Review of GPU-based Fast-FFD implementation

Mathilde Bateson, Élie Michel

December 7, 2016

This document is a review of [MRT+10] written as part of the validation process of the MVA lecture "Introduction to Medical Image Analysis" [DP16].

1 Introduction

The process of deforming a floating image to match a reference image is at the heart of many applications of medical imaging. Comparing two images requires to match what is common between them without losing what is different. Thus, this problem remains one of the main research topics and challenges.

Among the transformation types proposed by the literature to address this issue, Free Form Deformation (FFD) as introduced in [RSH+99] have been efficiently used in real applications. But a major issue of FFD-based registration methods is their computational cost. Hence, the studied paper [MRT+10] explores two different ways of addressing this problem. The first one is through hardware optimization and the second one is algorithmic.

Section 2 first gives an overview of FFD. Then Section 3 details the main contributions of the paper, namely how it improves the FFD. Section 4 reviews the evaluation process and the results of the method, in terms of both speed and accuracy. Section 5 studies the context in which these contributions have been published and how it evolved since then.

2 General FFD computation method

FFD has been developed to bring together the advantages of two classes of pre-existing methods for registration. First, methods based on voxel-based similarity measures such as (normalized) mutual information [MCV+97], which take into account image intensity and contrast changes, but are limited to either rigid or affine transformations. And second, the most commonly used nonrigid registration algorithms based on elastic [Thi98] or fluid deformations [BNG96], which usually assume a constant tissue intensity between images.

The main idea of FFD is to deform an underlying mesh of control points, which controls the shape of the 3D object. To capture local motion, the geometric transformation in the FFD is based on cubic B-splines, which are locally controlled spline functions. Thus the resulting transformation is smooth and continuous.

The resolution of the control point mesh defines the number of degrees of freedom and, consequently, the computational complexity, which is the bottleneck of this algorithm and the focus for improvement of this work.

3 Contribution

3.1 Hardware-based improvements

Leveraging on custom hardware is a natural way of trying to speed up a slow algorithm when one really needs it to be faster, whatever the mean. Thus, there have been attempts to improve the speed of FFD by implementing it on FPGAs, or using the power of some specific supercomputer.

But the studied paper adds to the goals the necessity of developing a method that does not require expensive specific tooling, and hence focuses on the use of Graphics Processing Units (GPU).

Although a bit more specific than general purpose processors, GPUs are present in many computers, including personal computers, and have been increasingly used for research-oriented computing tasks as reviewed in [OLG+07], growing beyond their original role of video-game centric graphics rendering. This new use, often called GPGPU, was made possible by the programmability of GPUs and has been notably fostered by initiatives of Nvidia and the CUDA framework that have been designed with tasks such as physics simulation in mind.

The implementation presented in the studied paper makes use of CUDA. Another advantage of this technology is that it is widely used, so widely understood and its hence makes the code easier to reuse, improve and maintain for others. It is particularly interesting in the case of this publication because its publication was accompanied by public code release and this code has then been effectively reused.

3.2 Algorithmic improvements

Beside implementing the method on a GPU, [MRT+10] proposes a revisited version of the FFD estimation algorithm that also aims at speeding it up. A summary of the changes introduced in the algorithm is available in Table 1.

3.2.1 Porting to GPU

The algorithmic changes are mainly motivated by the difference of specificities of the hardware in use. Indeed, what can make GPU-based computing efficient is its ability to take benefit of data-parallelized operations, i.e. similar computing operations applied to many different data.

So, as explained in [Har05], porting an algorithm from CPU to GPU does not only consists in rewriting it in another language. It consists in formulating the algorithm as those common operations, called for instance *kernels* in CUDA's vocabulary and apply them to a maximum of data points at the same time.

The paper reports that porting also includes memory-related considerations, but as well as these problems have become far less concerning as they were before for CPU registries, close memory optimization for GPUs is becoming less and less a problem.

Concurrent optimization So, in order to benefit from the GPU architecture, [MRT+10] decided to compute in a parallel way the deformation. This parallelization capability released them from the constraint of the classical algorithm imposing to compute control points one by one.

Shared gradient computation Optimizing all the control points at once made itself another improvement possible. Computation could be shared among voxels when computing the gradient of the cost function at each control point. Optimizing sequentially the voxel's neighboring control points as done in the classical FFD leads to the calculation of several time the same value. The studied paper proposed to first compute the gradient value for every voxel (in parallel, when using a GPU implementation), and then compute the gradient values on all the control points.

3.2.2 Gradient descent

They also used a different variant of gradient descent making iteration steps faster to compute by avoiding to recompute the whole gradient at each step. More generally, many methods of gradient descent exist and new variants are explored every year. The paper seems to have made an efficient enough choice, but new methods could still be explored.

3.2.3 Energy function

The paper takes its formulation of the cost function to minimize almost directly from [RSH+99] used as reference for the classical FFD. The energy function has two main terms. The first one is the normalized mutual information, as explored in [MCV+97]. The second one is a regularization penalty term fostering regular enough transformations.

Main steps of FFD	Classical FFD	Proposed method
Transforming the floating image by interpolating the control points	One control point optimized per step	Concurrent optimisation and update of control points
Calculating the objective function	A trade-off between image similarity and smoothness of the transformation	Same as in classic FFD, but computing the bending-energy values at the control point positions only
Optimising this objective function and updating the control points	Simple gradient ascent	Conjugate gradient-ascent; parallel computation of voxel-centric gradient instead of the node-centric gradient (reduce redundancy)

Table 1: Summary of the diffences between classical and proposed FFD algorithms.

As we've seen during the course, other terms can be added to the energy function. For instance, a stretching energy term can be introduced. But using the same energy function was important to be able to compare fairly the results.

4 Evaluation and Results

The purpose of the proposed method is to greatly improve the computation time while yielding same accuracy levels. Thus both these criteria were used in the paper to evaluate the improvement.

4.1 Computation time

The source of the great improvement over the classical FFD algorithm is two-fold. A x100 speed improvement was observed using the CPU formulations and implementations of the classical versus proposed FFD. A x10 speed improvement was observed using the GPU versus the CPU implementations of the proposed FFD.

What is curious is that these algorithmic changes originally motivated by the specificities of GPU architecture actually benefited to the CPU version as well. And this result seems to surprise the writers of the paper themselves.

4.2 Registration accuracy

Once proved that the proposed methodology greatly improves computation time, it is necessary to check that the same accuracy level is still observed. Thus the Fast-FFD and classical FFD registrations were compared on a dataset consisting of 20 T1-weighted brain MR images, 10 of which from patients with Alzheimer's disease (AD) patients, and 10 from age-matched control subjects. For each image, manual segmentations of different regions of interest (Left/Right amygdale, L/R hippocampus, etc.) were used as ground truth, and registration was performed against all other images. Dice similarity was used to measure the overlap.

Fast-FFD performed slightly better than classical FFD in most cases. The authors attributed this to the fact that fast FFD was fast enough to iterate until convergence, whereas classical FFD was so slow that it was stopped after 10 iterations, thus before convergence.

5 Discussion

This section adopts a more critical point of view over the original paper and explores how it relates to its research field and applications.

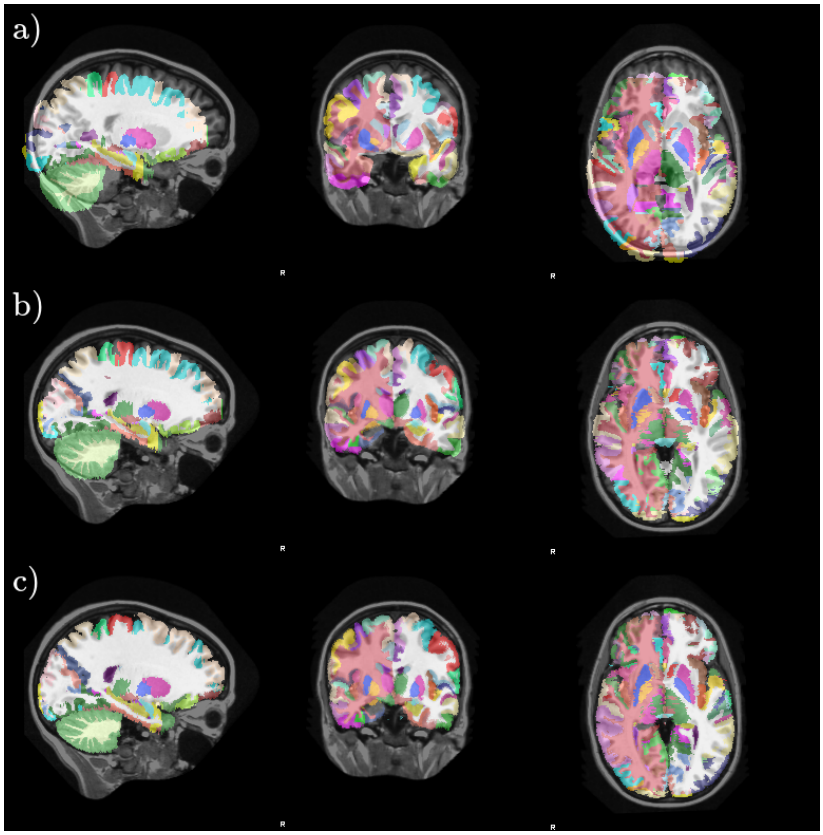

Figure 1: Result of segmentation propagation based on registration. a) Propagated labels without registration. b) Propagated labels with affine registration. c) Propagated labels with FFD registration. It is important to note that although for display purpose we only show one slice in this report, the registered data is tridimensional.

5.1 Examples and reproductibility

In order to attest from the reproductibility of these results, we looked for examples beyond the scope of the paper itself.

The first one that we show in Figure 1 is the result of segmentation propagation based on the transformation retrieval as shown in [Nif], a tutorial provided with the software suite released with the paper. It is a good illustration of the fact that beside the purely academic publication, time has been spend to effectively bring the method to application. This is especially important in our opinion for implementation papers as this one.

For our second example, we even tried to reproduce the results ourselves using the proposed implementation. The released code is clean enough to be understandable and it was easy to compile. We did not have easy access to

Figure 2: Unsuccessful attempt of registration between different expositions. The first image is the reference. The second image is the floating image. The last image is the output of the FFD as computed by the proposed implementation.

actual medical tridimensional data, thus we decided to stress-test the registration method by applying it to a non medical imaging related problem.

We tried to use it to register bracketing images, a use case of image registeration coming from camera related issues. The result shown in Figure 2 proposes a very abusive deformation of the image, while the original transformation was almost affine.

But this shows two things. First, it illustrates well the kind of deformation that FFD can model. Second, it proved to us that the implementation was usable in practice.

5.2 Other registration methods

FFD is not the only state-of-the-art method used for image registration. Listing all the alternative would be the point of a full survey, like for example [SDP13]. And even more recently, the wave of neural networks has reached this field and Convolutional Neural Networks (CNNs) have been applied to the problem of estimating transformation parameters [MWL16] in order to reach real-time registration, although it is still limited to 2D/3D registration. A recent comparison of such implementations [XLH+16] shows that, the method presented here is still competitive with some of the major actors, though it is consistently beaten by [HJBS13].

Since the publication of this paper, frameworks for transparent us of GPU for scientific computing have became more and more used and powerful, especially for their automatic differentiation features [BPRS15]. These frameworks could be advantageously used to implement Modat's Fast-FFD algorithm with at least the same computational power while benefiting from more flexibility in testing new variants of gradient descent or new hardware setups.

6 Conclusion

The reviewed paper proposed a fast FFD technique, performing registration of T1-weighted MR images in less than 1 min with the same level of accuracy as a classical serial implementation. The publication of the code of the resulting NiftyReg software led to wide reuse and improvement by a community of researchers (60 publications). For all these reasons, this project seems to us a good example of using available technology to bring concrete improvements in the clinical setting.

References

[BNG96] Morten Bro-Nielsen and Claus Gramkow. Fast fluid registration of medical images. In *Visualization in Biomedical Computing*, pages 265–276. Springer, 1996.

[BPRS15] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

[DP16] Hervé Delingette and Xavier Pennec. Master mva 2016-2017 first semester : Introduction to medical image analysis. `http://www-sop.inria.fr/asclepios/cours/MVA/Module1/index.htm`, 2016. Accessed: 2016-12-05.

[Har05] Mark Harris. Mapping computational concepts to gpus. In *ACM SIGGRAPH 2005 Courses*, page 50. ACM, 2005.

[HJBS13] Mattias P Heinrich, Mark Jenkinson, Michael Brady, and Julia A Schnabel. Mrf-based deformable registration and ventilation estimation of lung ct. *IEEE transactions on medical imaging*, 32(7):1239–1248, 2013.

[MCV+97] Frederik Maes, Andre Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging*, 16(2):187–198, 1997.

[MRT+10] Marc Modat, Gerard R Ridgway, Zeike A Taylor, Manja Lehmann, Josephine Barnes, David J Hawkes, Nick C Fox, and Sébastien Ourselin. Fast free-form deformation using graphics processing units. *Computer methods and programs in biomedicine*, 98(3):278–284, 2010.

[MWL16] Shun Miao, Z Jane Wang, and Rui Liao. A cnn regression approach for real-time 2d/3d registration. *IEEE transactions on medical imaging*, 35(5):1352–1363, 2016.

[Nif] Niftyreg segmentation propagation tutorial. `http://cmictig.cs.ucl.ac.uk/wiki/index.php/NiftyReg_Segmentation_Propagation_Tutorial`. Accessed: 2016-12-05.

[OLG+07] John D Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E Lefohn, and Timothy J Purcell. A survey of general-purpose computation on graphics hardware. In *Computer graphics forum*, volume 26, pages 80–113. Wiley Online Library, 2007.

[RSH+99] Daniel Rueckert, Luke I Sonoda, Carmel Hayes, Derek LG Hill, Martin O Leach, and David J Hawkes. Nonrigid registration using free-form deformations: application to breast mr images. *IEEE transactions on medical imaging*, 18(8):712–721, 1999.

[SDP13] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. Deformable medical image registration: A survey. *IEEE transactions on medical imaging*, 32(7):1153–1190, 2013.

[Thi98] J-P Thirion. Image matching as a diffusion process: an analogy with maxwell's demons. *Medical image analysis*, 2(3):243–260, 1998.

[XLH+16] Z Xu, CP Lee, MP Heinrich, M Modat, D Rueckert, S Ourselin, RG Abramson, and BA Landman. Evaluation of six registration methods for the human abdomen on clinically acquired ct. *IEEE transactions on bio-medical engineering*, 2016.