ECE 5470 Computer Vision Lab2 report

Binary image Processing

Weiran Wang

NetID: ww463

# Bound programming

Description:

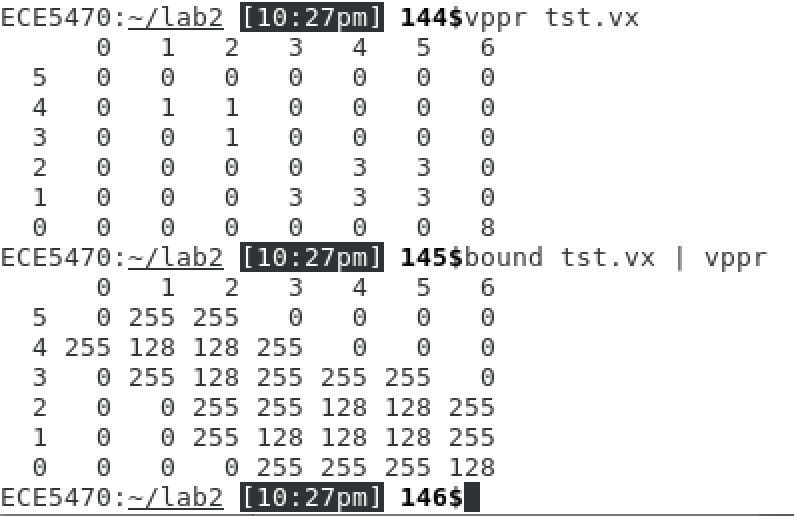
I get started by using the template provided in the vtemp. Vfstruct commands are used to declare Vfstructs to hold old and new images, Vfembed is used to copy image old to new and pad the boundary. To generate an image in which the segment boundary pixels are set to 255, interior region pixels are set to 128 , we can start by looking for the segment boundary. Segment boundary are those pixels whose value is 0 and have one or more of the 4 connected non-zero pixels. In other words, boundary pixels are pure black with value 0 and they are connected with the segmented part of the image. Pixels that have non-zero values belong to the interior region. Pixels that have value 0 and surrounded by all 0 value pixels are considered the background. After knowing the boundary, interior region and background region, I set boundary pixels to 255, interior region to 128 and the background pixels to 0.

Program:

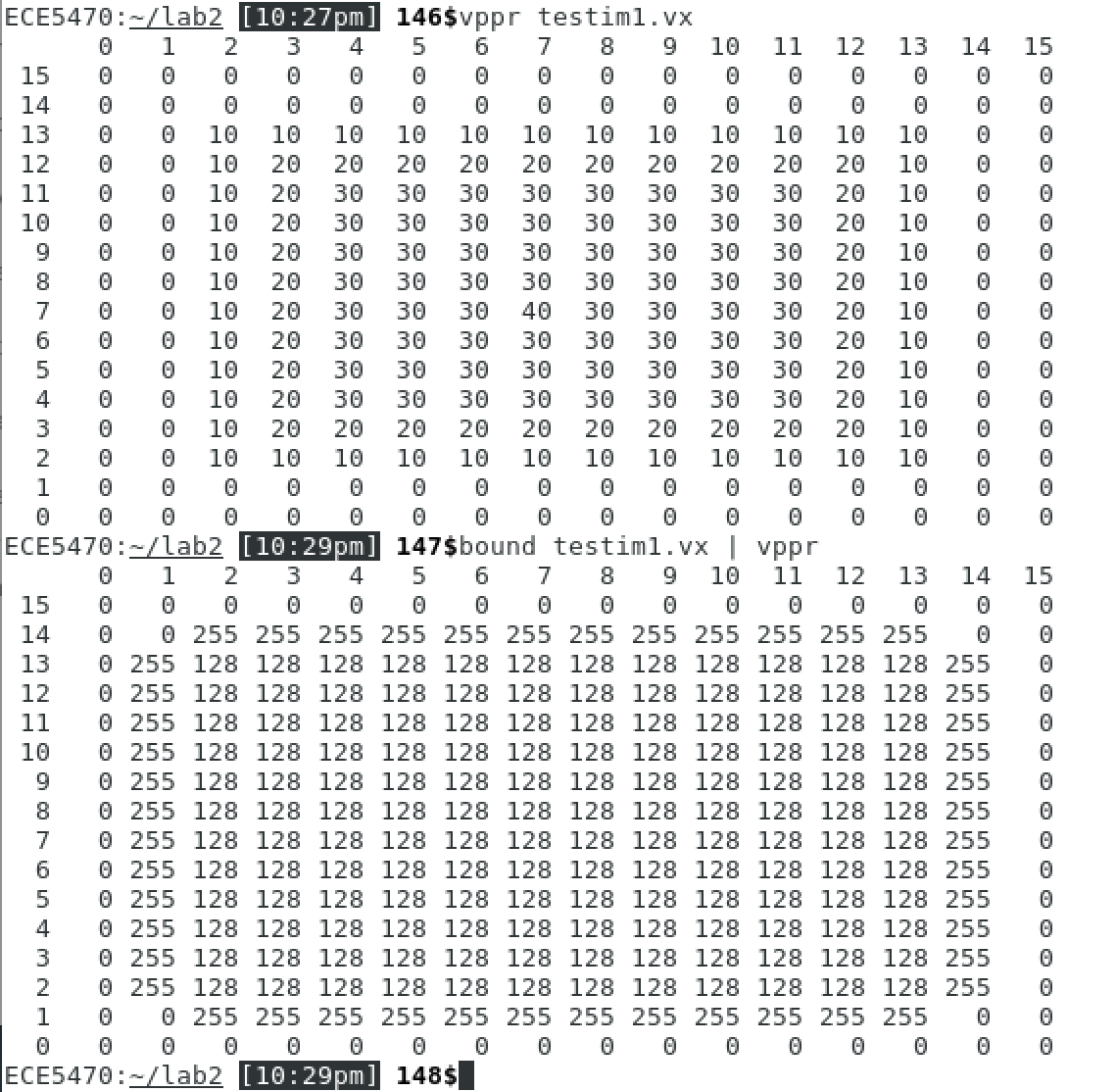


Small image manipulation:

As we can see from the screenshots below, output images after running bound.c script meet the requirement, segment boundary pixels are set to 255, interior region pixels are set to 128 and the background remains the same. This proves the program bound.c is working.



img 2.1



img 2.2

Large images:



img 2.3



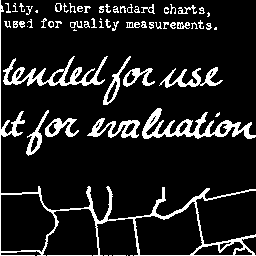
img 2.4



img 2.5



img 2.6

img 2.7 img2.8

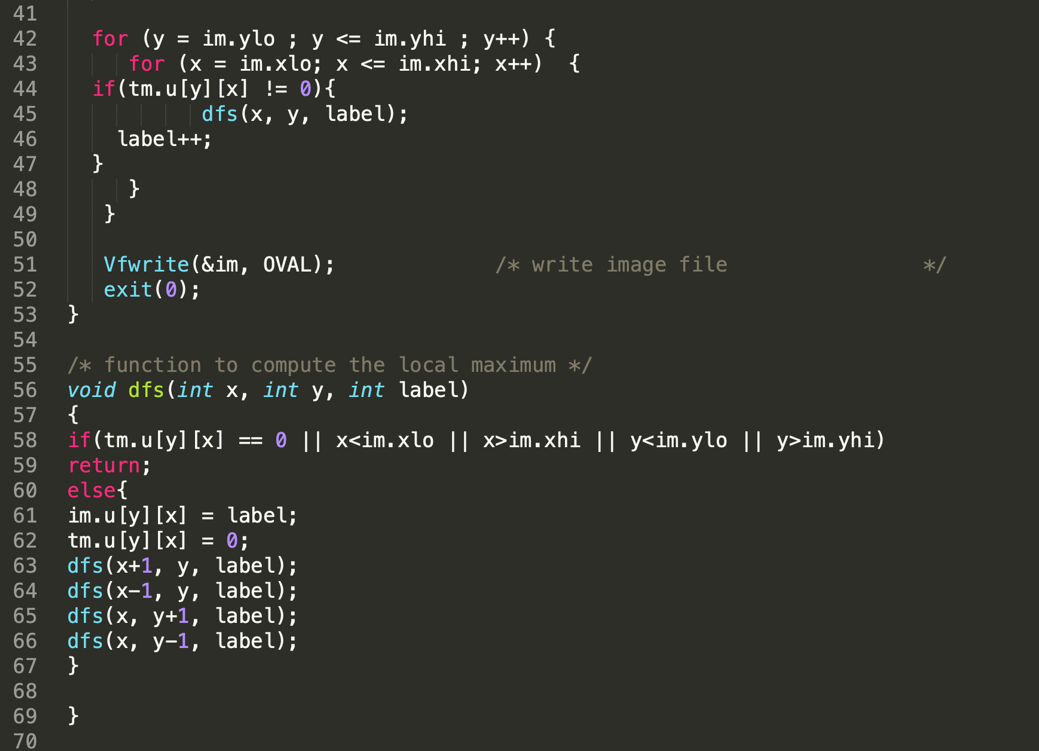
# Label Programming

Description:

The images setup is the same as the previous one, we have the new image im and old image tm. To label connected components in the image, I started by setting all pixels value to 0 for the new image and label each connected component by referring to the old image. The technique I use is call Depth First Search(DFS), this is achieved by recursively call the function dfs(x, y, label) in my program. We search for pixels whose value is not zero in the old image, if we found one, we label this particular pixel on the new image starting from 1,2…, and change the current pixel to 0 to avoid labeling again. Once this is done, we call function dfs(x, y, label) to check if there are non-zero pixels connected to (left, right, up ,down) the current pixel. If there are, then they should be labeled the same as the previous one, because they are connected components. Then we keep checking and labeling new pixels on four directions, so on and so forth. In the end, we will end up having an pixel island where all pixels have the same label because of the connectivity. Then, we increase the label by one and start looking for non-zero pixels in the old image again. Note that, all pixels that have been labeled are set to zero in the old image, so they will not be repeatedly labeled.

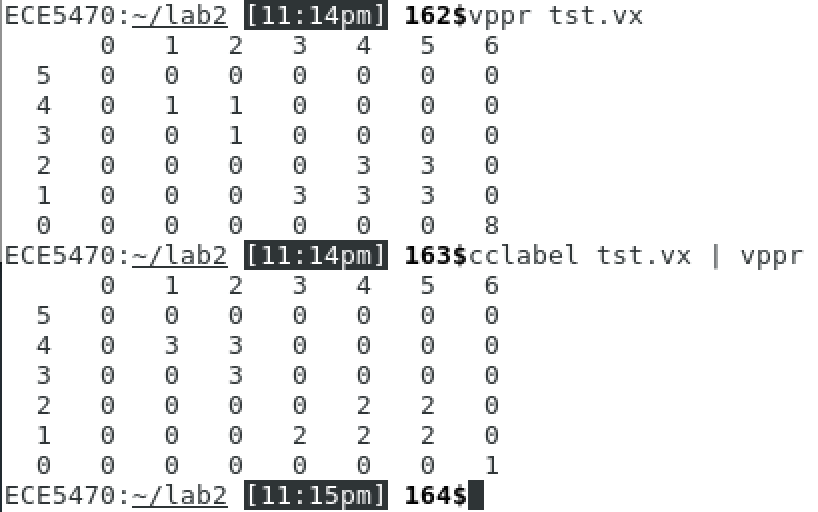
Program:



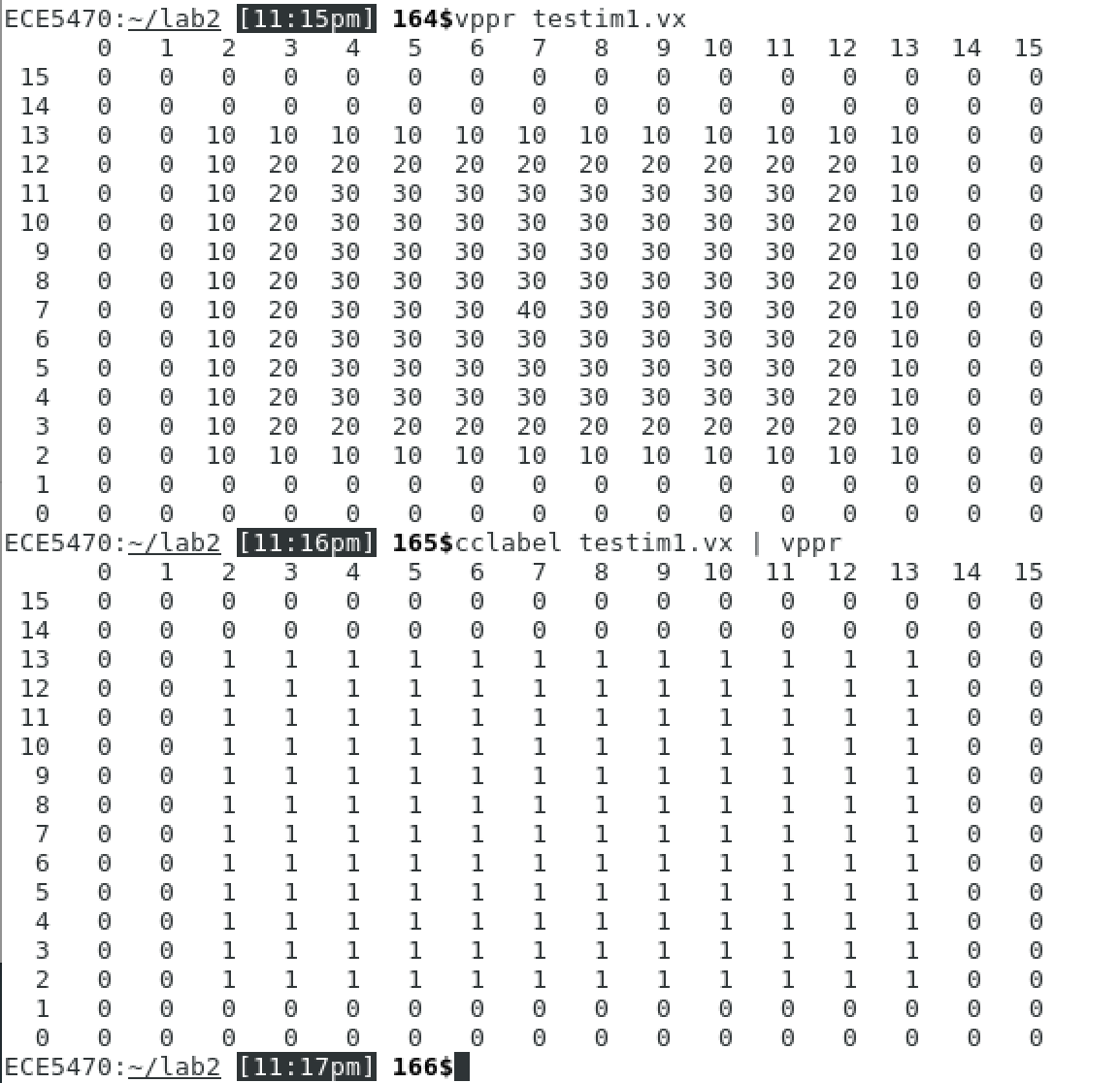


Small image manipulation:

As we can see below, cclabel.c works for small images, connected component are labelled in sequence.



img2.9



img 2.10

Large images:

Since label pixels value start from 1,2,3… so components that have small label value are very hard to see. To solve this issue, I convert pixel values by using vpix command, what used to be black now is white and vice versa. But some of them are still too vague to be seen, then I reinforce the pixel value by three times using vpix tf=3 command. As you can see below, img 2.11, img 2.15, img 2.18 are original images. img2.12, img2.16, img2.19 are converted images, img 2.13, img 2.17, img 2.20 are converted labeled images. img2.14, img2.21 are reinforced labeled images.



img 2.11



img 2.12



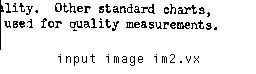
img 2.13



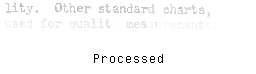
img 2.14



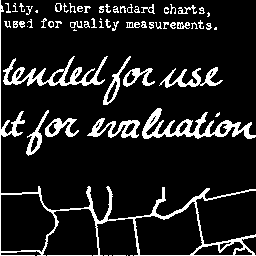
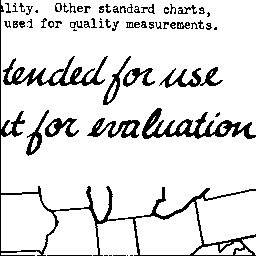
img 2.15



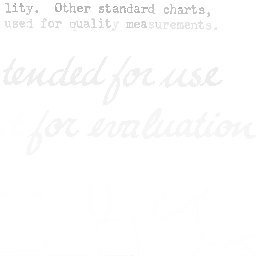
img 2.16



img 2.17

img 2.18 img2.19

img 2.20 img2.21