Investigation of Normalized Gradient Field based cross modality image registration

Thilan Costa & José Rincon

This report describes the attempt to recreate the results described in the paper *Intensity Gradient Based Registration and Fusion of Multi-modal Images* by E. Haber and J. Modersitzki (Methods of Information in Medicine, vol. 46(3), 2007).

1 Formulation of the problem

The paper formulates the registration problem between two images as an optimization problem where one tries to maximize the inner-product of the normalized gradient fields of the two images. The cost function D for a reference image R, and a transformed image T, where $R, T : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, is defined in the paper as follows

$$\begin{aligned} d &= \frac{\nabla T . \nabla R}{\|\nabla T\|_2 \|\nabla R\|_2} \\ D &= -\frac{1}{2n} d^T d \end{aligned} \tag{1}$$

where n is the total number of pixels within the image.

2 Problem formulation & Proposed solution

Given the cost function in equation (1), the authors indicate that the optimal transformation was found by using the *Gauss-Newton* method which requires one to compute the derivative of the function with respect to the parameters used in the transformation. However, the provided expressions for the derivative of the cost functions were found to be incomplete. Therefore, the derivatives for the optimization problem were formulated as follows:-

Given that R is an affine transformation of the image T, and $x = (x_1, x_2)$, then the ith entry of d in the above equation, corresponding to each pixel location of the images, can be written as

$$d_i = \frac{R_{x_1}(x_i) T_{x_1}(x_i - \phi(x)\theta) + R_{x_2}(x_i) T_{x_2}(x_i - \phi(x)\theta)}{\|\nabla T_x(x - \phi(x)\theta)\|_2 \|R_x(x_i)\|_2}$$

where

$$\phi(x) := \begin{pmatrix} x_1 & x_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & x_2 & 0 & 1 \end{pmatrix},$$

and

$$\theta := \begin{pmatrix} A_{11} & A_{21} & A_{12} & A_{22} & b_1 & b_2 \end{pmatrix}.$$

where the Affine transformation is given by

$$x' := \begin{pmatrix} A_{11} & A_{12} \\ A21 & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

Now the step direction for minimizing an objective function of the form

$$f_0(x) = \sum_{i=1}^{N} f_i(x)^2$$

using the Gauss-Newton method is given by

$$\Delta x = (\sum_{i=1}^{N} \nabla f_i(x) \nabla f_i(x)^T)^{-1} (\sum_{i=1}^{N} f_i(x) \nabla f_i(x)^T)^{-1}$$

It can be seen that the cost function proposed by the authors of the paper is of a similar form where $f_i(x) = d_i$. Therefore, the derivatives of d_i with respect to the parameters A and B (i.e. ∇d) are required to perform the *Gauss-Newton* method. The simplified expressions for the gradient of d that were derived are given in equation (2) given below

$$\nabla_\theta d = \sum_{i}^{N} d_i \phi(x_i)^T H(\frac{d_i}{\|\nabla T((x_i - \phi(x))\|_2^2} \nabla T(x_i - \phi(x)\theta) - \frac{1}{P_i} \nabla R(x_i)) \tag{2}$$

where

$$P_i = \|\nabla T(x_i - \phi(x)\theta)\|_2 \|\nabla R(\theta)\|_2$$

and H is the Hessian of $T(x_i - phi(x_i)\theta)$ evaluated at pixel x_i.

Using the above equations, the analytical gradient was coded using Matlab. To verify its accuracy, the numerical gradient with respect to the parameters was also estimated and compared with the analytical gradient given above. The tests showed that the gradients only matched in regards to the derivatives with respect to translations parameters (b_1, b_2). The reason for this discrepancy was not discovered and therefore it was decided to proceed forward with a transformation model that was limited to registering translations i.e.

$$\phi(x) := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$\theta := \begin{pmatrix} b_1 & b_2 \end{pmatrix}.$$

and the transformation model is given by

$$x' := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

Furthermore, due to time constraints, a simple gradient ascent method was used to converge toward the correct solution.

3 Data sources and optimization tools used

The data sources for the cross modality and same modality tests were obtained from the MRI and CT images provided for CS 673 assignments. The images were also slightly blurred to aid with registration and a multi-grid scheme was used. For optimization, a gradient ascent scheme was used. The steepest ascent direction (Δx) was computed using equation (2). The step size t was initially fixed to 0.5. The solution at each iteration was updated as

$$x = x + t\Delta x$$

The scheme was terminated after 1000 iterations or when the norm of the gradient at the solution was less than $10E-5$.

The drawback of the gradient ascent scheme is that the solution at each iteration may *zig-zag* and therefore the convergence rate maybe slower compared to using Gauss-Newton. However, it was considered acceptable since the focus was on accuracy as opposed to speed. In future, the solution can surely be enhanced by using a line search method (back-tracking or Amijo) for faster performance.

4 Results and Observations

The following table contains a summary of the results for a sample of test cases using images of the same modality. No results were recorded for multi-modal image registration since the proposed algorithm of this paper does not yield any accurate results.

Actual		NGF			MI	
x_1	x_2	Estimated x_1	Estimated x_2	$\|\nabla$at estimated point$\|_2$	x_1	x_2
1	1	0.9994	0.9986	9.67E-05	1.0096	1.0258
1	5	0.9995	4.9994	7.28E-05	1.007	5.0219
0	5	-7.2275	-3.5022	9.51E-05	0.031731	5.0185
-4	5	-4.0001	4.9995	7.52E-05	-4.0023	-5.0187
-4	-3	-3.9999	-3.0002	8.66E-05	-3.9999	-3.0213
-4	7	-3.8489	1.2342	9.78E-05	-4.0039	7.0259
-2	6	-2.0004	5.9987	8.31E-05	-2.0035	6.0288

As can be seen from the sample of results indicated in the above table, the NGF based estimations are in some cases inaccurate. But even these inaccurate estimate do seem to be at a local optimal given that the gradient is almost zero. On the other hand, the mutual information based estimates are more accurate throughout. It must also be noted that the mutual information based registration obtained results with high accuracy during cross modality image registration tests performed using the above sample test cases as well.

5 Analysis and conclusions

1. The proposed algorithm can be unpredictable in its performance of registering even small translations ($+/-$ 7 pixels) within the images of the same modality.
2. The iterative algorithm tends to have problems with converging to the correct solution due to local optima when registering images of the same modality.
3. The algorithm does not give good results for cross modality image registration.
4. In contrast, mutual information either outperformed or did as well as the proposed algorithm in almost all the cases that were tested.
5. Mutual information performs just as well for cross modality image registration.
6. The claim of the authors that the normalized gradient field based registration is a better option to mutual information cannot be accepted given the observed results.

Appendix

Main registration function

```
function [ estimated_params ] = NGF_registration(x_disp,y_disp )

%% Read the test images
f = imread('rire_t2.jpg');
fs = double( f(:,:,1) );

g = imread('rire_pd.jpg');
gs = double( g(:,:,1) );
%%

%% Deform the image for testing
%  Note: Only translations are considered for reasons explained in report
% x_disp = -3;
% y_disp = -5;

fprintf('Actual parameters = [%d, %d]\n',x_disp, y_disp);
```

```
actual_params = [ x_disp; y_disp]';

g = push_forward(fs, actual_params, [1, 1]);

%%

%% Scaling for multi-grid & Initial guess
scales = [0.125 0.25 0.5 1];
% scales = [1];

initial_guess = [0 0];

figure(1);
%%

%% Bluring the image for better registation
h=[0.05; 0.25; 0.4; 0.25; 0.05]';

f=conv2(h,h,f,'same');
g=conv2(h,h,g,'same');
%%

%% Iterate through each grid level to find optimal solution
for scale = scales

    % Scale the images for each grid
    fs = imresize(f, scale);
    gs = imresize(g, scale);

    new_params = initial_guess * scale;

    fs = double(fs);
    gs = double(gs);

    % Estimate new parameters using gradient descent
    new_params = gradient_ascent(fs,gs,new_params,[1, 1]*scale);

    initial_guess = new_params'/ scale;
end
%%

fprintf('Estimated parameters = [%d, %d]',initial_guess(1), initial_guess(2));

%% Output the estimated parameters
estimated_params = initial_guess;

end
```

Gradient Ascent function

```
function [ params ] = gradient_ascent(fs,gs,initial_guess,resolutions)
```

```
%% Set up variables
newf = fs;
[rows, cols] = size(gs);
counter = 1;
t = 0.5*resolutions(1); % Introduces an adapative aspect to t
new_params = initial_guess';
newf = push_forward(fs,new_params,resolutions);
df = Derivatives_NGF_v(fs, gs, resolutions);
old_params = initial_guess';
old_cost = 0;

%% Iterate through using Gradient ascent
while(norm(sum(df,2)) > 0.0001)

    d = ngf_D(newf,gs,resolutions);
    d = d(:);

    new_params = new_params + t*sum(df,2);


    newf = push_forward(fs,new_params,resolutions);

    if mod(counter,5)==0 %Display the both images every 5 iterations.
        C = imfuse(newf,gs,'falsecolor','Scaling','joint','ColorChannels',[1 2 0]);
        imshow(C)
    end

    df = Derivatives_NGF_v(newf, gs, resolutions);

    counter = counter  + 1;

    if(mod(counter,1000) == 0)
        if((resolutions(1) == 1) && (counter < 2000))
            %Increase t to speedup convergence
            t = 1;
        else
            break;
        end
    end

    old_params = new_params;
end

fprintf('Norm of gradient at grid scale %d was %d \n', resolutions(1), norm(sum(df,2)));

params = new_params;

end
```

Cost function d

```
function [ d ] = ngf_D( f, g, resolutions )
```

```
[row, column] = size(f);
n = row*column;

eps = 0.005;

[fx, fy] = gradient(f,resolutions(1), resolutions(2));
[gx, gy] = gradient(g,resolutions(1), resolutions(2));

norm_g_f = sqrt(fx.^2 + fy.^2 + eps.^2);
ngf_fx = fx./norm_g_f;
ngf_fy = fy./norm_g_f;

norm_g_g = sqrt(gx.^2 + gy.^2 + eps.^2);
ngf_gx = gx./norm_g_g;
ngf_gy = gy./norm_g_g;

d = ngf_fx.*ngf_gx + ngf_fy.*ngf_gy;

D = (1/(2*n))*d(:)'*d(:);

end
```

Analytical gradient

```
function [ df ] = Derivatives_NGF_v( f,g, resolutions )

eps = 0.005;

[rows,columns] = size(f);
t_y = -(rows-1)/2:(rows-1)/2; %defining vectors for mesh grid
t_x = -(columns-1)/2:(columns-1)/2; %defining vectors for mesh grid
MN = columns*rows;
[x_1,x_2] = meshgrid(t_x*resolutions(1),t_y*resolutions(2));

[Rx1, Rx2] = gradient(g, resolutions(1), resolutions(2));
[Tx1, Tx2] = gradient(f, resolutions(1), resolutions(2));
[Tx11, Tx12] = gradient(Tx1, resolutions(1), resolutions(2));
[Tx21, Tx22] = gradient(Tx2, resolutions(1), resolutions(2));

Tx1 = Tx1(:);
Tx2 = Tx2(:);
Rx1 = Rx1(:);
Rx2 = Rx2(:);

Tx11 = Tx11(:);
Tx12 = Tx12(:);
Tx21 = Tx21(:);
Tx22 = Tx22(:);

x_1 = x_1(:);
x_2 = x_2(:);
```

```
N0 = (Tx1.*Rx1 + Tx2.*Rx2);
N0 = [N0,N0]';

N1 = [-Rx1.*Tx11, -Rx1.*Tx12]';
N2 = [-Rx2.*Tx21, -Rx2.*Tx22]';

Di = sqrt(Tx1.^2 + Tx2.^2 + eps^2).*sqrt(Rx1.^2 + Rx2.^2 + eps^2);
Di = [Di,Di]';

C1 = [Tx11.*Tx1, Tx12.*Tx1]';
C2 = [Tx21.*Tx2, Tx22.*Tx2]';

C = C1 + C2;

norm_grad_T = sqrt(Tx1.^2 + Tx2.^2 + eps^2);
norm_grad_T = [norm_grad_T,norm_grad_T]';
norm_grad_R = sqrt(Rx1.^2 + Rx2.^2 + eps^2);
norm_grad_R = [norm_grad_R,norm_grad_R]';

A = (N1 + N2)./Di;

B = C.*N0./(norm_grad_T.^3.*norm_grad_R);

d = ngf_D(f,g,resolutions);

d = [d(:),d(:)]';
% d = 1;
df = (1/(rows*columns))*d.*(A+B);

% df =(A+B);
```