A Parallel-friendly Normalised Mutual Information Gradient for Free-Form Registration

Marc Modat\textsuperscript{a}, Gerard R. Ridgway\textsuperscript{a}, Zeike A. Taylor\textsuperscript{a}, Daves J. Hawkes\textsuperscript{a}, Nick C. Fox\textsuperscript{b} and Sébastien Ourselin\textsuperscript{a,b}

\textsuperscript{a}Centre for Medical Image Computing, University College London, UK, \textsuperscript{b}Dementia Research Centre, Institute of Neurology, University College London, UK.

ABSTRACT

Non-rigid registration techniques are commonly used in medical image analysis. However these techniques are often time consuming. Graphics Processing Unit (GPU) execution appears to be a good way to decrease computation time significantly. However for an efficient implementation on GPU, an algorithm must be data parallel. In this paper we compare the analytical calculation of the gradient of Normalised Mutual Information with an approximation better suited to parallel implementation. Both gradient approaches have been implemented using a Free-Form Deformation framework based on cubic B-Splines and including a smoothness constraint. We applied this technique to recover realistic deformation fields generated from 65 3D-T1 images. The recovered fields using both gradients and the ground truth were compared. We demonstrated that the approximated gradient performed similarly to the analytical gradient but with a greatly reduced computation time when both approaches are implemented on the CPU. The implementation of the approximated gradient on the GPU leads to a computation time of 3 to 4 minutes when registering $190 \times 200 \times 124$ voxel images with a grid including $57 \times 61 \times 61$ control points.

Keywords: Non-Rigid Registration, Normalised Mutual Information, Gradient, Graphics Processing Unit, Parallel Implementation

1. INTRODUCTION

Graphics Processing Unit- (GPU-) based execution is becoming more widely used in medical imaging, particularly when computation time is important, as is often the case in image registration. Non-rigid registration is a procedure commonly used in medical imaging and is often a pre-processing step in more complex studies. It can, for example, be used as intra-patient registration to align images provided by different imaging techniques or to quantify changes happening over time. It can be used as well for inter-patient studies where all data must be gathered into the same spatial frame of reference. It is important to decrease the computation time of such techniques to permit use on large cohorts. Some work has been done in the past to do so. As an example, the demons method\textsuperscript{1} can be seen as a simplification of the fluid algorithm,\textsuperscript{2} which leads to a significant computation time decrease. Concerning the Free-Form Deformation algorithm,\textsuperscript{3} Schnabel \textit{et al.}\textsuperscript{4} decreased the time requirement by deactivating unnecessary control points. Rohlfing \textit{et al.}\textsuperscript{5} exploited the redundancy within the cubic B-Splines computation to reduce the running time, and used high-performance computers. GPUs are easily programmable nowadays with dedicated languages, and their use could result in significant computation speed improvement. However, their computational power arises from highly parallel architectures, meaning their effectiveness hinges on the data parallelism of the algorithm; algorithms must be formulated to suit these parallel architectures.

An essential component of automatic registration algorithms is computation of a similarity measure, which quantifies the degree of correspondence between the images, and its gradient. The Normalised Mutual Information (NMI) is one such measure frequently used in registration algorithms, as it is suitable for multi-modal registration and appears to be robust to noise. However its gradient computation is expensive, both in terms of computation time and memory resources. The aim of this paper is to compare the analytical gradient of the

Send correspondence to Marc Modat, E-mail: m.modat@ucl.ac.uk
NMI with an approximation more suitable for parallel implementation. Using a Free-Form Deformation framework based on B-Splines we implemented two registration algorithms which differed only in their NMI gradient computation.

In the next section, we present the implementations used in the comparisons. We then examine the differences and similarities of the two methods in Section 3. In the final section we present a discussion of the results, and our conclusions.

2. METHOD

Our framework is related to the algorithm presented by Rueckert et al. A cubic B-Spline interpolation is used to deform a source image to match a target image. This guarantees $C^2$ continuous deformation but not necessarily a one to one mapping; we therefore use an objective function which includes the NMI similarity measure and an additional regularisation term which favours smooth transformations.

2.1 Deformation model

The source image is deformed using cubic B-Splines. A lattice of control points $\mu$ is superimposed on the reference image and the displacement field is then interpolated based on these control point positions. A movement of one control point will thus affect the position of all the voxels in its neighborhood and then provide a local deformation. A transformation $g$ maps locations in the target image to corresponding points in the source, such that $T(x)$ corresponds to $S(g(x))$. The deformed source image is also denoted by $S_g$. The new position of a voxel $x$ can be calculated from

$$g(x) = x + \sum_{i=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} \beta^3(u)\beta^3(v)\beta^3(w) \mu_{i+l,j+3m,k+3n} \tag{1}$$

where $\beta^3$ are the cubic spline basis functions, $u$, $v$ and $w$ are the relative positions of $x$ along the $x$, $y$ and $z$ axes, respectively. We point out that this model allows computation of each voxel’s new position independently. From the point of view of execution order, it is thus suitable for a parallel implementation in which each voxel is processed by a separate execution thread.

2.2 Similarity measure and its gradients

The NMI is a voxel intensity-based information theoretic metric based on the Shannon entropy formula. Optimising the NMI value maximises the shared information of two images. The NMI is defined as

$$\text{NMI} = (H(T) + H(S_g)) / H(T, S_g), \tag{2}$$

where $H(T)$ and $H(S_g)$ are the marginal entropies of images $T$ and $S_g$, respectively, and $H(T, S_g)$ is their joint entropy. It is possible to compute analytically the derivative of the NMI for each degree of freedom of the deformation as

$$\frac{\partial \text{NMI}}{\partial \mu_{i,j,k}^\xi} = \frac{\partial H(T)}{\partial \mu_{i,j,k}^\xi} + \frac{\partial H(S_g)}{\partial \mu_{i,j,k}^\xi} - \text{NMI} \times \frac{\partial H(T, S_g)}{\partial \mu_{i,j,k}^\xi} \tag{3}$$

where $\mu_{i,j,k}^\xi$ is the $\xi^{th}$ coordinate of the control point with coordinate $i, j, k$ along the $x, y$ and $z$ axes. The different entropy derivatives are computed using a joint histogram derivative. We use Parzen Window methods with a cubic B-Spline kernel to initially fill the joint histogram $H$. For every bin pair we have:

$$H(t, s) = \sum_{x=0}^{N-1} \beta^3(T(x) - t) \beta^3(S(g(x)) - s) \tag{4}$$

where $N$ is the number of voxels in the target image and $t$ and $s$ are the bin intensities in the target and source images, respectively.
To fill the derivative of the joint histogram the same scheme is applied:

\[
\frac{\partial H(t, s)}{\partial \mu^{\xi}_{ijk}} = \sum_{x=0}^{N-1} \beta^3 (T(x) - t) \frac{\partial \beta^3 (i - s)}{\partial i} \bigg|_{i=S(g(x))} \frac{\partial S(p)}{\partial p} \bigg|_{p=g(x)} \frac{\partial g(x)}{\partial \mu^{\xi}_{ijk}} .
\]  
(5)

For further information, we refer the reader to Loeckx.\textsuperscript{7}

This approach provides the mathematical value of the gradient but is extremely memory intensive, since for every degree of freedom a joint histogram must be allocated. A parallel implementation (for example using GPUs) would require sufficient memory for these to be allocated concurrently. By way of example, registration of a 200\textsuperscript{3} voxels image using 4 voxel control point spacing results in a total of 51\textsuperscript{3} control points. If we consider 64 bin joint histograms, 3 degrees of freedom per control point, and single precision, 51\textsuperscript{3} \times 3 \times 64\textsuperscript{2} \times 4 = 6.5 GB of memory are required. For comparison, an NVidia 8800GTX-based graphics card, which we use in this paper, has 768 MB of memory. Additionally control point-centric computation of the gradient in this manner leads to significant redundancy as each voxel’s influence is recomputed for each affected control point.

### 2.2.1 An approximation suited to parallel implementation

The second approach we implemented was based on the approximation of the NMI gradient proposed by Crum et al.\textsuperscript{8} For every voxel a local deformation is simulated and the impact of the movement on the similarity measure is quantified. In the target image we consider one voxel \(T(x, y, z)\) with intensity \(m\). In the source image the corresponding voxel \(S(g(x, y, z))\) has an intensity of \(s\). For the gradient calculation along the \(x\) axis, for example, we require intensities \(r\) and \(t\) of voxels \(S(g(x - 1, y, z))\) and \(S(g(x + 1, y, z))\), respectively. From the Joint-Entropy gradient \(E_x\) and the Mutual Information gradient \(F_x\),

\[
E_x = -\frac{1}{N} \log \left[ \frac{p_{mr}}{p_{mt}} \right],
\]  
(6)

\[
F_x = \frac{1}{N} \log \left[ \frac{p_{mr}}{p_r} \right],
\]  
(7)

the NMI gradient can be calculated from

\[
G_x = \frac{1}{H(A, B)^2} [H(A, B) \times F_x - \text{normal MI} \times E_x]
\]  
(8)

This technique was originally developed for algorithms like the fluid scheme,\textsuperscript{8} where one gradient value is obtained per voxel. However by applying a convolution window to the gradient image we can estimate the gradient for every control point \(\mu^{\xi}_{ijk}\). The chosen convolution window is a cubic B-Spline curve which matches the basis functions in the deformation model in terms of control-point spacing; it is equivalent to \(\frac{\partial g(x)}{\partial \mu^{\xi}_{ijk}}\) in equation 5.

### 2.3 Regularisation term and its gradient

In order to constrain the deformation of the source image to be smooth, a weighted penalty term \(P\) is added to the NMI value. The constraint we describe here, the bending-energy, was used for non-rigid registration by Rueckert et al.\textsuperscript{3} It is defined as

\[
P = \frac{1}{N} \sum_{x=0}^{N-1} \left( \frac{\partial^2 g(x)}{\partial x^2} \right)^2 + \left( \frac{\partial^2 g(x)}{\partial y^2} \right)^2 + \left( \frac{\partial^2 g(x)}{\partial z^2} \right)^2 + 2 \left( \frac{\partial^2 g(x)}{\partial x \partial y} \right)^2 + 2 \left( \frac{\partial^2 g(x)}{\partial x \partial z} \right)^2 + 2 \left( \frac{\partial^2 g(x)}{\partial y \partial z} \right)^2 .
\]  
(9)

Abbreviating equation 9 as

\[
P = \frac{1}{N} \sum_{x=0}^{N-1} A^2 + B^2 + C^2 + 2D^2 + 2E^2 + 2F^2,
\]  
(10)
the derivative of the penalty term involves a sum of derivatives each of which can be obtained using the chain rule, e.g.

$$\frac{\partial (A^2)}{\partial \mu_{ijk}} = \frac{\partial (A^2)}{\partial A} \frac{\partial A}{\partial \mu_{ijk}} = 2A \frac{\partial A}{\partial \mu_{ijk}},$$

leading to:

$$\frac{\partial P}{\partial \mu_{ijk}} = \frac{1}{N} \sum_{x=0}^{N-1} \left[ 2 \times \left( A \times \frac{\partial A}{\partial \mu_{ijk}} + B \times \frac{\partial B}{\partial \mu_{ijk}} + C \times \frac{\partial C}{\partial \mu_{ijk}} \right) + 4 \times \left( D \times \frac{\partial D}{\partial \mu_{ijk}} + E \times \frac{\partial E}{\partial \mu_{ijk}} + F \times \frac{\partial F}{\partial \mu_{ijk}} \right) \right].$$

### 2.4 Framework

Voxel-wise gradient computation and control point-wise deformation optimisation naturally lend themselves to parallel execution. A GPU scheme in which concurrent execution threads handle individual voxel gradient calculations and individual control point displacements was implemented using the CUDA API.

Once the gradient $\frac{\partial M}{\partial \mu_{ijk}}$ is calculated for every control point $\mu_{ijk}$ we perform a line descent in the direction of the overall gradient for all control points simultaneously. When displacing control points in the direction of the gradient produced no further metric improvement we recomputed the gradient and restarted the line ascent from the position with highest metric value. The optimisation stopped when a displacement of 0.01 mm produced no further metric improvement. The whole framework is presented in Fig 1. There is scope for further computation time reduction using a more sophisticated optimiser, however this was not investigated; the objective of our study was to evaluate the gradients’ accuracy, not their optimisation schemes.

### 3. EXPERIMENT

In order to compare the gradient computation efficiency and accuracy we used the two implementations to register the same dataset.

#### 3.1 Data

We used 3D-T1 weighted magnetic resonance imaging (inversion recovery (IR)-prepared spoiled GRASS sequence: TE 6.4 ms, TI 650 ms, TR 3000 ms, bandwidth 16 kHz) from 65 elderly subjects, of which 40 were clinically diagnosed with Alzheimer’s disease (AD) and 25 were age-matched healthy controls. The image dimensions were $190 \times 200 \times 124$ voxels with a $0.9375 \times 0.9375 \times 1.5$ mm spacing size.

To generate a pair of images related by a known, or ground truth, transformation we first non-linearly registered each subject image (source) to a specific template (target) as shown in Fig 2, step 1. The resultant warped images were then treated as the new target images in the pair, connected to their corresponding source image via the deformation field. For these registrations we used the diffeomorphic demons algorithm. This step provides a set of complex deformation fields to test our own non-rigid registration scheme, without any bias towards the spline model.

Subsequently we used our spline algorithm with both gradient implementations to register each source to its corresponding simulated target, attempting to recover the underlying deformation field (Fig 2, step 2). A multi-resolution approach with a 3 mm spacing between the nodes at the final stage and a constraint weighting factor of 0.005 was used for each registration.

In the following section we compare the ground truth with the recovered deformation fields using both gradients implemented on the CPU. The computational advantages of the approximate gradient formulation, in particular its suitability for parallel execution are also demonstrated with a GPU-based implementation. The reported computation times were obtained using an NVidia 8800GTX GPU, which included 128 processors and 768 MB of memory, as mentioned.
3.2 Results

The norm of the difference between the ground truth and each of the obtained deformation fields gives a value of the error at each voxel. Averaging these errors over all voxels (not excluded by a brain mask) gives an error value for each registration. Table 1 presents the mean and standard deviation of these registration errors over all subjects for each gradient implementation. A paired two-sample t-test comparing the errors (paired over subjects) resulted in a 95% confidence interval of (-0.0132, 0.0691) and a p-value of 0.197, showing that the difference is not significant at the 5% level. The average computation times are also presented in this table and show that the analytical gradient framework takes longer to reach the stopping criteria. The time difference between the approaches comes from two sources: the number of steps performed, and the difference in the gradient computation time (both higher for the analytical gradient). In fact, independently we find that the approximated gradient is roughly 5 times faster than the analytical gradient. This figure was obtained by running each gradient computation 100 times with $190 \times 200 \times 124$ voxel images and $36 \times 38 \times 38$ control point grids. The larger number of iterations for the analytical gradient produce additional subtle refinements before the stopping criteria is reached. These refinements do not improve the quality of the registration dramatically, but significantly increase its computation time.

Fig 3 presents an example of registration using both gradient computation methods. It can be seen that the difference between the two result images is very low.
Figure 2. Illustration of the steps in our experimental investigation. Step 1 involved registering each subject image to a template using the diffeomorphic demons algorithm. The resulting transformations then provided a ground truth against which we assessed the registrations achieved with analytic and approximate gradients in an FFD scheme. These latter registrations constituted step 2. Use of the demons algorithm to provide the ground truth deformations ensured there was no bias toward the spline deformation model.

<table>
<thead>
<tr>
<th>Error (std dev)</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical gradient</td>
<td>0.2855 mm (0.6454)</td>
</tr>
<tr>
<td>Approximated gradient</td>
<td>0.3135 mm (0.3487)</td>
</tr>
</tbody>
</table>

Table 1. Results comparison with both gradient implementations

4. DISCUSSION

In this paper we have compared the analytical gradient of the NMI with an approximated gradient. The mean registration errors were (statistically) not significantly different between the two gradient approaches.

However, the computation time differs dramatically between the two approaches for two reasons: primarily (1) the analytical gradient implementation performs more iterations per registration, and (2) the approximated gradient computation itself is 5 times faster.

The results of this study suggest that the approximated gradient should furnish similar accuracy to the analytical gradient, and may be employed without degradation of results. Moreover, as the approximated gradient is more parallel-friendly than the analytical one, mainly due to lower redundancy and reduced memory requirements, we were able to implement it for efficient GPU execution. The average computation time of the GPU implementation of the framework using the approximate gradient was 3.33 minutes.

REFERENCES

Figure 3. A source image (b) has been registered to a target image (a) using both gradient computation methods. Images (c) and (d) are the resultant warped images using respectively the real and the approximated gradient. Image (e) is the difference image before any non-rigid registration; (f) and (g) are the difference images between the target image and the result images using both methods.


