Title: Lab2 – Binary Image Processing Report

Name:Yuixang Long

NetID:yl3377

Part 1: The principle of the program for section 4

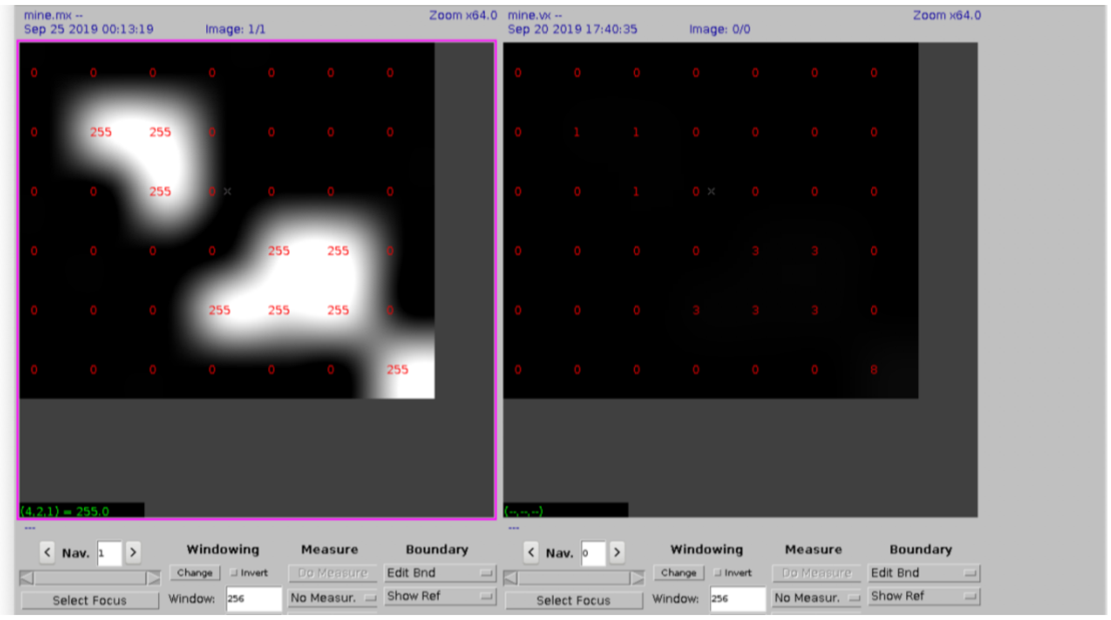
In the Section 4, it is required that we need to write a program to display the boundary of the image and set the different part of the image to different pixel values (The segment boundary pixels set to 255, the interior part of the object set to 128 and the background of the image set to zero). The main principle of my program is as followed:
1. The pixel value of the background in the image is always 0, so the pixel value that is not zero belongs to the object. As a result, I write a “for” circle to detect all the pixel value in the image, if the value is not zero, then set the pixel value (both im.u[y][x] and tm.u[y][x]) to 128, so the pixels of the object is all set to be 128, which includes the interior part of the object and the boundary of the object.
2. Then the program will detect the 4 pixel value which surround the pixel of the object (up, down, left, right). I one of the value of these four pixel is detected to be zero, that means this pixel is the boundary of the object because it separates the background and the object. Then set the value of this pixel (im.u[y][x])to be 255. As a result, the boundary of the object is detected. Work done!
In the program, the &im will take the input image firstly and then generates the output to the output image. We need to use the “Vfembed” function to extend the boundary of the whole image in the beginning, for the reason to avoid segmentation fault.

Part 2: The listing of the program for section 4

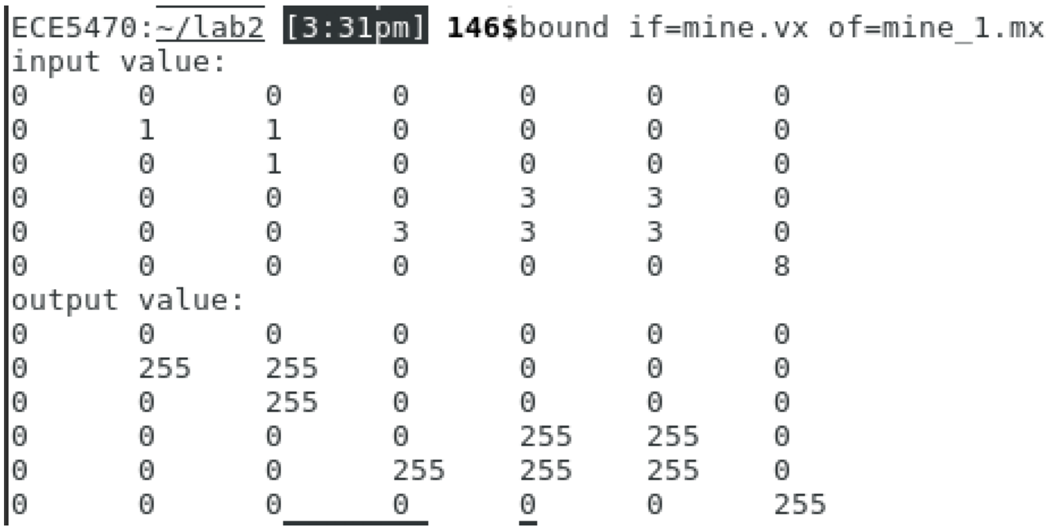
```
/*******************************************************************/
/* vtemp      Compute local max operation on a single byte image   */
/*******************************************************************/

#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);
  }
  fprintf(stderr, "input value:\n");
  for (y = im.yhi;y>=im.ylo;y--){
     for (x = im.xlo;x<=im.xhi;x++){
                if(im.u[y][x]>=100){fprintf(stderr, "%d    ",im.u[y][x]);}
                else if((im.u[y][x]>=10) && (im.u[y][x]<100)){fprintf(stderr, "%d     ",im.u[y] [x]);}
                else {fprintf(stderr, "%d      ",im.u[y][x] );}
                   }
                fprintf(stderr,"\n");
     }
  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][x] = 128;
                   im.u[y][x] = 128;}
                   }

     }

  for (y = im.ylo ; y <= im.yhi ; y++) {  /* compute the function */
     for (x = im.xlo; x <= im.xhi; x++)  {     /***********************/
                if(tm.u[y][x]==128&(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;}
                   }


     }
 fprintf(stderr, "output value:\n");
  for (y = im.yhi;y>=im.ylo;y--){
     for (x = im.xlo;x<=im.xhi;x++){
                if(im.u[y][x]>=100){fprintf(stderr, "%d    ",im.u[y][x]);}
                else if((im.u[y][x]>=10) && (im.u[y][x]<100)){fprintf(stderr, "%d     ",im.u[y][x]);}
                else {fprintf(stderr, "%d      ",im.u[y][x] );}
                   }
                fprintf(stderr,"\n");
     }
   Vfwrite(&im, OVAL);              /* write image file             */
   exit(0);
}
```

Part 3: The result of the program for a small size image

The input image is “mine.vx”, and the output image is “mine_1.mx”. The screenshot are as followed.

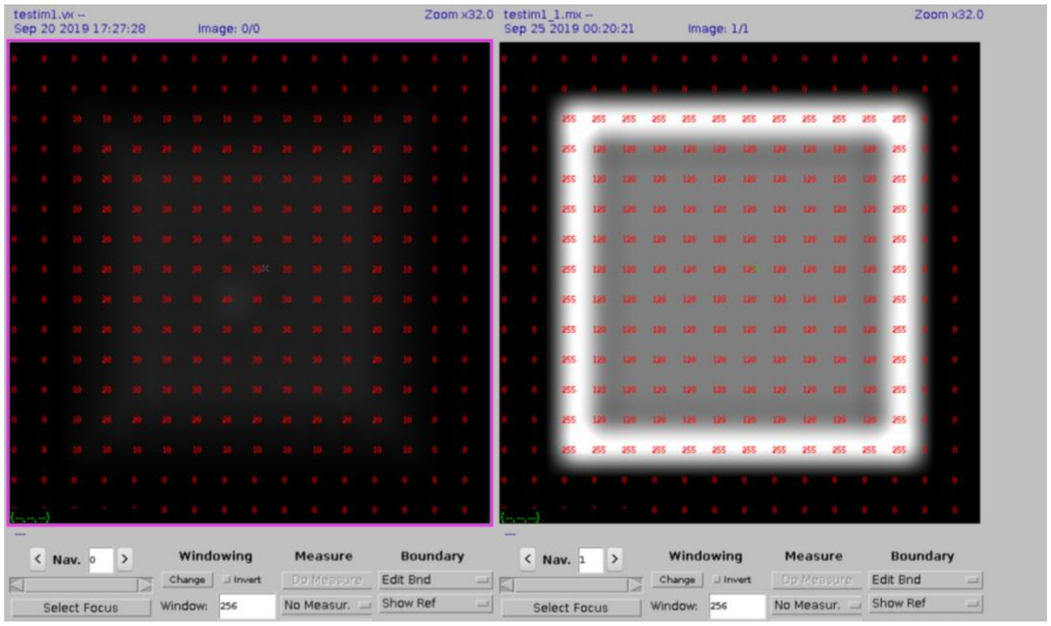


As shown in the two images, the boundary is detected by the program, and the pixel value of the boundary is set to be 255, but all the object pixel are boundary, so the interior is not set. To see the function of the program to detect the boundary and interior of the object, we can turn to the next part, which runs on full size image. The input and output value displayed in the terminal is as followed: (input is mine.vx, output is mine_1.vx)



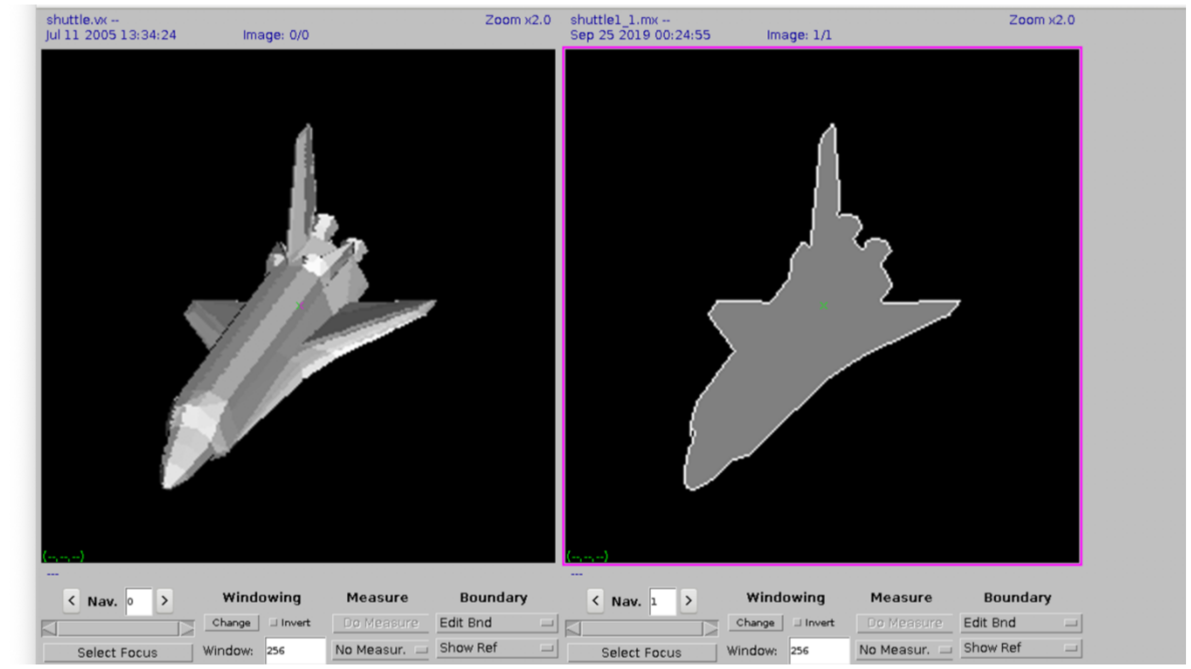
```
ECE5470:~/lab2 [3:31pm] 146$bound if=mine.vx of=mine_1.mx
input value:
0       0       0       0       0       0       0
0       1       1       0       0       0       0
0       0       1       0       0       0       0
0       0       0       0       3       3       0
0       0       0       3       3       3       0
0       0       0       0       0       0       8
output value:
0       0       0       0       0       0       0
0       255     255     0       0       0       0
0       0       255     0       0       0       0
0       0       0       0       255     255     0
0       0       0       255     255     255     0
0       0       0       0       0       0       255
```

The followed is another example: the input is testim1.vx, the output is testim1_1.mx.

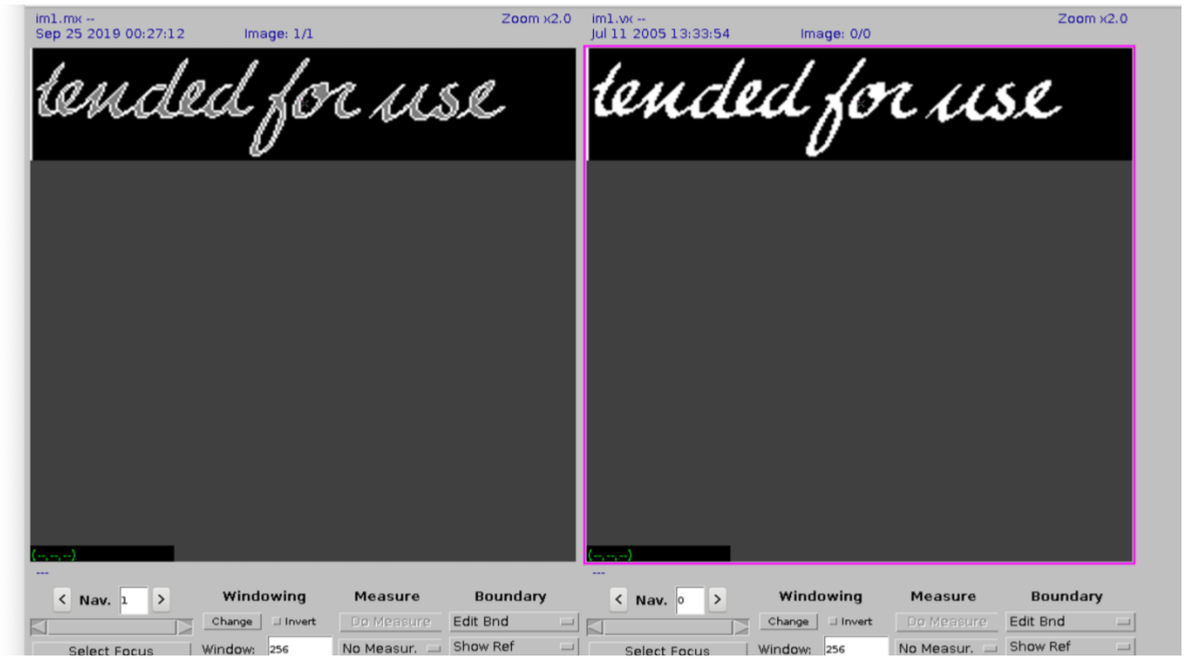


Part 4: The result of the program for a full size image

The input image is “shuttle.vx”, and the output image is “shuttle1.mx”, the result is as followed:



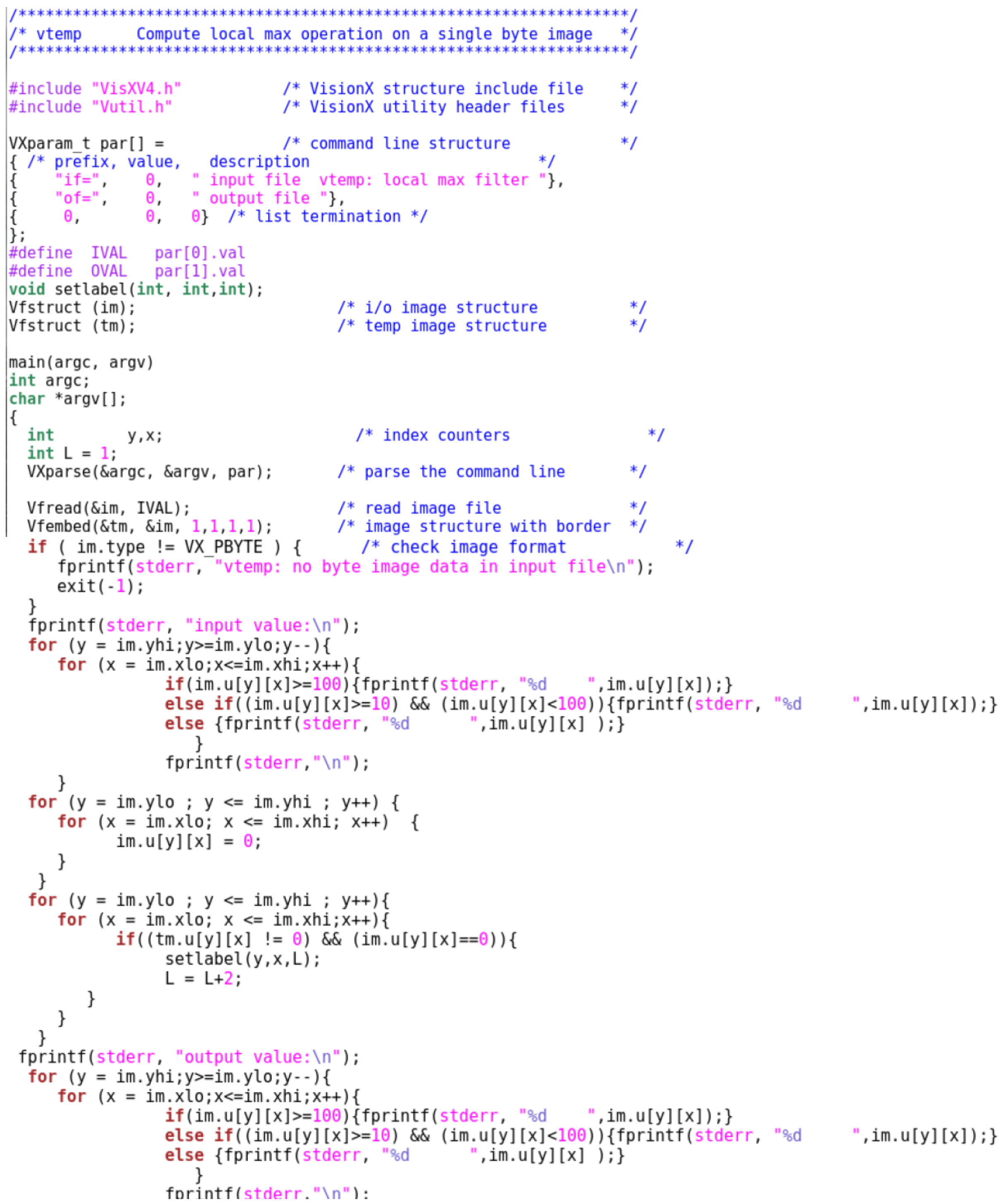
And this is another example: The input image is “im1.vx”, the output is “im1.mx”



Part 5: The principle of the program for section 5

In the Section 5, it is required that we need to write a program to generate a output image which labels the different object part (every part is formed by connected pixels) in the input image. And the pixels from the same connective part have the same label, different connective part have different sequential label numbers from 1 to n. The main principle of the program is as followed:
1. Firstly, the program takes the input image and store it in “&im”. Then use Vfembed command to extend the boundary of the im, and store the new image to “&tm”. Now because we have the image stored in “&tm”, we can set the pixels of the image stored in “&im” all to zero. The reason we do this is because we need to generate the labeled image and store it in “&im”.
2. Then we search the image in “&tm”, until we found the object pixel whose pixel value is not zero. Meanwhile the pixel must not be labeled. When we found that, we call the set label function “setlabel(y,x,L)”,   “L” stand for the label number. After that, we do increment to L, in order to make the label more noticeable, we set the increment to be 2, so the label is 1, 3, 5…
3. The “setlabel” function implements the recursive method, firstly, we set the object pixel value to be L, and search the four pixels around the pixel “tm.u[y][x]”, when one of them is connect to the “tm.u[y][x]”, we call the “setlabel” function again until there is no connective pixels. This is the recursive method. After all the pixels are searched, the program exits.

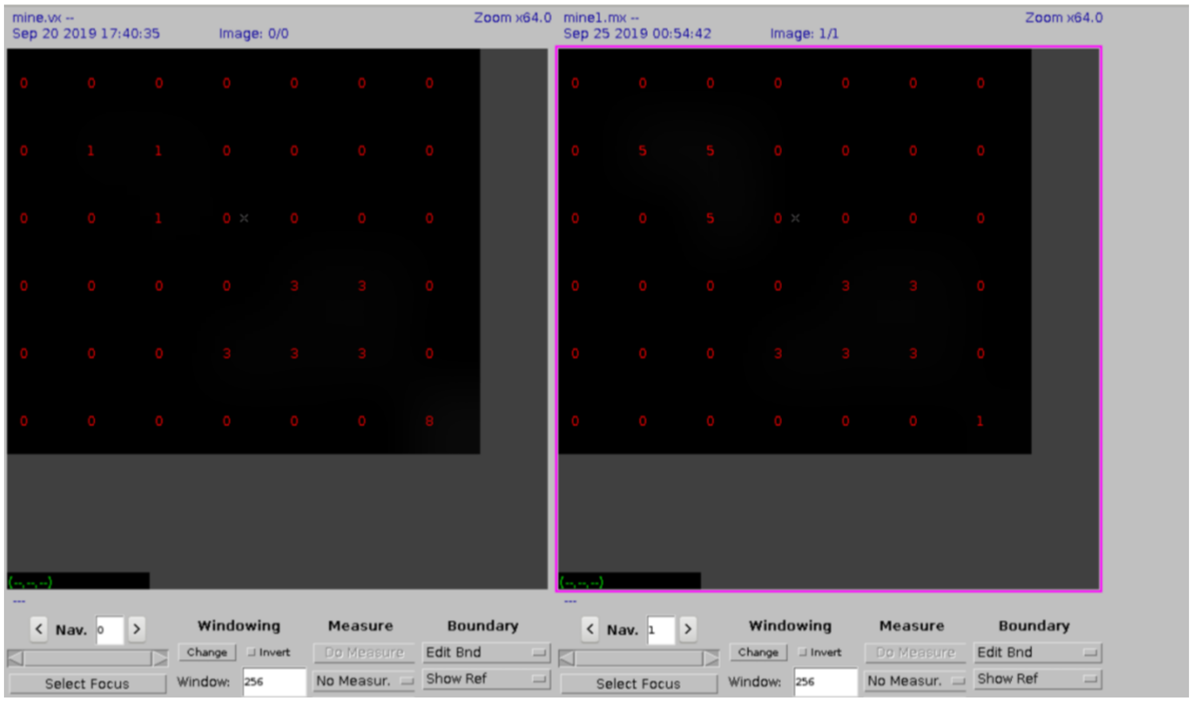
Part 6: The listing of the program for section 5



```
/*******************************************************************/
/* vtemp      Compute local max operation on a single byte image   */
/*******************************************************************/

#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               */
  int L = 1;
  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);
  }
  fprintf(stderr, "input value:\n");
  for (y = im.yhi;y>=im.ylo;y--){
     for (x = im.xlo;x<=im.xhi;x++){
                if(im.u[y][x]>=100){fprintf(stderr, "%d    ",im.u[y][x]);}
                else if((im.u[y][x]>=10) && (im.u[y][x]<100)){fprintf(stderr, "%d     ",im.u[y][x]);}
                else {fprintf(stderr, "%d      ",im.u[y][x] );}
                   }
                fprintf(stderr,"\n");
     }
  for (y = im.ylo ; y <= im.yhi ; y++) {
     for (x = im.xlo; x <= im.xhi; x++)  {
           im.u[y][x] = 0;
     }
   }
  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(y,x,L);
                L = L+2;
        }
     }
   }
 fprintf(stderr, "output value:\n");
  for (y = im.yhi;y>=im.ylo;y--){
     for (x = im.xlo;x<=im.xhi;x++){
                if(im.u[y][x]>=100){fprintf(stderr, "%d    ",im.u[y][x]);}
                else if((im.u[y][x]>=10) && (im.u[y][x]<100)){fprintf(stderr, "%d     ",im.u[y][x]);}
                else {fprintf(stderr, "%d      ",im.u[y][x] );}
                   }
                fprintf(stderr,"\n");
     }
   Vfwrite(&im, OVAL);             /* write image file                */
   exit(0);
}

/* function to compute the local maximum */
void setlabel(int y, int x, int L)
{
    im.u[y][x] = L;
    if ((tm.u[y+1][x] != 0) && (im.u[y+1][x] == 0)){
        setlabel(y+1,x,L);}
    if ((tm.u[y-1][x] != 0) && (im.u[y-1][x] == 0)){
        setlabel(y-1,x,L);}
    if ((tm.u[y][x-1] != 0) && (im.u[y][x-1] == 0)){
        setlabel(y,x-1,L);}
    if ((tm.u[y][x+1] != 0) && (im.u[y][x+1] == 0)){
        setlabel(y,x+1,L);}
}
```

Part 7: The result of the program for a small size image

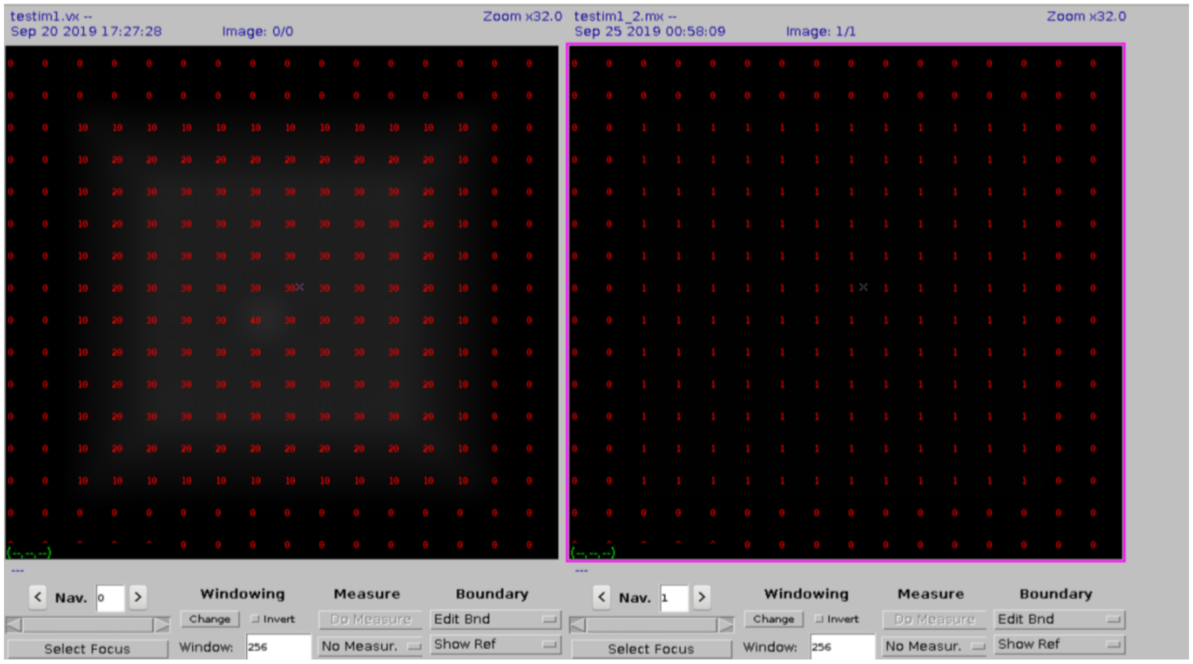
The input image is “mine.vx”, and the output image is “mine.mx”. The screenshot are as followed.



The input and output value displayed in the terminal is as followed: (input is mine.vx, output is mine_1.vx)

```
input value:
0       0       0       0       0       0       0
0       1       1       0       0       0       0
0       0       1       0       0       0       0
0       0       0       0       3       3       0
0       0       0       3       3       3       0
0       0       0       0       0       0       8
output value:
0       0       0       0       0       0       0
0       5       5       0       0       0       0
0       0       5       0       0       0       0
0       0       0       0       3       3       0
0       0       0       3       3       3       0
0       0       0       0       0       0       1
```

The followed is another example: the input is testim1.vx, the output is testim1_2.mx.



```
testim1.vx --                                             Zoom x32.0
Sep 20 2019 17:27:28          Image: 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   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
---
```

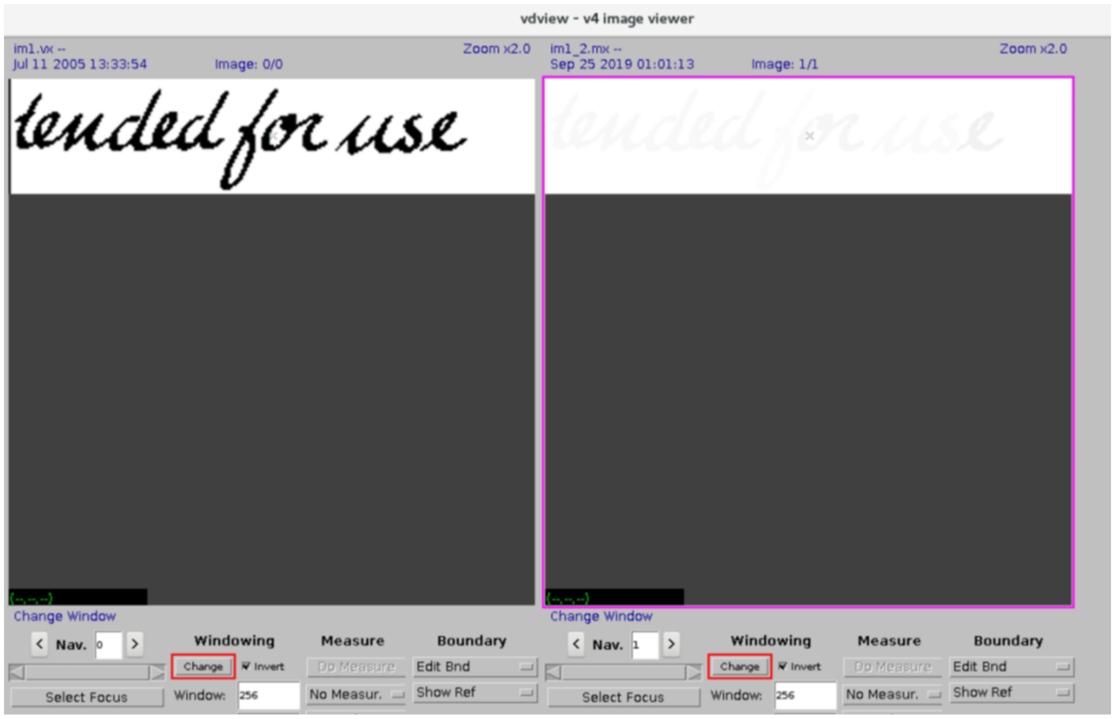
```
testim1_2.mx --                                           Zoom x32.0
Sep 25 2019 00:58:09          Image: 1/1

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   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
---
```

Nav.: 0 — Windowing: Window: 256 — Measure: No Measur. — Boundary: Edit Bnd, Show Ref

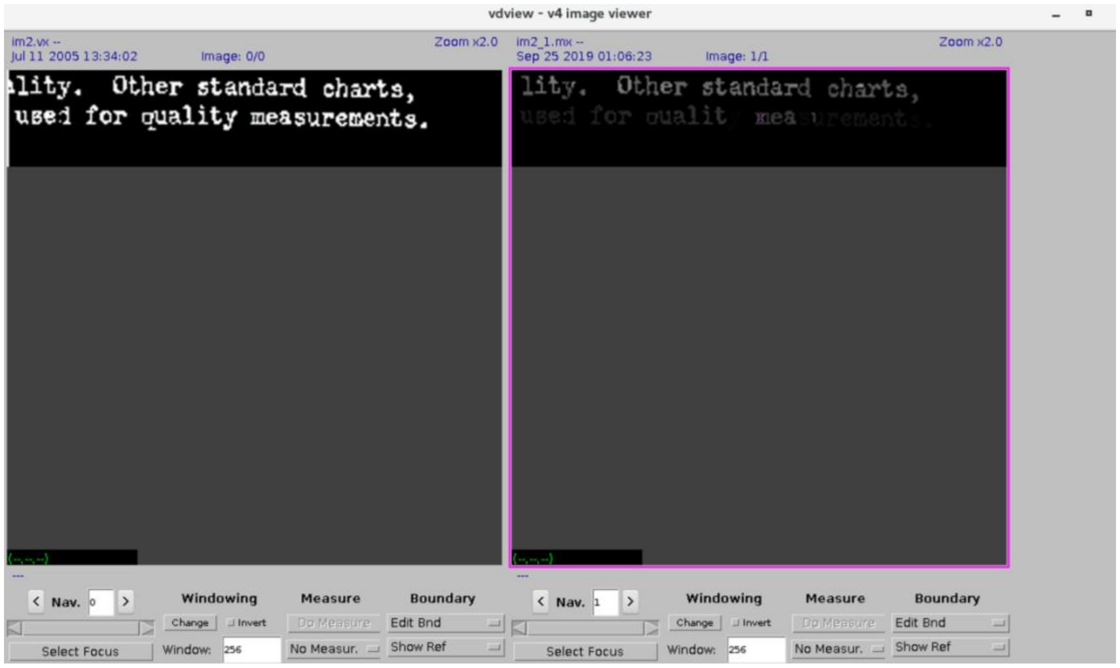
Nav.: 1 — Windowing: Window: 256 — Measure: No Measur. — Boundary: Edit Bnd, Show Ref

Part 8: The result of the program for a full size image

The input image is "im1.vx", and the output image is "im1_2.mx", the result is as followed:



Here is another example: the input image is "image2.vx", and the output image is "image2_1.mx".


vdview - v4 image viewer
im2.vx --
Jul 11 2005 13:34:02
Image: 0/0
Zoom x2.0
lity. Other standard charts,
used for quality measurements.
im2_1.mx --
Sep 25 2019 01:06:23
Image: 1/1
Zoom x2.0
Nav.
Windowing
Change
Invert
Window: 256
Measure
Do Measure
No Measur.
Boundary
Edit Bnd
Show Ref
Select Focus