Medical Image Analysis
Third hand-in

Mariuta Nicolae (rqt629) September 29, 2015

Exercise 1

For the non-rigid registration I have mostly followed the steps show in the lecture.

I started my loading the images and making a rescaling, by reducing 4 times the size of the images so I can be able to process them on my computer. I1 contains the baseline image and I2 is the follow-up.

```
%load the images and rescale so I can compute
B1 = my_load_mgh('nu1.mgz');
B1 = imresize3d(B1,0.25,[],'linear','bound');
B2 = my_load_mgh('nu2.mgz');
B2 = imresize3d(B2,0.25,[],'linear','bound');
I1 = B1;
I2 = B2;
```

After this, I tried to apply the rigid registration I did for the previous assignment and unfortunately, I got errors on it because as I shown last week, there were some issues with my rigid registration. I have continued the implementation without making any rigid registration.

```
%apply rigid registration with my cost function but it is not working
P = [0 0 0 0 0 0];
fdf = @(P)my_cost_function(P,I1,I2);
odf = fminunc(fdf,P);
I2 = my_3d_affine (I2, odf(4), odf(5),odf(6), odf(1), odf(2), odf(3));
```

I continued by creating the evaluation grid which is stored into the pts variable and the control grid that is stored in variable p. I also defined the spacing and other variables needed for the interpolation. The variable rTrivial that is needed in the cost function contains the values of the pixels at the evaluation grid coordinates. After this I applied the spline interpolation to calculate the variables needed for the cost function: the new alignment of new points and the indices.

```
%define spacing for the spline interpolation
spacing = [2 2 2];
offset = -spacing;
%create the evaluation grid
[gridX,gridY,gridZ] = ndgrid(1:2:64,1:2:64,1:2:64);
pts = [gridX(:) gridY(:) gridZ(:)];
%create the control points grid
[cpts] = ndgrid(1:4:64,1:4:64,1:4:64);
p = zeros([size(cpts)+2 3]);
%rTrivial values in the baseline grid
rTrivial = interp3(I1, gridX,gridY,gridY,'linear',0);
%make initial interpolation
[~,dfdp,idx] = SplineInterpolation(pts,p,offset,spacing);
```

After this, I used the cost function that I created for the rigid registration where I mostly use the functions created by Akshay for the spline interpolation, normalization and calculating the gradients of the control points.

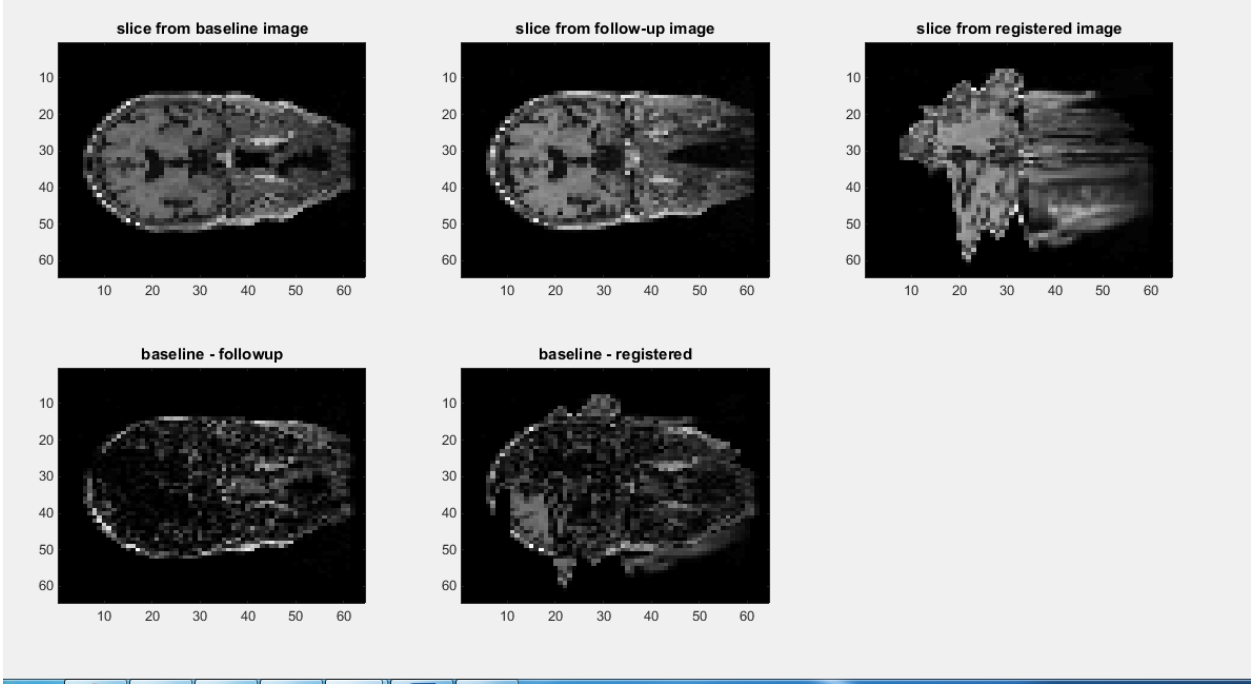
After this, I used the minFunc package to make the optimization of my cost function and calculate the optimal control points that have to be used for the registration.

```
%make optimization
options.Method = 'lbfgs';
options.MaxIter = 100;
controlPoints =
minFunc(@non_rigid_cost,p(:),options,pts,rTrivial,I2,dfdp,idx,szP,spacing);
```

In the last stage, I used the optimal control points to apply the spline interpolation again and obtained the non-rigid registered image.

```
%use optimization control points to spline interpolate the follow-up image
[x y z] = ndgrid(1:64,1:64,1:64);
ptsfinal = [x(:) y(:) z(:)];
controlPoints = reshape(controlPoints,szP);
deltaPoints = SplineInterpolation(ptsfinal,controlPoints,offset,spacing);
newPts = ptsfinal + deltaPoints;
registeredImage = interp3(I2,
reshape(newPts(:,2),size(I2)),reshape(newPts(:,1),size(I2)),reshape(newPts(:,
3),size(I2)));
```

In the end, I have plotted the results by showing a splice from the baseline, followup and the registered image, and also the difference images.



Unfortunately, the results are not good and the sum of squared error that I calculated is also a lot larger for registered image. I think it is because I did not make the rigid registration at the beginning since my code was not running. However, I worked pretty much together with Lou for this assignment and she got improved results since her rigid registration is working properly. However, I got a good understanding of how this algorithm described by Daniel Reuckert for the non-rigid registration works.

Code annexes

The main file

```
clear;
%compile the code files
% mex -O 'SplineInterpolation.cpp'
% mex -O 'PNorm_det.cpp'
%mex -O 'dDdPFunc.cpp'
%load the images

%load the images and rescale so I can compute
B1 = my_load_mgh('nu1.mgz');
B1 = imresize3d(B1,0.25,[],'linear','bound');
B2 = my_load_mgh('nu2.mgz');
B2 = imresize3d(B2,0.25,[],'linear','bound');
I1 = B1;
I2 = B2;


% %apply rigid registration with my cost function but it is not working
% P = [0 0 0 0 0 0];
% fdf = @(P)my_cost_function(P,I1,I2);
% odf = fminunc(fdf,P);
% I2 = my_3d_affine (I2, odf(4), odf(5),odf(6), odf(1), odf(2), odf(3));


%define spacing for the spline interpolation
spacing = [2 2 2];
offset = -spacing;
%create the evaluation grid
[gridX,gridY,gridZ] = ndgrid(1:2:64,1:2:64,1:2:64);
pts = [gridX(:) gridY(:) gridZ(:)];
%create the control points grid
[cpts] = ndgrid(1:4:64,1:4:64,1:4:64);
p = zeros([size(cpts)+2 3]);
%rTrivial values in the baseline grid
rTrivial = interp3(I1, gridX,gridY,gridY,'linear',0);
%make initial interpolation
[~,dfdp,idx] = SplineInterpolation(pts,p,offset,spacing);


szP = size(p);
```

```
%nonrigid cost function mostly for testing
d = my_ssd(I1,I2);
[f,df] = non_rigid_cost(p,pts,rTrivial, I2, dfdp, idx, szP, spacing);

%make optimization
options.Method = 'lbfgs';
options.MaxIter = 100;
controlPoints =
minFunc(@non_rigid_cost,p(:),options,pts,rTrivial,I2,dfdp,idx,szP,spacing);


%use optimization control points to spline interpolate the follow-up image
[x y z] = ndgrid(1:64,1:64,1:64);
ptsfinal = [x(:) y(:) z(:)];
controlPoints = reshape(controlPoints,szP);
deltaPoints = SplineInterpolation(ptsfinal,controlPoints,offset,spacing);
newPts = ptsfinal + deltaPoints;
registeredImage = interp3(I2,
reshape(newPts(:,2),size(I2)),reshape(newPts(:,1),size(I2)),reshape(newPts(:,
3),size(I2)));

%calculate sum of square difference at the end
dfinal = my_ssd(I1,registeredImage);

%display the results
subplot(2,3,1),imagesc((squeeze(I1(:,:,32)))),colormap('gray'),title('slice
from baseline image');
subplot(2,3,2),imagesc((squeeze(I2(:,:,32)))),colormap('gray'),title('slice
from follow-up image');
subplot(2,3,3),imagesc((squeeze(registeredImage(:,:,32)))),colormap('gray'),t
itle('slice from registered image');
subplot(2,3,4),imagesc(abs((squeeze(I1(:,:,32))) -
(squeeze(I2(:,:,32))))),colormap('gray'),title('baseline - followup');
subplot(2,3,5),imagesc(abs((squeeze(I1(:,:,32))) -
(squeeze(registeredImage(:,:,32))))),colormap('gray'),title('baseline -
registered');
```

The cost function

```
function [f,df] = non_rigid_cost(p,pts,rTrivial, ImMove, dfdp, idx, szP,
spacing)

%reshape the control points grid
p = reshape(p,szP);
Np = numel(p)/3;
offset = -spacing;

%apply spline interpolate and add delta to the evaluation points
deltaPoints = SplineInterpolation(pts, p, offset, spacing);
```

```
pts = pts + deltaPoints;

%dummy variables
det = ones(length(pts),1);
sourceVoxelSize = [1,1,1];
mx = 10;

%apply normalization
[f, d(:,1), d(:,2), d(:,3)] = PNorm_det(pts, rTrivial+2,ImMove+2,[0 130 0
130], [mx mx], [0 0 0], sourceVoxelSize, 2, double(det));
%calculate the gradients
df = dDdPFunc(d,dfdp,idx,Np);

df = df(:);

end
```