Lab 6: Three-Dimensional Image Processing

ECE 5470

Computer Vision

Name: Yanling Wu

NetID: yw996

Section 2: Three-dimensional Filtering

In this section, we should generate one 3D byte test image first. And then try to smooth the polygons. Finally, experiment with simple three-dimensional morphological filtering.

Q1: How does t1f.vd compare with t1.vd? Would you always want to perform this action on polygons?

The figure 1.1 shows the image of t1.vd and the y-view of it and the figure 1.2 shows the image of t1f.vd and the y-view of it. From these four figures, we can see the t1f.vd becomes smoother than t1.vd. It is more obvious if comparing the y-view of these two files. There are more layers on y-view of t1.vd. So, the method of "*v3pflit*" command can successfully make the coarse surface of polygons smoother. In this method, we use default parameter value, which is the new location of a vertex depends 90% on the locations of neighboring vertices and 10% on its original location.

I do not think we always want to perform this action on polygons. Because sometimes we need to see the original picture and the layers of the original design image. For example, sometimes, the size of protrusions around the polygons is very important for making the mechanical parts, but if we use this way to process the polygons, the size will change.

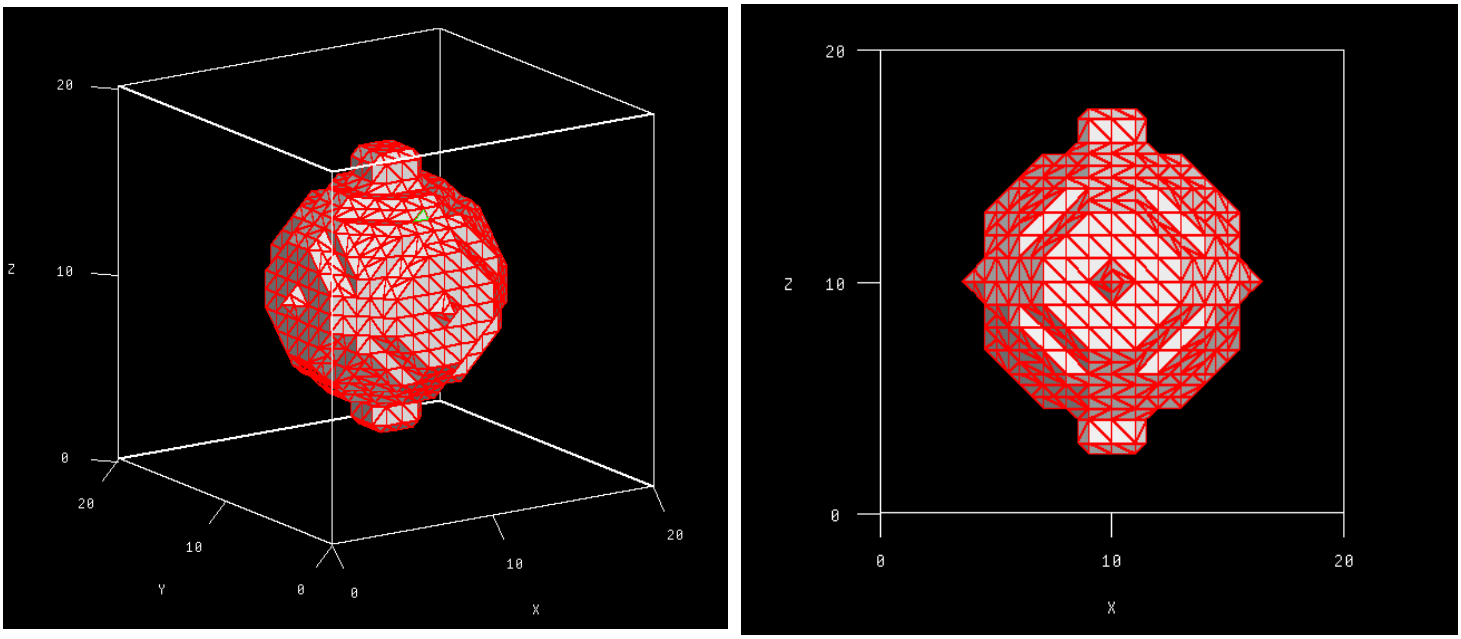

Figure 1.1 the 3D image of t1.vd and the y-view of it

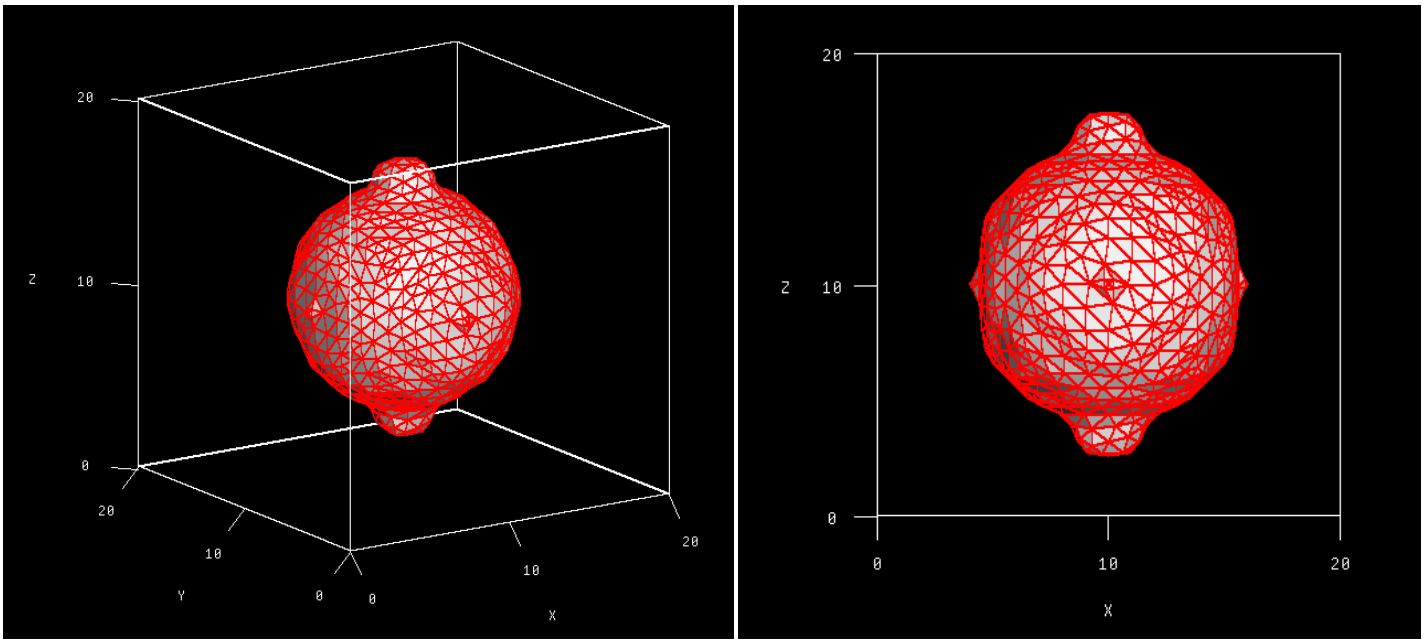

Figure 1.2 *Figure 1.1 the 3D image of t1f.vd and the y-view of it*

Q2: Does vmorph remove the protrusions? If not, can you think of a different set of parameters that would remove them using vmorph? What is the problem with removing the protrusions using morphological filtering?

Vmorph does not remove the protrusions of its bottom and top though it does remove the protrusions around the object. When I change “s=3,3,3” to “s=10,10,10” and then execute same commands. The protrusions of bottom and top are removed shown as the following images in figure 1.3. The problem of using morphological filtering is that it might generate some protrusions in other place of the object like the red-marked area of figure 1.4. There should be no protrusion in that area. But after applied the morphological filter, there are protrusions in that area.

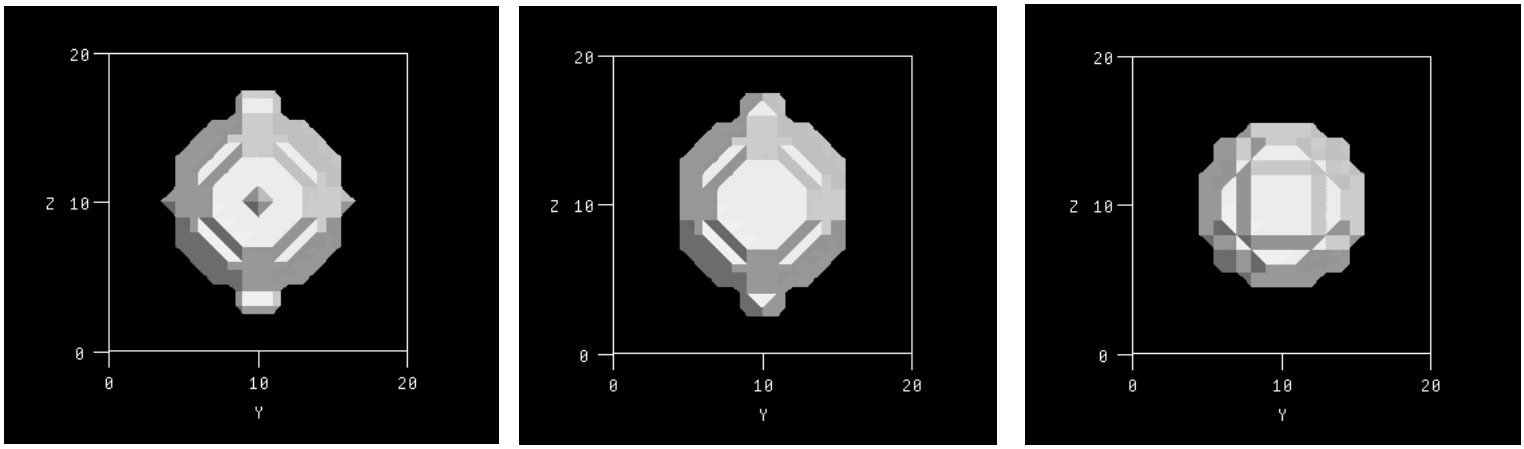

Figure 1.3 Left: the original x-view image; Middle: the x-view image (s=3,3,3); Right: the x-view image (s=10,10,10) .

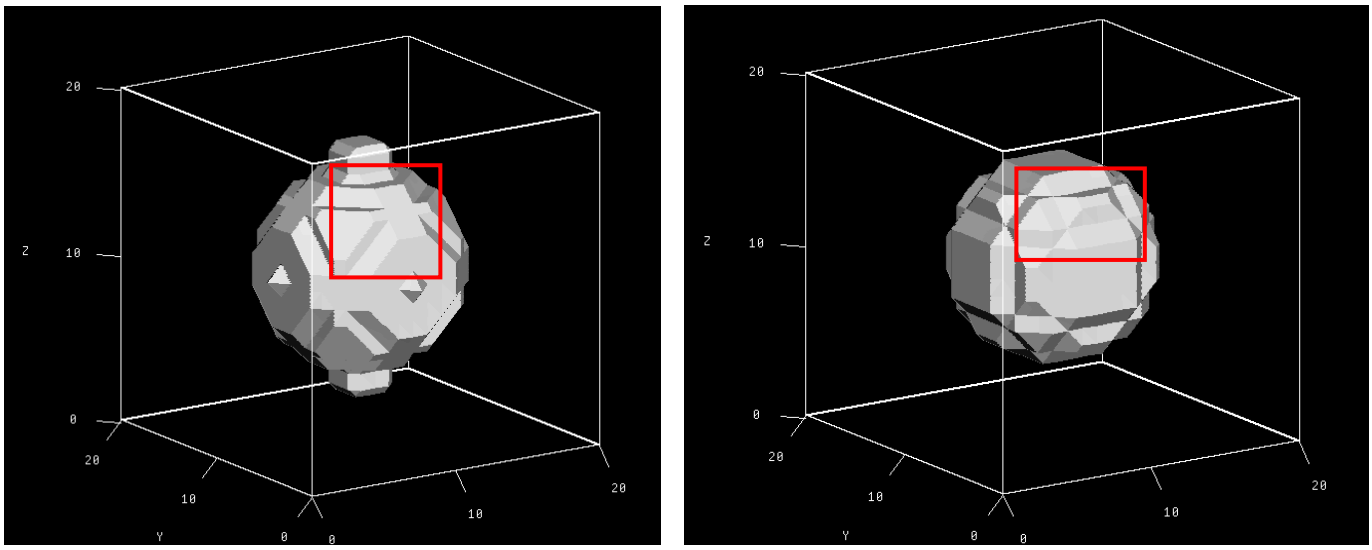

Figure 1.4 the comparison images of o-view

Section 3: Three-dimensional Edge Detector

In this section, we need to realize the edge detector of three dimension. First, we apply what we program to test image. Then, we add Gaussian noise to the image and process this image using some filters and detect its edge.

Firstly, we slice the three-dimensional image to twenty two-dimensional images as the left image below. And apply the edge detector program to this images and results shows below. I used the sobel-edge-detector to detect the edge. And I use the criteria that if one of the other 26 points of one cube whose central point is the one that we want to decide whether it is edge is background point and I set it as the edge. Hence, the result is good, it can detect the edge of this 3D object.

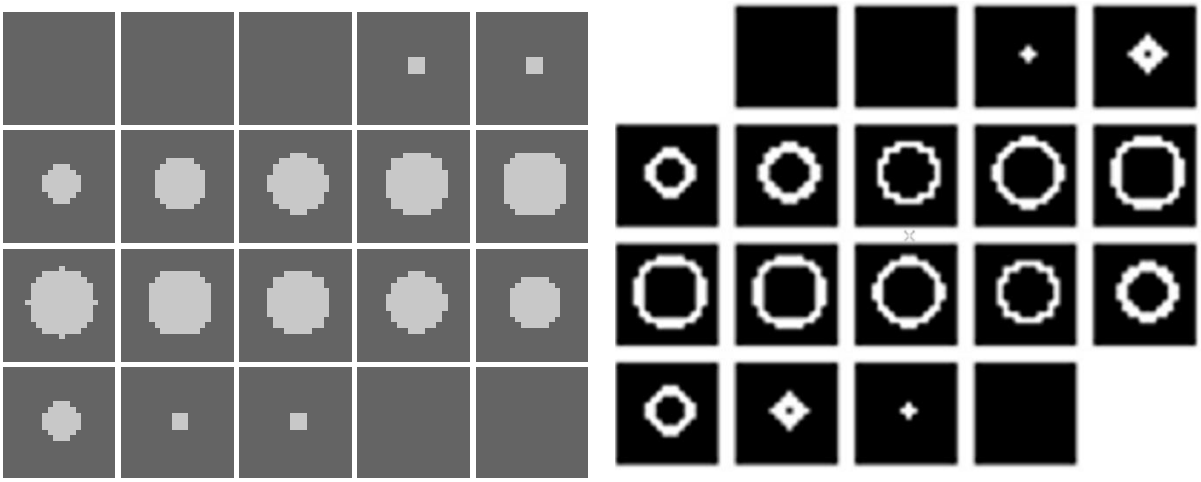

Figure 2.1 The original 2D images and the edge detector image

Secondly, we add the Gaussian noise to the original images. The images with Gaussian noise show below. The left one is an amplified image and the pixel values become inconsistent compared the one without noise. The right one is the images collection of 20 images of 3D objects.

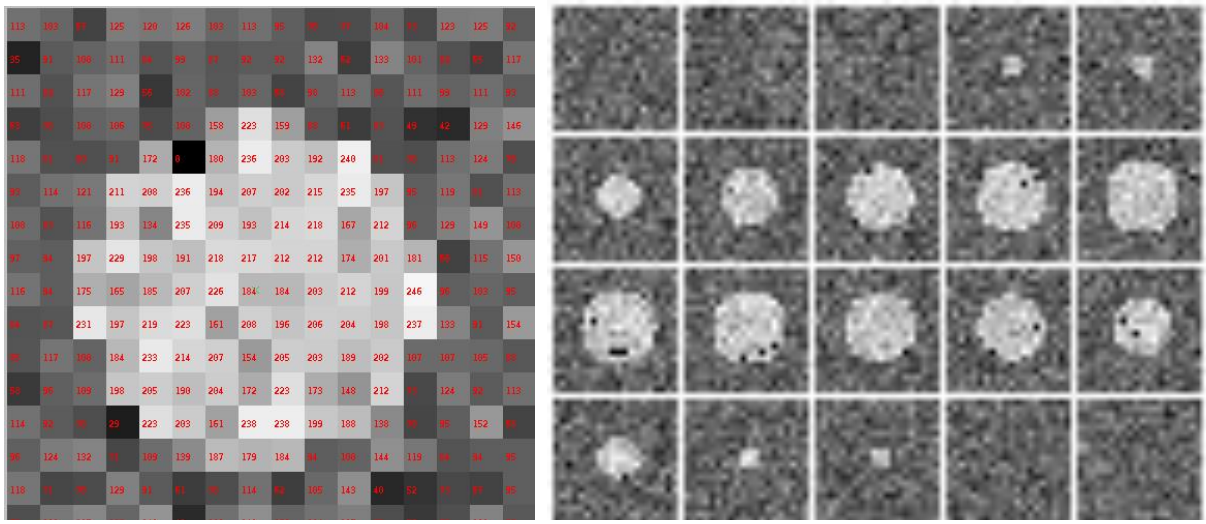

Figure 2.2 The image with Gaussian noise and the 20 images of them

Thirdly, I used one low frequency pass filters, median-filter, to process the images with noise. Note that we should filter before doing edge detector. Because the noise will have bad influence on the effect of edge detection. The left image of figure 2.3 shows the original image with noise after processed by mean-filter. We can see that the mean-filter could remove the noise from the middle image. The middle image is the one that I used my sobel-edge-detector to process it whose effect is not so good because of noise. The left one is processed by the sobel command in VM whose effect is better than mine.

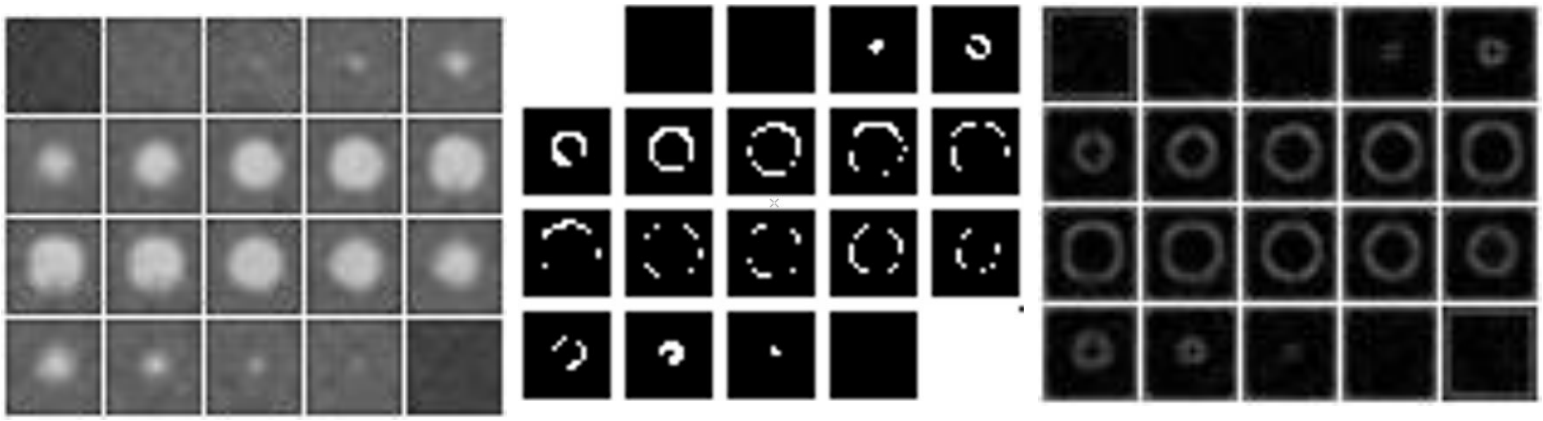

Figure 2.3 The comparison of edge detection without filter and with mean-filter

Codes are attached on code appendix.

Section 4: Three-dimensional Segmentation Evaluation

The dcompare.vx, dmregion.vx and dnodseg.vx show below figures. There are many vital parameters to measure the quality of my segmentation on campare.txt. Strict one is FPC and SIM also can measure the quality of segmentation. The FPC of my segmentation is 0.54482 and the SIM is 0.68140 and the maximum of these two parameters is 1, which means that my segmentation is not that good and From dcompare.vx, we can see that I segmented some part which should not be nodule part, which is the reason why my result is not so good.

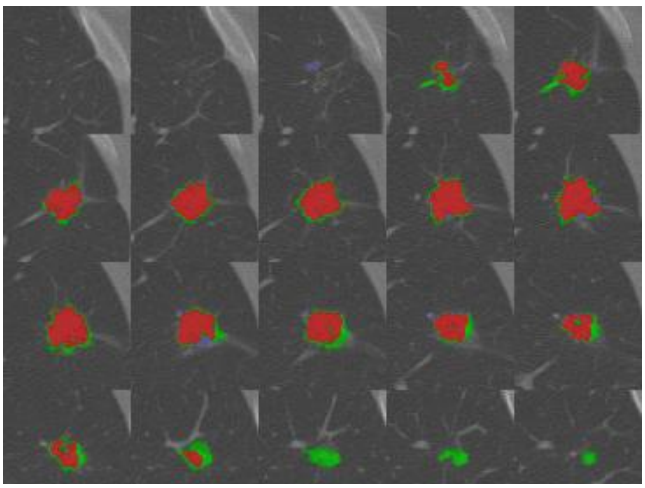

Figure 3.1 dcompare.vx

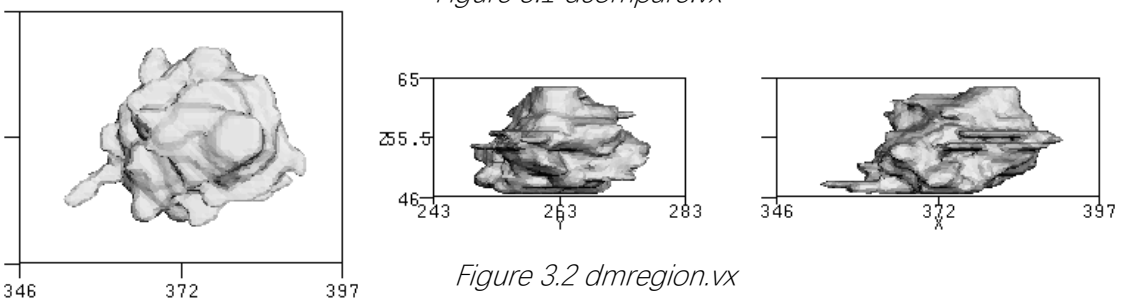

Figure 3.2 dmregion.vx

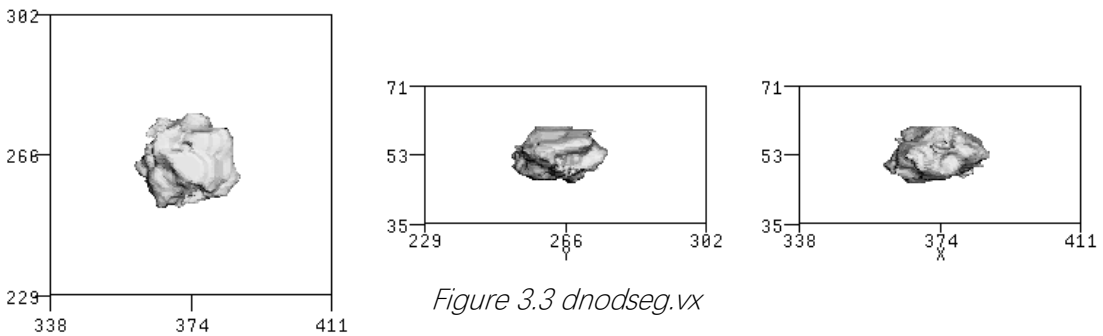

Figure 3.3 dnodseg.vx

Code Appendix:

v3dmedian.c

```
/*****************************************************************/
/* Compute function on a 3D image structure          */
/* Yanling Wu                                                    */
/* NetID: yw996                                                  */
/* Lab6                                                              */
/* Syntax:                                                     */
/*        v3df if=infile of=outfile [-v]                         */
/*****************************************************************/

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

VXparam_t par[] =            /* command line structure               */
{
{    "if=",    0,   " input file  v3dmean: compute local mean"},
{    "of=",    0,   " output file "},
{    "-v",     0,   " visible flag"},
{     0,       0,    0}
};
#define  IVAL   par[0].val
#define  OVAL   par[1].val
#define  VFLAG  par[2].val

//define the function to sort the tuple of numbers and find the median
int find_median(int nums[27]){
 //sort the tuples
 int i, j;
 for (i = 1; i < 27; i++ ) {
     int j = i;
     int target = nums[i];
     for (j = i; j>=0; j--) {
         if(j > 0 && nums[j-1] > target)
             nums[j] = nums[j-1];
         else{
             nums[j] = target;
             break;
         }
     }
 }
 return nums[13];
```

```
41. }
42.
43. int main(argc, argv)
44. int argc;
45. char *argv[];
46. {
47.  V3fstruct (im);
48.  V3fstruct (tm);
49.  int x,y,z;               /* index counters                */
50.  int xx,yy,zz;            /* window index counters         */
51.  int median, count;
52.  int nums[27];
53.     VXparse(&argc, &argv, par); /* parse the command line         */
54.
55.     V3fread( &im, IVAL);        /* read 3D image                  */
56.     if ( im.type != VX_PBYTE || im.chan != 1) { /* check  format  */
57.        fprintf (stderr, "image not byte type or single channel\n");
58.        exit (1);
59.     }
60.
61.     V3fembed(&tm, &im, 1,1,1,1,1,1); /* temp image copy with border */
62.     if(VFLAG){
63.        fprintf(stderr,"bbx is %f %f %f %f %f %f\n", im.bbx[0],
64.                  im.bbx[1],im.bbx[2],im.bbx[3],im.bbx[4],im.bbx[5]);
65.     }
66.
67.     for (z = im.zlo; z <= im.zhi; z++) {/* for all pixels */
68.       for (y = im.ylo; y <= im.yhi; y++) {
69.         for (x = im.xlo; x <= im.xhi; x++) {
70.           count = 0;
71.               for (zz = -1; zz <= 1; zz++) {/* compute the function */
72.                 for (yy = -1; yy <= 1; yy++) {
73.                   for (xx = -1; xx <= 1; xx++) {
74.                     nums[count] = tm.u[z + zz][y + yy][x + xx];
75.                     count++;
76.                   }
77.                 }
78.               }
79.               median = find_median(nums);
80.               im.u[z][y][x] = median;
81.         }
82.       }
83.    }
84.    V3fwrite (&im, OVAL);
```

```
85.    exit(0);
86. }
```

v3dedge_sobelc

```
1.  /*************************************************************/
2.  /* Example VisX4 program v3df                                */
3.  /*            Compute function on a 3D image structure       */
4.  /* Yanling Wu                                                    */
5.  /* NetID: yw996                                                  */
6.  /* Lab6                                                          */
7.  /* Syntax:                                                   */
8.  /*        v3df if=infile of=outfile [-v]                     */
9.  /*************************************************************/
10.
11. #include "VisXV4.h"          /* VisionX structure include file       */
12. #include "Vutil.h"           /* VisionX utility header files         */
13. #include <math.h>
14.
15. VXparam_t par[] =            /* command line structure               */
16. {
17. {    "if=",    0,   " input file  v3dmean: compute local mean"},
18. {    "of=",    0,   " output file "},
19. {    "-v",     0,   " visible flag"},
20. {    "th",     0,   "threshold"},
21. {     0,       0,    0}
22. };
23. #define  IVAL   par[0].val
24. #define  OVAL   par[1].val
25. #define  VFLAG  par[2].val
26. #define  TFLAG  par[3].val
27.
28. Int main(argc, argv)
29. int argc;
30. char *argv[];
31. {
32. V3fstruct (im);
33. V3fstruct (tm);
34. int        x,y,z;              /* index counters                */
35. int        xx,yy,zz;           /* window index counters         */
36. int        sum, grad;
37. float gradth;
38. int th;
```

```
int gy[3][3][3], gx[3][3][3], gz[3][3][3];
int hx[3]={1,2,1}, hy[3]={1,2,1},hz[3]={1,2,1};
int hpx[3]={1,0,-1},hpy[3]={1,0,-1},hpz[3]={1,0,-1};
int sumx, sumy, sumz;
    VXparse(&argc, &argv, par); /* parse the command line         */

    V3fread( &im, IVAL);        /* read 3D image                  */
    if ( im.type != VX_PBYTE || im.chan != 1) { /* check  format  */
       fprintf (stderr, "image not byte type or single channel\n");
       exit (1);
    }

    V3fembed(&tm, &im, 1,1,1,1,1,1); /* temp image copy with border */
    if(VFLAG){
       fprintf(stderr,"bbx is %f %f %f %f %f %f\n", im.bbx[0],
                 im.bbx[1],im.bbx[2],im.bbx[3],im.bbx[4],im.bbx[5]);
    }
    th = (TFLAG ? atoi(TFLAG) : 50);
 for(xx=0;xx<=2;xx++){ //build the kernel
     for(yy=0;yy<=2;yy++){
         for(zz=0;zz<=2;zz++){
             gx[xx][yy][zz]=hpx[xx]*hy[yy]*hz[zz];
             gy[xx][yy][zz]=hx[xx]*hpy[yy]*hz[zz];
             gz[xx][yy][zz]=hx[xx]*hy[yy]*hpz[zz];
             }
     }
 }

    for (z = im.zlo; z <= im.zhi; z++) {/* for all pixels */
      for (y = im.ylo; y <= im.yhi; y++) {
        for (x = im.xlo; x <= im.xhi; x++) {
          sumx = 0; sumy = 0; sumz = 0;
             for (zz = -1; zz <= 1; zz++) {/* compute the function */
               for (yy = -1; yy <= 1; yy++) {
                 for (xx = -1; xx <= 1; xx++) {
                   sumx += gx[zz+1][yy+1][zz+1]*tm.u[z-zz][y-yy][x-xx];
                   sumy += gy[zz+1][yy+1][zz+1]*tm.u[z-zz][y-yy][x-xx];
                   sumz += gz[zz+1][yy+1][zz+1]*tm.u[z-zz][y-yy][x-xx];

                 }
               }
             }
           sumx /= 27;
           sumy /= 27;
```

```
83.            sumz /= 27;
84.            gradth = sqrt(sumx*sumx+ sumy*sumy+sumz*sumz);
85.            if(gradth > th)
86.                im.u[z][y][x]=255;
87.           else
88.                im.u[z][y][x] = 0; //threshold at 50
89.         }
90.       }
91.    }
92.    V3fwrite (&im, OVAL);
93.    exit(0);
94. }
```