Accelerating 3D Non-Rigid Registration using Graphics Hardware

Nicolas Courty and Pierre Hellier

University of South Brittany - IRISA/INRIA
France

Abstract

In the context of image-guided surgery, there is an increasing need for fast implementation of $3D$ image analysis processes. Among the various image analysis tasks, non-rigid image registration is particularly needed and is also computationally prohibitive. This paper presents a GPU (Graphical Processing Unit) implementation of the popular Demons algorithm [15] using a Gaussian recursive filtering [4]. Results shows acceleration factor up to 9 compared to a software implementation, which allows to use such an algorithm in a functionnal intra-operative context.

Key words: Non-rigid registration, $3D$ image processing, GPU implementation.

1 Introduction

In the last decade, it has become increasingly common to use image-guided navigating systems to assist surgical procedures [5]. The reported benefits are improved accuracy, reduced intervention time, improved quality of life, reduced morbidity, reduced intensive care and reduced hospital costs. Image-guided systems can help the surgeon plan the operation and provide accurate information about the anatomy during the intervention. Image-guided systems are also useful for minimally invasive surgery, since the intraoperative images can be used interactively as a guide. Current surgical procedures rely on complex preoperative planning, including various multimodal examinations: anatomical, vascular, functional explorations for brain surgery. Once all information have been merged, it can be used for navigation in the operating theatre (OR)

Email address: courty@univ-ubs.fr,phellier@irisa.fr (Nicolas Courty and Pierre Hellier).

URL: http://www.irisa.fr/visages (Nicolas Courty and Pierre Hellier).

using image-guided surgery systems. Image-guided surgery involves the rigid registration of the patient's body with the preoperative data. With an optical tracking system, and Light Emitting Diodes (LED), it is possible to track the patient's body, the microscope and the surgical instruments in real time.

Unfortunately, the assumption of a rigid registration between the patient's body and the preoperative images only holds at the beginning of the procedure. This is because soft tissues tend to deform during the intervention. This is a common problem in many image-guided interventions, the particular case of neurosurgical procedures can be considered as a representative case. When dealing with neurosurgery, this phenomenon is called "brain shift". The magnitude of soft tissue deformation shows striking differences at each stage of surgery. Brain shift must be considered as a spatio-temporal phenomenon, and should be estimated continuously, or at least at key moments, to update the preoperative planning. To do so, intraoperative images (like intraoperative Magnetic Resonance Images or $3D$ ultrasound images) are acquired during surgery and can be used to estimate the deformation of soft tissues. Non-rigid image registration method is then needed. The literature on non-rigid image registration is large, we thus refer the reader to comprehensive surveys on this domain [11,12,17]. Methods usually differ by the similarity measure (the modeling between data and unknowns) and the type of regularization (the regularity of the estimated deformation field). All published methods are computationally expensive. Reported computation times vary between several minutes (10 for the fastest method) and several hours and are in all cases incompatible with an application in the operating room.

This paper proposes a fast non-rigid registration method implemented on GPU and compatible with the image-guided surgery requirements. The contribution of the paper is twofold: firstly, $3D$ image processing is expressed as operations on $2D$ textures. Secondly, we propose an efficient recursive filtering scheme implemented on GPU that is shown to be 10 times more efficient than the software implemented version. The paper is organized as follow: after a short presentation on previous work on using the GPU in general purpose applications (section 2), we briefly recall some theoritical background on the used registration method (section 3), then our GPU implementation is proposed (section 4) along with some results (section 5) and a conclusion.

2 Related Work

Though first designed for computer graphics industry, graphics processing units have revealed over the last years to be high performance computing platforms at low cost. With their increased programmability, it is now possible to consider execution of non-graphic applications on such boards. Several

groups have explored these possibilities for a wide variety of computationally expensive problems (see [6] for a survey). Considering $3D$ image processing, some work has been done on visualization [10,1], segmentation [14], and filtering [9]. The use of graphics board to speed-up medical applications has also drawn attention in the domain of tomography [16] and non-linear warping of volumes with thin-plates splines [2]. To our knowledge, our work constitutes the only attempt to implement a non-linear registration on commodity PC graphics boards. Among other, our recursive filtering method constitutes an efficient approach to volume processing on GPU.

3 $3D$ non-rigid registration method

3.1 Overview

We have chosen to use the Demon's method for its proved effectiveness and computational efficiency. In particular, this method has been shown to efficiently register an atlas toward a subject [3] and to register brains of different subjects [8]. This method was proposed by Thirion [15]. At each demon's location (usually the grid of demons is dense, i.e. every voxel is a demon), force are computed so as to repulse the model toward the data. The force depends on the polarity of the point (inside or outside the model), the image difference and gradients. This amounts to a minmax problem: maximization of similarity and regularization of solution. For small displacements, it has been shown that the demon's method and optical flow are equivalent. The method alternates between the computation of forces and the regularization of the deformation field until convergence. Here is a synopsis of the algorithm:

```
DO
  1. Compute spatial and temporal gradients
  2. Compute dense grid of demons
  3. Regularize incremental deformation field
      using Gaussian filtering
  4. Update deformation field
  5. Compute deformed image using trilinear interpolation
UNTIL CONVERGENCE
```

The convergence condition can be expressed with the mean square error (MSE) between the reconstructed volume and the source volume. When the MSE decreases less than a threshold, the algorithm is stopped. In this paper, the 5 steps described above are implemented using GPU. Most of the computation time is due to the Gaussian filtering of the deformation field (step 3). For an optimal implementation, we have chosen the recursive implementation of the

Gaussian filtering proposed in [4].

3.2 Recursive filtering

The recursive Gaussian filtering makes it possible to compute infinite impulse filters with a bounded complexity. Deriche [4] proposes to approximate the Gaussian filter with 4^{th} order cosines-exponential functions. It is shown that the approximation is good for Gaussian filters with a standard deviation lower than 10. Separability is one of the attractive feature of the Gaussian filtering. Therefore, the three components of the deformation field will be processed successively. For a $1D$ signal x, the causal and anti-causal parts of the filtered signal y are expressed as:

$$y(i) = \sum_{k=0}^{k=3} b_k x(i-k) - \sum_{k=1}^{k=4} a_k y(i-k).$$

Numeric coefficients a_k and b_k are given in [4] for the Gaussian filter and its derivatives. The main advantage of recursive filtering is that the number of operations is bounded and independent of the standard deviation of the Gaussian filter. The latter is particularly appealing when filtering $3D$ images since a classical implementation is computationally expensive for large standard deviation. Let us finally note that this method allows to minimize the number of texture fetching within the fragment program responsible for the filtering (which is one of the most time-consuming operation on the GPU).

4 Implementation

In this section, we present an original and efficient implementation of Thirion Demon's algorithm [15]. As presented in section 3.1, one iteration of convergence loop can be divided into 5 main operations. In our implementation, we factorized the two first steps (computation of temporal and spatial gradients and computation of demons in each voxel) in a unique fragment program, followed by the regularization of the field (Gaussian filtering of the 3 field components) and finally the reconstruction of a final volume thanks to a trilinear interpolation of the current volume. The implementation of the trilinear interpolation and the demons computation on GPU are straightforward and will not be described in this section. We mainly concentrate on the representation of the volume (4.1) and the recursive filtering scheme (4.2).

4.1 Mapping the 3D volume on a 2D texture

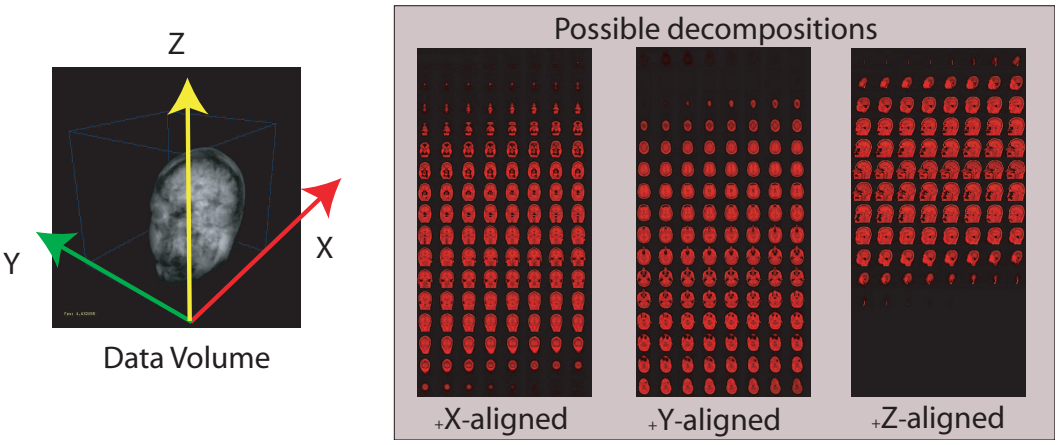

Figure 1. Possible decompositions of a $3D$ volume along three orthogonal axis. For each axis, two decompositions are possible if axial symetry is considered.

In our implementation, the data volume is not represented as a $3D$ texture, but instead as a big texture containing all the slices from the volume. This technique can be reffered to as flat $3D$ texture, as first introduced by Harris [7]. Figure 1 is an illustration of the different possible decompositions along the three axis. There are two major advantages to such a decomposition: first of all, it is possible to process the entire volume in a single rendering pass, and a render-to-$3D$-texture extension is no more needed. Secondly, as far as $3D$ float texture are not yet supported on the current generation of graphics board, it would have not been possible to handle the velocity field volume for instance.

Nevertheless, this transformation is not straightforward. Two major problems arise: texture size limitation (4096×4096 for the current generation) and unicity of such a decomposition. Let us investigate the possible decomposition for a volume containing $D_x \times D_y \times D_z$ voxels, where D_k is the volume dimension along axis k. This $3D$ data should be mapped onto a $2D$ texture of size $N_i \times N_j$, under the constraint that $N_i \leq 2^{12}$ and $N_j \leq 2^{12}$. Such a mapping might not be doable, let us find the conditions where it can be performed:

Let us first assume that the image plane are square dyadic images, i.e:

$$\exists p \in \mathbb{N} \quad \text{such that} \quad D_x = D_y = 2^p.$$

This assumption is not very restrictive when considering medical images, since this is very standard with actual scanners.

A solution to this problem can be seen as finding $n \in [1, p-1]$ such that:

$$N_j = D_z \times 2^{p-n} \; and \quad N_i = 2^{p+n}$$

under the constraint $N_i \leq 2^{12}$ *and* $N_j \leq 2^{12}$.

D_z can be bounded as: $\exists k \in \mathbb{N}$ such that $2^{k-1} < D_z \leq 2^k$, what leads to:

$$p - n + k \leq 12 \text{ and } \quad p + n \leq 12$$

Therefore, adding and subtracting the two equations gives:

$$k \leq 24 - 2p \text{ and } \quad n \leq \frac{k}{2}$$

Since $n \in [1, p-1]$, the mapping can be performed if $D_z \leq 2^{24-2p}$. For instance, if $p = 8$ (i.e., $D_x = D_y = 256$), this amount to $D_z \leq 256$. Practically, this mapping is therefore doable.

From there, it is possible to access the whole volume in fragment programs by using a correct look-up function. Let us note $Zx = p - n$ and $Zy = p + n$ computed from the previous equation. The look-up functions are given by:

```
float2 fromVolumeToUV(float3 coord){
    float X = fmod (coord.y,Zx);
    float Y = (coord.y - X)/Zx;
    return float2(X*Dx+coord.x+0.5,
                  Y*Dy+coord.z+0.5);}
```

```
float3 fromUVtoVolume(float2 uv){
    float x = fmod (uv.x,Dx);
    float z = fmod (uv.y,Dy);
    return float3(x, (uv.x-x)/Dx
                     +(uv.y-z)/Zy, z) ;}
```

Let us finally note that those functions are given for a particular decomposition along a particular axis. It is easy to retrieve for each decomposition the given look-up functions.

4.2 Recursive volume filtering on GPU

The regularization of the field is the most critical part in the registration loop. It consists of the Gaussian filtering of the 3 components of the deformation field. In order to speed up this process, the recursive filtering scheme presented in 3.2 was implemented on the GPU. As stated previously, for each axes a *causal* and an *anti-causal* part has to be computed, which sums up in parsing the volume in one direction and its opposite. In order to factorize those two traversals, the volume is transferred to the graphics board memory as a texture containing slices of the volume. This texture contains two decompositions of the volume conduced along one axis and its opposite (on the red and green components of the texture), which allows to handle simultaneously the causal and anti-causal parts. To simulate the traversal process, we simply set up the view frustum to process one slice in a rendering pass. This process in then repeated for each slices along the given direction (which amounts to a full sweep of the volume). Then the *causal* and *anti-causal* parts are added in a

single rendering pass, along with a reorientation of the whole volume on the texture to prepare the next step. Figure 2 is an illustration of this process. At the end of the third axis processing, causal and anti-causal part are added, which ends up the convolution process.

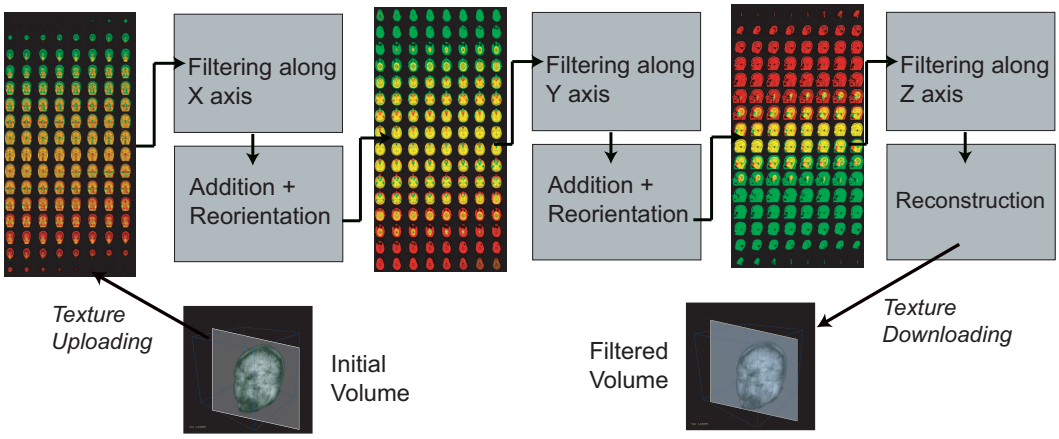

Figure 2. Different steps in the recursive filtering process. Boxes correspond to fragment shaders

5 Results

The registration method was tested on $3D$ magnetic resonance (MR) T1 images presented in figure 3. In order to implement our library, we used the Cg toolkit [13] for desiging fragment shaders. The computer used for tests was an Athlon XP 2500+ equipped with a PCI-Express Quadro FX 1400 with 128 Mo of video memory. The library was written in C++ with OpenGL. First experiments have shown important performances enhancement compared to a CPU optimized version. Next table sums up the times spent within the different parts of the loop.

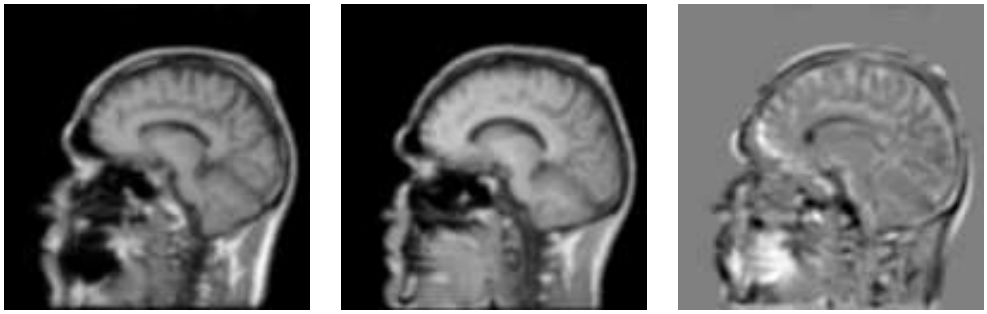

Figure 3. MR images used in the experiments. From left to right: the source volume, the target volume and the initial centered difference image. These images are $2D$ slices of 128^3 $3D$ data.

	Average Performances (sec)		
	CPU	GPU	speedup factor
Computing demons	0.6	0.4	$\times 1.5$
Field regularization	28	2.5	$\times 10$
Trilinear interpolation	0.66	0.31	$\times 2$
Overall time	551	59	$\times 9$

Two brains of different subjects are registered and results are given in figure 4. The size of both volume was 128^3. Registration was performed in around 19 iterations. Results of the cpu and gpu implementations were satisfactory and exhibit small differences. This is due to some implementation details that are different between the software and hardware implementations, like boundary conditions and floating-double precision.

6 Conclusion

This paper presented a GPU implementation of a non-rigid $3D$ image registration method. Results obtained on 128^3 volumes indicate that the GPU implementation is 9 times faster for comparable implementations and parameter tuning: the CPU computation takes around 9 minutes, while the GPU computation takes 59 secondes. Those results are encouraging since they demonstrate the applicability of such methods in an image-guided surgery context. The paper presented two contributions: $3D$ image processing is expressed as operations on $2D$ textures. In addition, an efficient recursive filtering scheme implemented on GPU was presented.

In our implementation, the size of the volume remains a critical aspect, as far as it was not possible to store in video memory all the data needed to process a 256^3 volume. We expect future generation boards to provide sufficient amount of memory (512 Mo) to cope with this difficulty, and larger amount of pipelines to improve in a very significant way our algorithm. Lastly, though a PCI-Express card was used, we expect major improvements to our implementation provided that all the volumes needed for the registration may stay in the video memory. This would allow to minimize the data transfer to/from video memory which remains time-consuming when dealing with large volumes.

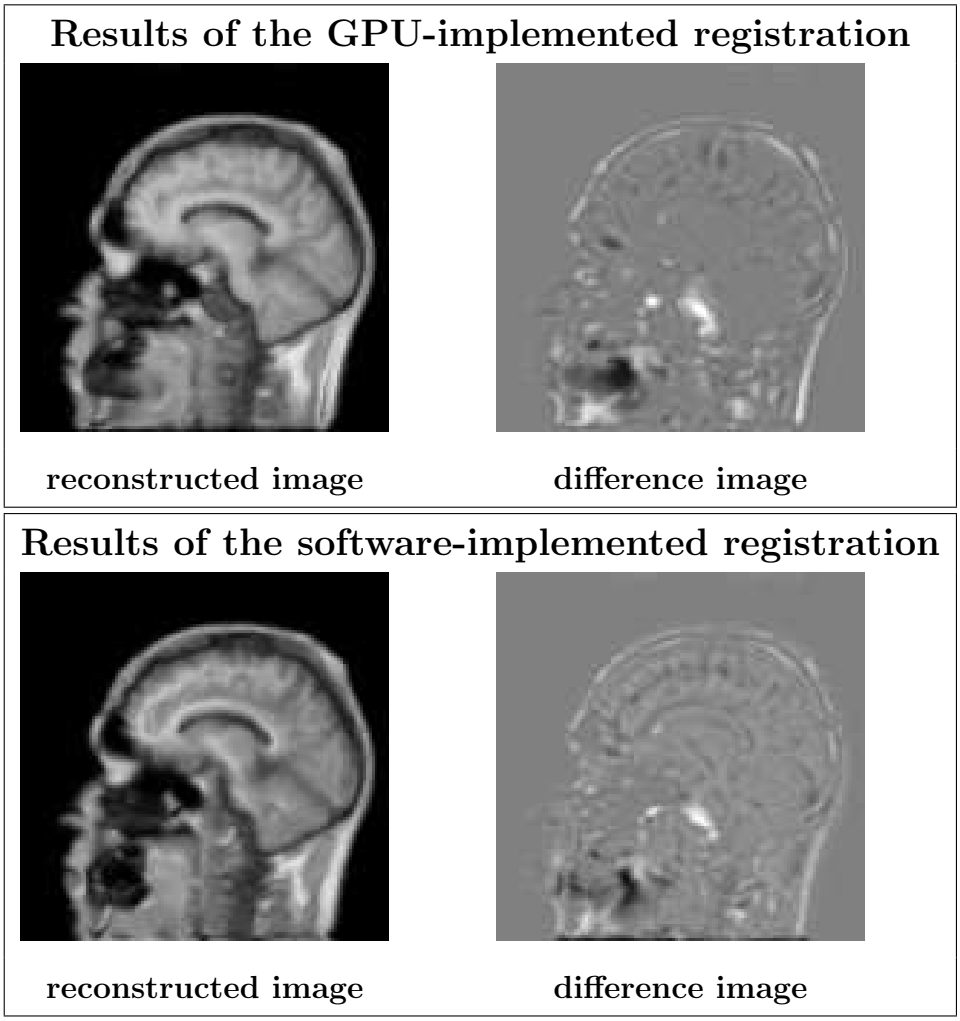

Figure 4. Results of the registration of brains of different subjects. Top row: results of the gpu-implementation, reconstructed image and difference image. Bottom raw: results of the CPU implementation. The reconstructed image is the target image, registered toward the source image using backward trilinear interpolation. It must thus be compared with the source image.

References

[1] C. Hansen A. Lefohn, J. Kniss and R. Whitaker. Interactive deformation and visualization of level set surfaces using graphics hardware. *Proc. of IEEE Visualization 2003*, 2003.

[2] D. Deyc D. Levina and P. Slomka. Acceleration of 3d, nonlinear warping using standard video graphics hardware: implementation and initial validation. *Computerized Medical Imaging and Graphics*, 28:471–483, 2005.

[3] B. Dawant *et al.* Automatic 3-d segmentation of internal structures of the head in mr images using a combination of similarity and free-form transformations: Part i, methodology and validation on normal subjects. *IEEE Trans. Medical Imaging*, 18(10):909–916, 1999.

[4] R. Deriche. Recursively implementing the gaussian and its derivatives. Tech. Rep. 1893, INRIA, http://www.inria.fr/RRRT/RR-1893.html, 1993.

[5] N.L. Dorward. Neuronavigation - the surgeon's sextant. *Journal of neurosurgery*, 11(2):101–103, 1997.

[6] GPGPU. General purpose computation on GPUs. http://www.gpgpu.org, 2004.

[7] M. Harris, W. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In W. Mark and A. Schilling, editors, *Proc. of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware*, pages 92–101, Switzerland, July 2003.

[8] P. Hellier *et al.* Retrospective evaluation of inter-subject brain registration. *IEEE Transactions on Medical Imaging*, 22(9):1120–1130, 2003.

[9] M. Hopf and T. Ertl. Accelerating 3D Convolution using Graphics Hardware. In *Proc. of IEEE Visualization 1999*, pages 471–474, 1999.

[10] J. Krueger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *Proc. of IEEE Visualization 2003*, 2003.

[11] H. Lester and S. Arridge. A survey of hierarchical non-linear medical image registration. *Pattern Recognition*, 32:129–149, 1999.

[12] J. Maintz and MA. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[13] W. Mark, R. Glanville, K. Akeley, and M. Kilgard. Cg: a system for programming graphics hardware in a c-like language. *ACM Trans. on Graphics (TOG)*, 22(3):896–907, 2003.

[14] A. Sherbondy, M. Houston, and S. Napel. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. *Proc. of IEEE Visualization 2003*, 2003.

[15] JP. Thirion. Image matching as a diffusion process: an analogy with Maxwell's demons. *Medical Image Analysis*, 2(3):243–260, 1998.

[16] F. Xu and K. Mueller. Accelerating popular tomographic reconstruction algorithms on commodity pc graphics hardware. To appear in *IEEE Tra. of Nuclear Science*, 2005.

[17] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.