EECE 8395: Medical Image Segmentation

Vanderbilt University

Prof. Jack H. Noble

Livewire 3D

In this demo, we will see how to implement Livewire in a multislice fashion to do 3D segmentation. First we create our function that will attach a new callback to the current figure and wait for the set of contours to be completed.

```
function LiveWireSegment3D()
global done3d
done3d=0;

hfig = gcf;
iptaddcallback(hfig,'KeyPressFcn',@LiveWire3DKeyPressCallback);
h = guidata(hfig);
h.lcntrs = zeros(1,h.img.dim(3));
h.cntrs = zeros(2,1000,h.img.dim(3));
guidata(hfig,h);


while ~(done3d)
    pause(0.5);
end
```

The callback let's the user use the escape key to indicate when they are finished and the 'l' key to conduct the LiveWire segmentation on the current slice. My LiveWireSegment function uses our sobel filter/canny edge combination as the default edge cost function if one is not provided as input.

```
function LiveWire3DKeyPressCallback(h,e)
global done3d
if (strcmp(e.Key,'escape'))
    done3d=1;
elseif strcmp(e.Key,'l')
    inp = guidata(h);
    LiveWireSegment(inp.img,[],inp.direction,inp.slc);
end
return;
```

We also modify our MouseButtonDownCallback so that when the LiveWire contour is completed (it is stored in donepath), this is copied into a data-structure that stores all the contours for all the image slices for the figure.

```
function MouseButtonDownCallback(h,e)
…
h = guidata(gcf);
h.lcntrs(h.slc) = donepathcnt+1;
h.cntrs(:,1:h.lcntrs(h.slc)-1,h.slc) = donepath(:,1:donepathcnt);
```

```
h.cntrs(:,h.lcntrs(h.slc),h.slc) = donepath(:,1);
guidata(gcf,h);
…
```

And we modify our DisplayVolume function so that it plots that set of contours as we page through the image slices.

```
function DisplayVolume(img,direction,slc)
…
if isfield(inp,'cntrs')
    hold on;

plot(inp.cntrs(1,1:inp.lcntrs(inp.slc),inp.slc),inp.cntrs(2,1:inp.lcntrs(inp.slc),inp.slc),'r');
end

…
```

Now we can try it:

```
img = ReadNrrd('0522c0001\img.nrrd');
slcim = img;
fp = [260,150,70];
sp = [320,220,107];

slcim.data = img.data(fp(1):sp(1),fp(2):sp(2),fp(3):sp(3))/2+100;
slcim.dim = size(slcim.data);
figure(1); close(1); hfig = figure(1); colormap(gray(256));
DisplayVolume(slcim,3,19);
LiveWireSegment3D();
```

That works well, we just need to be able to output the contour

```
function cntrs = LiveWireSegment3D()
…
h = guidata(hfig);
cntrs = h.cntrs;
return;
```

We can also try a custom cost function, where we select two thresholds (here 105 and 193) that correspond to an intensity between the optic nerve and CSF and an intensity between the optic nerve and bone.

```
function LiveWire3DKeyPressCallback(h,e)
global done3d
if (strcmp(e.Key,'escape'))
    done3d=1;
elseif strcmp(e.Key,'l')
    inp = guidata(h);
    costim = 50+repmat(min(abs(inp.img.data(:,:,inp.slc)-
        105),abs(inp.img.data(:,:,inp.slc)-193)),[1,1,8]);
    costim(:,:,5:8) = costim(:,:,5:8)*sqrt(2);
    LiveWireSegment(inp.img,costim,inp.direction,inp.slc);
```

```
end
return;
```

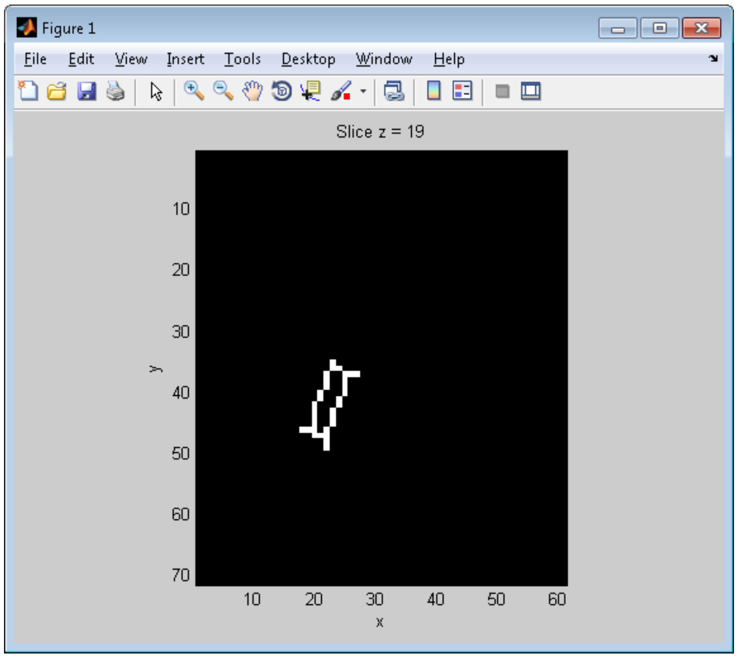
Next we need to convert our contours to a volumetric representation. This is done similarly to our connected component labelling algorithm. First we initialize some variables:

```
cntr2vol = slcim;
cntr2vol.data(:)=0;
neibs = [-1 0;+1 0;0 -1; 0 +1];
slcnum=19;
```

'neibs' defines the graph neighborhood. So we are going treat the image as a 4-connected graph. Next we loop over our contour points and fill in the image at those locations.

```
tot = sum(cntrs(1,:, slcnum)~=0);
for j=1:tot
   cntr2vol.data(cntrs(1,j, slcnum),cntrs(2,j, slcnum), slcnum)=1;
end

cntr2vol.data = cntr2vol.data*255;
figure(1); close(1); hfig = figure(1); colormap(gray(256));
DisplayVolume(cntr2vol,3,19);
hold on;
plot(cntrs(1,1:tot,slcnum),cntrs(2,1:tot,slcnum),'r')
cntr2vol.data = cntr2vol.data/255;
```



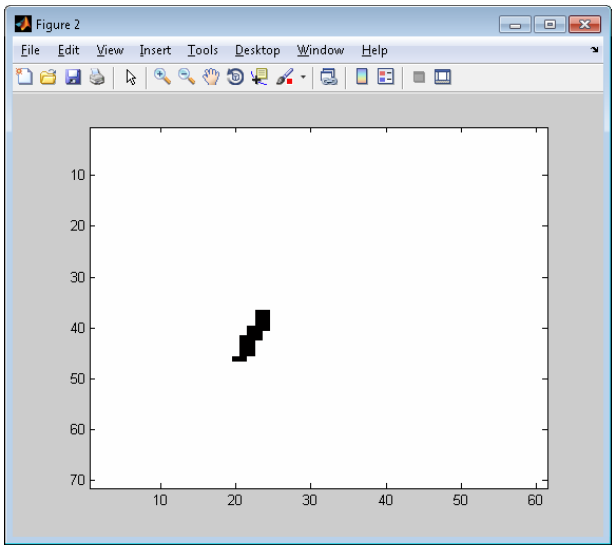
A few more helper variables:

```
done = zeros(cntr2vol.dim(1),cntr2vol.dim(2));
list1 = zeros(cntr2vol.dim(1)*cntr2vol.dim(2),2);
list2 = zeros(cntr2vol.dim(1)*cntr2vol.dim(2),2);
fillval=0;
```

```
lenlist1=1;
lenlist2=0;
```

We start the ‘floodfill’ at pixel [1,1]:

```
list1(1,:) = [1,1];
done(1,1)=1;
      while (lenlist1)
            cur = list1(lenlist1,:);
            lenlist1 = lenlist1-1;
            for j=1:length(neibs)
                nd =cur + neibs(j,:);
                if nd(1)>0 && nd(2)>0 && nd(1)<=cntr2vol.dim(1) &&
nd(2)<=cntr2vol.dim(2) && ~done(nd(1),nd(2))
                    if cntr2vol.data(nd(1),nd(2),slcnum) == 0
                        lenlist1 = lenlist1+1;
                        list1(lenlist1,:) = nd;
                    else
                        lenlist2 = lenlist2+1;
                        list2(lenlist2,:) = nd;
                    end
                    done(nd(1),nd(2))=1;
                end
            end
        end
figure(2); close(2); hfig = figure(2); colormap(gray(256));
image(done'*255)
```



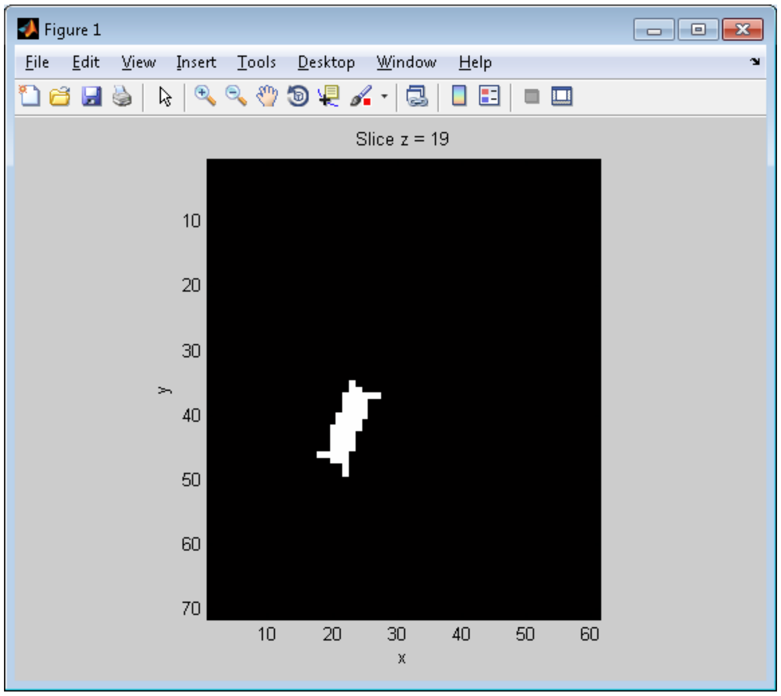
That one loop fills out the background up to the border of the contoured region, but now the foreground needs to be filled. To account for objects that might have holes, we iteratively swap between filling background and foreground until all the pixels in the image are visited.:

```
while (lenlist1)
      while (lenlist1)
            cur = list1(lenlist1,:);
            lenlist1 = lenlist1-1;
            % this if statement fills in the foreground when we are in a
foreground fill iteration
```

```
            if cntr2vol.data(cur(1),cur(2),slcnum)==0 && fillval
                cntr2vol.data(cur(1),cur(2),slcnum)=1;
            end
            for j=1:length(neibs)
                nd =cur + neibs(j,:);
                  % allows filling in the foreground on foreground fill iterations
                if nd(1)>0 && nd(2)>0 && nd(1)<=cntr2vol.dim(1) &&
nd(2)<=cntr2vol.dim(2) && ~done(nd(1),nd(2))
                    if cntr2vol.data(nd(1),nd(2),slcnum) == 0
                        lenlist1 = lenlist1+1;
                        list1(lenlist1,:) = nd;
                    else
                        lenlist2 = lenlist2+1;
                        list2(lenlist2,:) = nd;
                    end
                    done(nd(1),nd(2))=1;
                end
            end
        end
        list1 = list2;
        lenlist1 = lenlist2;
        lenlist2 = 0;
        fillval = 1 - fillval;
end
```

The outer while loop allows swapping list1 and list2 at each iteration. It also allows switching between background and foreground filling operations by doing `fillval = 1 - fillval;`. The extra if statement allows filling in the foreground on foreground fill iterations.

```
cntr2vol.data = cntr2vol.data*255;
figure(1); close(1); hfig = figure(1); colormap(gray(256));
DisplayVolume(cntr2vol,3,19);
hold on;
plot(cntrs(1,1:tot,slcnum),cntrs(2,1:tot,slcnum),'r')
```



Now to create a 3D mask, we just need to do this over all the slices. With the mask we can do things like measure Dice scores between two masks or create a 3D surface using the marching cubes (isosurface) algorithm.