BE 537 - Grand Challenge 1

James Wang, Michael Lautman, Shreel Vijayvargiya

March 22, 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

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)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% myView extended to display affine transformation results %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   inputs
%       fixed - fixed image
%       moving - moving image
%       spacing - voxel spacing
%       A - 3 x 3 rotation, scaling and shearing matrix
%       b - 3 x 1 translation matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Objective function for 3D Affine Transform  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inputs
%   p - 12 x 1 parameter vector
%   I - fixed image
%   J - moving image
```

```
%   varargin
%       dJ/dy - gradient of moving image in y direction
%       dJ/dx - gradient of moving image in x direction
%       dJ/dz - gradient of moving image in z direction
% outputs
%   E - value of the objective function
%   g - gradient of the objective function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3.1.3 Testing the Correctness of Gradient Computation

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

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

The maximum relative error we found using an epsilon of $1e-4$ was 0.01%.

```
% 3.1.3 Testing the Correctness of Gradient Computation
clear;

[image1,spacing] = myReadNifti('sub001_mri.nii');
[image2,spacing2] = myReadNifti('sub002_mri.nii');
p = [1,0,0,0,1,0,0,0,1,0,0,0]';

% Gaussian LPF
sigma = 1;
smoothedimage1 = myGaussianLPF(image1,sigma);
smoothedimage2 = myGaussianLPF(image2,sigma);

% analytical gradient
[E,g] = myAffineObjective3D(p,smoothedimage1,smoothedimage2);

% numerical gradient
epsilon = 1e-4;
gnumer = ones(12,1);

for j = 1:12
    % Create ej vector
    ej = zeros(12,1);
    ej(j) = 1;

    % Add/subtract ej*epsilon vector to p
    pup = p + ones(12,1).*ej*epsilon;
    pdown = p - ones(12,1).*ej*epsilon;

    % Compute E terms for numerical gradient approximation
    [Eup,~] = myAffineObjective3D(pup,smoothedimage1,smoothedimage2);
    [Edown,~] = myAffineObjective3D(pdown,smoothedimage1,smoothedimage2);

    % Compute dE/dpj
    gnumer(j) = (Eup - Edown)/(2*epsilon);
end

% Compute relative error
diffvector = g - gnumer;
relerr = 100.*(diffvector./g);
```