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:
Received 23 February 2009
Received in revised form
28 August 2009
Accepted 3 September 2009

Keywords:
GPU
Non-rigid registration
Free-form deformation
Normalised mutual information

ABSTRACT

A large number of algorithms have been developed to perform non-rigid registration and it is a tool commonly used in medical image analysis. The free-form deformation algorithm is a well-established technique, but is extremely time consuming. In this paper we present a parallel-friendly formulation of the algorithm suitable for graphics processing unit execution. Using our approach we perform registration of T1-weighted MR images in less than 1 min and show the same level of accuracy as a *classical* serial implementation when performing segmentation propagation. This technology could be of significant utility in time-critical applications such as image-guided interventions, or in the processing of large data sets.

1. Introduction

In the field of medical image analysis, image registration remains one of the main research topics and challenges. Image registration consists of deforming a floating image to match a reference image. The most active area of research is non-rigid registration (NRR), in which attempts are made to locally "warp" one image into correspondence with another. Example problems are matching 3D MRI scans of two different patients, or two scans of the same patient before and after surgery. While a huge amount of research has been devoted to the methodological development [1,2], very little research has focused on the computational burden of the proposed algorithms. One of the most widely used NRR algorithms, free-form deformation (FFD) [3], has not reached its full clinical utility as a result; FFD's computation time on a single data set can extend to several hours. If such constraints could be removed, or alleviated a new range of clinical applications, which require real-time or near real-time computation could be attempted. Such applications arise, for instance, in the context of real-time image-guided surgery: new patient information acquired during surgery, such as ultra-sound images, could be used efficiently to update a previously developed surgical plan.

The bottleneck of the FFD algorithm is the cubic B-Spline computation, and consequently work has been done to speed up this part using various architectures. Jiang et al. [4] used a FPGA-based implementation which lead to a speed-up of 3.2 times compared with a 2.666 GHz CPU execution. Rohlfing and Maurer [5] reduced computation time by more than 50 times using 64 CPUs of a shared-memory supercomputer. More recently, Rohrer et al. [6] presented a multicore implementation of the B-Spline computation based on a Cell Broadband Engine ™ (Cell/B.E.) platform. Their architecture performed 40% faster than serial execution on a standard computer.

These techniques provide considerable computation time improvements, however they require either high technical knowledge or hardware with prices inhibiting wide adoption. We propose the use of graphics processing units (GPUs) as a

* *Corresponding author*. Tel.: +44 (0) 20 7679 0320; fax: +44 (0) 20 7679 0255.
E-mail address: m.modat@ucl.ac.uk (M. Modat).
0169-2607/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2009.09.002

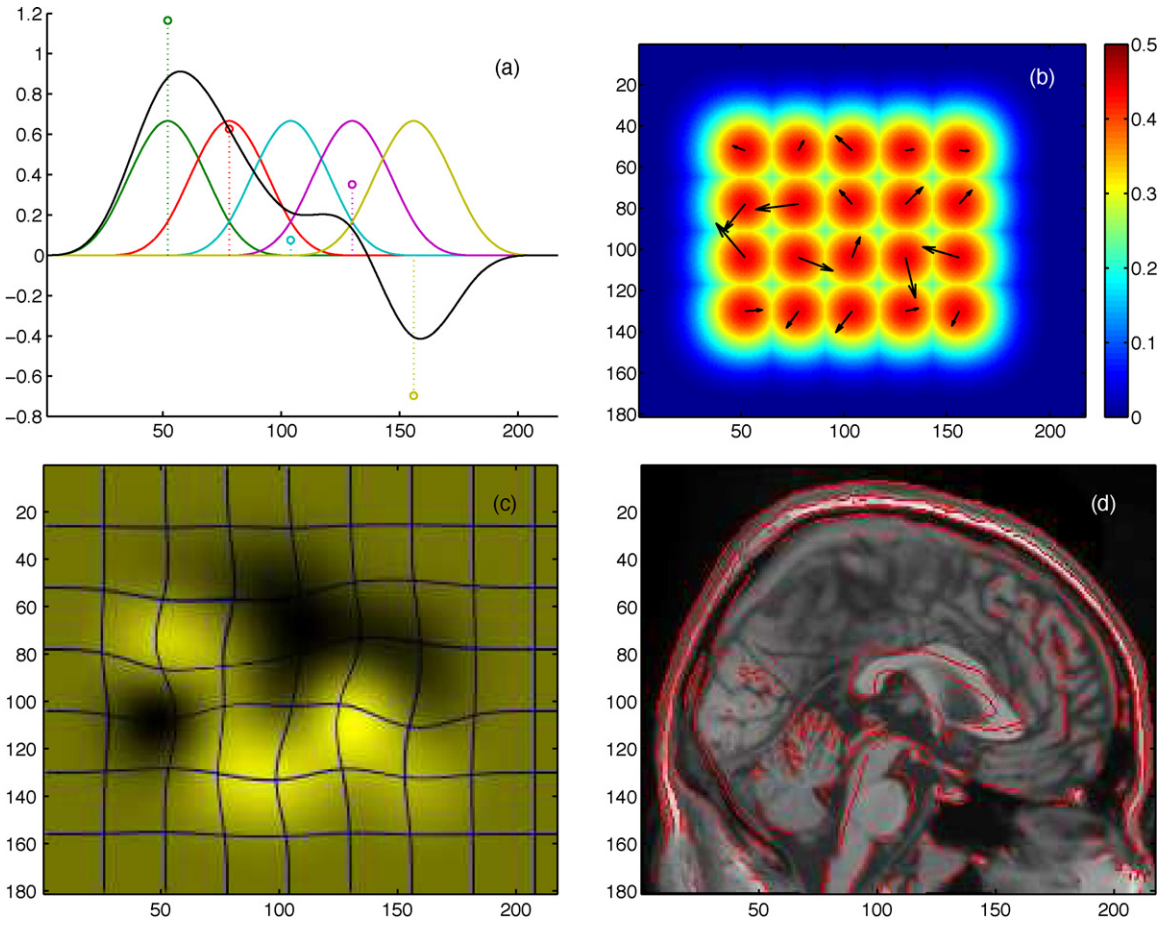

Fig. 1 – From splines to image warps. (a) A weighted sum of uniformly spaced cubic B-Spline basis functions used to construct a C2 continuous curve in one dimension. (b) The previous five basis functions are combined with another four to generate a two-dimensional tensor product; two weighted sums of these 2D basis functions are used to model the *x* and *y* components of a displacement vector field. (c) The *x* displacement field in yellow has been used to deform a regular grid, overlaid in blue. (d) The same transformation illustrated using a brain image: the red edges from the original MRI are overlaid on a grayscale image of the warped result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

cost effective high performance solution. Moreover we advocate use of NVidia Corporation's CUDA API, which requires only knowledge of the C language and very little awareness of the hardware.

In this article we present a data parallel formulation of the FFD algorithm and describe its execution on GPU architecture using the CUDA API. The formulation affords particularly efficient memory use, allowing much improved use of computational resources. The resulting system provides significant speed improvements, without resorting to theoretical or numerical approximations.

In the first section we present the methodology and its GPU-based implementation. In the second we present the computation time benefit from such an implementation, and evaluate the formulation's accuracy. The time benefit is simply assessed by comparing the computation time of a serial and a parallel implementation of the same algorithm. The accuracy is evaluated by comparing the result of segmentation propagation using our GPU implementation and the *classical* serial FFD formulation.

2. Method

2.1. *The free-form deformation algorithm*

The main requirement for an algorithm to benefit from GPU execution is data parallelism. The FFD algorithm comprises three components, which may be considered independently: transformation of the floating image using the splines and an interpolation function; evaluation of an objective function; and optimisation against this function. Individually, these components may be formulated in a data parallel manner as they mainly consist of voxel-wise computations. However, difficulties associated with GPU memory constraints mean certain aspects are not easily implemented in practice.

2.1.1. *Cubic B-Splines interpolation*

The FFD algorithm consists of locally deforming an image volume using cubic B-Splines. This technique has the desirable feature of guaranteeing a C^2 continuous deformation (see