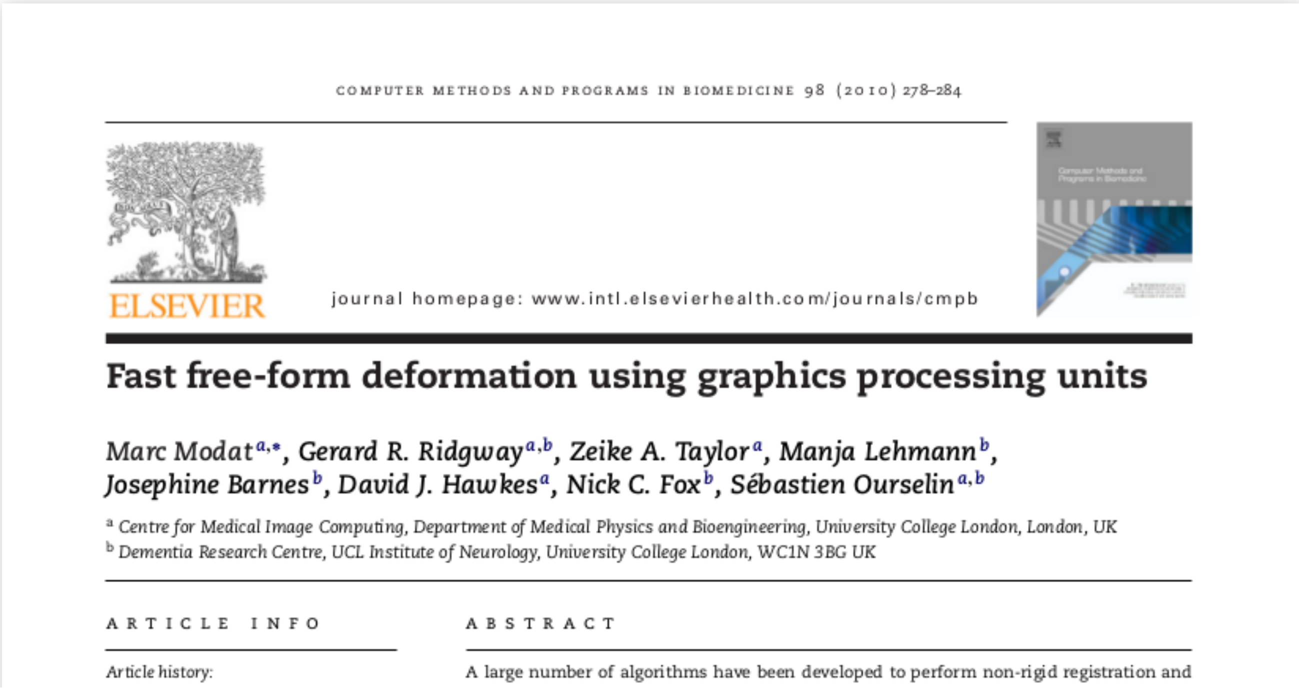
Review of GPU-based Fast-FFD implementation

Mathilde Bateson, Élie Michel

December 7, 2016


COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE 98 (2010) 278–284

journal homepage: www.intl.elsevierhealth.com/journals/cmpb

Fast free-form deformation using graphics processing units

Marc Modat[a,*], Gerard R. Ridgway[a,b], Zeike A. Taylor[a], Manja Lehmann[b], Josephine Barnes[b], David J. Hawkes[a], Nick C. Fox[b], Sébastien Ourselin[a,b]

[a] Centre for Medical Image Computing, Department of Medical Physics and Bioengineering, University College London, London, UK
[b] Dementia Research Centre, UCL Institute of Neurology, University College London, WC1N 3BG UK

ARTICLE INFO

Article history:

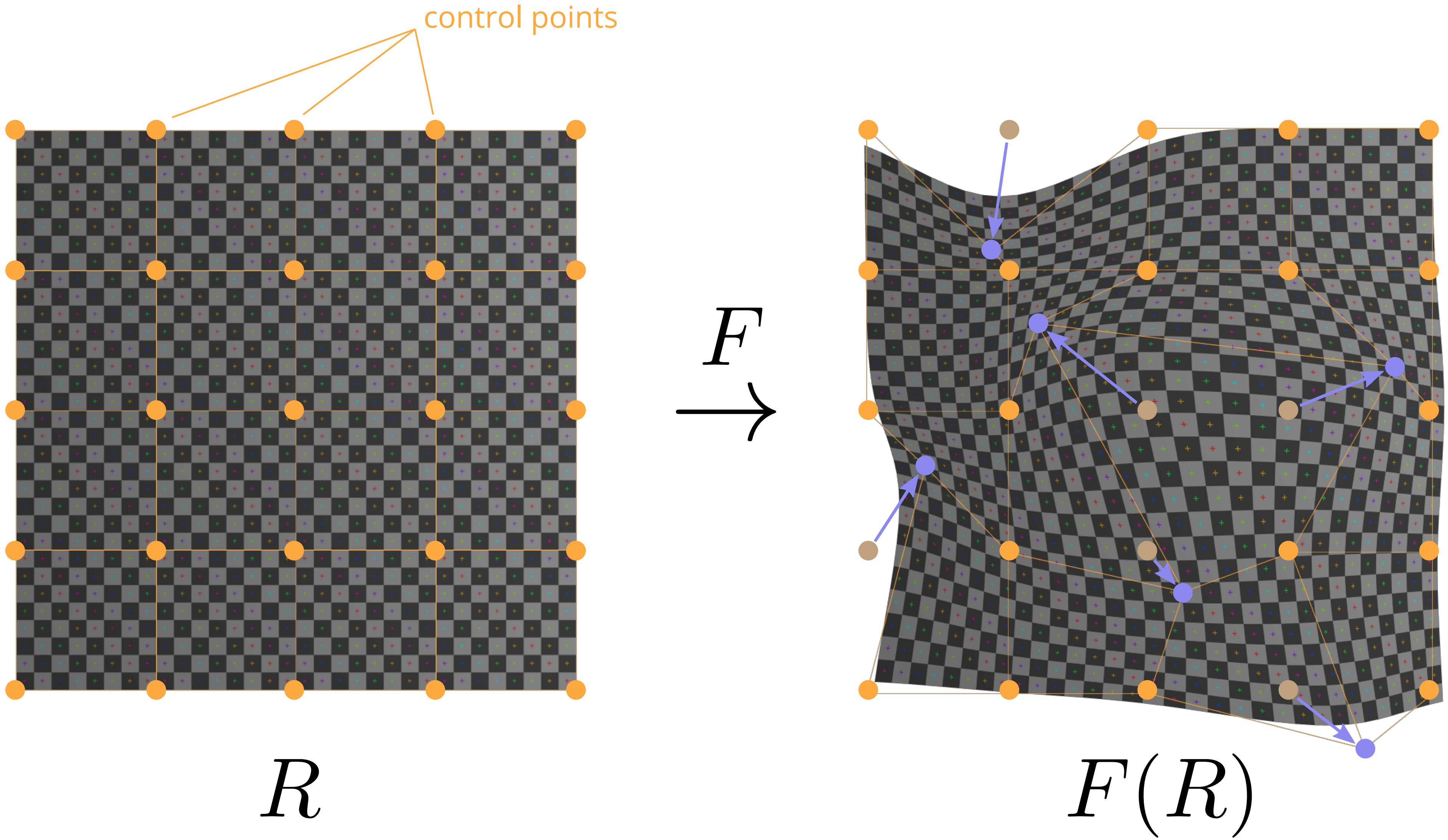
ABSTRACT

A large number of algorithms have been developed to perform non-rigid registration and

1. Free-Form Deformation (FFD)

[Rueckert99]

1.1. FFD principle



1. Free-Form Deformation (FFD)

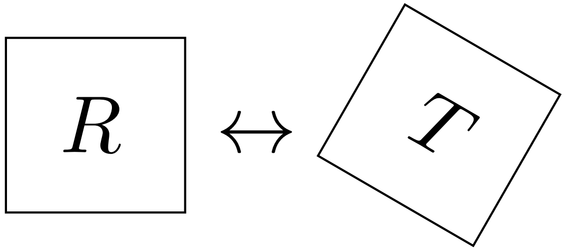
1.2. FFD computation

[Rueckert99]

Gradient Ascent

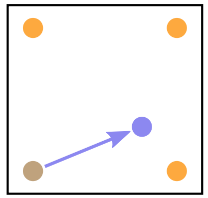
with cost

$$C = (1 - \alpha) \times \text{NMI} + \alpha \times R$$



Normalized Mutual Information [Maes97]

$$\text{NMI} = \frac{H(R) + H(F(T))}{H(R, F(T))}$$



Second order regularization

$$R = -\frac{1}{N} \sum_{u\text{voxel}} \left(\frac{\partial^2 T(u)}{\partial x^2}\right)^2 + \left(\frac{\partial^2 T(u)}{\partial y^2}\right)^2 + \left(\frac{\partial^2 T(u)}{\partial z^2}\right)^2 + 2\left[\left(\frac{\partial^2 T(u)}{\partial xy}\right)^2 + \left(\frac{\partial^2 T(u)}{\partial yz}\right)^2 + \left(\frac{\partial^2 T(u)}{\partial xz}\right)^2\right]$$

Gradient Ascent
takes

5 hours

for
181x127x181 voxels
40x44x40 control points

this makes it
Hard to apply

$\Rightarrow$ [Modat10] addresses this issue.

2. Fast-FFD

2.1. Improvement directions

[Modat10]

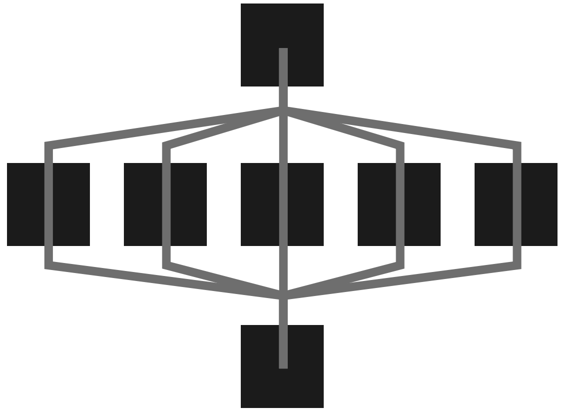
Hardware

Graphics Porcessing Units (GPU)

$\Rightarrow$

Software

Parallelization of the algorithm



2. Fast-FFD

2.2. Hardware — Why GPU?

[Modat10]

~~FPGA~~
~~HPC~~

More expensive, harder to get

GPU

- Present in almost any Personal Computer

  $\Rightarrow$ *leveraging on existing hardware*

- Widely used for Scientific Computing

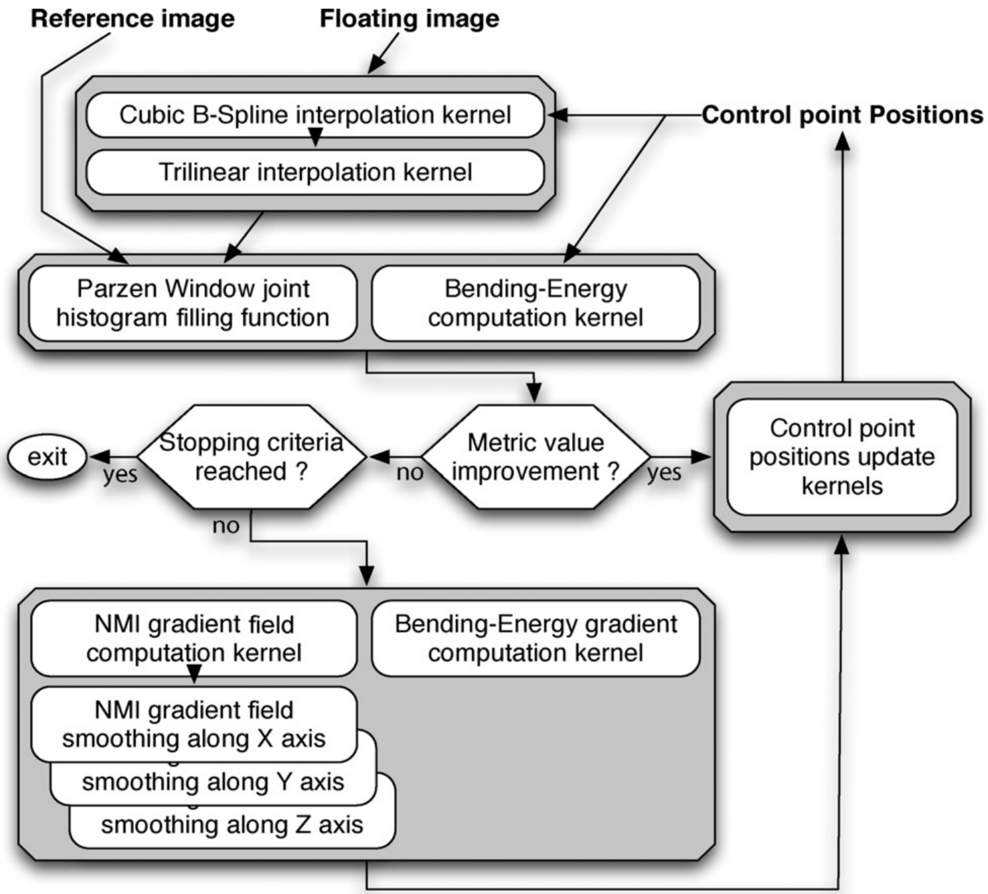
  $\Rightarrow$ *leveraging on existing tools and community*

[Harris05] [Owens07] [Baydin15]

2. Fast-FFD

2.3. Algorithmic changes

[Modat10]



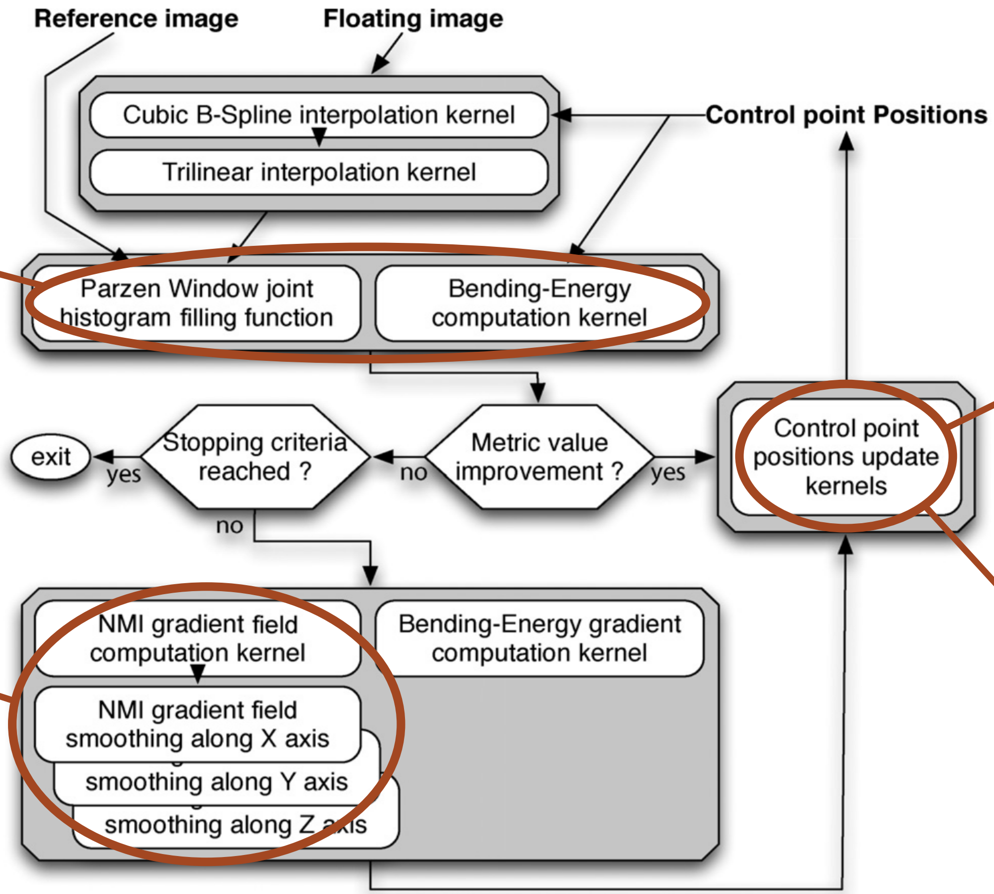
2. Fast-FFD

2.3. Algorithmic changes

[Modat10]

parallel computing

shared gradient computing



concurrent optimization

conjugate gradient ascent

[Modat10]

Classical FFD	Fast-FFD on CPU	Fast-FFD on GPU
5 h	3 min 18 s	$<$20 s

3. Results

3.1. Computation Time

[Modat10]

Classical FFD	Fast-FFD on CPU	Fast-FFD on GPU
5 h	3 min 18 s	<20 s

unexpectedly important improvement

3. Results

3.2. Accuracy

[Modat10]

Table 2 – Average (standard deviation) results of the segmentation propagation. For each propagation, the Dice similarity value between the manual and the propagated segmentations has been computed.

Mask area	Affine only	*Classical* FFD	Fast-FFD
Left amygdala	0.531 (0.163)	0.759 (0.089)	0.776 (0.066)
Left entorhinal cortex	0.203 (0.189)	0.296 (0.164)	0.372(0.155)
Left fusiform gyrus	0.398 (0.103)	0.483 (0.096)	0.499(0.098)
Left hippocampus	0.429 (0.157)	0.658 (0.093)	0.686(0.075)
Left medial-inferior temporal gyrus	0.626 (0.070)	0.699 (0.061)	0.709(0.064)
Left parahippocampal gyrus	0.399 (0.146)	0.527 (0.094)	0.637(0.070)
Left superior temporal gyrus	0.607 (0.069)	0.742 (0.057)	0.737(0.048)
Left temporal lobe	0.748 (0.052)	0.832 (0.046)	0.827(0.041)
Right amygdala	0.571 (0.139)	0.779 (0.072)	0.787 (0.058)
Right entorhinal cortex	0.170 (0.177)	0.266 (0.169)	0.334 (0.162)
Right fusiform gyrus	0.450 (0.111)	0.542 (0.119)	0.534 (0.113)
Right hippocampus	0.479 (0.162)	0.631 (0.120)	0.710 (0.086)
Right medial-inferior temporal gyrus	0.662 (0.062)	0.763 (0.059)	0.760 (0.053)
Right parahippocampal gyrus	0.276 (0.208)	0.323 (0.189)	0.340 (0.275)
Right superior temporal gyrus	0.624 (0.055)	0.780 (0.048)	0.775 (0.040)
Right temporal lobe	0.733 (0.119)	0.811 (0.128)	0.813 (0.125)

comparable results

4. Follow ups and Reproductibility

4.1. Open Source implementation

```
$ git clone git://git.code.sf.net/p/niftyreg/git niftyreg
```

- Code release

  $\Rightarrow$ *important for an implementation paper*

  $\Rightarrow$ *consistent with the willing of accessibility*

- We tested it

  $\Rightarrow$ *easy to compile and run*

  $\Rightarrow$ *integrated with other tools*

  $\Rightarrow$ *available documentation*

The publication had concrete consequences

4. Follow ups and Reproductibility

4.2. Comparison to other methods

[Xu16] Evaluation of six registration methods

$\Rightarrow$ *Results have effectively been reroduced*

$\Rightarrow$ *Studied method still competitive*

TABLE I
METRICS ON 400 REGISTRATIONS FOR ALL TESTED METHODS (MEAN ± STD)

Method	DSC	MSD (mm)	HD (mm)	Time (min)
FSL	0.12 ± 0.19	37.92 ± 44.11	84.28 ± 59.96	951.73 ± 201.20
ANTS-CC	0.18 ± 0.21	27.15 ± 32.65	62.92 ± 44.60	411.60 ± 74.20
ANTS-QUICK-MI	0.27 ± 0.25	15.96 ± 19.22	49.66 ± 32.96	50.18 ± 21.93
IRTK	0.28 ± 0.26	19.07 ± 26.50	55.58 ± 39.26	220.27 ± 91.79
NIFTYREG	0.35 ± 0.29	15.72 ± 19.16	59.59 ± 42.60	116.91 ± 34.94
DEEDS	0.49 ± 0.26	8.63 ± 16.16	40.15 ± 32.11	3.73 ± 0.77

Note that ANTS-CC, ANTS-QUICK-MI, and NIFTYREG used two CPU cores for each registration process. The mean DSC across four large organs (liver, spleen, kidneys) is 0.19, 0.31, 0.43, 0.48, 0.55, and 0.70 for FSL, ANTS-CC, ANTS-QUICK-MI, IRTK, NIFTYREG, and DEEDS, respectively.

only the CPU version was tested

5. Evolution of registration methods

[Xu16] Evaluation of six registration methods

→ [Heinrich13]

[Sotiras13] Survey of registration methods

Going further?

[Miao16] Application of CNNs to registration

[Baydin15] Automatic differentiation

theano TensorFlow torch

Thank you!

Questions?