Lab 2 Binary Image Processing Report

ECE 5470

Computer Vision

Name: Yanling Wu

NetID: yw996

Instruction:

In this lab, we are familiar with the basic VisionX utility commond and VisionX programming tools. First of all, we followed the instruction to do some exercise and be familiar with some examples of software tools that will be applied to following lab. In the body of lab, it was divided into two parts and we need to use these tools that we have been familiar with to develop and test two binary image processing algorithms. One is to recognize the boundary of the binary image. Another is to label the different parts of the images.

Part one: A First Program---bound

Description of the program:

1. Firstly, we embed the image im into the image tm and add a one pixel border into the image. By this way, we can now use the image im to mark the boundary of the image as output since we already have a copy of original image in the image tm.
2. Search every pixel of input image, tm, and judge whether this pixel is boundary, interior or background. The criteria is:
    a) if one pixel's value does not equal to zero (tm[x, y] != 0) and only if one of its around pixel equals to zero(tm[x+1, y] == 0 || tm[x-1, y] == 0 || tm[x, y+1] == 0 || tm[x, y-1] == 0), this pixel will be defined as boundary and mark this value as 255 (im[x, y] =255);
    b) Exception these pixels whose value do not equal to zero, others are interior parts and then mark these value as 128 (im[x, y] = 128);
    c) Otherwise, if the pixels' values equal to zero, set the pixel value of the output image to zero (im[x, y] = 0).
3. Write the image file and exit program.

Code:

bound.c

```
/*****************************************************************/
/*bound   Compute local boundary operation on a single byte image */
/*****************************************************************/
/*@copyright                                                      */
```

```
/*Name: Yanling Wu                                                 */
/*NetID: yw996                                                     */
/*LAB Number: LAB 2                                                */
/*****************************************************************/

#include "VisXV4.h"            /* VisionX structure include file    */
#include "Vutil.h"             /* VisionX utility header files      */

VXparam_t par[] =              /* command line structure            */
{ /* prefix, value,   description                         */
{    "if=",    0,   " input file  vtemp: local max filter "},
{    "of=",    0,   " output file "},
{     0,       0,   0}  /* list termination */
};
#define  IVAL   par[0].val
#define  OVAL   par[1].val

main(argc, argv)
int argc;
char *argv[];
{
Vfstruct (im);                      /* i/o image structure          */
Vfstruct (tm);                      /* temp image structure         */
int        y,x;                     /* index counters               */
  VXparse(&argc, &argv, par);       /* parse the command line       */

  Vfread(&im, IVAL);                /* read image file              */
  Vfembed(&tm, &im, 1,1,1,1);       /* image structure with border  */
  if ( im.type != VX_PBYTE ) {      /* check image format           */
     fprintf(stderr, "vtemp: no byte image data in input file\n");
     exit(-1);
  }
  for (y = im.ylo ; y <= im.yhi ; y++) {  /* compute the function */
     for (x = im.xlo; x <= im.xhi; x++) { /************************/
           if ( tm.u[y][x] != 0 && (tm.u[y-1][x] == 0 || tm.u[y+1][x] == 0 ||
                                tm.u[y][x-1] == 0 || tm.u[y][x+1] == 0 )){
          im.u[y][x] = 255;
       }
       else if ( tm.u[y][x] != 0 ){
          im.u[y][x] = 128;
       }
```

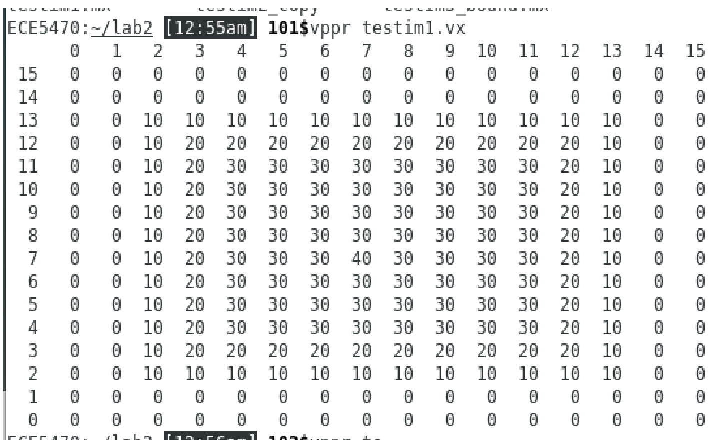
```
        else{
            im.u[y][x] = 0;
        }

      }
    }

    Vfwrite(&im, OVAL);              /* write image file                  */
    exit(0);
}
```

Result:

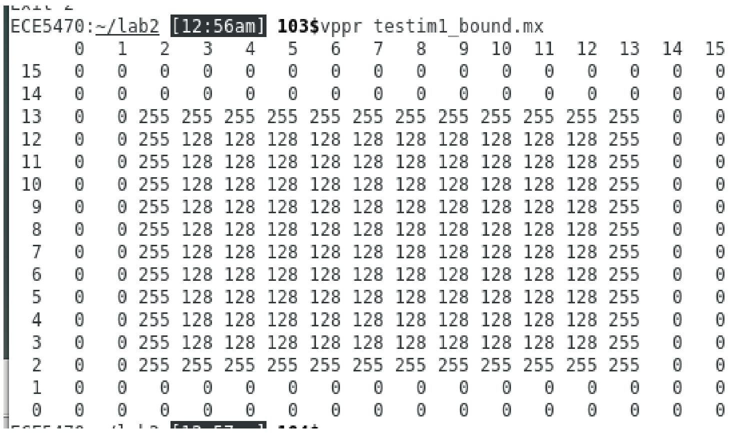
First, run "*vcc bound.c -o bound*" to compile this program and then debug this program with three small test images. Finally, test it with larger images.

Debug using small test images

Test image1(pixel value at terminal):



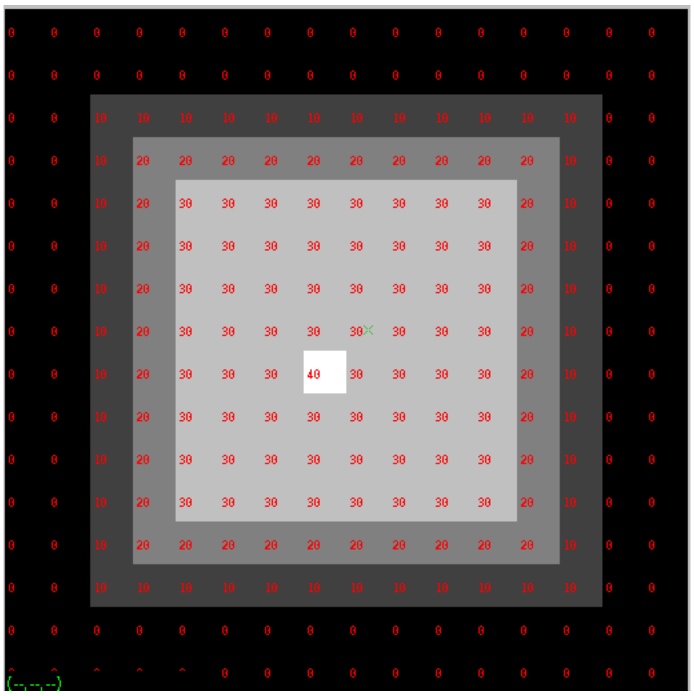
```
ECE5470:~/lab2 [12:55am] 101$vppr testim1.vx
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
 12   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
 11   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
 10   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  9   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  8   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  7   0   0  10  20  30  30  30  40  30  30  30  30  20  10   0   0
  6   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  5   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  4   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  3   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
  2   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```



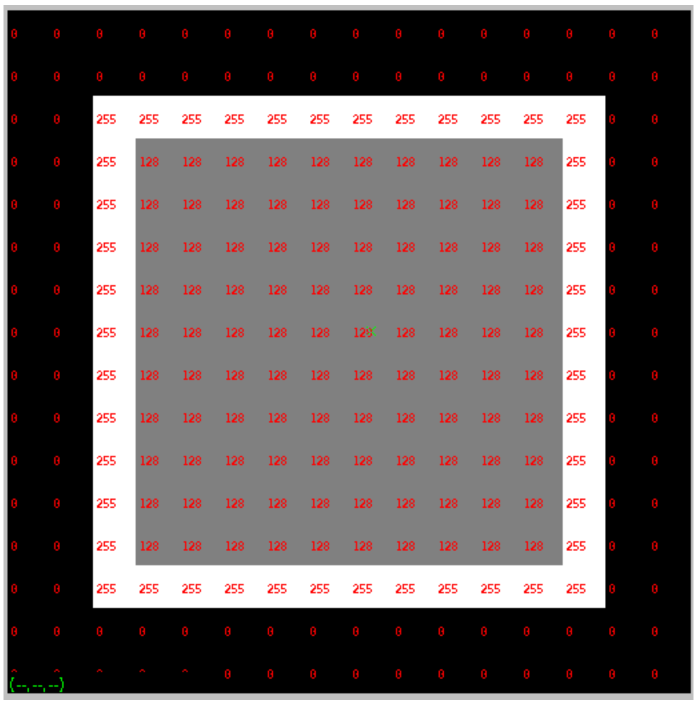
```
ECE5470:~/lab2 [12:56am] 103$vppr testim1_bound.mx
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0 255 255 255 255 255 255 255 255 255 255 255 255   0   0
 12   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
 11   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
 10   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  9   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  8   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  7   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  6   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  5   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  4   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  3   0   0 255 128 128 128 128 128 128 128 128 128 128 255   0   0
  2   0   0 255 255 255 255 255 255 255 255 255 255 255 255   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Figure 1.1 the pixel value of compare of Testim1 and the Testim1_bound

Test image1(images and pixel values):



```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   10  10  10  10  10  10  10  10  10  10  10  10  0   0
0   0   10  20  20  20  20  20  20  20  20  20  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  40  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  30  30  30  30  30  30  30  30  20  10  0   0
0   0   10  20  20  20  20  20  20  20  20  20  20  10  0   0
0   0   10  10  10  10  10  10  10  10  10  10  10  10  0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
(--,--,--)  ^   ^   ^   0   0   0   0   0   0   0   0   0   0   0
```

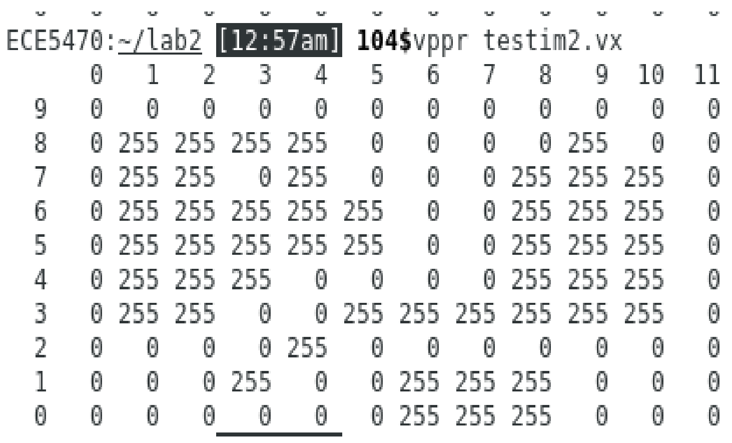


```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   255 255 255 255 255 255 255 255 255 255 255 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 128 128 128 128 128 128 128 128 128 128 255 0   0
0   0   255 255 255 255 255 255 255 255 255 255 255 255 0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
(--,--,--)  ^   ^   ^   ^   0   0   0   0   0   0   0   0   0   0
```

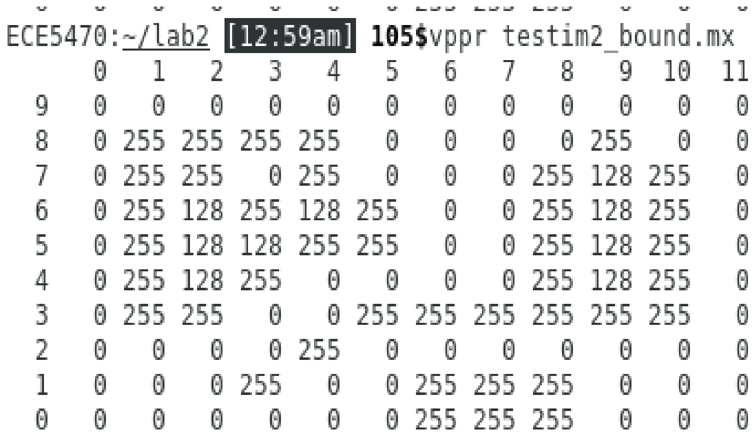
Figure 1.2 the image comparation of the testim1 and the testim1_bound

From the Figure 1.1, we can see the change of pixel values after bound processing clearly and Figure 1.2 shows the image change visually.

Test image2(pixel value at terminal):



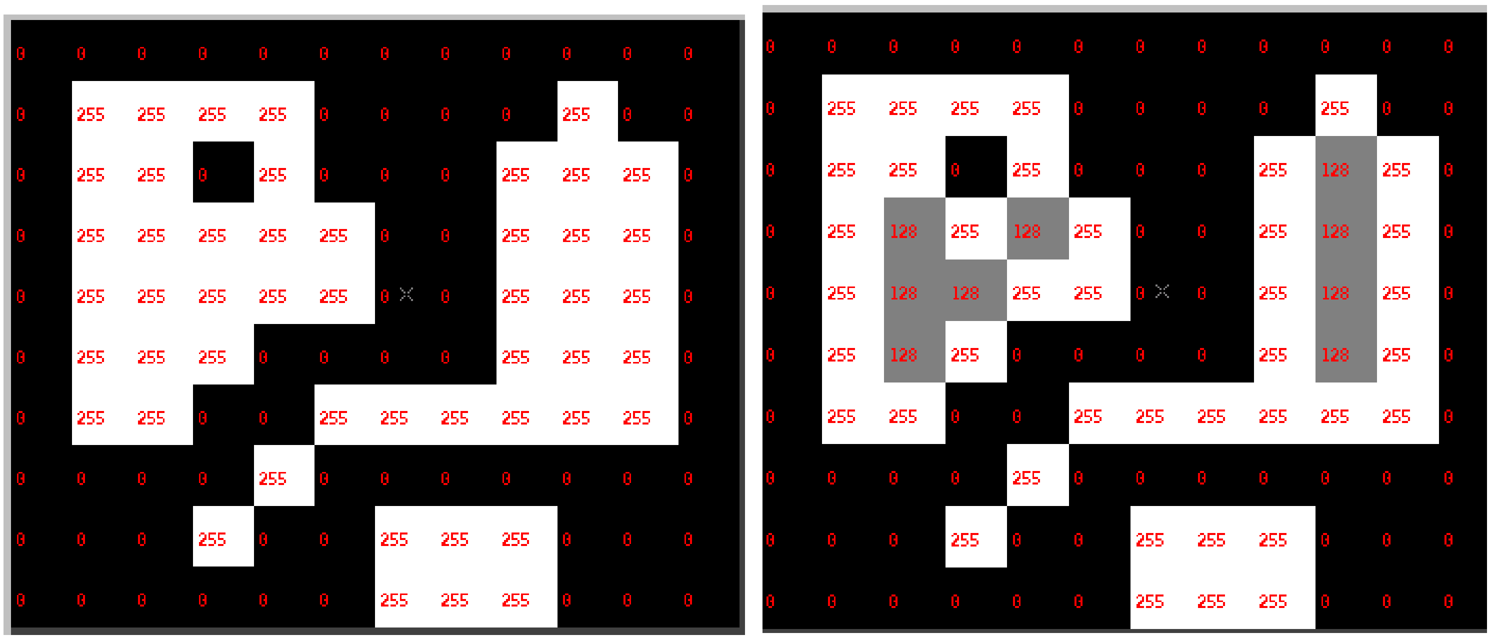
```
ECE5470:~/lab2 [12:57am] 104$vppr testim2.vx
      0   1   2   3   4   5   6   7   8   9  10  11
 9    0   0   0   0   0   0   0   0   0   0   0   0
 8    0 255 255 255 255   0   0   0   0 255   0   0
 7    0 255 255   0 255   0   0   0 255 255 255   0
 6    0 255 255 255 255 255   0   0 255 255 255   0
 5    0 255 255 255 255 255   0   0 255 255 255   0
 4    0 255 255 255   0   0   0   0 255 255 255   0
 3    0 255 255   0   0 255 255 255 255 255 255   0
 2    0   0   0   0 255   0   0   0   0   0   0   0
 1    0   0   0 255   0   0 255 255 255   0   0   0
 0    0   0   0   0   0   0 255 255 255   0   0   0
```



```
ECE5470:~/lab2 [12:59am] 105$vppr testim2_bound.mx
      0   1   2   3   4   5   6   7   8   9  10  11
 9    0   0   0   0   0   0   0   0   0   0   0   0
 8    0 255 255 255 255   0   0   0   0 255   0   0
 7    0 255 255   0 255   0   0   0 255 128 255   0
 6    0 255 128 255 128 255   0   0 255 128 255   0
 5    0 255 128 128 255 255   0   0 255 128 255   0
 4    0 255 128 255   0   0   0   0 255 128 255   0
 3    0 255 255   0   0 255 255 255 255 255 255   0
 2    0   0   0   0 255   0   0   0   0   0   0   0
 1    0   0   0 255   0   0 255 255 255   0   0   0
 0    0   0   0   0   0   0 255 255 255   0   0   0
```

Figure 2.1 the pixel value of compare of Testim1 and the Testim1_bound

Test image2(images and pixel values):



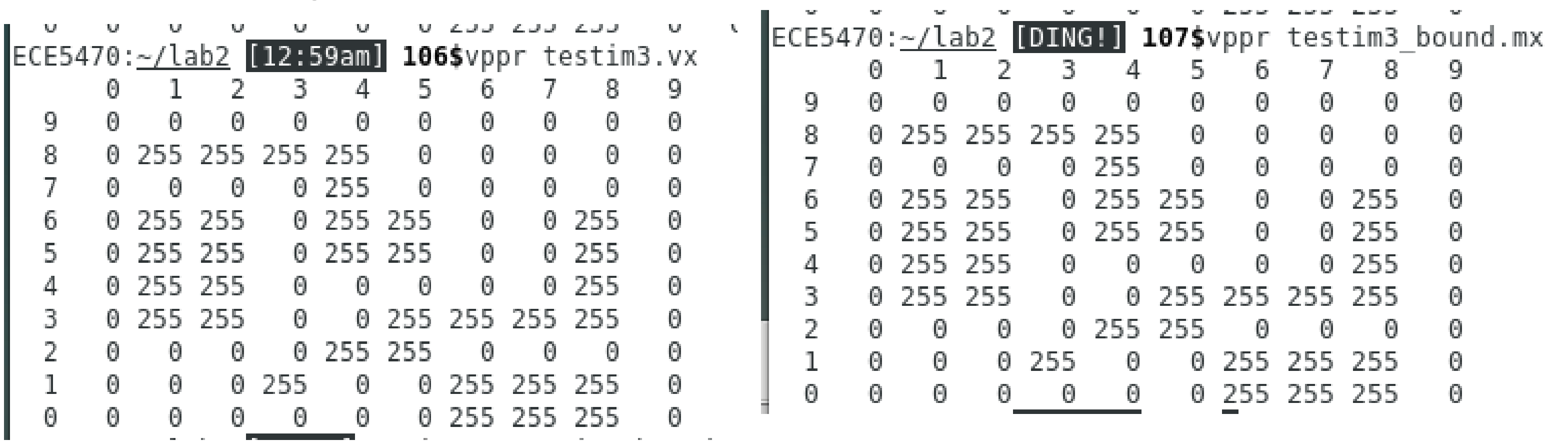
```
0   0   0   0   0   0   0   0   0   0   0   0
0 255 255 255 255   0   0   0   0 255   0   0
0 255 255   0 255   0   0   0 255 255 255   0
0 255 255 255 255 255   0   0 255 255 255   0
0 255 255 255 255 255   0   0 255 255 255   0
0 255 255 255   0   0   0   0 255 255 255   0
0 255 255   0   0 255 255 255 255 255 255   0
0   0   0   0 255   0   0   0   0   0   0   0
0   0   0 255   0   0 255 255 255   0   0   0
0   0   0   0   0   0 255 255 255   0   0   0
```

```
0   0   0   0   0   0   0   0   0   0   0   0
0 255 255 255 255   0   0   0   0 255   0   0
0 255 255   0 255   0   0   0 255 128 255   0
0 255 128 255 128 255   0   0 255 128 255   0
0 255 128 128 255 255   0   0 255 128 255   0
0 255 128 255   0   0   0   0 255 128 255   0
0 255 255   0   0 255 255 255 255 255 255   0
0   0   0   0 255   0   0   0   0   0   0   0
0   0   0 255   0   0 255 255 255   0   0   0
0   0   0   0   0   0 255 255 255   0   0   0
```

Figure 2.2 the image comparation of the testim1 and the testim1_bound

From the Figure 2.1, we can see the change of pixel values after bound processing clearly and Figure 2.2 shows the image change visually.

Test image3 (pixel value at terminal):

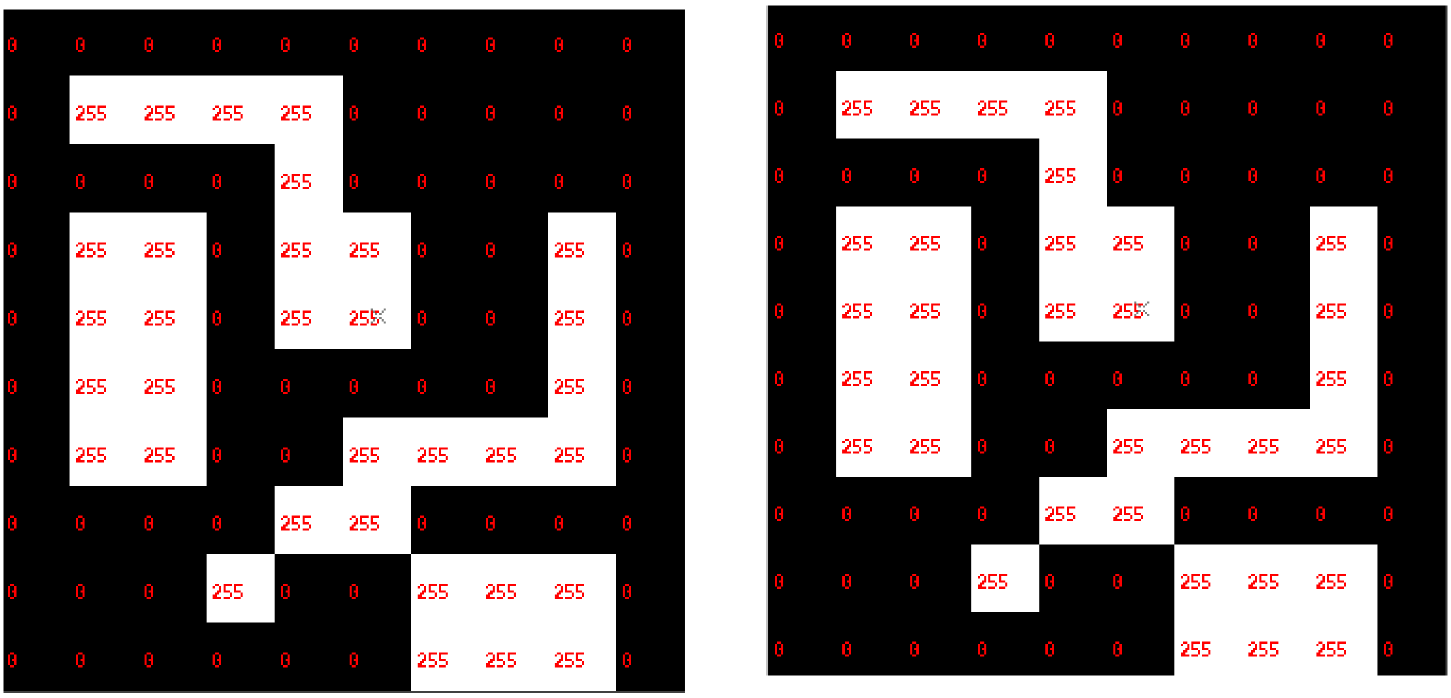


```
ECE5470:~/lab2 [12:59am] 106$vppr testim3.vx
      0   1   2   3   4   5   6   7   8   9
  9   0   0   0   0   0   0   0   0   0   0
  8   0 255 255 255 255   0   0   0   0   0
  7   0   0   0   0 255   0   0   0   0   0
  6   0 255 255   0 255 255   0   0 255   0
  5   0 255 255   0 255 255   0   0 255   0
  4   0 255 255   0   0   0   0   0 255   0
  3   0 255 255   0   0 255 255 255 255   0
  2   0   0   0   0 255 255   0   0   0   0
  1   0   0   0 255   0   0 255 255 255   0
  0   0   0   0   0   0   0 255 255 255   0
```

```
ECE5470:~/lab2 [DING!] 107$vppr testim3_bound.mx
      0   1   2   3   4   5   6   7   8   9
  9   0   0   0   0   0   0   0   0   0   0
  8   0 255 255 255 255   0   0   0   0   0
  7   0   0   0   0 255   0   0   0   0   0
  6   0 255 255   0 255 255   0   0 255   0
  5   0 255 255   0 255 255   0   0 255   0
  4   0 255 255   0   0   0   0   0 255   0
  3   0 255 255   0   0 255 255 255 255   0
  2   0   0   0   0 255 255   0   0   0   0
  1   0   0   0 255   0   0 255 255 255   0
  0   0   0   0   0   0   0 255 255 255   0
```

Figure 3.1 the pixel value of compare of Testim1 and the Testim1_bound

Test image3 (images and pixel values):



0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	255	255	0	255	255	0	0	255	0
0	255	255	0	255	255	0	0	255	0
0	255	255	0	0	0	0	0	255	0
0	255	255	0	0	255	255	255	255	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	0	0	255	255	255	0
0	0	0	0	0	0	255	255	255	0

0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	255	255	0	255	255	0	0	255	0
0	255	255	0	255	255	0	0	255	0
0	255	255	0	0	0	0	0	255	0
0	255	255	0	0	255	255	255	255	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	0	0	255	255	255	0
0	0	0	0	0	0	255	255	255	0

Figure 3.2 the image comparation of the testim1 and the testim1_bound

Actually, this test image is a little tricky, because all pixels are edges of image. So after applied my program to this image, the image does not change anything. Overall, these three image can prove that my program of bound is correct. Then, I will apply this program to process the full size image.

Testing the full size image:

The below figures are the pictures of gif that I used the command *vcapt if=im3.vx c=" input image im3.vx" of=im3_p1.vx* and *vxport -gif im3_p1.vx* to outputted picture. All in all, the bound process to this full-size image is successful and it can make the edge of the pictures.

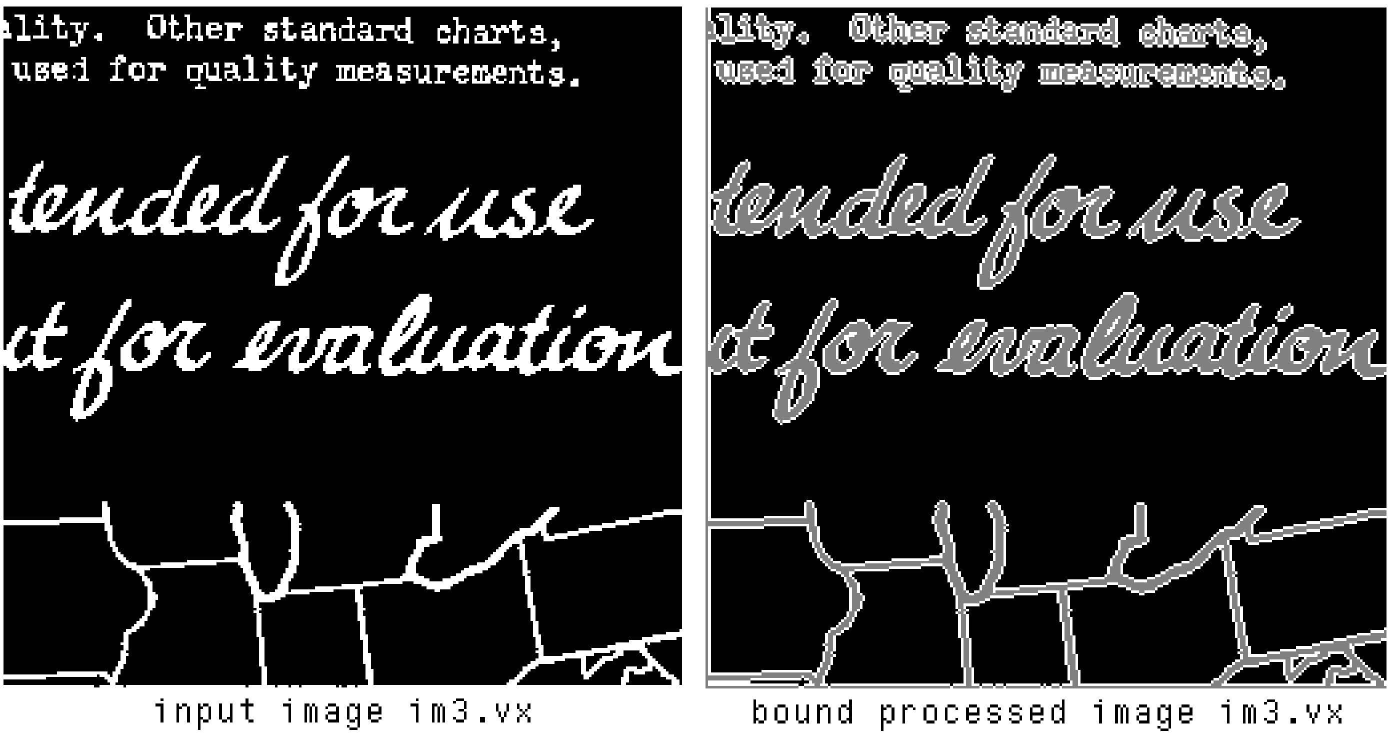


Figure 4.1 the compare of the full size image

Part Two: A Second Program---cclabel

Description of the program:

1. Firstly, we embed the image im into the image tm and add a one pixel border into the image. By this way, we can now use the image im to mark the boundary of the image as output since we already have a copy of original image in the image tm.
2. We set all pixel label of output image to zero (im[x, y] = 0) and set the label to 1(n=1).
3. Search every input image, tm, until an object pixel is found, which means the pixel value does not equal to zero (tm[x, y] != 0).
4. If the object pixel is not labeled(im[x, y] == 0), then call the recursive

function setlabel(x, y, n), where (x, y) is the location of pixel object and n is the label value.

5. Increase the label value(n++).
6. Repeat the step 3, 4, 5 for all unlabeled object pixels.

The “setlabel(x, y, n)” recursive function:

a) Set the pixel value whose location is (x, y) of output image to n (im[x, y] = n);
b) If the pixel above the given pixel is an object pixel (tm[y+1, x] != 0) and is not labeled (im[y+1, x] ==0), then call setlabel(x, y+1, n).
c) If the pixel below the given pixel is an object pixel (tm[y-1, x] != 0) and is not labeled (im[y-1, x] ==0), then call setlabel(x, y-1, n).
d) If the pixel left the given pixel is an object pixel (tm[y, x-1] != 0) and is not labeled (im[y, x-1] ==0), then call setlabel(x-1, y, n).
e) If the pixel right the given pixel is an object pixel (tm[y, x+1] != 0) and is not labeled (im[y, x+1] ==0), then call setlabel(x+1, y, n).

Code:

cclabel.c

```
/*********************************************************************/
/*cclabel      Compute label operation on a single byte image      */
/*********************************************************************/
/*@copyright                                                       */
/*Name: Yanling Wu                                                  */
/*NetID: yw996                                                      */
/*LAB Number: LAB 2                                                 */
/*********************************************************************/
```

```

#include "VisXV4.h"           /* VisionX structure include file    */
#include "Vutil.h"            /* VisionX utility header files      */

VXparam_t par[] =             /* command line structure            */
{ /* prefix, value,   description                         */
{    "if=",    0,   " input file  vtemp: local max filter "},
{    "of=",    0,   " output file "},
{     0,       0,   0}  /* list termination */
};
#define  IVAL   par[0].val
#define  OVAL   par[1].val

void setlabel(int ,int ,int );
Vfstruct (im);                      /* i/o image structure          */
Vfstruct (tm);                      /* temp image structure         */

main(argc, argv)
int argc;
char *argv[];
{
int        y,x;                     /* index counters               */
  VXparse(&argc, &argv, par);       /* parse the command line       */

  Vfread(&im, IVAL);                /* read image file              */
  Vfembed(&tm, &im, 1,1,1,1);       /* image structure with border  */
  if ( im.type != VX_PBYTE ) {      /* check image format           */
     fprintf(stderr, "vtemp: no byte image data in input file\n");
     exit(-1);
  }
  //set all pixels (labels) to 0
  for (y = im.ylo ; y <= im.yhi ; y++) {
        for (x = im.xlo; x <= im.xhi; x++)
          im.u[y][x] = 0;
   }
  int n =1;

  for (y = im.ylo ; y <= im.yhi ; y++) {
     for (x = im.xlo; x <= im.xhi; x++)  {
       if(tm.u[y][x] != 0 && im.u[y][x] == 0){
          setlabel(x,y,n);
          n++;
       }
     }
```

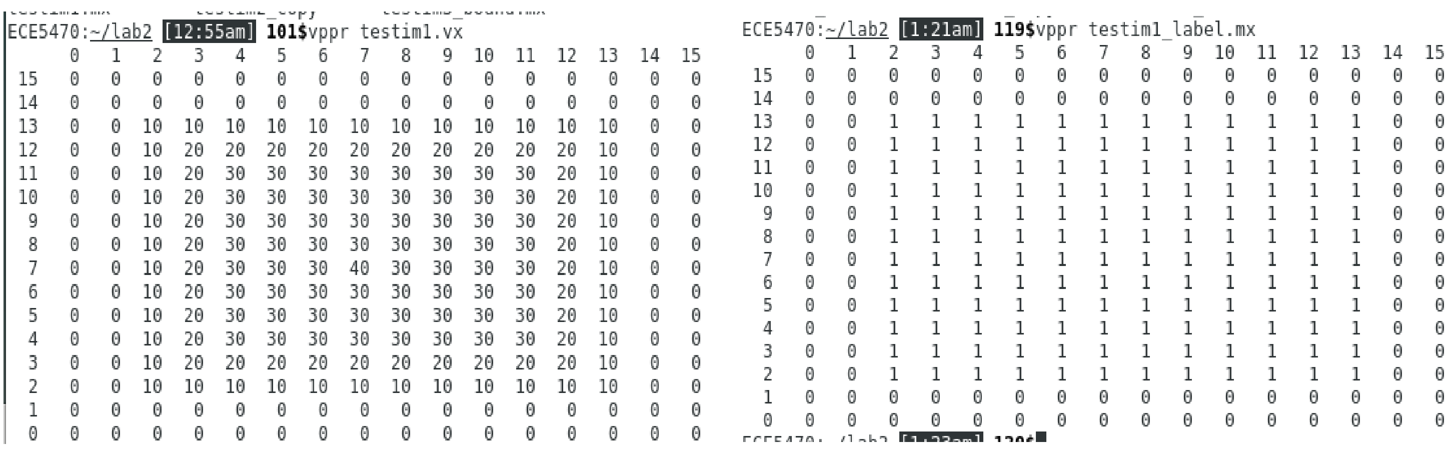
```c
    }

    Vfwrite(&im, OVAL);             /* write image file                */
    exit(0);
}

/*Set label recursive function*/
void setlabel(int x,int y,int L){
     im.u[y][x] = L;
     if ( tm.u[y+1][x] != 0 && im.u[y+1][x] == 0 )
         setlabel(x,y+1,L);
     if( tm.u[y-1][x] != 0 && im.u[y-1][x] == 0)
         setlabel(x,y-1,L);
     if(tm.u[y][x-1] != 0 && im.u[y][x-1] == 0)
         setlabel(x-1,y,L);
     if(tm.u[y][x+1] != 0 && im.u[y][x+1] == 0)
         setlabel(x+1,y,L);


}
```

Result:

Debug using small test images:

First, run "*vcc cclabel.c -o bound*" to compile this program and then debug this program with three small test images. Finally, test it with larger images.

Test image1(pixel value at terminal):

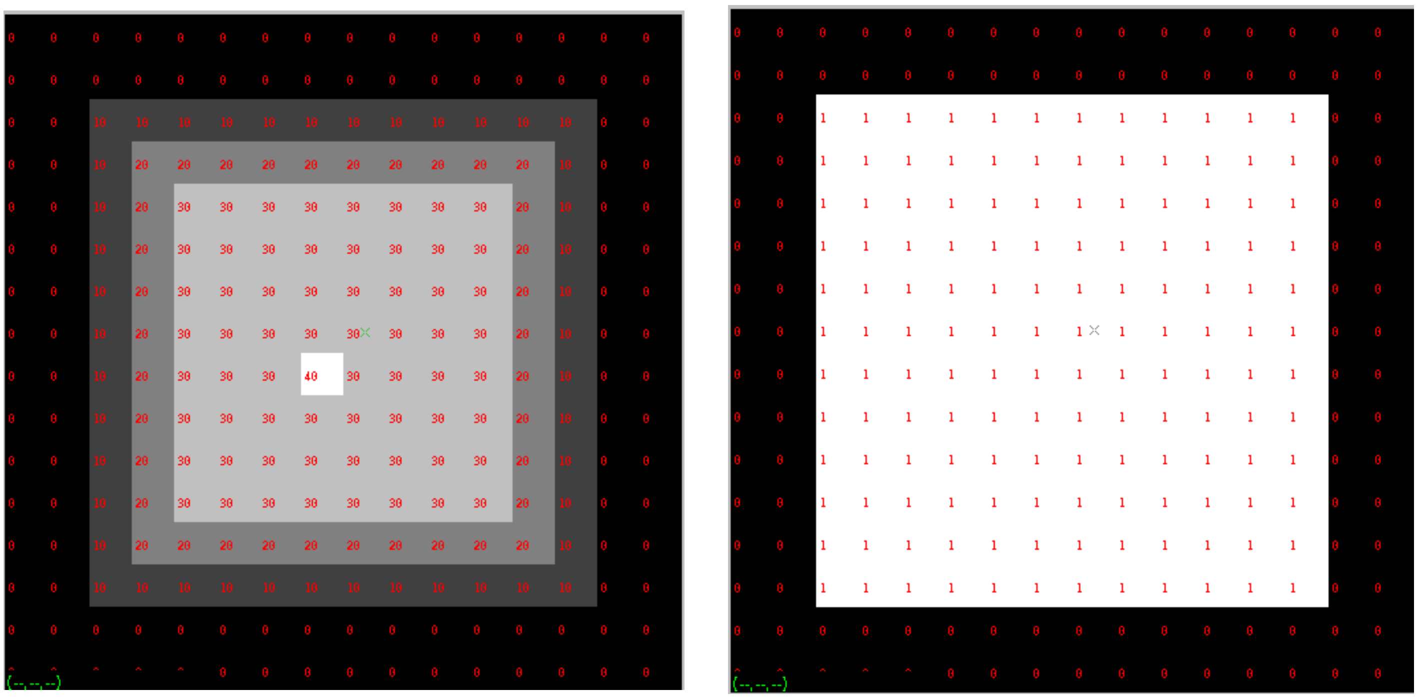


```
ECE5470:~/lab2 [12:55am] 101$vppr testim1.vx
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
 12   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
 11   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
 10   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  9   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  8   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  7   0   0  10  20  30  30  30  40  30  30  30  30  20  10   0   0
  6   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  5   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  4   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  3   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
  2   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
ECE5470:~/lab2 [1:21am] 119$vppr testim1_label.mx
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
 12   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
 11   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
 10   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  9   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  8   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  7   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  6   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  5   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  4   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  3   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  2   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Figure 5.1 the pixel value of compare of Testim1 and the Testim1_label

Test image1(image and pixel value):



```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  10 10 10 10 10 10 10 10 10 10 10 10 0  0
0  0  10 20 20 20 20 20 20 20 20 20 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 40 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 30 30 30 30 30 30 30 30 20 10 0  0
0  0  10 20 20 20 20 20 20 20 20 20 20 10 0  0
0  0  10 10 10 10 10 10 10 10 10 10 10 10 0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
^  ^  ^  ^  ^  0  0  0  0  0  0  0  0  0  0  0
(--,--,--)
```

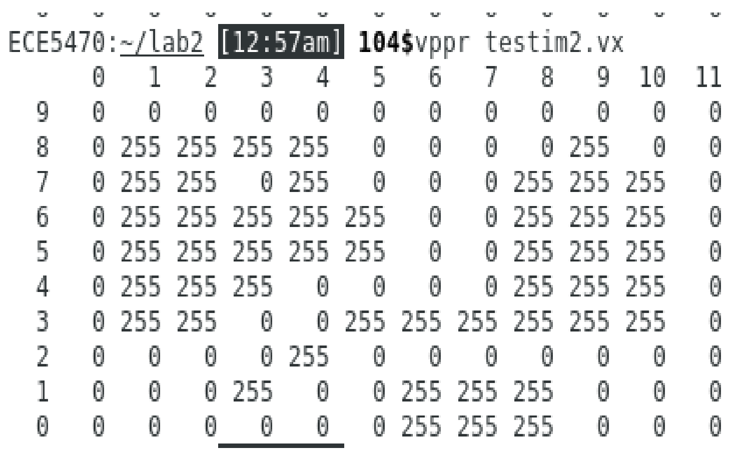
```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
^  ^  ^  ^  ^  0  0  0  0  0  0  0  0  0  0  0
(--,--,--)
```

Figure 5.2 the image comparation of the testim1 and the testim1_label

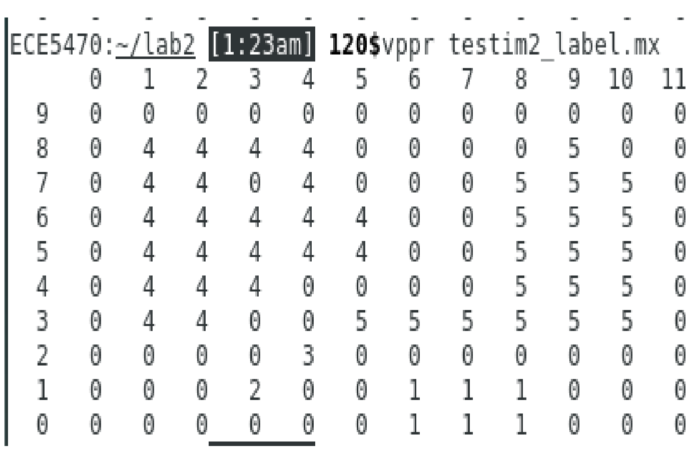
Figure 5.1 shows that the pixel value change at the terminal clearly and figure 5.2 shows this change visually.

In fact, there are some ambiguities in this cclabel program, which is about how to label and the criteria. If I suppose that if one pixel is not zero and they are connected, then they would be in same label. But if my criteria is one pixel is not zero and the adjacent pixel has the same pixel value, then they can be recognized the same label. I choose the former as my criteria. So my program would be as the figure5.2 shows.

Test image2 (pixel value at terminal):



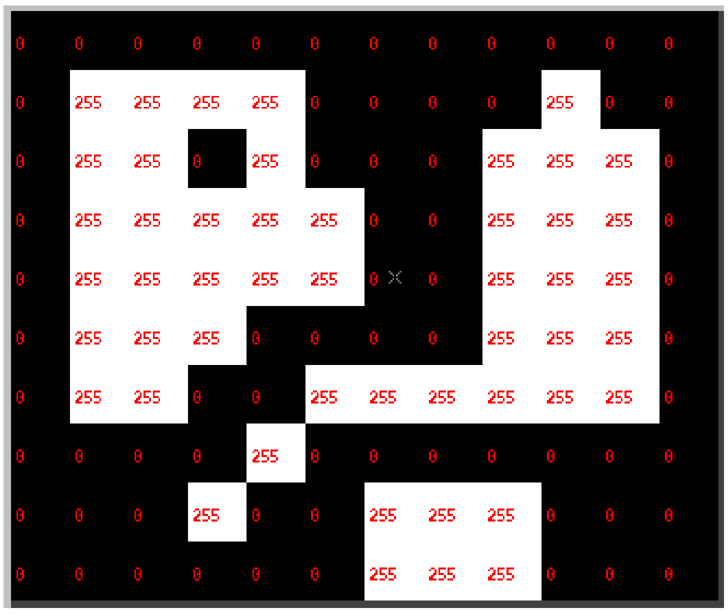
```
ECE5470:~/lab2 [12:57am] 104$vppr testim2.vx
        0   1   2   3   4   5   6   7   8   9  10  11
   9    0   0   0   0   0   0   0   0   0   0   0   0
   8    0 255 255 255 255   0   0   0   0 255   0   0
   7    0 255 255   0 255   0   0   0 255 255 255   0
   6    0 255 255 255 255 255   0   0 255 255 255   0
   5    0 255 255 255 255 255   0   0 255 255 255   0
   4    0 255 255 255   0   0   0   0 255 255 255   0
   3    0 255 255   0   0 255 255 255 255 255 255   0
   2    0   0   0   0 255   0   0   0   0   0   0   0
   1    0   0   0 255   0   0 255 255 255   0   0   0
   0    0   0   0   0   0   0 255 255 255   0   0   0
```



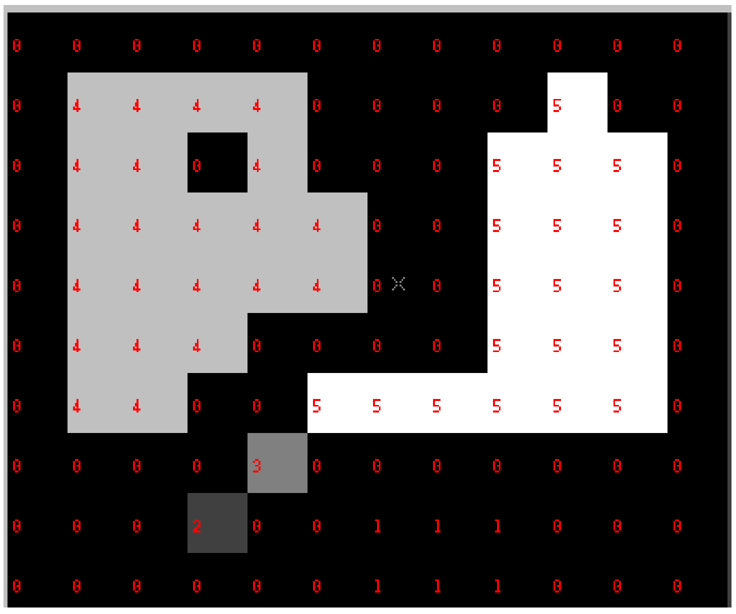
```
ECE5470:~/lab2 [1:23am] 120$vppr testim2_label.mx
        0   1   2   3   4   5   6   7   8   9  10  11
   9    0   0   0   0   0   0   0   0   0   0   0   0
   8    0   4   4   4   4   0   0   0   0   5   0   0
   7    0   4   4   0   4   0   0   0   5   5   5   0
   6    0   4   4   4   4   4   0   0   5   5   5   0
   5    0   4   4   4   4   4   0   0   5   5   5   0
   4    0   4   4   4   0   0   0   0   5   5   5   0
   3    0   4   4   0   0   5   5   5   5   5   5   0
   2    0   0   0   0   3   0   0   0   0   0   0   0
   1    0   0   0   2   0   0   1   1   1   0   0   0
   0    0   0   0   0   0   0   1   1   1   0   0   0
```

Figure 6,1 the pixel value of compare of Testim2 and the Testim2_label

Test image2(image and pixel value):



```
0   0   0   0   0   0   0   0   0   0   0   0
0   255 255 255 255 0   0   0   0   255 0   0
0   255 255 0   255 0   0   0   255 255 255 0
0   255 255 255 255 255 0   0   255 255 255 0
0   255 255 255 255 255 0   0   255 255 255 0
0   255 255 255 0   0   0   0   255 255 255 0
0   255 255 0   0   255 255 255 255 255 255 0
0   0   0   0   255 0   0   0   0   0   0   0
0   0   0   255 0   0   255 255 255 0   0   0
0   0   0   0   0   0   255 255 255 0   0   0
```

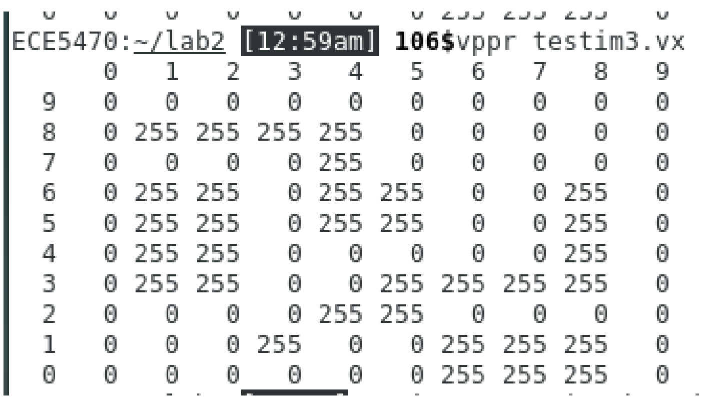


```
0   0   0   0   0   0   0   0   0   0   0   0
0   4   4   4   4   0   0   0   0   5   0   0
0   4   4   0   4   0   0   0   5   5   5   0
0   4   4   4   4   4   0   0   5   5   5   0
0   4   4   4   4   4   0   0   5   5   5   0
0   4   4   4   0   0   0   0   5   5   5   0
0   4   4   0   0   5   5   5   5   5   5   0
0   0   0   0   3   0   0   0   0   0   0   0
0   0   0   2   0   0   1   1   1   0   0   0
0   0   0   0   0   0   1   1   1   0   0   0
```

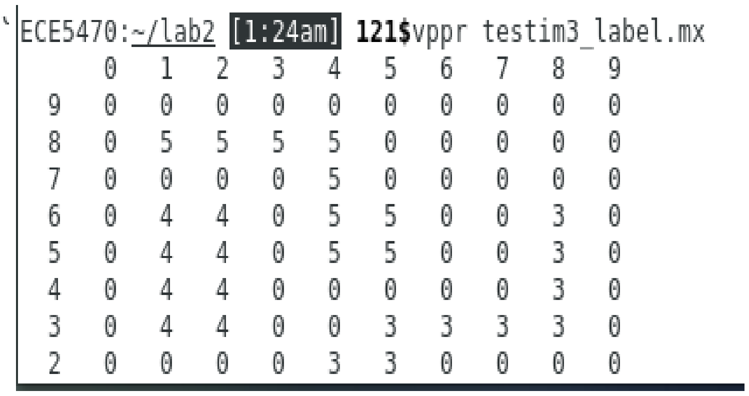
Figure 6.2 the image comparation of the testim2 and the testim2_label

Figure 5.1 shows that the pixel value change at the terminal clearly and figure 5.2 shows this change visually and instinctively.

Test image3 (pixel value at terminal):



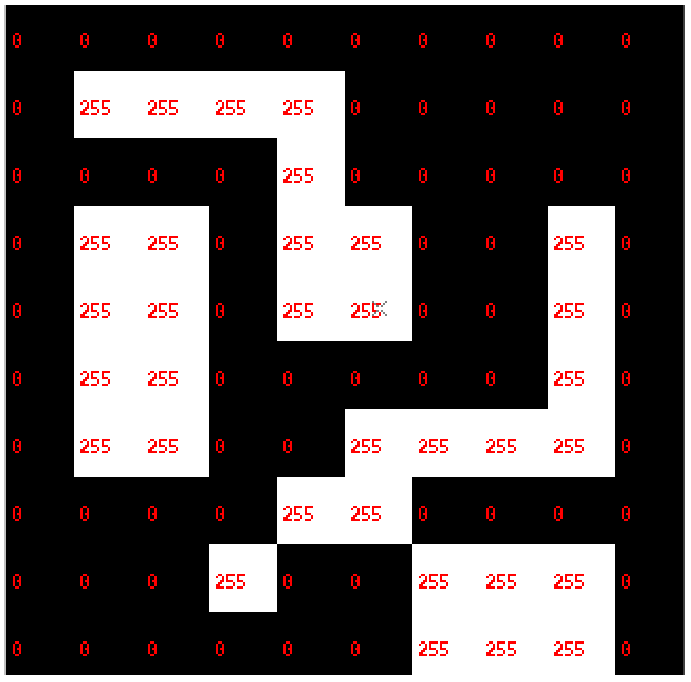
```
ECE5470:~/lab2 [12:59am] 106$vppr testim3.vx
      0   1   2   3   4   5   6   7   8   9
  9   0   0   0   0   0   0   0   0   0   0
  8   0 255 255 255 255   0   0   0   0   0
  7   0   0   0   0 255   0   0   0   0   0
  6   0 255 255   0 255 255   0   0 255   0
  5   0 255 255   0 255 255   0   0 255   0
  4   0 255 255   0   0   0   0   0 255   0
  3   0 255 255   0   0 255 255 255 255   0
  2   0   0   0   0 255 255   0   0   0   0
  1   0   0   0 255   0   0 255 255 255   0
  0   0   0   0   0   0   0 255 255 255   0
```



```
ECE5470:~/lab2 [1:24am] 121$vppr testim3_label.mx
      0   1   2   3   4   5   6   7   8   9
  9   0   0   0   0   0   0   0   0   0   0
  8   0   5   5   5   5   0   0   0   0   0
  7   0   0   0   0   5   0   0   0   0   0
  6   0   4   4   0   5   5   0   0   3   0
  5   0   4   4   0   5   5   0   0   3   0
  4   0   4   4   0   0   0   0   0   3   0
  3   0   4   4   0   0   3   3   3   3   0
  2   0   0   0   0   3   3   0   0   0   0
```

Figure 7.1 the pixel value of compare of Testim3 and the Testim3_label

Test image3(image and pixel value):



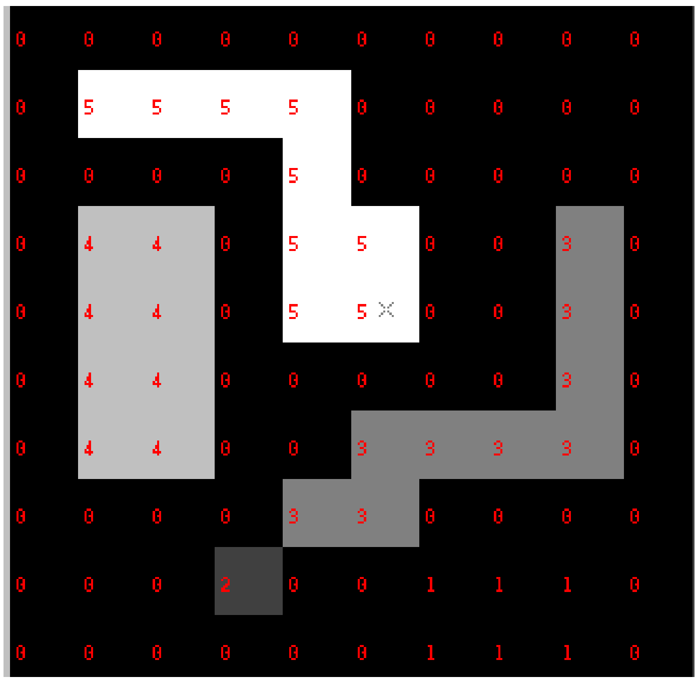


Figure 7.2 the image comparation of the testim3 and the testim3_label

Figure 6.1 shows that the pixel value change at the terminal clearly and figure 6.2 shows this change visually and instinctively. Also, the label is actually the pixel value so we can see there are different pixel values in the cclabel processed image.

To sum up, these test images prove that my cclabel program can give the correct label to different images.

Testing the full size image:

The figure 8.1 are the pictures of gif that I used the command *vcapt if=im3.vx c=" input image im3.vx" of=im3_p1.vx* and *vxport -gif im3_p1.vx* to outputted picture. All in all, the label process to this full-size image is successful. Because the comparation is unclear in cclabel processed image im3.vx, I also output the inverted version of these two picture as figure 8.2 shown. In those picture, we can see the degree of dark and white are different in different parts of pictures.

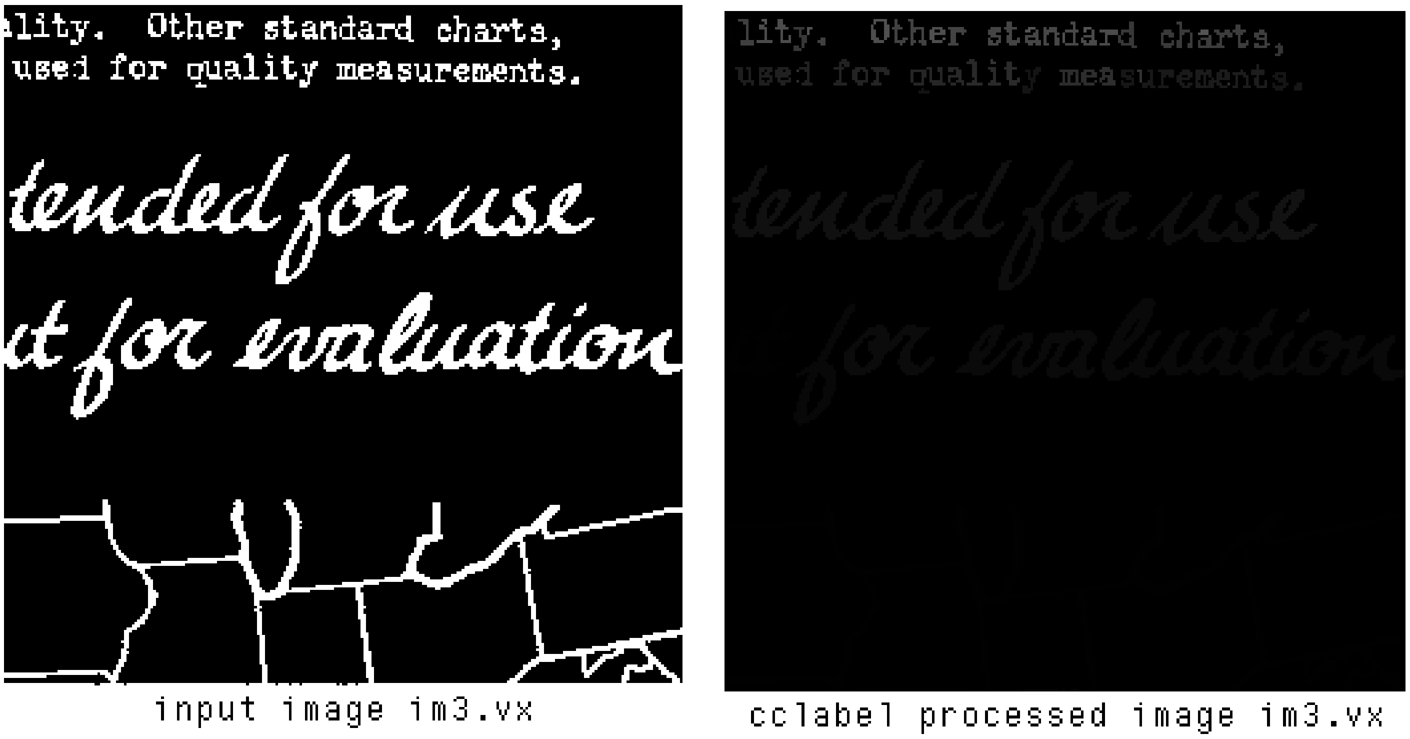
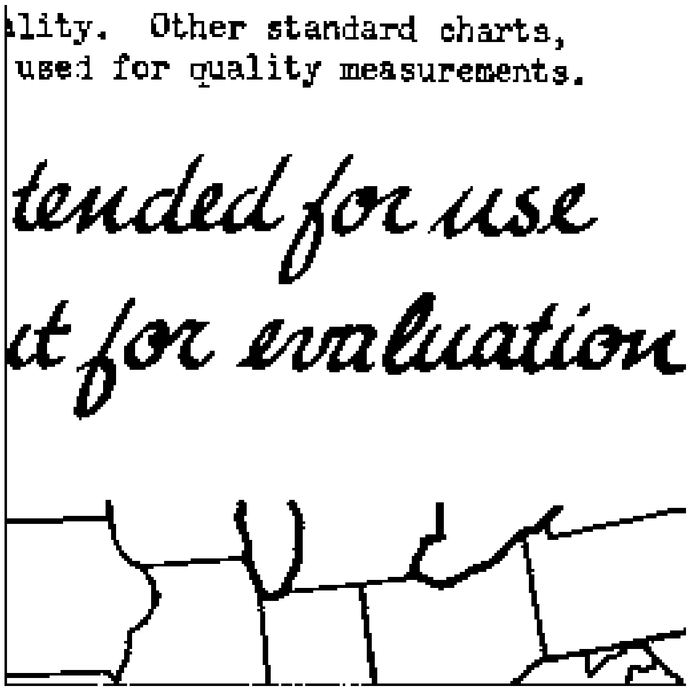
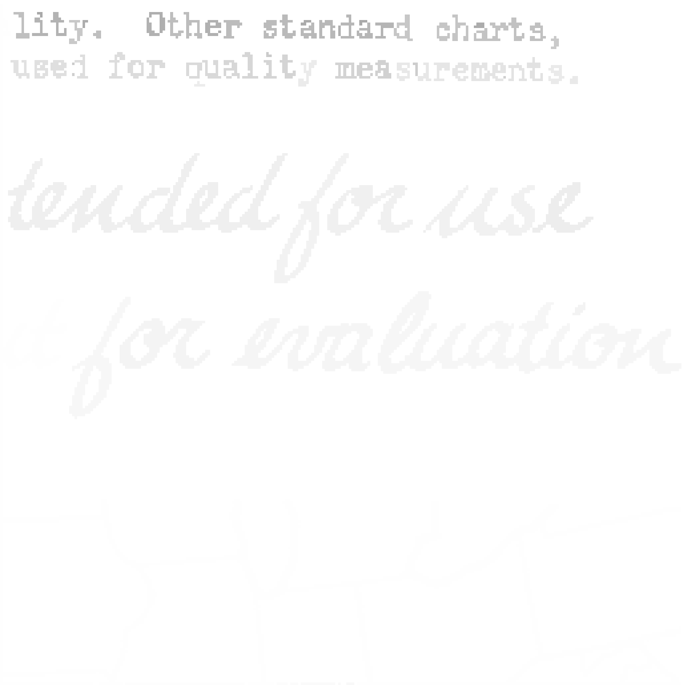

lity. Other standard charts,
used for quality measurements.
tended for use
t for evaluation
input image im3.vx
lity. Other standard charts,
used for quality measurements.
tended for use
for evaluation
cclabel processed image im3.vx

Figure 8.1 the compare of the full size image


lity. Other standard charts,
used for quality measurements.

tended for use
t for evaluation

inverted image im3.vx


lity. Other standard charts,
used for quality measurements.

tended for use
t for evaluation

inverted labeled image im3.vx

Figure 8.2 the compare of the inverted full size image