BE 537 - Grand Challenge 1

James Wang, Michael Lautman, Shreel Vijayvargiya

March 23, 2015

Introduction

For this project we explored methods for performing groupwise registration between a set of brain images. We performed registration on a set of 3D brain images solving for transformations from the set of images into a common reference space. The quality of a registration was established by measuring how closely a set of labeled features in the images corresponded when projected into the common frame via the transformations we built.

3.1 The Basic Component

To configure the environment in matlab we have included a setup script.

```
% Setup the environment
addpath(genpath('./train/'))
addpath(genpath('./test/'))
addpath(genpath('./segmentation template')
addpath(genpath('./NIFTI_20110921/'))
addpath(genpath('./starter_code/'))
```

3.1.1 Extend myView to Display Registration Results

Our project uses a visualizer that takes in a fixed image, a moving image, the voxel spacing in the image, a rotation matrix and a translation vector.

```
function myViewAffineReg(fixed, moving, spacing, A, b)
```

3.1.2 3D Affine Registration Objective Function

We compute the objective function for 3D registration using the equation below.

$$E(A,b) = \int_{\Omega} [I(x) - J(Ax+b)]^2 dx$$

```
function [E,g] = myAffineObjective3D(p,I,J,varargin)
```

3.1.3 Testing the Correctness of Gradient Computation

We verify our analytic gradient computation by computing a numerical gradient approximation that utilizes the central finite difference approximation.

$$\frac{\partial E}{\partial p_j}|_p \simeq \frac{E(p+\epsilon e_j) - E(p-\epsilon e_j)}{2\epsilon}$$

We then test our numerical gradient computation on an image `'sub001_mri.nii'` using the Matlab script `'p313.m'`. shows that The maximum relative error found using an $\epsilon = 1e-4$ and $\sigma = 1$ is $\approx 0.52\%$.

3.1.4 Testing our objective function by registering two images

We show that our objective function is able to compute the transformation between two images, `'sub001_mri.nii'` and `'sub003_mri.nii'`. Using the script `'p314.m'`, we solve for the A and b that register the images into a common coordinate frame.

$$A = \begin{vmatrix} 0.9569 & 0.1349 & 0.0145 \\ -0.1584 & 0.8760 & 0.3057 \\ 0.0276 & -0.2628 & 1.0851 \end{vmatrix} \quad b = \begin{vmatrix} -2.6059 \\ -1.0698 \\ 4.1131 \end{vmatrix}$$

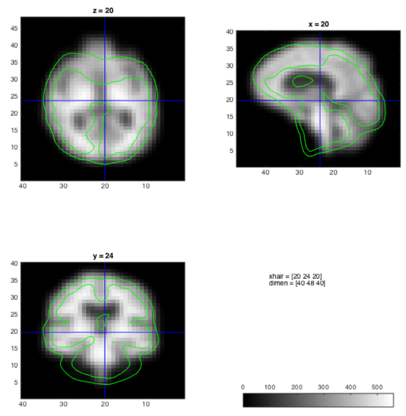

Figure 1: Before Registration

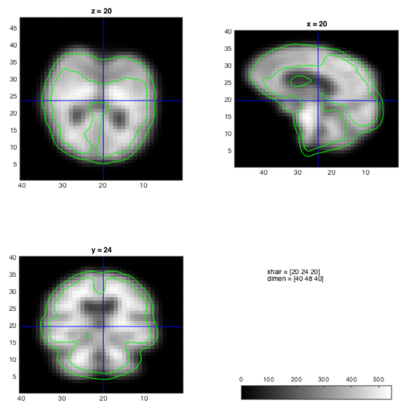

Figure 2: After Registration

3.1.5 Multi-Resolution Affine Registration

Using the script `'p315.m'`, demonstrate another method for registration that to compute the transformation between two images, `'sub001_mri.nii'` and `'sub003_mri.nii'`. This method down samples the images for fast registration between them and gradually increases the resolution of the images, using the coordinate transform just solved for as initial conditions for the next layer of registration.

$$A = \begin{vmatrix} 0.9674 & 0.1352 & 0.0159 \\ -0.1554 & 0.8860 & 0.3120 \\ 0.0171 & -0.2672 & 1.0999 \end{vmatrix} \quad b = \begin{vmatrix} -5.5977 \\ -2.8699 \\ 7.9202 \end{vmatrix}$$

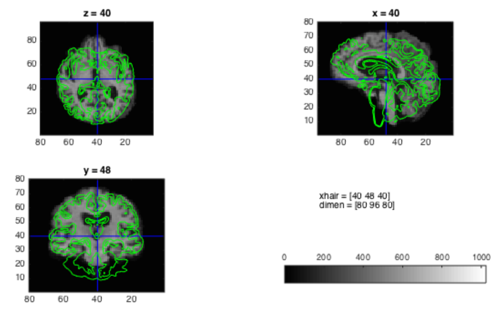

Figure 3: Before Multi-Layered Registration

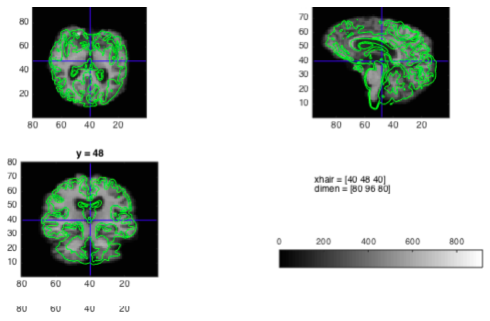

Figure 4: After Multi-Layered Registration

3.1.6 Speedups in the Objective Function

Using `my_interp3_precompute` and `my_interp3` for interpolation did nor result in a significant speedup. Using the `tic; toc;` function in Matlab for `myAffineRegMultiRes3D` with `iter = [20 10 5]` and `sigma = 1` for images `'sub001_mri.nii'` and `'sub002_mri.nii'`, we got a baseline time of 102.61 seconds. By precomputing the gradient we got the time down to 95.03 seconds. Replacing `interpn` with `my_interp3` however, increased the time to 173.45 seconds.

3.1.7 Baseline Groupwise Registration

Using `iter = [20 10 5]; sigma = 1.; fixed_img = sub021−sub040;`
We get `min(meanov) = .2827; max(meanov) = 0.3110; Mean(meanov) = 0.3025;`.

3.2 Extensions to the Base Component

3.2.1 Baseline Groupwise Registration

In this, histogram of moving and fixed was matched before finding optimum values of A and b. This method works the best if the intensity ranges of the images are different. So to test it, we used sub0021 as our fixed image, and used sub001, sub002 and sub003 as moving images. Before matching, mean overlap was `0.2399` and after matching, it improved to `0.2624`. However, when this scheme was applied to all the images, no improvement was seen. This is because histogram matching leads may lead to loss in intensity information. So method of histogram matching is unpredictable.
Source code: `myHistMatch.m`

3.2.2 Registration to an Unbiased Population-Derived Template

At 5 iterations, provided an improvement to mean overlap results relative to images with poor registration results (`min(meanovl) = 0.2827`), but did not improve results for images with good registration results (`max(meanovl) = 0.3110`). Multi-resolution affine transformation seems to be good enough that an average template doesn't offer much improvement.
Source code: `p322.m`

3.2.3 Defining a Mask

Using the given segmentations for training images, mask template was derived by using the unbiased population approach (`p323mask.m`). This mask was used to compute objective function. $J(Ax + b)$ was computed only at those points where $mask \neq 0$. This increased the computation speed significantly and allowed optimization of objective function at the highest resolution. This gave significant improvement in mean overlap

of the images increasing it to 0.3643 from 0.3111 with sub030 used as the fixed image.
When generating the mask template, we notice that the mean overlap goes down and then it becomes a constant. Initial template generated by just averaging the segmentation gives 0.3643 of mean overlap which goes down to 0.3468 after using the template generated after second iteration, and then it becomes constant for the templates generated in later iterations. $A0 = 0.3643$, $A1 = 0.3513$, $A2 = 0.3468$, $A3 = 0.3468$, $A4 = 0.3468$, $A5 = 0.3468$.
Source code: `myAffineObjective3DwithMask.m` and `myAffineMultiRes3DwithMask.m` and `p323mask.m`

3.2.4 Analytic Hessian Computation

The analytic hessian was computed within the objective function. For sub001 as the fixed image and sub002 as the moving image, maximum relative error was found to be 0.0584%. This shows that the results are still valid. This enhancement resulted in a significant speedup compared to `myAffineObjective3D.m`. This increased the speed of registration dramatically allowing us to run additional iterations at higher resolutions. Before the enhancement, we saw a baseline registration take $95.03 seconds$. After the enhancement, we saw the time drop to $33.81 seconds$. This speedup also enabled us to run many more tests as can be seen below
Source code: `myAffineObjective3DwithHessian.m`, `p324.m`

1 Table of results for enhancements

Name	Description	meanovl
3.2.1 Histogram Matching	Match histogram of fixed and moving image before performing registration	0.2846
3.2.2 Registration to an Unbiased Population-Derived Template	Create template by averaging input image intensities and registering to this average iteratively (j=5)	A0=0.2896 A5=0.3014
3.2.3 Defining a Mask	Set intensity to 0 for regions outside of the hippocampus using the segmentation images	.3643
3.2.4 Analytic Hessian Computation	Compute analytic Hessian in the objective function to speed up computations and perform more iterations at higher resolution	0.3111

2 Experiments

Fixed	Iter	Sigma	Hessian	Histogram	Mask	Template	Overlap
sub021	[20,10,0]	1	Yes	Yes	No	No	0.2306
sub030	[20,10,0]	1	Yes	Yes	No	No	0.2529
sub030	[20,10,0]	1	No	No	No	No	0.2889
sub021	[20,10,5]	1	Yes	No	No	No	0.3067
sub030	[20,10,5]	1	No	No	No	No	0.3106
sub030	[20,10,5]	1	Yes	No	No	No	0.3110
sub021	[20,10,5]	2	No	No	No	No	0.2798
sub030	[20,10,5]	2	No	No	No	No	0.2875
sub030	[40,20,5]	1	Yes	No	No	No	0.3105
sub030	[40,20,5]	2	Yes	No	No	No	0.2921
sub030	[80,40,5]	1	Yes	No	No	No	0.3111
sub030	[80,40,5]	1	Yes	Yes	No	No	0.2846
sub030	[80,40,5]	2	Yes	No	No	No	0.2909
sub030	[80,40,5]	2	Yes	Yes	No	No	0.2536
sub030	[80,40,20,5]	1	Yes	No	No	No	0.3107
sub030	**50**	**1**	**Yes**	**No**	**Yes**	**No**	**0.3643**
sub030	50	1	Yes	Yes	Yes	No	0.3643
A5_template	[20, 10, 5]	1	Yes	No	No	Yes	0.3014