Lab6

THREE DIMENSIONAL IMAGE PROCESSING

Blanca Bai

Cornell University Electrical and Computer Engineering

Table of Contents

2 Three dimensional filter

2.1 Generate a 3D byte test image

```
t1make
```

2.2 View the polygon file t1.vd from vview

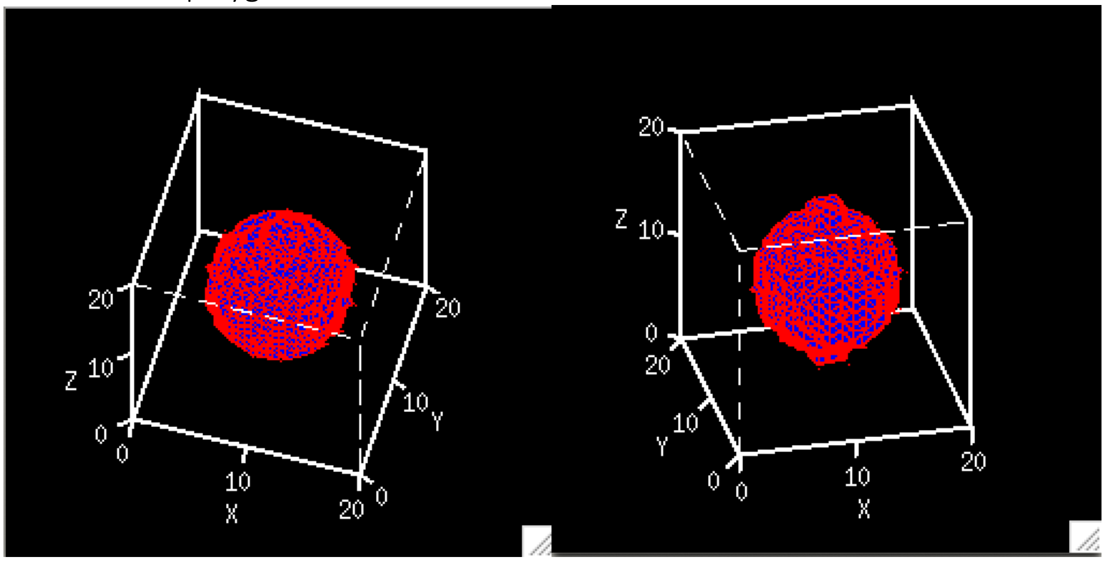


Figure 1,2: polygon files ti.vd

We can rotate the 3D image to view from different perspectives using the shortcut “J” and “L” in my keyboard. (yes, my keyboard is j and l)

De-select showpoints and showboundaries options, then under the render options you can select lightmodel to shade the polygon.

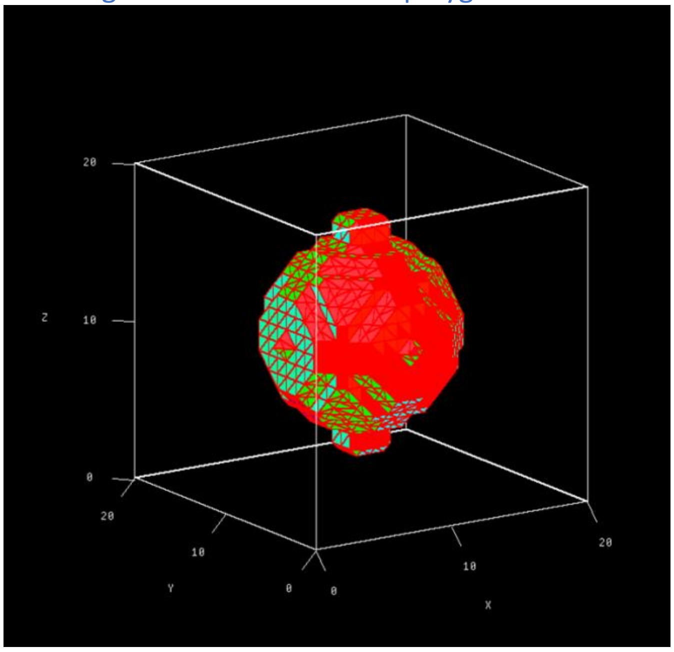


Figure 3: polygon files ti.vd under lightmodel

The appearance of the polygon we generated is coarse. Now we smooth the polygons.

```
v3pfilt - a t1.vd - o t1f.vd v3d t1f.vd cf = v3dcf &
```

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

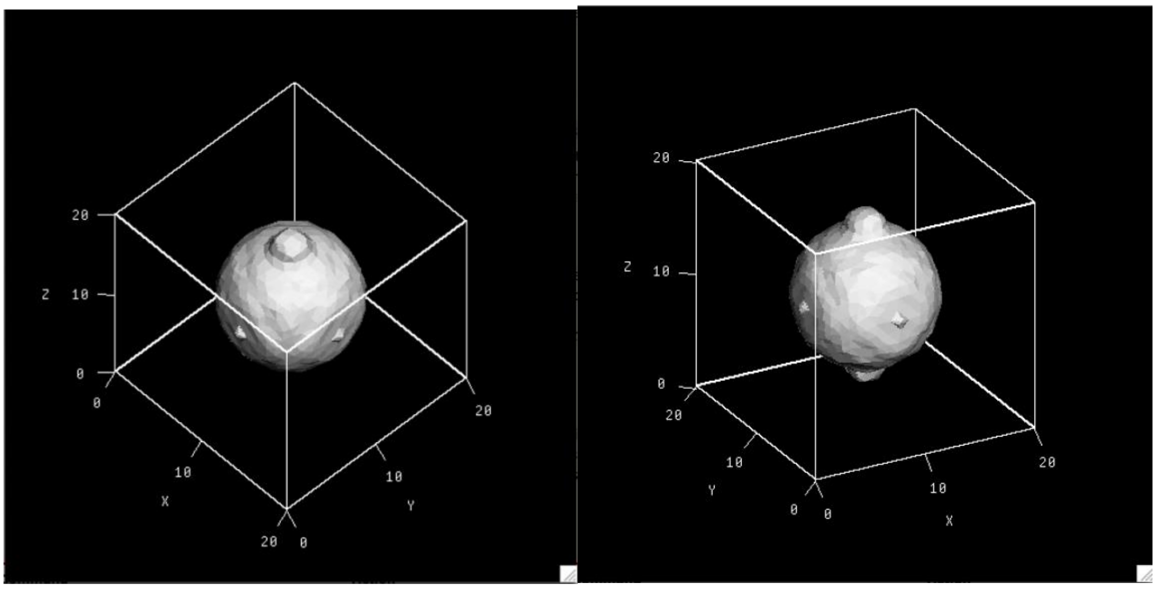


Figure 4,5: smoothed polygon files ti.vd

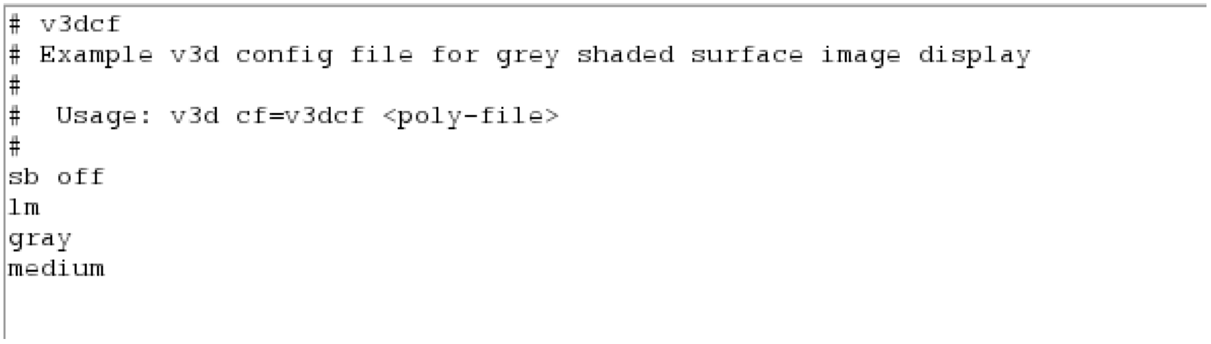
tif.vd is smoother than ti.vd, it remove some sharp edges, because V3pfilt smooths a polygonal surface in a 3D polygon file. The algorithm modifies the position of each vertex as the weighted sum of neighboring vertices and itself.

Usually I perform this action on the polygons, but not always, because some information of sharp edge will be removed, if I need those information, I won't perform smooth on the polygons result.

2.4 v3dcf config file

you create a file called .v3drc then v3d will read it automatically when it starts as if had specified it with a cf= parameter.

```
cp v3dcf.v3drc
```

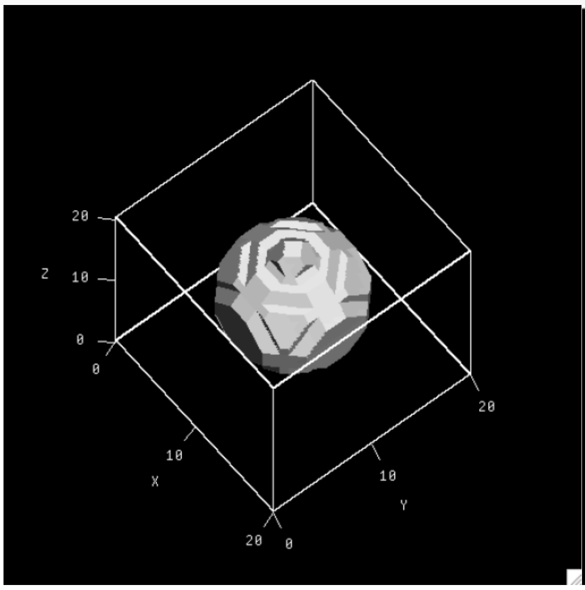


```
# v3dcf
# Example v3d config file for grey shaded surface image display
#
#  Usage: v3d cf=v3dcf <poly-file>
#
sb off
lm
gray
medium
```

Figure 6: v3dcf config file content

2.5 experiment with simple three-dimensional morphological filtering.

```
vmorph t1.vx - ed t = s s = 3, 3, 3 | vdim - c of = n3.vx v3pol - t
if = n3.vx of = n3.vd
```



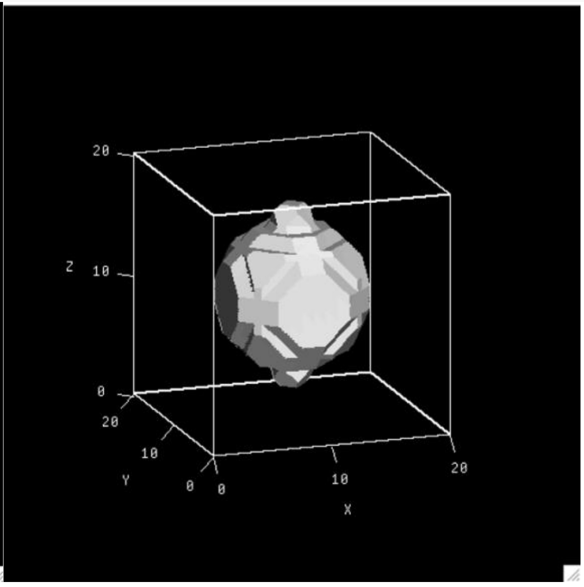


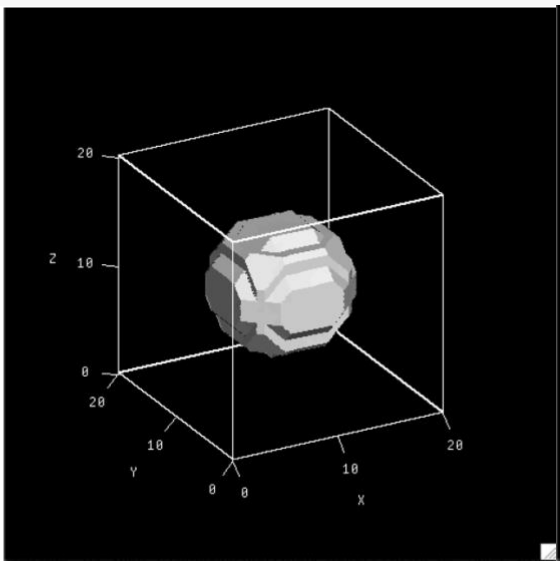
Figure7,8: input n3.vd

2.5 Does vmorph remove the protrusions? If not, can you think of a different set of parameters that would remove them using vmorph?

the vmorph remove four protrusions in xoz, yoz plane. But doesn't remove the protrusions on the top and button.

I can change the dimension of kernel, and also change the type of kernel from eclipse to rectangle.

```
vmorph t1.vx - ed t = c s = 5, 5 | vdim - c of = nc55.vx v3pol - t
if = nc55.vx of = nc55.vd
```



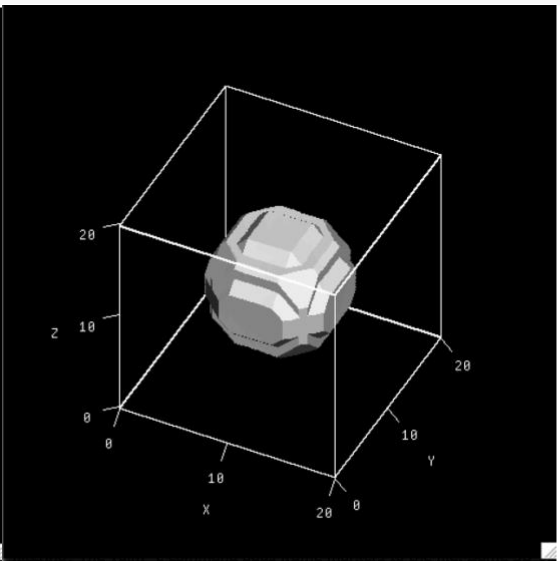


Figure 9, 10: protrusions removed output image the kernel type to be generated, where 's' corresponds to a spherical/ellipsoidal kernel and 'c' to a cubic/rectangular kernel (default is spherical/ellipsoidal).

2.5 What is the problem with removing the protrusions using morphological filtering?
the kernel of morphological filter are automatically generated, only include spheres, ellipsoids, rectangular parallel pipeds, as well as 2D ellipses and rectangles.

The vdim -c command can adds frame markers to the file.

3 Three Dimension Edge Detector

3.1 how the 3D image is stored in a 3D array and that byte images are the supported input image format.
The 3D image are treated as a single image, when reading this 3D image, there is no need to loop for multiple times to read one image.
3D image is stored in a 3 dimension array, assuming it is im.u[x][y][z], index x, y , z is the value of a point mapping to x axis, y axis, and z axis.

3.2 generate a test image to test a three-dimensional edge detector.
take image t1.vx and add some noise,

```
vpix t1.vx lo = 100 hi = 200 th = 1 of = t1g.vx
vpix t1g.vx gn = 25 of = t1n.vx
```

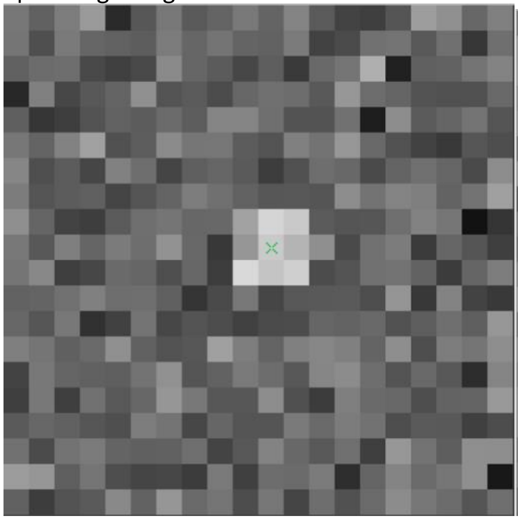
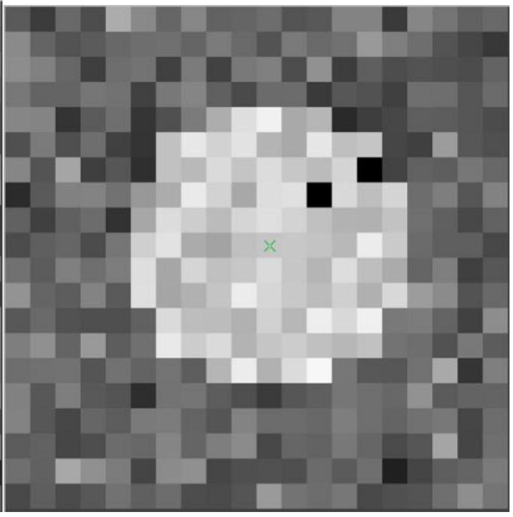



Figure 11, 12: noised t1n.vx

3.3 Write an edge detection program on 3D images.
In t1g.vx: the pixel value of object is 200, the background is 100.

In t1n.vx, the image contains so many noise, so we need to use a low-pass filter to remove the noise in this image, and then use our edge detection algorithm to detect the edge.

3.4 Test your program with the test image t1g.vx and t1n.vx

Source Code:

```

int main(argc, argv) int argc;
char * argv[]; {
    V3fstruct(im);
    V3fstruct(tm);
    int x, y, z; /* index counters                 */
    int xx, yy, zz; /* window index counters          */
    int x_grad, y_grad, z_grad; /* X, Y, Z gradiant value */
    int grad_m;
    int threshold;
    VXparse( & argc, & argv, par); /* parse the command line        */
    printf("sadffsadsaf");
    threshold = (THRESH ? atoi(THRESH) : 10);
    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.b
bx[3], im.bbx[4], im.bbx[5]);
    }
    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++) {
                x_grad = 0;
                y_grad = 0;
                z_grad = 0;
                grad_m = 0;
                for (yy = y - 1; yy <= y + 1; yy++) {
                    for (zz = z - 1; zz <= z + 1; zz++) {
                        x_grad += tm.u[zz][yy][x + 1] - tm.u[zz][yy][x - 1];
                    }
                }
                for (xx = x - 1; xx <= x + 1; xx++) {
                    for (zz = z - 1; zz <= z + 1; zz++) {
                        y_grad += tm.u[zz][y + 1][xx] - tm.u[zz][y - 1][xx];
                    }
                }
                for (yy = y - 1; yy <= y + 1; yy++) {
                    for (xx = x - 1; xx <= x + 1; xx++) {
                        z_grad += tm.u[z + 1][yy][xx] - tm.u[z - 1][yy][xx];
                    }
                }
                grad_m = cbrt(x_grad * x_grad + y_grad * y_grad + z_grad * z_grad);
                if (grad_m > threshold) {
                    im.u[z][y][x] = 255;
                } else {
                    im.u[z][y][x] = 128;
                }
            }
        }
    }
    for (z = im.zlo; z <= im.zhi; z++) { /* for all pixels */
        for (y = im.ylo; y <= im.yhi; y++) {
            im.u[z][y][im.xlo] = 128;
            im.u[z][y][im.xhi] = 128;
        }
    }
    for (z = im.zlo; z <= im.zhi; z++) { /* for all pixels */
```

```
        for (x = im.xlo; x <= im.xhi; x++) {
            im.u[z][im.ylo][x] = 128;
            im.u[z][im.yhi][x] = 128;
        }
    }
    for (x = im.xlo; x <= im.xhi; x++) { /* for all pixels */
        for (y = im.ylo; y <= im.yhi; y++) {
            im.u[im.zlo][y][x] = 128;
            im.u[im.zhi][y][x] = 128;
        }
    }
    V3fwrite( & im, OVAL);
    exit(0);
```

```
v3dedge t1g.vx of = v3dedget1g.vx
vtile v3dedget1g.vx n = 5, 4 of = v3dedget1gt.vs
```

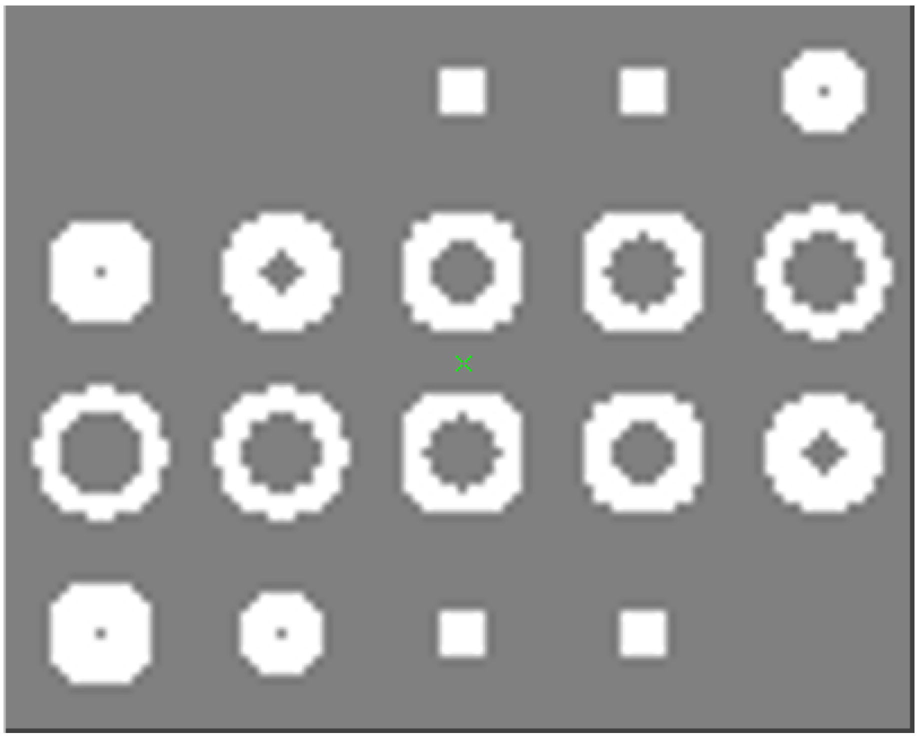


Figure 13: edge detection result of tig.vx

So we use mean filter to remove the noise,

```
vtile t1n.vx n = 5, 4 of = t1nt.vs
```

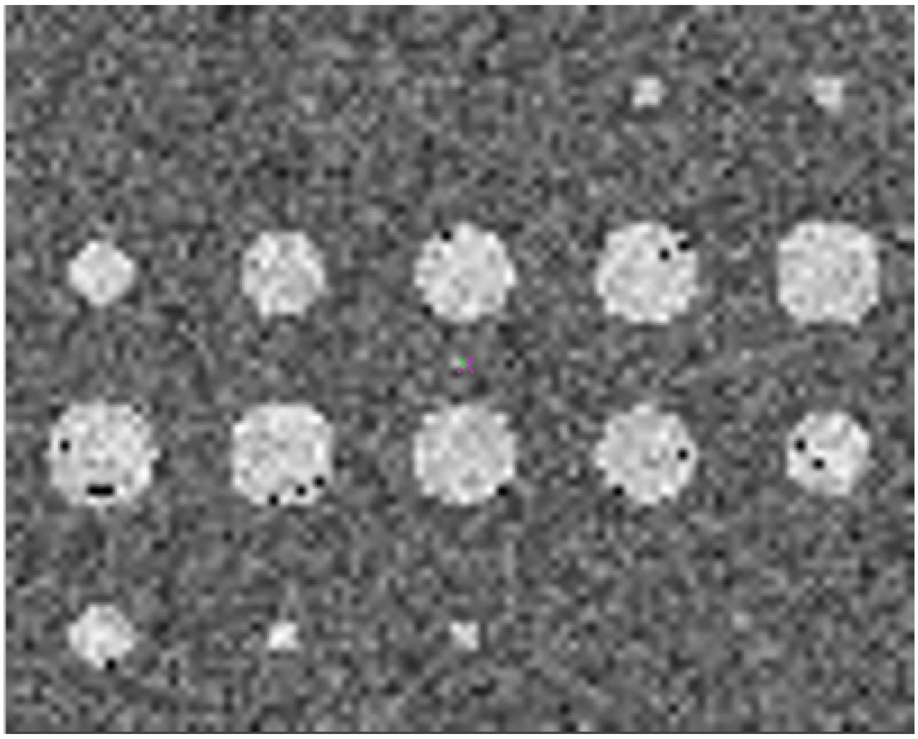


Figure 14: noisy t1nt.vs

```
vcc v3dmean.c - o v3dmean
v3dmean t1n.vx of = v3dmeant1n.vx
vtile v3dmeant1n.vx n = 5, 4 of = v3dmeant1nt.vs
```

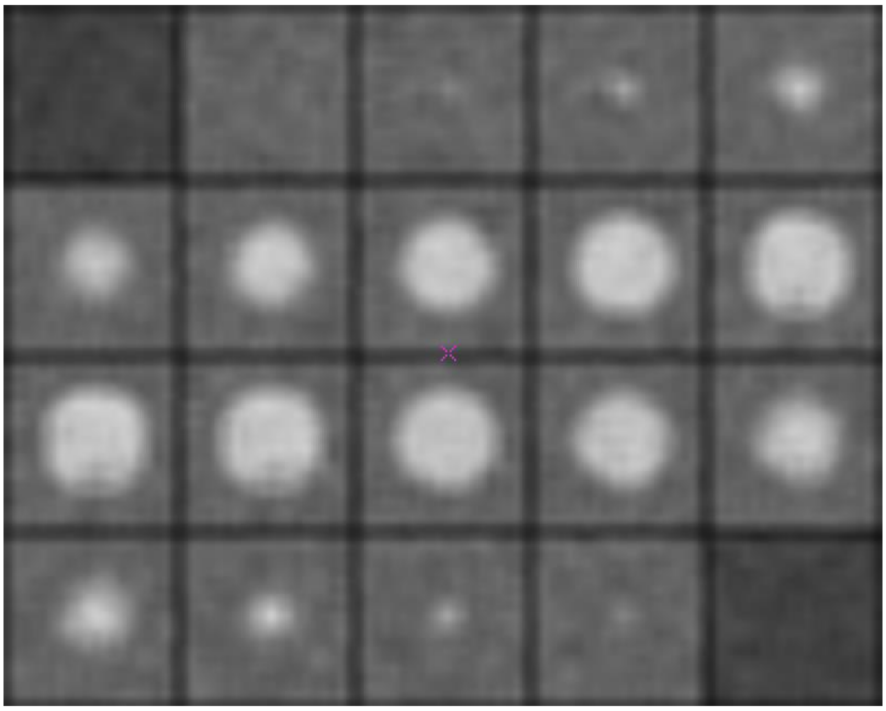


Figure 15: mean filter denoised t1nt.vx

Then I experiment my 3dedge algorithm on this denoised image, to get a better result, I hard code the threthold as 55. If the threthold is lower, such as 53, there will be a lot noise, if the threthold is higher, such as 56, more edge will lose.

```
threshold = 55;
```

```
v3dedge v3dmeant1n.vx of = v3dedgemeant1n.vx
vtile v3dedgemeant1n.vx n = 5, 4 of = 111111. vx
```

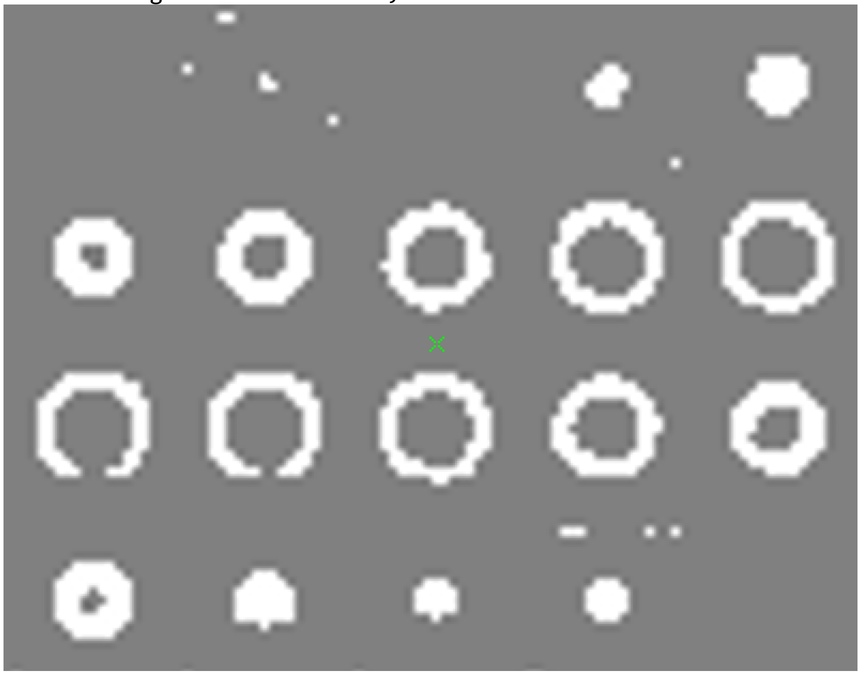


Figure 16: edge detection result after mean filter denoise image

3.5 Consider the best way to process the output of your edge detector and display the 3D images.

```
vtile < 3 D - image > -ib n = 5, 4 - xb of =
```

4. Three-dimensional Segmentation Evaluation

4.1 display the image ctimage.vs with vdview

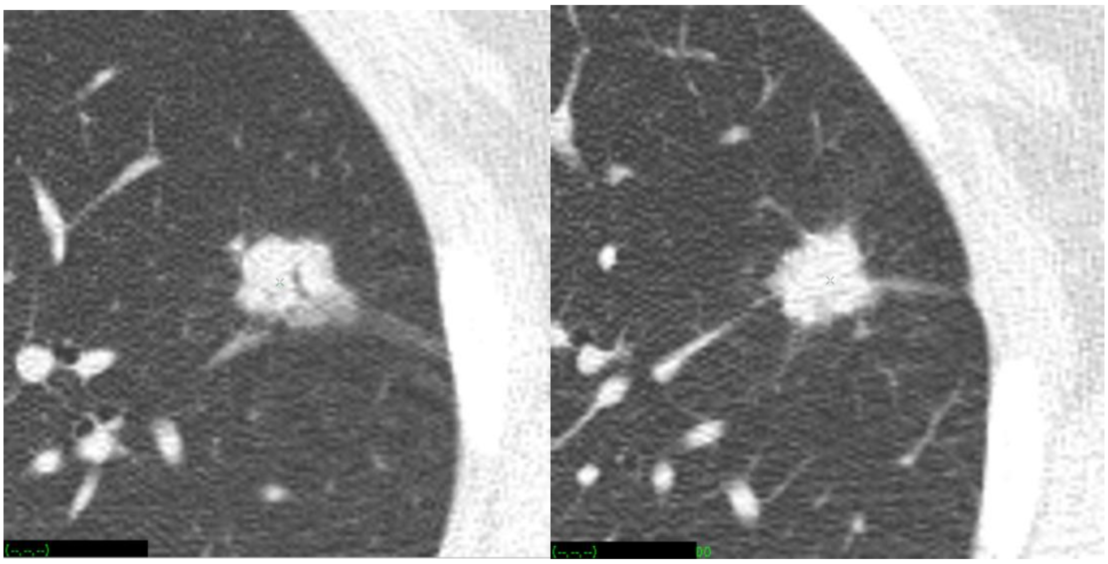


Figure 17, 18: input ctimage.vs

4.2 mark the boundary of this nodule in all image slices

Done

4.3 run the script analseg

Run the script analseg and review the contents of this script to gain an understanding of what it does.

Noseg.vx

```
vtile nodseg.vx n = 6, 6 of = nodsegtt.vs
```

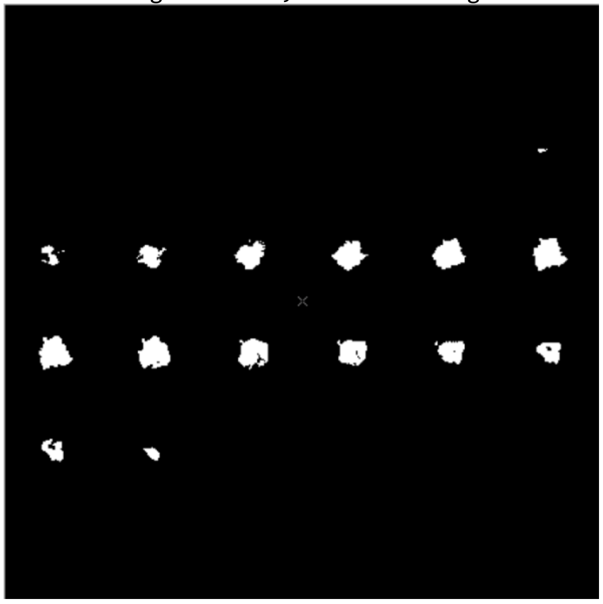


Figure 19: labeled boundary noseg.vx

4.4 Review the images: compare.vx, dcompare.vx, nodseg.vd and mregion.vd

```
vtile compare.vx n = 6, 6 of = comparet.vs
```

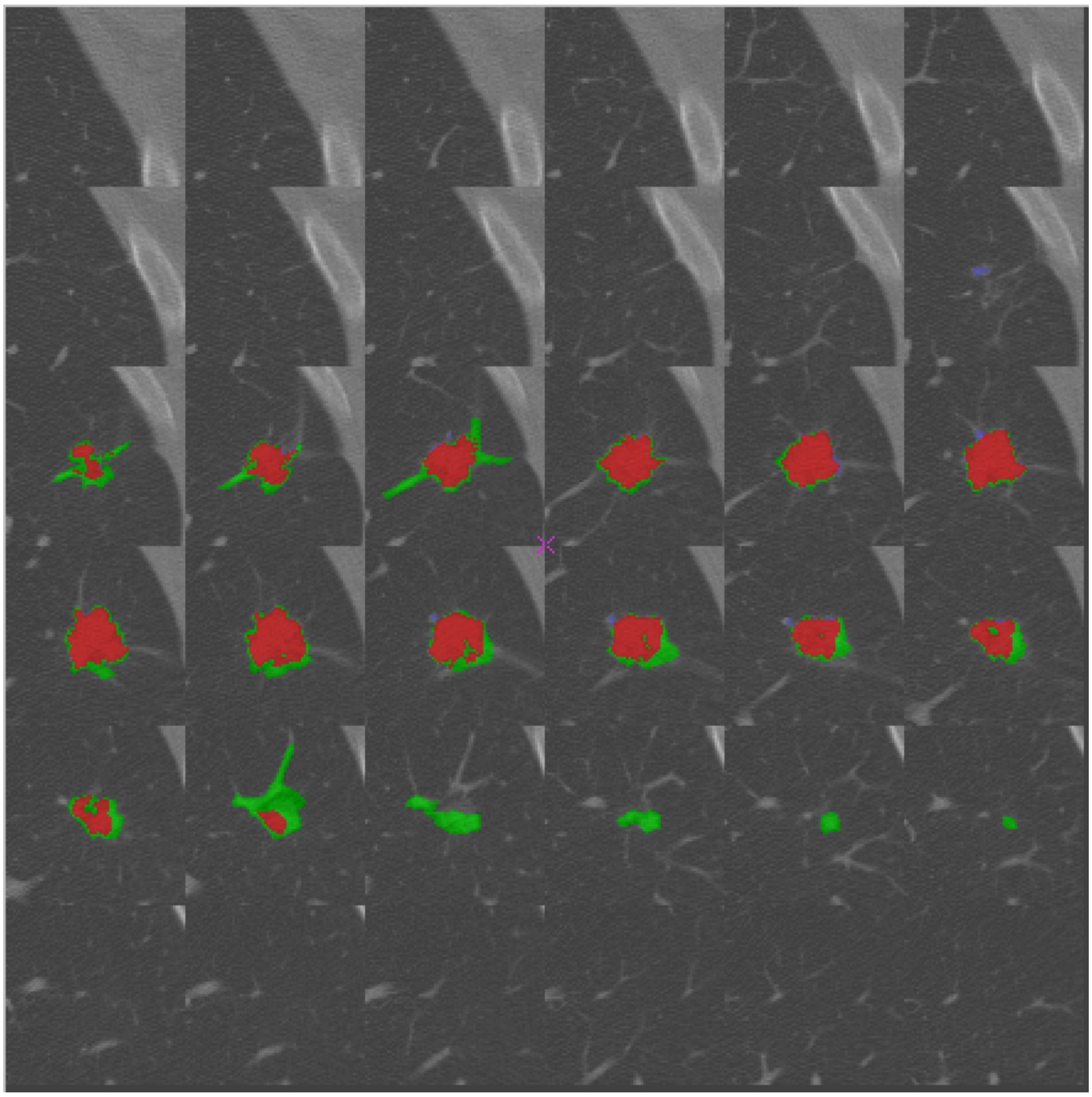
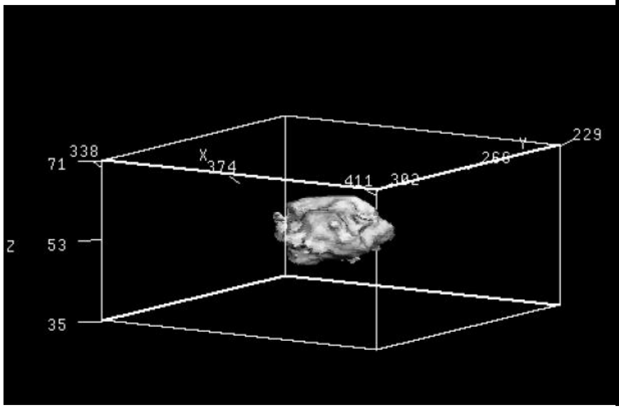


Figure 20: comparison results of labeled boundary image comparet.vs

nodseg.vd



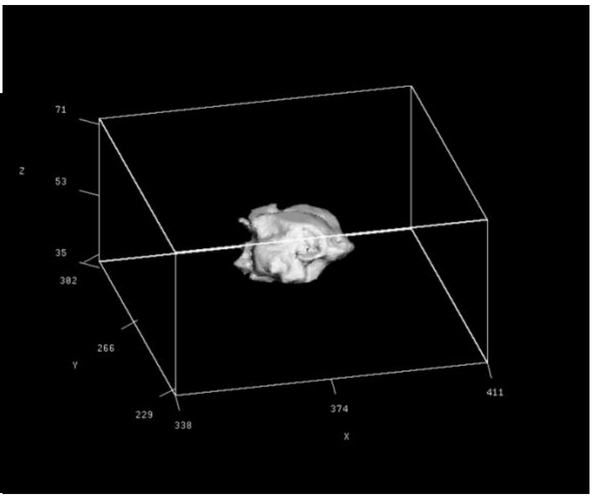


Figure 21, 22 : nodseg.vd

Mregion.vd

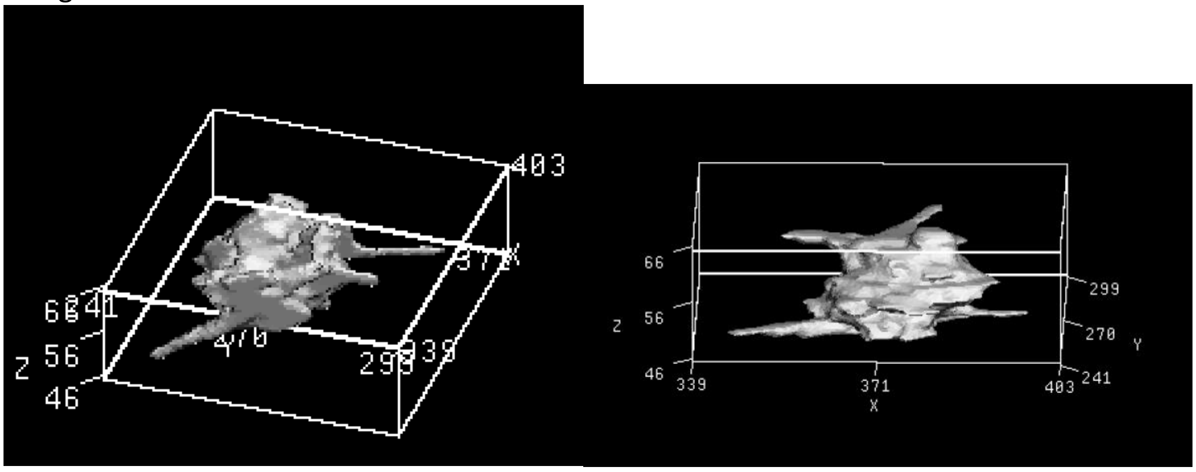


Figure 23, 24 : mregion.vd

4.5 discusses the quality of my segmentation

View dcompare.vx, dmregion.vx, dnodseg.vx

Decompare.vx

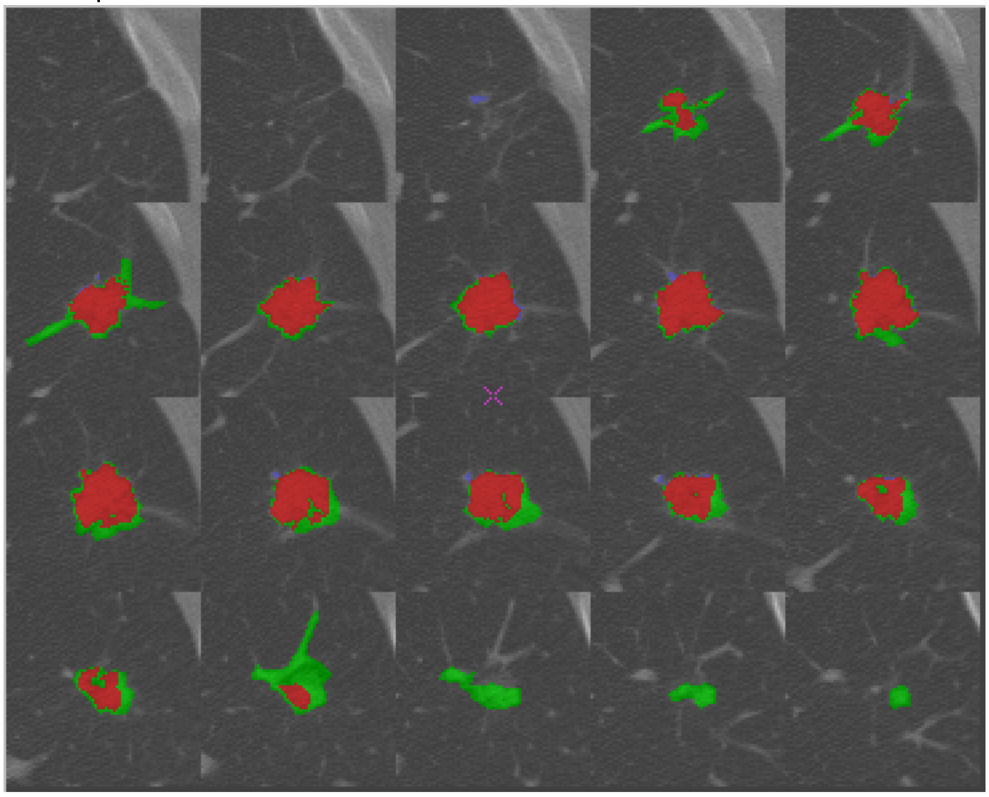


Figure 25 : decompare.vx

dmregion.vx

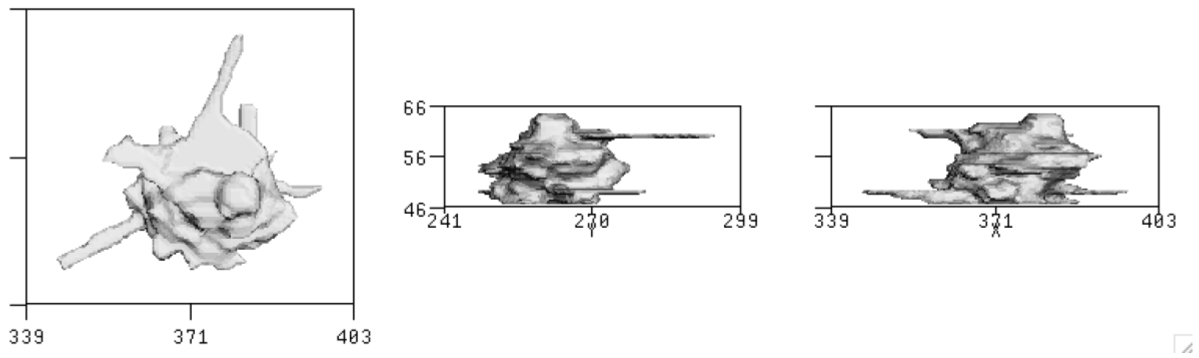


Figure 26 : dmregion.vx

dnodseg.vx

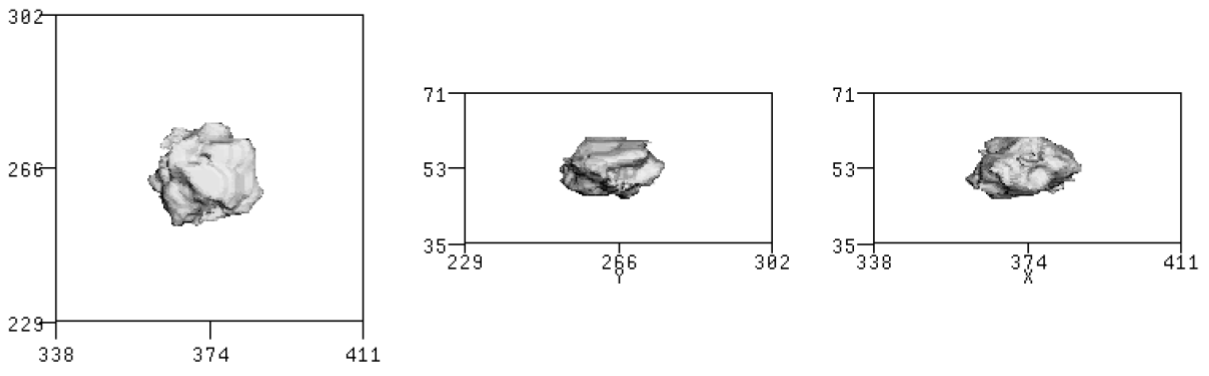
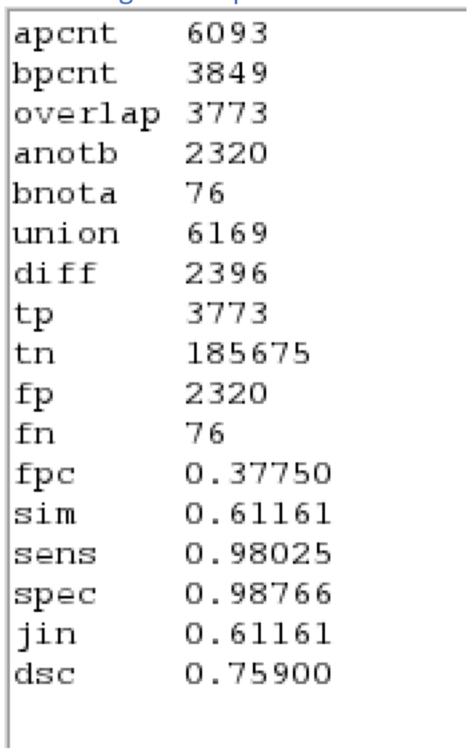


Figure 27 : dnodseg.vx

According to compare.txt:



```
apcnt    6093
bpcnt    3849
overlap  3773
anotb    2320
bnota    76
union    6169
diff     2396
tp       3773
tn       185675
fp       2320
fn       76
fpc      0.37750
sim      0.61161
sens     0.98025
spec     0.98766
jin      0.61161
dsc      0.75900
```

Figure 28 : compare.txt

The number of pixels in ground truth is 6093, and the number of pixels in my segmentation is 3849, so my boundary is less than actual aoundary. Some more areas should be include into my segmentation. However bnota valua is 76, it is low, which means the accuracy in my segmentation is good, almost all pixels in my segmentation belongs to the ground truth. But my anotb value is 2320, it is high, which mean I haven't all ground truth.