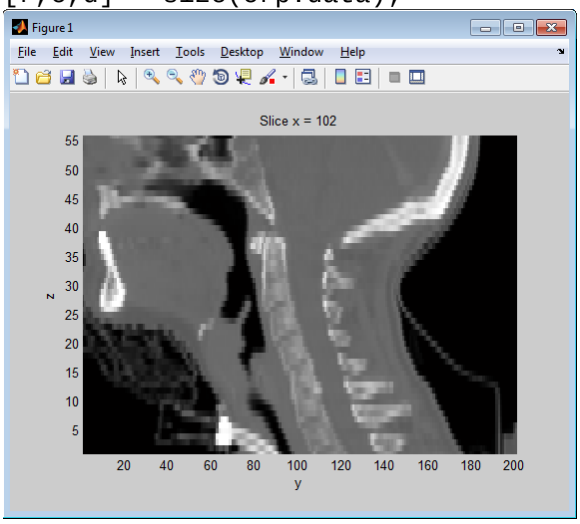
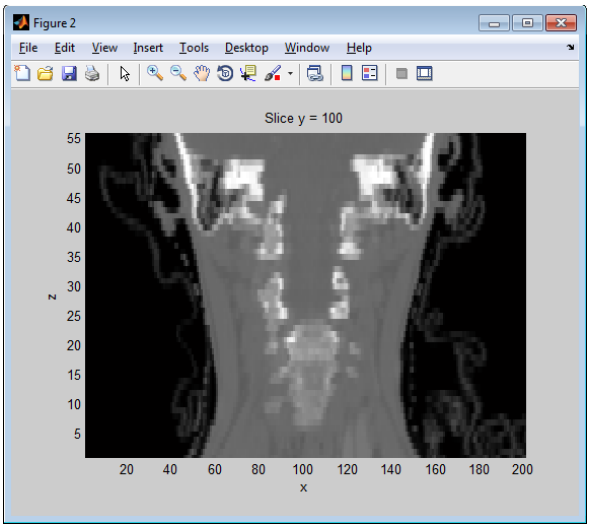
EECE 8395: Medical Image Segmentation
Vanderbilt University
Prof. Jack H. Noble

4D path search

In this demo, we will see how to implement a 4D path to do 3D tubular structure segmentation. First we load and crop our image.

```
function test_4DGraphSearch2
global r c d img rset searchbox Done Pointer mndef vardef;

im = ReadNrrd('0522c0001\img.nrrd');

crp = im;
crp.data = im.data(160:360,140:340,30:85);
img = crp.data;
crp.dim = size(crp.data);
crp.data = crp.data/10+100;
figure(1); close(1); figure(1); colormap(gray(256));
DisplayVolume(crp,1,102);
figure(2); close(2); figure(2); colormap(gray(256));
DisplayVolume(crp,2,100);
[r,c,d] = size(crp.data);
```





Next we choose what radii we want to represent in the graph:

```
rset = [4.75 5.5 6.6 8.14 10.12];
```

This gives us a set of 5 radii that are finely spaced for low radii then become coarsely spaced for larger radii. Next we define our seed and endnodes.

```
seedxyz = [105,120,10];
seednode = (seedxyz(3)-1 + ((seedxyz(1)-1)*c + (seedxyz(2)-1))*d
)*length(rset) + 1;
endxyz = [105,105,37];
for i=1:length(rset)
    endnode(i) = (endxyz(3)-1 + ((endxyz(1)-1)*c + (endxyz(2)-
1))*d)*length(rset) + i;
end
```

Note we have a different endnode for every possible radii. We could do the same for the seednodes but for simplicity here we have one seed with radius = 4.62 mm. We also define a search box to limit the search range to speed up the search a bit.

```
searchbox = [80,120, 80,140, 10,37];
```

Finally we initialize our global ‘Done’ and ‘Pointer’ variable that work with the helper functions `@NodeMark,@IsNodeMarked,@SetPointer,@GetPointer` the same way as our LiveWire implementation.

```
Done = zeros(r*c*d*length(rset),1);
Pointer = zeros(r*c*d*length(rset),1);
```

We need to define NodeNeibs using our neighborhood structure and cost function. Here I used a 9-connected coordinate neighborhood where each voxel at z=k has 9 coordinate neighbors with z=k+1, so that the path always traverses up in the z direction. For every coordinate neighbor there are also multiple nodes corresponding to different radii. To constrain the radius to change smoothly from node to node, we constrain radii neighbors to only include neighboring radii indices. So a node with radius 5.5 has connections to nodes with radius [4.75, 5.5, 6.6]. And a

node with radius 10.12 has connections to nodes with radius [8.14 10.12]. For the cost function, first we try a modification of the one Li and Yezzi proposed:

```
…
w=1;
lambda1=1;
lambda2=1;
...
Cost = w + (lambda1*(mni - mndef)*(mni - mndef) + lambda2*(vari -
vardef)*(vari - vardef))/(radius*radius);
```

where

```
[mndef,vardef] = SphereStat(seedxyz(1),seedxyz(2),seedxyz(3),1);
```

We then can perform the graph search:

```
[nodepath,cost] =
GraphSearch(@NodeNeibs,@NodeMark,@IsNodeMarked,@SetPointer,@GetPointer,seedno
de,endnode);
```

Once it executes, we need to extract the resulting path:

```
pnts = floor((nodepath'-1)/(c*d*length(rset)))+1;
pnts(:,2) = floor((nodepath'-1-(pnts-
1)*d*c*length(rset))/(d*length(rset)))+1;
pnts(:,3) = floor((nodepath'-1-d*length(rset)*(pnts(:,2)-1+c*(pnts(:,1)-
1)))/length(rset))+1;
pnts(:,4) = mod(nodepath'-1,length(rset))+1;
```

Then we can build a mask that corresponds to the resulting segmentation.

```
msk = zeros(size(crp.data));
for i=1:length(pnts)
      for xx=max(1,floor(pnts(i,1)-
      rset(pnts(i,4)))):min(r,ceil(pnts(i,1)+rset(pnts(i,4))))
      for yy=max(1,floor(pnts(i,2)-
      rset(pnts(i,4)))):min(r,ceil(pnts(i,2)+rset(pnts(i,4))))
      for zz=max(1,floor(pnts(i,3)-
      rset(pnts(i,4)))):min(r,ceil(pnts(i,3)+rset(pnts(i,4))))
            if (xx-pnts(i,1))*(xx-pnts(i,1))+(yy-pnts(i,2))*(yy-
            pnts(i,2))+(zz-pnts(i,3))*(zz-pnts(i,3)) <
            rset(pnts(i,4))*rset(pnts(i,4))
                  msk(xx,yy,zz) = 1;
            end
      end
      end
      end
end
```

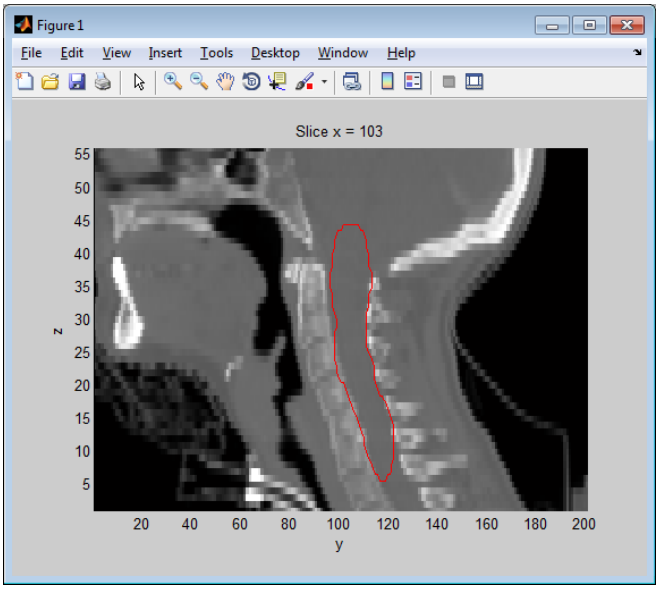
And we can display it on our image

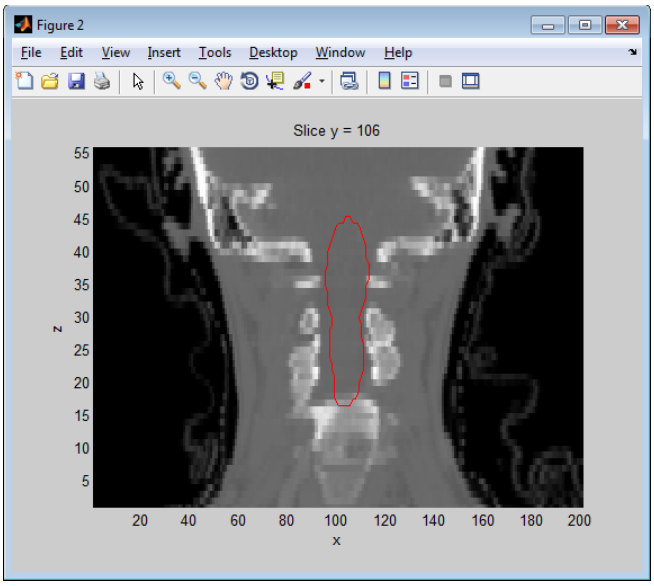
```
h = figure(1); hold on;
inp = guidata(h);
inp.cntrs = zeros(2,1000,r);
for i=1:r
```

```
    if sum(sum(squeeze(msk(i,:,:))))>0
        cntr = contour(squeeze(msk(i,:,:))',[0.5,0,5]);
        inp.cntrs(:,1:length(cntr),i) = cntr;
    end
end
guidata(h,inp);


h = figure(2); hold on;
inp = guidata(h);
inp.cntrs = zeros(2,1000,r);
for i=1:c
    if sum(sum(squeeze(msk(:,i,:))))>0
        cntr = contour(squeeze(msk(:,i,:))',[0.5,0.5]);
        inp.cntrs(:,1:length(cntr),i) = cntr;
    end
end
guidata(h,inp);
```

where we need to modify our display volume function slightly to account for when we have multiple contours on a single slice. Our original implementation would not permit this

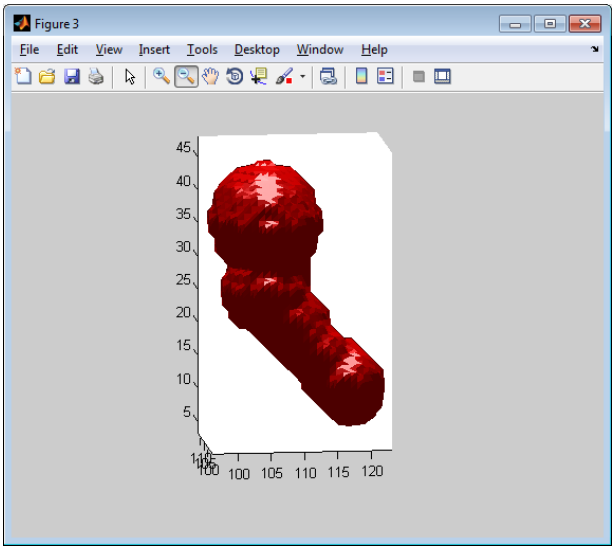
```
function DisplayVolume(img,direction,slc)
…
if isfield(inp,'cntrs')
    hold on
    j=1;
    c = inp.cntrs(:,:,inp.slc);
    while j<length(c) && c(2,j)>0
        len = c(2,j);
        plot(c(1,j+1:j+len),c(2,j+1:j+len),'r');
        j = j+len+1;
    end
end
…
```





We can also isosurface the result to produce a mesh:

```
M = isosurface(msk,0.5);
figure(3); clf;
DisplayMesh(M)
```



The result is kind of reasonable but not great. Why? The cost function doesn't really make a lot of sense. We don't need the mean intensity of the sphere to match the first one, and we also don't need the variance to match the first one. Let's try a cost function designed to find a tube with the large radius that contains low intensity variance:

```
w=1;
lambda1=1;
lambda2=1e-4;
```

```
Cost = w + lambda1/rset(kk) + lambda2*(vari);
…
```

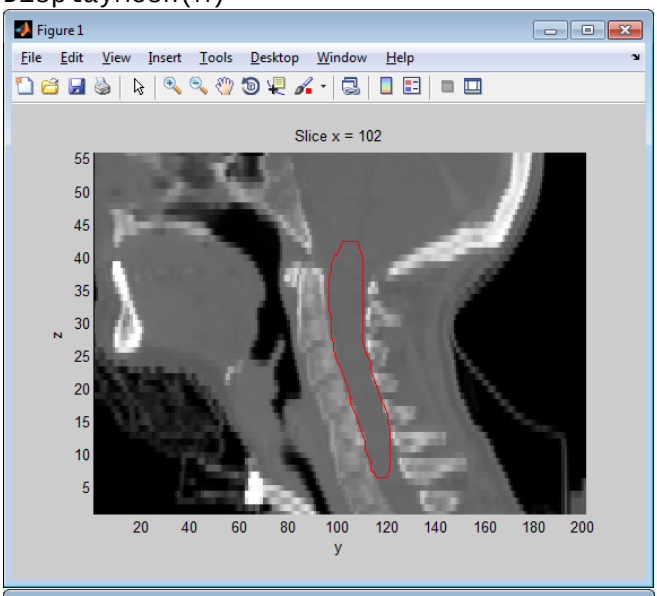
Then redo the graph search using the new cost function defined in NodeNeibs2:

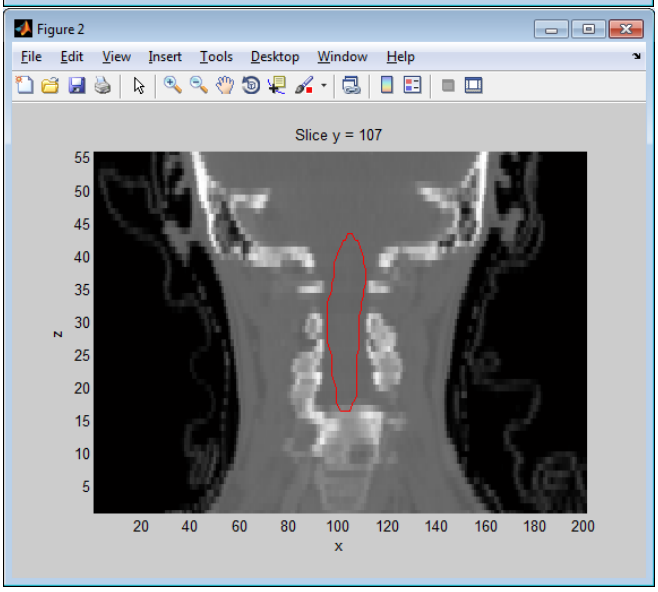
```
Done = zeros(r*c*d*length(rset),1);
Pointer = zeros(r*c*d*length(rset),1);
[nodepath,cost] =
GraphSearch(@NodeNeibs2,@NodeMark,@IsNodeMarked,@SetPointer,@GetPointer,seednode,endnode);
pnts = floor((nodepath'-1)/(c*d*length(rset)))+1;
pnts(:,2) = floor((nodepath'-1-(pnts-
1)*d*c*length(rset))/(d*length(rset)))+1;
pnts(:,3) = floor((nodepath'-1-d*length(rset)*(pnts(:,2)-1+c*(pnts(:,1)-
1)))/length(rset))+1;
pnts(:,4) = mod(nodepath'-1,length(rset))+1;

msk = zeros(size(crp.data));
for i=1:length(pnts)
      for xx=max(1,floor(pnts(i,1)-
      rset(pnts(i,4)))):min(r,ceil(pnts(i,1)+rset(pnts(i,4))))
      for yy=max(1,floor(pnts(i,2)-
      rset(pnts(i,4)))):min(r,ceil(pnts(i,2)+rset(pnts(i,4))))
      for zz=max(1,floor(pnts(i,3)-
      rset(pnts(i,4)))):min(r,ceil(pnts(i,3)+rset(pnts(i,4))))
            if (xx-pnts(i,1))*(xx-pnts(i,1))+(yy-pnts(i,2))*(yy-
            pnts(i,2))+(zz-pnts(i,3))*(zz-pnts(i,3)) <
            rset(pnts(i,4))*rset(pnts(i,4))
                  msk(xx,yy,zz) = 1;
            end
      end
      end
      end
end
```

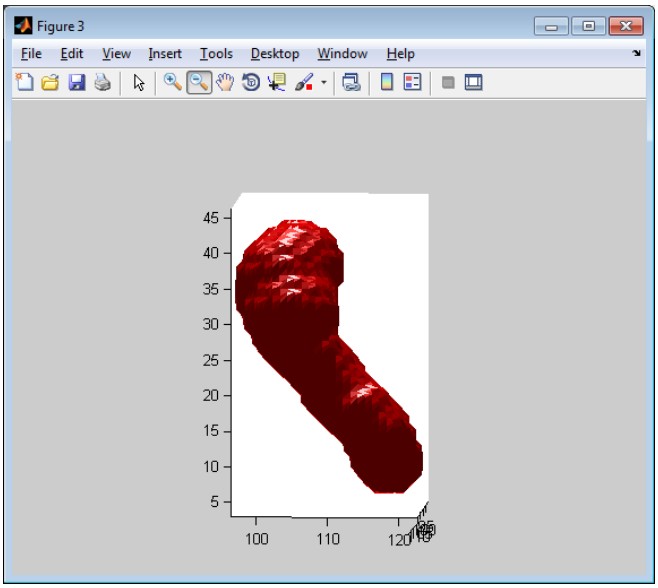
And display it:

```
h = figure(1); hold on;
inp = guidata(h);
inp.cntrs = zeros(2,1000,r);
for i=1:r
    if sum(sum(squeeze(msk(i,:,:))))>0
        cntr = contour(squeeze(msk(i,:,:))',0.5);
        inp.cntrs(:,1:length(cntr),i) = cntr;
    end
end
guidata(h,inp);


h = figure(2); hold on;
inp = guidata(h);
inp.cntrs = zeros(2,1000,r);
for i=1:c
```

```
    if sum(sum(squeeze(msk(:,i,:))))>0
        cntr = contour(squeeze(msk(:,i,:))',0.5);
        inp.cntrs(:,1:length(cntr),i) = cntr;
    end
end
guidata(h,inp);

M = isosurface(msk,0.5);
figure(3); clf;
DisplayMesh(M)
```







This is a much better result in the ‘x’ imaging plane but still not great in the ‘y’ plane. Why? Because the spinal cord is not perfectly round, it is elliptical in shape.

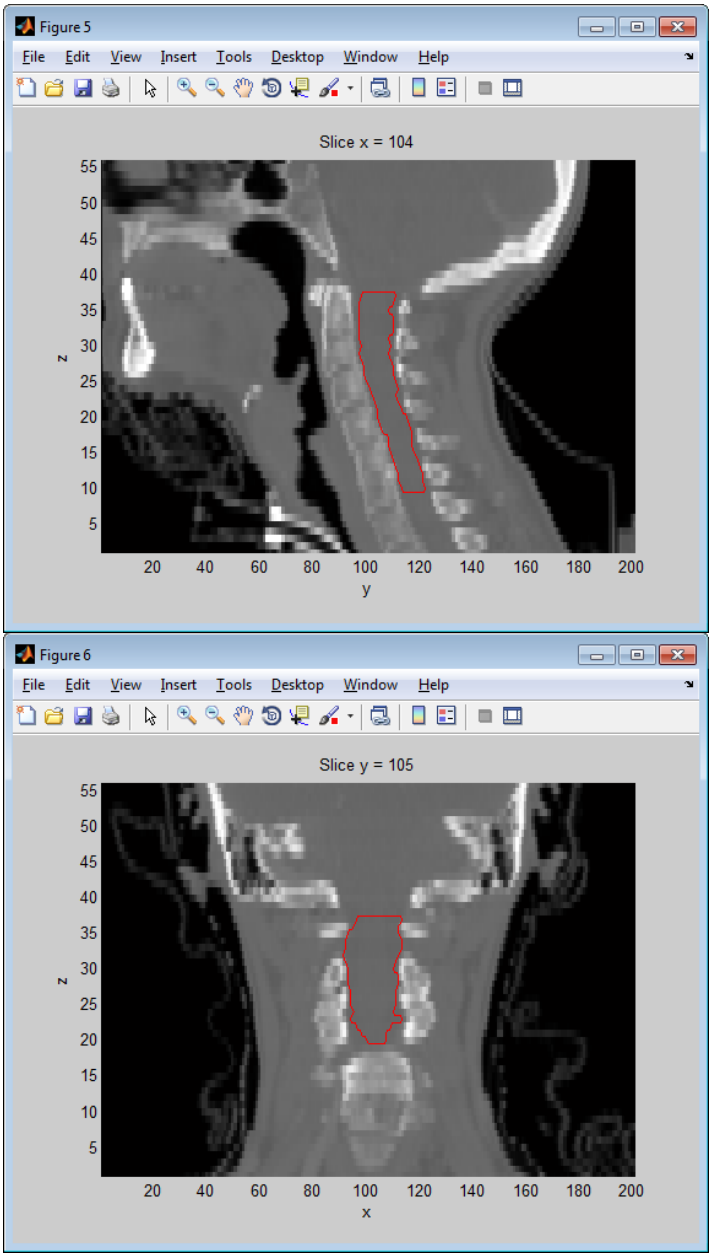
Let’s try a 5D graph with different radii for x and y to allow an elliptical shape. We create NodeNeibs3 using a similar cost function as NodeNeibs2 but with 5D nodes. Then, we do:

```
seednode = ((seedxyz(3)-1 + ((seedxyz(1)-1)*c + (seedxyz(2)-1))*d
)*length(rset))*length(rset) + 1;
for i=1:length(rset)
    for j=1:length(rset)
        endnode((i-1)*length(rset)+j) = ((endxyz(3)-1 + ((endxyz(1)-1)*c +
(endxyz(2)-1))*d)*length(rset)+i-1)*length(rset) + j;
    end
end

Done = zeros(r*c*d*length(rset)*length(rset),1);
Pointer = zeros(r*c*d*length(rset)*length(rset),1);
[nodepath,cost] =
GraphSearch(@NodeNeibs3,@NodeMark,@IsNodeMarked,@SetPointer,@GetPointer,seedn
ode,endnode);

pnts = floor((nodepath'-1)/(c*d*length(rset)*length(rset)))+1;
pnts(:,2) = floor((nodepath'-1-(pnts-
1)*d*c*length(rset)*length(rset))/(d*length(rset)*length(rset)))+1;
pnts(:,3) = floor((nodepath'-1-d*length(rset)*length(rset)*(pnts(:,2)-
1+c*(pnts(:,1)-1)))/(length(rset)*length(rset)))+1;
pnts(:,4) = floor((nodepath'-1-length(rset)*length(rset)*(pnts(:,3)-
1+d*(pnts(:,2)-1+c*(pnts(:,1)-1))))/(length(rset)))+1;
pnts(:,5) = mod(nodepath'-1,length(rset))+1;

msk = zeros(size(crp.data));
for i=1:length(pnts)
    for xx=max(1,floor(pnts(i,1)-
rset(pnts(i,4)))):min(r,ceil(pnts(i,1)+rset(pnts(i,4))))
```

```
    for yy=max(1,floor(pnts(i,2)-
rset(pnts(i,5)))):min(r,ceil(pnts(i,2)+rset(pnts(i,5))))
        zz = pnts(i,3);
        if (xx-pnts(i,1))*(xx-
pnts(i,1))/(rset(pnts(i,4))*rset(pnts(i,4)))+(yy-pnts(i,2))*(yy-
pnts(i,2))/(rset(pnts(i,5))*rset(pnts(i,5)))<=1
            msk(xx,yy,zz) = 1;
        end
    end
    end
end

h = figure(1); hold on;
inp = guidata(h);
inp.cntrs = zeros(2,1000,r);
for i=1:r
    if sum(sum(squeeze(msk(i,:,:))))>0
        cntr = contour(squeeze(msk(i,:,:))',0.5);
        inp.cntrs(:,1:length(cntr),i) = cntr;
    end
end
guidata(h,inp);


h = figure(2); hold on;
inp = guidata(h);
inp.cntrs = zeros(2,1000,r);
for i=1:c
    if sum(sum(squeeze(msk(:,i,:))))>0
        cntr = contour(squeeze(msk(:,i,:))',0.5);
        inp.cntrs(:,1:length(cntr),i) = cntr;
    end
end
guidata(h,inp);

M = isosurface(msk,0.5);
figure(3); clf;
DisplayMesh(M)
```

In NodeNeibs3 our cost function is

```
Cost = w + 2*lambda1/(radius_x + radius_y) + lambda2*(vari);
```



The 5D graph is more difficult to solve and takes about 6 minutes on my computer. But, building a graph that permits representing elliptical cross sections allows much more accurate results.