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

# Graph Cut 3D

This demo is an example of a 3D graph cut segmentation with user input.

```
img = ReadNrrd('0522c0001\img.nrrd');
crp = img;
crp.data = (img.data(200:320,140:225,52:80))/10+100;
crp.dim = size(crp.data);
% img.data = img.data/10+100;
figure(1); close(1); figure(1); colormap(gray(256));
DisplayVolume(crp);
crp.data = round((crp.data-100)*10);
```

User collected seeds from the GetSeeds function: `fore; back;`

We use the user seeds to build our probability distribution functions:

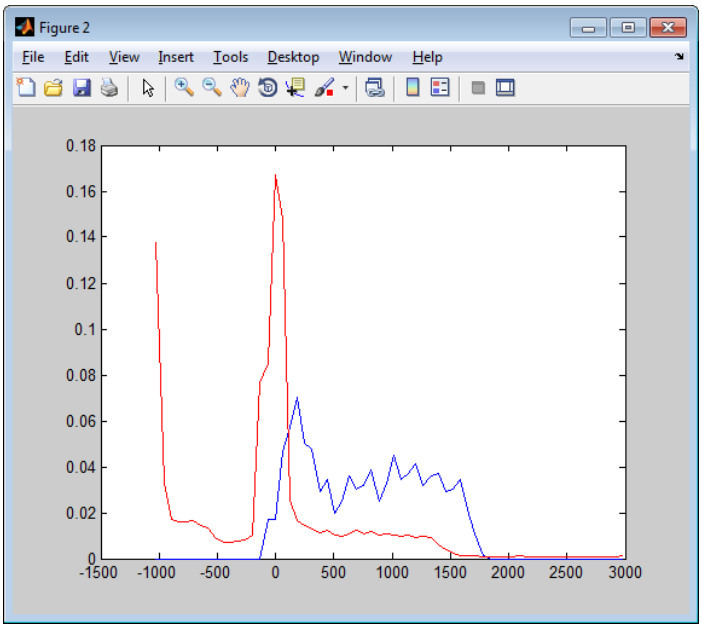
```
mn = min(crp.data(:));
mx = max(crp.data(:));
numbins=64;
binsize = (mx-mn+0.001)/numbins;
bins = [mn:binsize:mx];

hist_fore = zeros(1,numbins);
for i=1:size(fore,1)
    for j=max([1,floor(fore(i,1)-
fore(i,4))]):min([crp.dim(1),ceil(fore(i,1)+fore(i,4))])
        for k=max([1,floor(fore(i,2)-
fore(i,4))]):min([crp.dim(2),ceil(fore(i,2)+fore(i,4))])
            if (j-fore(i,1))*(j-fore(i,1))+(k-fore(i,2))*(k-
fore(i,2))<fore(i,4)*fore(i,4)
                hist_fore(floor((crp.data(j,k,fore(i,3))-mn)/binsize+1)) =
...
                    hist_fore(floor((crp.data(j,k,fore(i,3))-mn)/binsize+1))
+1;
            end
        end
    end
end
hist_fore = hist_fore/sum(hist_fore);
hist_back = zeros(1,numbins);
for i=1:size(back,1)
    for j=max([1,floor(back(i,1)-
back(i,4))]):min([crp.dim(1),ceil(back(i,1)+back(i,4))])
        for k=max([1,floor(back(i,2)-
back(i,4))]):min([crp.dim(2),ceil(back(i,2)+back(i,4))])
```

```
            if (j-back(i,1))*(j-back(i,1))+(k-back(i,2))*(k-
back(i,2))<back(i,4)*back(i,4)
                hist_back(floor((crp.data(j,k,back(i,3))-mn)/binsize+1)) =
...
                    hist_back(floor((crp.data(j,k,back(i,3))-mn)/binsize+1))
+1;
            end
        end
    end
end
hist_back = hist_back/sum(hist_back);
```

So that we don't have a zero probability, we add a small number to our pdfs:

```
hist_fore = hist_fore + 0.001;
hist_back = hist_back + 0.001;
```




Now we can do our Graph Cut:

```
function result = GraphCut3D(img,hist_fore,hist_back,binsize,sigma,lambda)
global Parent Tree Active EdgeCaps Edges Edge_Lens  r c d Orphans Orphan_cnt
PLengths;

[r,c,d] = size(img);
```

Initialize all our helper variables and set Edge capacities

```
FIFOInit(2*r*c*d);
Tree = zeros(r*c*d+2,1);
PLengths = zeros(r*c*d+2,1);
PLengths(1:r*c*d)=1;
Parent = zeros(r*c*d+2,1);
Active = zeros(r*c*d+2,1);
nodes = [1:r*c*d]';
X = mod(nodes-1,r)+1;
Z = floor((nodes-1)/(r*c))+1;
```

```
Y = floor((nodes-1-(Z-1)*r*c)/r)+1;
Edges = [X<r,X>1,Y<c,Y>1,Z<d,Z>1].*[nodes+1, nodes-1, nodes+r, nodes-r,
nodes+r*c, nodes-r*c];
Edge_Lens = sum(Edges'>0)';
EdgeCaps = zeros(r*c*d,8);
D = -ones(r,c,d);
D(1:end-1,:,:) =  exp(-((crp.data(1:end-1,:,:)-
crp.data(2:end,:,:)).^2)/(2*sigma*sigma));
EdgeCaps(:,1) = (1-lambda)*reshape(D,[r*c*d,1]);%x+1
D = -ones(r,c,d);
D(2:end,:,:) =  exp(-((crp.data(1:end-1,:,:)-
crp.data(2:end,:,:)).^2)/(2*sigma*sigma));
EdgeCaps(:,2) = (1-lambda)*reshape(D,[r*c*d,1]);%x-1
D = -ones(r,c,d);
D(:,1:end-1,:) =  exp(-((crp.data(:,1:end-1,:)-
crp.data(:,2:end,:)).^2)/(2*sigma*sigma));
EdgeCaps(:,3) = (1-lambda)*reshape(D,[r*c*d,1]);%y+1
D = -ones(r,c,d);
D(:,2:end,:) =  D(:exp(-((crp.data(:,1:end-1,:)-
crp.data(:,2:end,:)).^2)/(2*sigma*sigma));
EdgeCaps(:,4) = (1-lambda)*reshape(D,[r*c*d,1]);%y-1
D = -ones(r,c,d);
D(:,:,1:end-1) =  exp(-((crp.data(:,:,1:end-1)-
crp.data(:,:,2:end)).^2)/(2*sigma*sigma));
EdgeCaps(:,5) = (1-lambda)*reshape(D,[r*c*d,1]);%z+1
D = -ones(r,c,d);
D(:,:,2:end) =  exp(-((crp.data(:,:,1:end-1)-
crp.data(:,:,2:end)).^2)/(2*sigma*sigma));
EdgeCaps(:,6) = (1-lambda)*reshape(D,[r*c*d,1]);%z-1

EdgeCaps(:,7) = -lambda*log(hist_back(floor((img(:)'-mn)/binsize)+1));%s
EdgeCaps(:,8) = -lambda*log(hist_fore(floor((img(:)'-mn)/binsize)+1));%t
```

We need to define the hard constraints on the user supplied seeds:

```
K = max(sum((EdgeCaps(:,1:6).*(EdgeCaps(:,1:6)>0))'))+1;


for i=1:size(fore,1)
    for j=max([1,floor(fore(i,1)-
fore(i,4))]):min([crp.dim(1),ceil(fore(i,1)+fore(i,4))])
        for k=max([1,floor(fore(i,2)-
fore(i,4))]):min([crp.dim(2),ceil(fore(i,2)+fore(i,4))])
            if (j-fore(i,1))*(j-fore(i,1))+(k-fore(i,2))*(k-
fore(i,2))<fore(i,4)*fore(i,4)
                EdgeCaps(j+r*((k-1)+c*(fore(i,3)-1)),7) = K;
                EdgeCaps(j+r*((k-1)+c*(fore(i,3)-1)),8) = 0;
            end
        end
    end
end
for i=1:size(back,1)
    for j=max([1,floor(back(i,1)-
back(i,4))]):min([crp.dim(1),ceil(back(i,1)+back(i,4))])
        for k=max([1,floor(back(i,2)-
back(i,4))]):min([crp.dim(2),ceil(back(i,2)+back(i,4))])
```

```
            if (j-back(i,1))*(j-back(i,1))+(k-back(i,2))*(k-
back(i,2))<back(i,4)*back(i,4)
                EdgeCaps(j+r*((k-1)+c*(back(i,3)-1)),7) = 0;
                EdgeCaps(j+r*((k-1)+c*(back(i,3)-1)),8) = K;
            end
        end
    end
end
```

Clean up our edges at our border voxels:

```
for i=1:r*c*d
    EdgeCaps(i,1:Edge_Lens(i)) = EdgeCaps(i,Edges(i,1:6)>0);
    Edges(i,1:Edge_Lens(i)) = Edges(i,Edges(i,1:6)>0);
end
```

Now we can initialize our search tree

```
msk = EdgeCaps(:,7)>EdgeCaps(:,8);
FIFOInsert(nodes);
Active(1:r*c*d)=1;
Tree(msk)=1;
Tree(~msk)=2;
Tree(r*c*d+1)=1;
Tree(r*c*d+2)=2;
Parent(msk)=r*c*d+1;
Parent(~msk)=r*c*d+2;

Orphans = zeros(1,10000);
Orphan_cnt = 0;
```

And perform the cut:

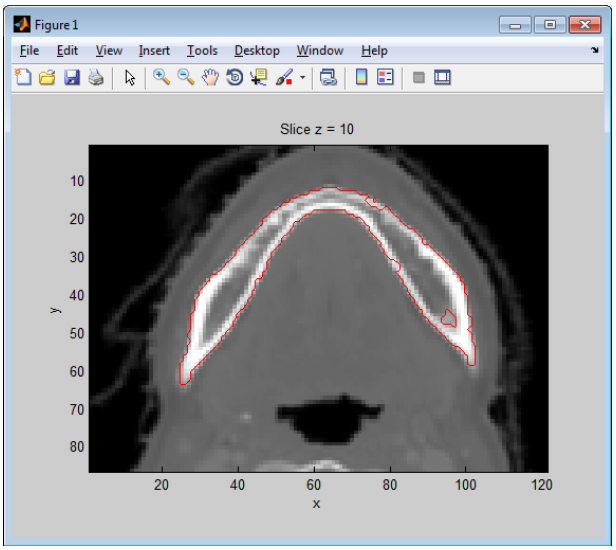
```
while (1)
    P = Grow();
    if isempty(P)
        break;
    end
    Augment(P);
    Adoption();
end
result = reshape(Tree(1:r*c*d),[r,c,d]);
return
```

Finally we call our new function to perform the cut:

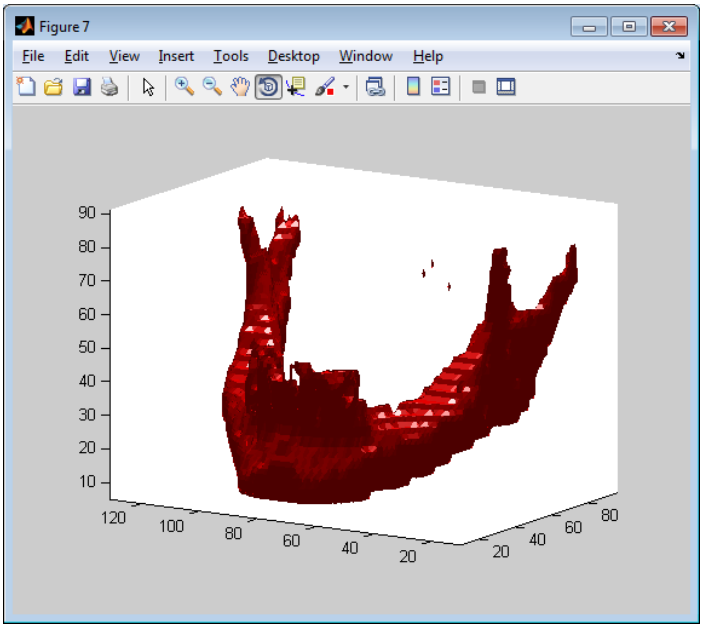
```
res = GraphCut3D(crp.data,hist_fore,hist_back,binsize,200,.025);
```

while it runs we can prepare to display the result:

```
inp = guidata(gcf);
inp.cntrs = zeros(2,1000,d);
for i=1:d
    cntr = contour(res(:,:,i)',1.5);
    inp.cntrs(:,1:size(cntr,2),i) = cntr;
end
guidata(gcf,inp);
```




```
figure();
msh = isosurface(res,1.5);
msh.vertices = msh.vertices.*repmat(crp.voxsz,[length(msh.vertices),1]);
DisplayMesh(msh)
```




```
man = ReadNrrd('0522c0001\structures\mandible.nrrd');
auto = man;
auto.data = res;
auto.data(auto.data(:)==2)=0;
Dice(auto,man)

      0.8764
```