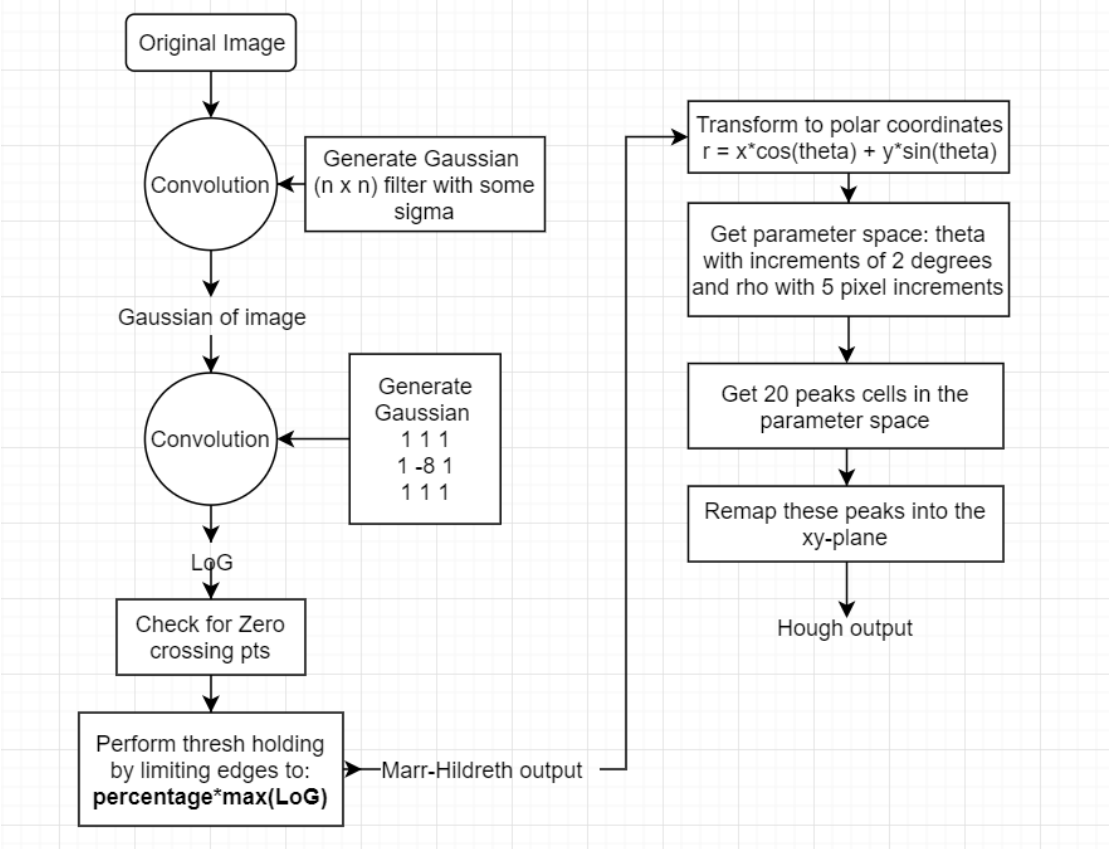


c) Cells marked (not ordered):

X	Y
431	82
385	90
393	88
433	80
424	84
227	2
416	86
401	86
122	1
314	22
242	4
463	64
467	62
431	78
395	90
412	84
459	66
302	1
542	46
253	6

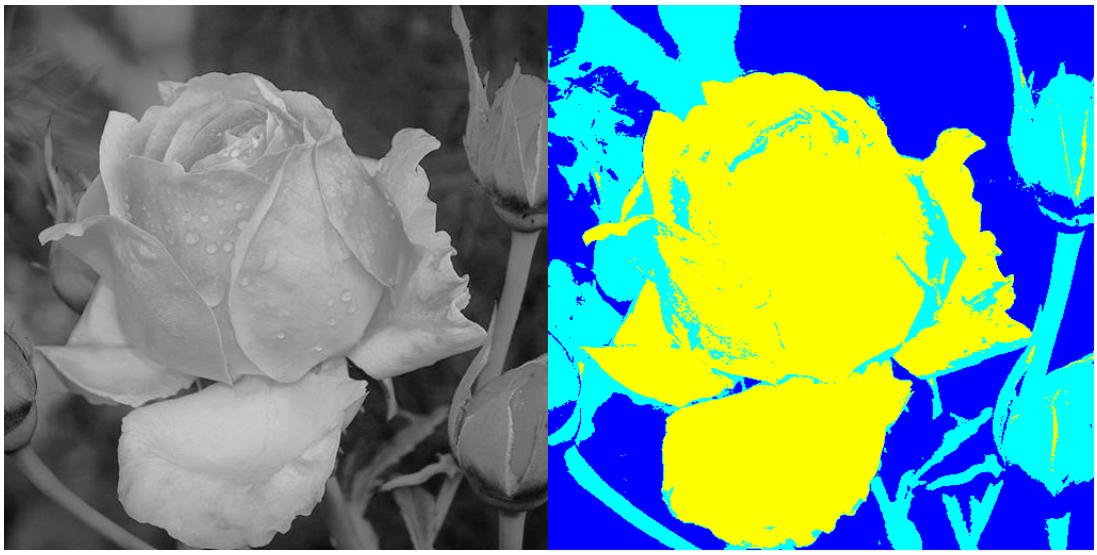
Possible cells of aircraft body

Flow Diagram of part 2



Part 3

For this final part we attempt to perform image segmentation using Otsu's algorithm. We segment the image in 3 sections:

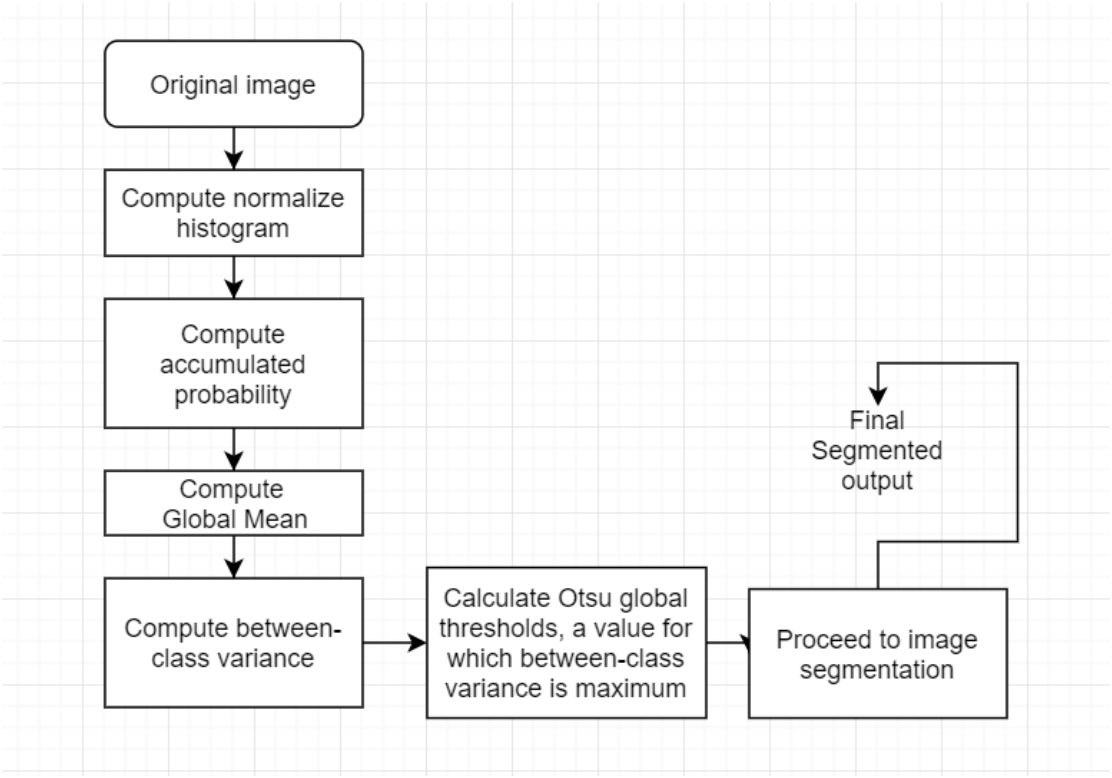


Original image

Segmented Image

Image Segmentation: Segment the image using T's. This will produce n groups of pixels. G1 consisting of all pixels with gray level values >T1 and G2 consisting of pixels with T2<values <=T1. And group 3 <=T2.

Flow Diagram of part 3



Final Discussions

Part 1

In the part number one we dealt we color image processing. A new topic tackled for first time. We perform 2 simple tasks, the first task is to split the image in its distinct channels. This process has no mayor difficulty. One of the technicalities you need to keep in mind while programming was the system being used by the language or function brought to your code. Some confussion occur if you process an image that is channeled as bgr instead of rgb. In this case inversion of "b" and "r" is required. The final is just complementary to the first. We just need to perform our well known histogram equalization to one of the channels and then rejoin them. A trivial process.

Part 2

In this part implementation of our Marr-Hildreth algorithm is somehow sum of simple processes. We first get the LoG of our image, which can be done in a single step or in two separated steps. The part of the zero crossing is a little trickier, you need decide what to do after a zero crossing point is found. We just gave the value of the pixel's neighbor pixel P(i,j+1). Then thresholding is applied and our image then is to be converted into a binary image.

Following the first edge detection algorithm we proceed to perform Hough edge linking. In this process we transport ourselves to the (rho,theta) domain. Get maximum values in this domain and remap them into the (x,y) world (lines that share similar theta). We should stress the fact that when remapping of this peaks occur; it is not unique. So expecting as many lines as peaks is a terrible misconception. We also create clear image (black) so could have a better appreciation of the lines.

Part 2

In Otsu's global threshold algorithm there are not big issues. One of the things to keep in mind is that you should be clear in the number of classes you wish to obtain, otherwise no segmentation will occur. The simplest version of the algorithm, which is easier to implement, just performs binary segmentation. This is a simple algorithm to implement as showed in the flow diagram, but quite useful as showed in the results.