Fast Free-Form Deformation using the Normalised Mutual Information gradient and Graphics Processing Units

Marc Modat[1], Zeike A. Taylor[1], Josephine Barnes[2], David J. Hawkes[1], Nick C. Fox[2], and Sébastien Ourselin[1]

[1] Centre for Medical Imaging Computing, Department of Medical Physics and Bioengineering, University College London, UK,
[2] Dementia Research Centre, Institute of Neurology, University College London, UK.

Abstract. Non-rigid registration is a tool commonly used in medical image analysis. However techniques are usually time consuming. In this paper we present a fast registration framework which is a modification of the well-known Free-Form Deformation (FFD) algorithm. Our algorithm uses the analytical Normalized Mutual Information gradient which leads to a highly parallel framework. Its implementation is therefore suitable for execution via Grapics Processing Units. We apply the new method to estimate the brain atrophy on Alzheimer's disease subjects and show that accuracy is similar to the classical FFD, however the computation time is dramatically decreased.

1 Introduction

In longitudinal studies of atrophy the Boundary Shift Integral [1] (BSI) technique is widely used in imaging studies [2]. However this method is labour-intensive as it requires segmentation of both the brain baseline and repeat scans. Importantly segmentations are only semi-automatic and thus require significant operator time. The clinical trial sizes are increasing and consequently the time spent segmenting the brain scans. Using the Jacobian Integration [3] (JI) the segmentation time can be reduced by half as only one brain segmentation is necessary. The JI requires a non-rigid registration pre-step to compute the Jacobian determinant map. The most common non-rigid frameworks used in clinical trials are the fluid [4] and the Free-Form Deformation [5] (FFD) algorithms. FFD has been shown to perform well in Alzheimer's disease patient atrophy estimation [6]. Although FFD appears to be more accurate than the BSI [3], it is very time consuming.

Some groups have implemented supercomputer- [7] or FPGA-based [8] solutions to accommodate time constraints. However these kinds of hardware are either high-cost or require specialised skills. Graphics Processing Unit- (GPU-) based computation is a more accessible high performance alternative which has been shown to be effective for computationally expensive applications [9]. However since GPUs are highly parallel devices this effectiveness depends on the level

of parallelism in the algorithm. We present a modified FFD framework which is more suitable for parallel execution. This is principally achieved by driving the transformation with the analytical Normalised Mutual Information gradient which allows reinterpolation of the whole image at each step.

In the next section we present the Fast Free-Form Deformation (F3D) algorithm and its implementation on rapid GPU hardware. Then we apply the framework to real Alzheimer's disease subjects and controls to estimate their brain atrophy. We compare our results with those obtained with the classical FFD algorithm.

2 Method

Our framework is related to the non-rigid registration scheme proposed by Rueckert *et al.* [5]. A cubic B-Splines interpolation is used to transform a source image to match a target image. This transformation is driven by the Normalised Mutual Information (NMI) and its normalised gradient.

2.1 Cubic B-Splines interpolation

The cubic B-Splines framework is well documented elsewhere [5], and the details are omitted for brevity. However we note that a particularly favourable property of the framework is that any deformation produced with a grid of density n can be exactly produced on a grid of density $2n-1$. This property has been used in a pyramidal approach in our implementation. However cubic B-Splines suffer from being extremely computationally expensive. For this reason in the classical approach only one control point is optimised at a time which means the whole image does not have to be fully interpolated at each step. However the computation of each voxel's position and their new intensities are fully independent and thus their computation is suitable for parallel implementation. For this reason at each step we optimise all control points and interpolate the whole image.

2.2 Optimisation using the NMI gradient

To optimise the control point position we use Normalised Mutual Information [10, 11] (NMI) which is a voxel intensity-based information theoretic metric based on the Shannon entropy formula: $H = p(e) . \log(p(e))$ where $p(e)$ is the probability of an event e.

By optimizing the Mutual Information(1) (MI) or the NMI(2) we aim to maximise the information one image has about another image. MI and NMI are defined as:

$$MI = H(A) + H(B) - H(A, B) \tag{1}$$

$$NMI = (H(A) + H(B)) / H(A, B) \tag{2}$$

In (1) and (2) $H(A)$ and $H(B)$ are the marginal entropies respectively of the image A and B and $H(A, B)$ denotes their joint entropy. The probability computation is based on a joint histogram which indicates the probability of each

combination of intensity in the A and B images. These metrics are suitable for muti-modal registration as they are not only based on the voxel intensity but on the quantity of information given by the intensity. The metric is maximised when A and B are aligned.

To align our two images we displace the B-Splines control points to maximise the metric value. To do so we use the gradient of the NMI at each control point. In the classical FFD implementation the control points are displaced in each direction, and the local changes are taken into account to update the result image and the joint histogram. From this update, the new metric value is calculated from which the metric gradient in one direction is obtained. Summation of gradient in each direction yields a single resultant gradient at each control point. Once the gradient is known a steepest gradient descent is performed by updating the result image with the local changes. This technique is used for each control point and to guarantee a good registration the whole loop is performed several times. However this scheme is not suitable for our method as we aim to optimise all the node positions and interpolate the result image at once. Consequently, we use the analytical gradient for each voxel proposed by Crum *et al.* [12].

To the best of our knowledge there is no implementation of the FFD using such a gradient formulation. It is possible to compute anatically the NMI gradient for each voxel by considering an infinitesimal displacement at a voxel resolution. In the Target image we consider one voxel Target(x,y,x) with intensity m. In the Source image the corresponding voxel Source(x,y,z) has an intensity of s. For the gradient calculation along the x axis for example we need Source($x-1$,y,z) and Source($x+1$,y,z) which intensities are respectively equal to r and t. From the Joint-Entropy gradient, $E_x = -\frac{1}{N}\log\left[p_{mr}/p_{mt}\right]$ and the MI gradient ,$F_x = \frac{1}{N}\log\left[\frac{p_{mr}}{p_r}/\frac{p_{mt}}{p_t}\right]$, the NMI gradient(3) can be calculated from:

$$G_x = \frac{1}{H\left(A,B\right)^2}\left[H\left(A,B\right)\times F_x - MI\times E_x\right] \tag{3}$$

where N is the number of voxels in our images and p_{mr}, p_{mt}, p_r and p_t are extracted from the joint histogram. p_{mt} is the probability of having the intensity m in image Target and the intensity t in the Source image and p_r is the probability of having intensity r in the Target image. A similar procedure is used in the other directions. For further information refer to [12].

Once the NMI gradient is known for each voxel we use a Gaussian convolution to smooth the gradient information. A normalisation step described in the next part is then used to convert the gradient information to displacements. The smoothing allows voxels close to a control point to have more impact than voxels further away. Moreover it allows generation of a smooth displacement, wherein the larger the standard deviation value of the Gaussian window, the smoother the deformation field, as shown on Fig. 1. For this figure we generate a sphere and a cube and then smooth them. We perform the registration between these two images with different standard deviation and apply the deformation field to a regular grid to visualise the deformation. All the registrations were performed with the same parameters: 3 levels of spacing (20, 10 and 5 mm), 16 histogram

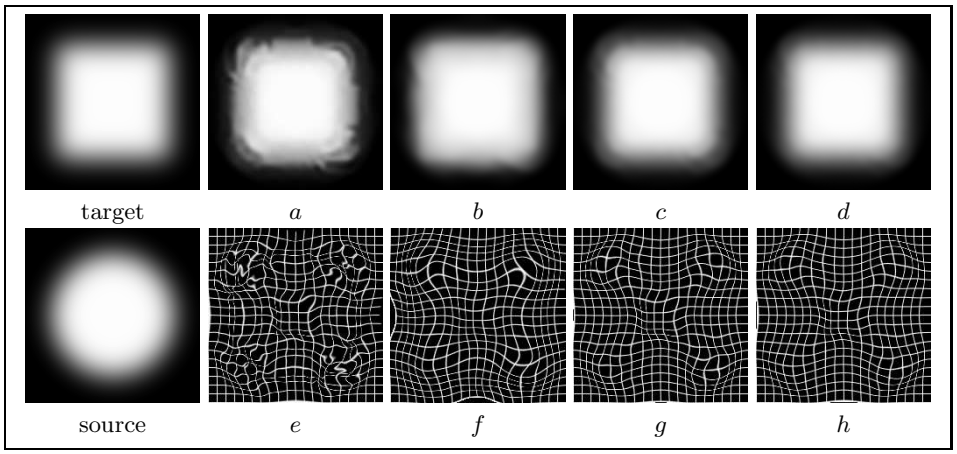

Fig. 1. Influence of the kernel size on the registration smoothness. Images a, b, c and d are the result images from the registration of the source image into the target image using the F3D with a Gaussian kernel radius of 1, 2, 3 and 4 respectively. Images e, f, g and h are the resulting deformation fields and are displayed with the same organisation.

intensity bins and iteration until convergence of the metric value. The size of the Gaussian window is defined by its width ω which include 99% of the integral of the Gaussian function. ω takes values which are integer multiples m of δ ($\omega = 2m\delta$). In this paper we refer to m as the kernel radius. In Fig. 1 values of $m = 1, 2, 3, 4$ were used.

2.3 Normalisation of the NMI gradient

Usually with the FFD algorithm the number of iterations performed for each level is manually defined, since the large computation time prohibits many iterations. In our framework, we iterate until the NMI value does not show any improvement of the registration. As described previously, our framework is driven by the forces derived from the local gradient. These values are numerically low (on the order of 10^{-6}), and lead to an excessive number of iterations before convergence. Therefore we nomalise the gradient values using a user-defined step size (half the δ in our implementation). If the NMI value descreases when moving the control point we consider the displacement to be too large and the step size is divided by 2. The gradient field is then normalised using the new step size value. This approach is used until the displacement change falls below a set threshold of 10^{-2} mm. This simple technique improves our convergence speed while generating displacement of reasonnable order for human brain MRIs resolution (in the order of 1 mm).

2.4 Implementation on parallel hardware

The F3D implementation was achieved using CUDA [13] which is an Application Programming Interface developed by NVidia to simplify interface between CPU

(host) and GPU (device). Our framework comprises four steps, organised as in Fig. 2.

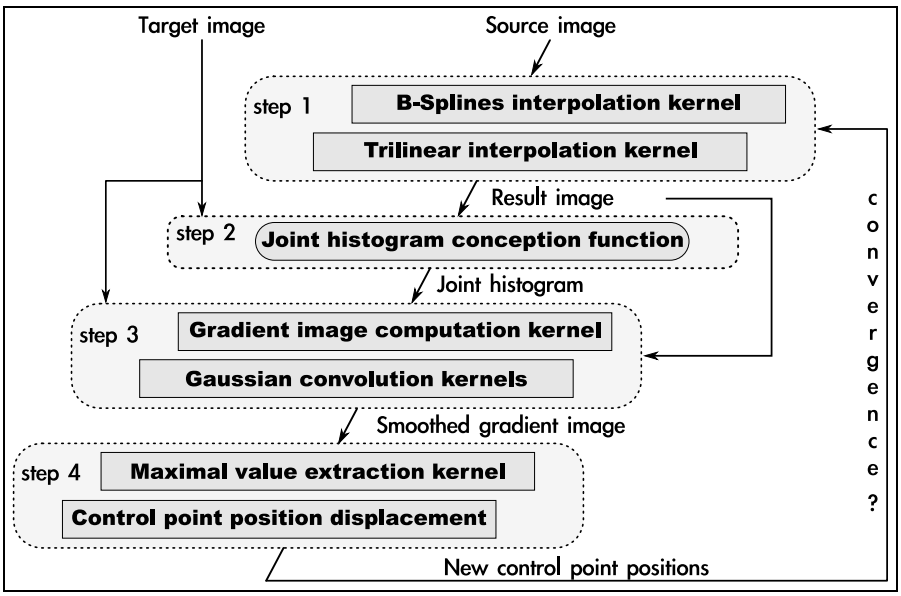

Fig. 2. Organization of our implementation

The first step performs result image interpolation via cubic B-Splines and trilinear interpolation to define the new voxel position and intensity. As already stated the computation of each voxel displacement and intensity interpolation is independent and their parallel hardware implementation is therefore straightforward. However the calculations are demanding in terms of dynamic memory resources, requiring allocation of around 22 registers per computational thread. As GPU memory is limited, a higher register requirement per thread dictates that fewer threads may be executed concurrently, resulting in sub-optimal use of the device's computational resources. The ratio of active threads to maximum allowed (hardware dependent) is referred to as occupancy [13], and an efficient implementation should maximise this. A single kernel requiring 22 registers leads to an occupancy of 42%. For this reason this step has been split into two kernels, the first dealing with the B-Splines interpolation only and the second with trilinear interpolation. Register requirements then fall to 16 and 12 respectively, and occupancies increase to 67% and 83%. Such a technique allows a computation time improvement of 36.8% in our case.

The second step involves filling the whole joint histogram and computing the different entropy values. A GPU implementation of this step did not show significant computation time improvement compare with a classic implementation. Furthermore this step occupies only around 2.2% of the entire computation time.

Moreover a GPU implementation necessitates use of single precision which for this step proves detrimental to accuracy. For these reasons this step is executed on CPU rather than on GPU. This choice does not affect the computation time even with the data transfer between device and host.

In the third step the gradient value is computed for each voxel and then smoothed. As for the first step, we distributed the computation across several kernels. The first kernel computes the gradient value using the joint histogram, the target image and the result image. Then the gradient is smoothed using three different kernels where each of them deals with one axis. For these kernels it appears that computing the Gaussian function values on the fly is faster than precomputing and fetching them from memory.

The last step normalises and updates the control point positions. A first kernel is used to extract the maximal gradient value from the whole field. The field is split into several parts, from each the maximal value is extracted. Subsequently, the largest value from the extracted maximals is kept. A last kernel is then used to update the control point position based on the normalised gradient value.

Table 1 presents the computation profile for each kernel.

Step	GPU kernel	Registers	Occupancy(%)	Computational load(%)
1	B-Splines Interpolation	16	67	25.5
	Trilinear Interpolation	12	83	10.8
2	*none*			2.2
3	Gradient computation	15	67	33.0
	Smoothing, X-axis	16	67	6.7
	Smoothing, Y-axis	15	67	7.0
	Smoothing, Z-axis	16	67	9.0
4	Maximal value	5	100	5.8
	Point displacement	21	50	< 0.01

Table 1. Computation profile for each kernel. Values were obtained with $181 \times 217 \times 181$ voxels images and a $37 \times 44 \times 37$ grid.

The GPU we use is an NVidia 8800GTX, which includes 128 processors and 768 MB of memory. The memory size is a limitation as it prohibits loading very large image sets with a small δ size. However we manage to run tests on 256^3 voxels images with a $\delta = 2.5$ voxels along each axis. These specifications are largely acceptable, for MRI brain images for example.

3 Results

3.1 Application to brain atrophy estimation

Boyes *et al.* presented a study [3] in which the Jacobian Integration (JI) using a FFD algorithm was compared with a widely-used brain atrophy measure, the

Boundary Shift Integral [1] (BSI), for evaluating brain atrophy. In this paper we perform similar tests in order to compare our results with the classical FFD implementation. The JI consists in a first time in the affine then non-rigid registration of a repeat image in the space of its baseline. In a second time the determinant of the jacobian map is computed from the deformation field or directly from the B-Splines grid. Then the determinant values included into the baseline brain mask are averaged to obtain the JI value. JI < 1 indicates atrophy whereas JI > 1 indicates expansion.

The comparison has been done using a cohort of 38 clinically Alzheimer's disease (AD) subjects and 22 aged matched healthy controls. Each subject had two baselines, scans on the same day and a repeat scan approximately one year later (mean(SD) 364(14) days). The data acquisition was performed on a 1.5 T Signa Unit (GE Medical Systems, Milwaukee) with a inversion recovery (IR)-prepared spoiled GRASS sequence: TE 6.4 ms, TI 650 ms, TR 3000 ms, bandwidth 16 kHz, $256 \times 256 \times 128$ matrix with a field of view of $240 \times 240 \times 186$ mm.

Three different tests were performed. For each the non-rigid registration parameters were: 2.5 mm δ along each axis at the final stage, 64 histogram intensity bins, 5δ radius for the Gaussian kernel radius and no smoothing of the input images. The first test performed involved the calculation of the JI between the two baselines from the same patient. As the two scans were performed the same day it is assumed that the brain volume is the same in each. Consequently we expect a JI value of 1. The mean errors are presented in Table 2(a) where the results from BSI and JI-FFD are directly extracted from [3]. The second test was similar except that one baseline was shrunk by decreasing the voxel size by a factor of 0.01 in each direction. This operation simulates an atrophy by a factor of 0.9703 (0.99^3). The error is calculated from $e = |\text{JI} - 0.9703|$. In this case only a rigid registration is performed before the non-rigid instead of an affine registration. See table 2(b) for results. The third experiment involves calculation

	Same day scans (a)	One baseline scaled (b)	Mean atrophy (c)	
	mean %error [SD]	mean %error [SD]	mean %atrophy [SD]	
			Controls	AD subjects
BSI	0.28 [0.38]	0.78 [0.48]	0.68 [0.57]	2.09 [1.14]
JI FFD	0.25 [0.27]	0.23 [0.28]	0.43 [0.52]	2.05 [0.98]
JI-F3D	0.29 [0.23]	0.26 [0.20]	0.39 [0.45]	2.15 [0.93]

Table 2. Average errors and mean atrophy rates for the BSI and the JI using FFD and F3D

of the JI between the repeat image and the two corresponding baselines. The obtained values were normalised by the number of days between the repeat and the baselines scans. The mean atrophy rates and standard deviation are presented in table 2(c). The Fig. 3 shows the result for each baseline-repeat pair for the AD patients and the controls. Then the JI measures of atrophy are used to

separate the AD subjects and the controls. The two series of baseline were analysed separately using linear regression models relating the particular variable to probable AD-control status with Huber/White/sandwich estimate of variance (robust standard errors) [14, 15] to assess subject group discrimination. Multiple linear regression was used to determine whether the FFD rates of atrophy were independently associated with probable AD-control status while adjusting for BSI rates.

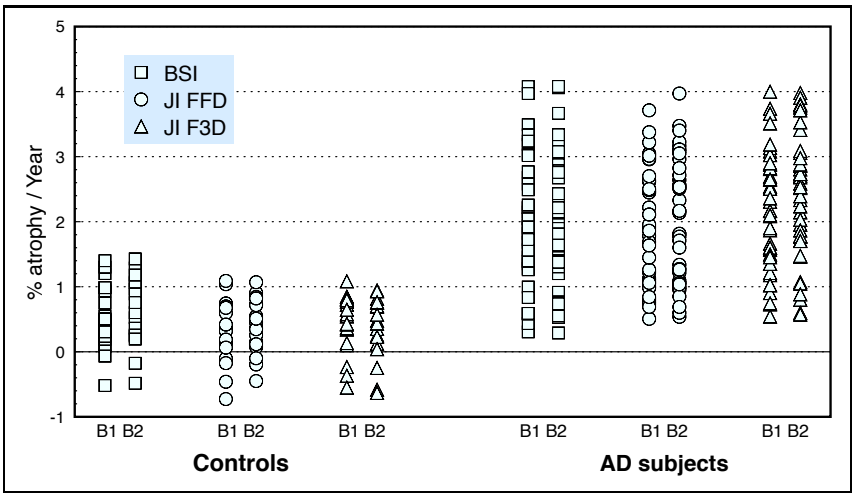

Fig. 3. Control and AD rates of atrophy for both set of baseline (B1,B2) using BSI and JI with FFD and F3D.

Using atrophy rates derived from each series at baseline to repeat, all three methods were predictors of AD subjects from controls ($p < 0.001$). Mutually-adjusted linear regression models showed that after adjusting for FFD, BSI atrophy rates were not independently associated with AD-controls status ($p > 0.05$). However, adjusting for BSI, FFD was independently associated with AD-controls status ($p < 0.001$) inferring that FFD is a superior group discriminator. The equivalent comparisons of BSI and F3D showed that BSI atrophy rates were independently associated with AD-control status ($p < 0.02$). When assessing F3D adjusting for BSI, the F3D was independently associated for both baseline series ($p < 0.001$). This suggests that F3D and BSI both independently add to the discrimination of the two groups.

3.2 Computation time benefit

In their study Boyes *et al.* performed the registration on full brain image. As a consequence we followed the same protocol to avoid any bias. Based on recent discussion with Daniel Rueckert the computation time for such analysis and

image sizes is between 3 and 5 hours on standard high-end PC. When the source image is skull stripped the computation time falls to approximately 30 minutes.

The mean computation time of the F3D is 2.35 minutes to generate the result presented here. However when performing registration with the source images skull stripped the computation time of the F3D falls to 0.64 minute.

4 Discussion and Conclusion

The first two test based both on ground true show that the JI-F3D performs as well as the JI-FFD. Indeed they both do not show significant differences with the ground true, or between results. The third test demonstrates that JI-F3D, as JI-FFD, provides a strong discrimination between the normal controls and the AD subjects. However, JI-FDD and JI-F3D do not produce the exact same output. This difference might come from the stopping criteria; a user-defined number of step for FFD, compared to a convergence of the metric value for F3D.

We presented a modified FFD framework suitable for a multi-thread implementation. This framework has been implemented for PGU execution using CUDA. We validated the F3D and demonstrate that it is able to discriminate between AD subjects and controls based on their brain atrophy rate. The same discrimination has been previously obtained using BSI or the JI with the FFD. However using the F3D the time required decreases dramatically.

The F3D framework may be useful in inter-subject applications such as atlas creation which is highly time consuming. This will be the focus of our future work.

Acknowledgment

We are thankful to Karsten Ostergaard Noe for his advice about GPU implementation and to Ged Ridgway for interesting chat and advices.

References

1. Freeborough, P., Fox, N.: "The Boundary Shift Integral: An Accurate and Robust Measure of Cerebral Volume Changes from Registered Repeat MRI". IEEE Trans. Med. Imag. **16** (1997) 623–623
2. Schott, J., Price, S., Frost, C., Whitwell, J., Rossor, M., Fox, N.: "Measuring atrophy in Alzheimer disease: A serial MRI study over 6 and 12 months". Neurology **165** (2005) 119–124
3. Boyes, R., Rueckert, D., Aljabar, P., Whitwell, J., Schott, J., Hill, D., Fox, N.: "Cerebral atrophy measurements using Jacobian integration: Comparison with the boundary shift integral". Neuroimage **32** (2006) 159–169
4. Christensen, G., Rabbitt, R., Miller, M.: "Deformable Templates Using Large Deformation Kinematics". IEEE Trans. on Imag. Proc. **5** (1996) 1435–1447
5. Rueckert, D., Sonoda, L., Hayes, C., Hill, D., Leach, M., Hawkes, D.: "Nonrigid Registration Using Free-Form Deformations: Application to Breast MR Images". IEEE Trans. Med. Imag. **18** (1999) 712–721

6. Camara, O., Schnabel, J., Ridgway, G., Crum, W., Douiri, A., Scahill, R., Hill, D., Fox, N.: "Accuracy assessment of global and local atrophy measurement techniques with realistic simulated longitudinal Alzheimers disease images". Neuroimage (in Press)
7. Rohlfing, T., Maurer, C.: "Nonrigid Image Registration in Shared-Memory Multiprocessor Environments With Application to Brains, Breasts, and Bees". IEEE Trans. on Information Technology in Biomedecine **7** (2003) 16–25
8. Jiang, J., Luk, W., D.Rueckert: "FPGA-based computation of free-form deformations in medical image registration". In: Proc. IEEE Intl Conf. Field-Programmable Technology. (2003) 234– 241
9. Taylor, Z., Cheng, M., Ourselin, S.: "High-speed nonlinear finite element analysis for surgical simulation using graphics processing units". IEEE Trans. Med. Imag. **27** (2008) 650–663
10. Maes, F., Collignon, A., Vandermeulen, D., Marechal, G., Suetens, R.: "Multimodality image registration by maximization of mutual information". IEEE Trans. Med. Imag. **16** (1997) 187–198
11. Studholme, C., Hill, D., Hawkes, D.: "An overlap invariant entropy measure of 3D medical image alignement". Pattern Recognit. **32** (1997) 71–86
12. Crum, D., Hill, D., Hawkes, D.: "Information theoretic similarity measures in non-rigid registration". In: Inf. Process. Med. Imaging. (2003) 378–387
13. NVIDIA Corporation: "NVIDIA CUDA Programming Guide Version 1.1". (2007)
14. Huber, P.: "The behavior of maximum likelihood estimates under nonstandard conditions". In: Proceeding of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume 1. (1967) 221–223
15. White, H.: "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity". Econometrica **48** (1980) 817–830