GPU Accelerated Image Registration in Two and Three Dimensions

A. Köhn, J. Drexl, F. Ritter, M. König and H. O. Peitgen

MeVis - Center for Medical Diagnostic Systems and Visualization
Universität Bremen

Abstract. Medical image registration tasks of large volume datasets, especially in the non-rigid case, often put a heavy burden on computing resources. GPUs are a promising new approach to address computational intensive image processing tasks. We investigate recently introduced GPU hardware features that accelerate 2D and 3D rigid and non-rigid registration tasks. Our implementation is entirely GPU based.

1 Introduction

Strzodka et al. [1] describe an implementation of a 2D non-rigid registration algorithm on a graphics processor (GPU) using DirectX 9 compatible hardware. They use the regularized gradient flow (RGF) algorithm originally introduced by Clarenz et al. [2]. RGF is a fast state-of-the-art registration algorithm set in the framework of variational calculus, regularization theory, and deterministic annealing (multiscale) optimization methods, similar to the "velocity-based diffusion registration"of Fischer and Modersitzki [3].

We explore the extension of RGF to 3D on a GPU. Furthermore, we also provide 2D/3D rigid registration.

In contrast to the work of Hastreiter and Soza [4] who used the trilinear interpolation capabilities of graphics hardware to accelerate the deformation of the moving volume but otherwise left the computation on the CPU, our implementation is entirely GPU-based.

We provide performance benchmarks for our implementations and compare them to available CPU based implementations.

2 The Algorithms

Image registration is the problem of finding a coordinate transformation Φ to align a template image f_T to a reference image f_R, so that $f_R \approx f_T \circ \Phi$. We focus on the sum-of-squares difference (SSD) metric:

$$E = \frac{1}{2} \int_{\Omega} \underbrace{[f_R(x) - f_T(\Phi(x))]}_{=:e(x)}^2 dx = \min! \tag{1}$$

with $\Omega = [0;1]^d$, d as the image dimension [2].

Rigid Registration with the SSD criterion: Let $\Phi(x,p)$ be a coordinate transform parameterised by the vector p. The gradient descent algorithm to compute an optimal p^* for (1) can be written as a differential equation:

$$\dot{p} = -\nabla E, \qquad p(0) = p_0 \tag{2}$$

with the gradient of E with respect to p given by:

$$\nabla E = \int_\Omega -e(x) J^T \nabla f_T(\Phi(x,p)) dx \tag{3}$$

and J as the Jacobian: $J_{ij} = \frac{\partial}{\partial p_j}\Phi_i$ of the coordinate transform.
For rigid registration, Φ is a linear rigid transformation (3D: three translations and three rotations). In our code we employ an explicit formula for the computation of the Jacobian, as we found that using finite difference approximations can lead to convergence problems.

Nonlinear Registration via RGF: In this case we deal with a free-form deformation parameterised by a displacement field $u(x)$: $\Phi(x) = x + u(x)$.

The "Gradient Flow"is the Gateaux-Derivative of E with respect to u:

$$E'[u] = -e(x)\nabla f_T(x + u(x))$$

The RGF algorithm consists in updating the smoothed gradient flow:

$$\dot{u} = -B\,E'[u], \qquad u(0) = u_0 \tag{4}$$

to find an optimal u^*, with a d-dimensional linear smoothing operator B. The smoothing (i.e. regularization) is necessary to bring a certain amount of "coherence"into the free-form deformation Φ.

Time-Discretization and Annealing: The time discretization of the differential equations is done by using the explicit Euler rule [2]. The step size of the explicit Euler step is determined by employing the well-known Armijo rule [2]. A deterministic annealing strategy is used to avoid local minima. The strategy works on a Gaussian image pyramid of f_R and f_T, the pyramid scale plays the role of the annealing temperature [2].

Choosing the smoothing operator B in RGF: Following a suggestion of Clarenz [2], we regularize the gradient flow with a Gaussian filter instead of using a multigrid smoother [1]. The main advantage of a multigrid smoother is that more global deformations can get corrected. But this is also achieved by using a multiscale approach, which justifies the use of a simple filtering instead of the comparably expensive multigrid smoother. A similar approach can be found in [5].

Table 1. comparison of voxel accumulation methods (256*256*128 fp16 texture with 1 channel)

method	memory	RT switches	time GPU1	time GPU2
A	18 mb (16+2)	255	130.79 ms	343.36 ms
B	160 kb (128+32)	1024	52.16 ms	42.6 ms
C1	160 kb	8	23.73 ms	12.29 ms
C2	160 kb	8	18.14 ms	9.84 ms

3 GPU implementation issues

3.1 General outline

The GPU Programming is done entirely in the OpenGL GPU language GLSL. We used a GeForce 6800 PCI-E 256MB(GPU1) and a GeForce 6800 GT AGP 256MB (GPU2) board for the benchmarks.

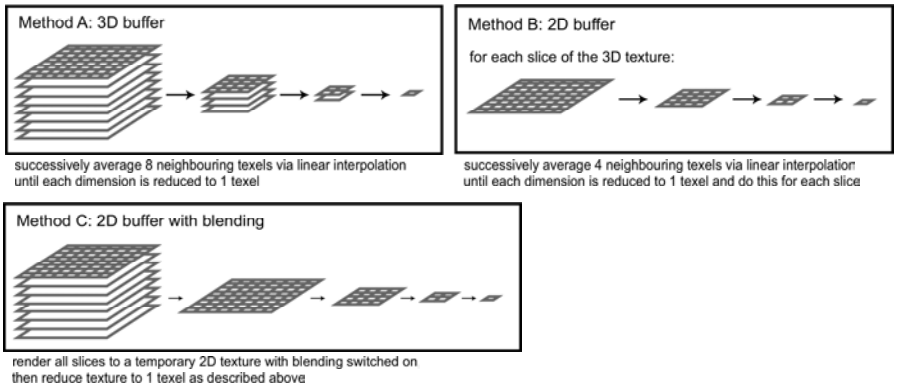
Render to texture: A particular problem arises when attempting to implement image processing algorithms that employ iteration over images. To pass result images from one iteration step as the input to the next iteration step one employs a technique called "render to texture". We use the new Framebuffer Objects (part of the OpenGL 2.0 specification). In [1] so-called Pbuffers have been used for this purpose. According to benchmarkings done by us, Framebuffer objects are 10-20 times faster than Pbuffers.

3.2 Rigid registration

Warping: For the rigid case, we use hardware accelerated image warping of the moving image f_T (via the OpenGL texture matrix) with activated linear interpolation.

Voxel accumulation in 3D: To compute spatial discretized versions of E or a component of the gradient vector ∇E, the GPU has to accumulate the voxels of a volume into a scalar value. We want to take maximal advantage of GPU parallelism and hardware acceleration in the 3D case. We have investigated three methods for this problem and compared them for the task of computing E. The basic idea of all methods is again a pyramidal scheme and employing the linear interpolation hardware of the GPU (Figure 1). Method C also exploits the blending capabilities of graphics hardware, which enable one to linear combine the already written pixel/voxel with the new computed one. Method C is further improved by sampling multiple slices in one pass (method C2). This saves rendering passes and utilizes the graphics pipeline more efficiently. See Table 1 for a comparison of the voxel accumulation methods. The results show that method C is the best choice. Comparing the results of the two GPUs, one can also observe that method C scales the best when using faster hardware.

Fig. 1. Voxel accumulation methods



Using Multiple Render Target (MRT): Since there is currently no support to render to a slice of a 3D texture, one has to use an expensive workaround by rendering to a 2D texture and copying the contents back to the 3D texture. One can avoid this for the rigid case by computing ∇E in one pass. But since $dim(\nabla E) = 6$, and a texture can only hold four components, we would have to compute ∇E in two passes, wasting computational resources. Instead, employing the MRT feature of current GPUs allows rendering to four textures in one pass.

3.3 Non-rigid registration

The displacement field u is represented as a 2D respectively 3D texture in GPU memory, but on a coarser resolution than the images. Since the regularization removes higher spatial frequencies from the displacement field this results in virtually no loss of information regarding the Nyquist Theorem. The required upsampling during the transformation of the moving image f_T is done by hardware using linear interpolation. Update of the displacement field is performed by an explicit Euler step in the differential equation (4). To implement this on the GPU, we again employ blending $u := u + \eta\dot{u}$, with stepsize η determined by Armijo's rule. To evaluate the metric, we use the same approach as described above for the rigid case.

4 Results

We presented some methods to significantly improve performance for rigid and non-rigid registration algorithms using state-of-the-art graphics hardware. Table 2 shows some results. Please note that the GPU timings for the 3D non-rigid case contains expensive copy operations, hence we provided estimated timings without taking them into account (in brackets). We compared our approach with a simplex (Nelder-Mead) based and a steepest descent based algorithm for the rigid case. The later uses finite differences to calculate the gradient. We provided

Table 2. Results of our registration implementations

	rigid		non-rigid	
	2D	**3D**	**2D**	**3D**
resolution	256*256	256*256*128	256*256	256*256*128
iterations	77	76	67	16
GPU time	105 ms	5.24 s	200 ms	62.5 s (9.2 s)
t/iter	1.36 ms	69 ms	2.98 ms	3.9 s (575 ms)
iterations	47 / 58	219 / 36	42	-
CPU time	656 ms / 6.1 s	180 s / 260 s	4.21 s	-
t/iter	14 ms / 105 ms	822 ms / 7.2 s	100 ms	-

CPU: Pentium 4 3.4 GHz 2GB RAM - GPU: GeForce 6800 PCI-E 256MB

both timings; the first belongs to the simplex based registration. We only had CPU code for the 2D non-rigid case.

5 Discussion

GPU based image registration achieves a significant speedup for the 2D case and the 3D rigid case. For 2D rigid we obtained a speed advantage of factor 10 for the GPU per iteration, and a speed advantage of factor 12 for 3D rigid. This allows us to register a 256x256x128 voxels volume in approx. 5 seconds. Our 3D non-rigid registration however is not as fast as one would expect. The reason for this is a memory copy bottleneck: Current NVIDIA and ATI drivers don't support rendering to a slice of a 3D volume so we have to render to a 2D texture and copy the results back to the 3D texture. This amounts to 80%-90% of the total computation time. Therefore, a huge speedup can be expected with the introduction of future drivers that support the render-to-3D-slice interface.

Acknowledgments. We'd like to thank Dr. Robert Strzodka (Stanford University, Computer Science) for sharing his ideas with us, and Tobias Böhler (MeVis) for the 2D RGF CPU code.

References

1. Strzodka R, Droske M, Rumpf M. Image Registration by a Regularized Gradient Flow - A Streaming Implementation in DX9 Graphics Hardware. Computing 2004.
2. Clarenz U, Droske M, Rumpf M. Towards fast non-rigid registration in Inverse Problems, Image Analysis and Medical Imaging. AMS Special Session Interaction of Inverse Problems and Image Analysis 2002;313:67–84.
3. Modersitzki J. Diffusion registration. Oxford University press; 2005. p. 153.
4. Soza G, Bauer M, Hastreiter P, Nimsky Ch, Greiner G. Non-Rigid Registration with Use of Hardware-Based 3D Bézier Functions. MICCAI 2002;p. 549–556.
5. Stefanescu R, Pennec X, Ayache N. Grid-enabled non-rigid registration of medical images. Parallel Processing Letters 2004;14:197–216.