caesar preprint

Gradient Flow Registration A Streaming Implementation in DX9 Graphics Hardware

Robert Strzodka, Marc Droske, Martin Rumpf

Document info:

Type:	Preprint
Group:	caesar-smm
Preprint ID:	32
Date:	17.01.2003

this page intentionally left blank

GRADIENT FLOW REGISTRATION
A STREAMING IMPLEMENTATION IN DX9 GRAPHICS HARDWARE

R. Strzodka

research center caesar, D-53111 Bonn
strzodka@caesar.de

M. Droske, M. Rumpf

Duisburg University, D-47048 Duisburg
{droske,rumpf}@math.uni-duisburg.de

ABSTRACT

The analysis of image time series requires a correlation of the information between two images. The gradient flow registration is a method for correlating this information by successively minimizing an appropriate energy along its gradient. A graphics hardware implementation of this approach to image registration is presented. The gradient flow formulation makes use of a robust multi-scale regularization, an efficient multi-grid solver and an effective time-step control. The locality of the involved operations implies a data-flow which is very well suited for an acceleration in the streaming architecture of the DX9 graphics hardware. Therefore the implementation promises very high performance, however the appropriate graphics hardware is not available before February. Currently the examples have been computed on a emulator, but the implementation will run unchanged on the soon released graphics hardware.

1. INTRODUCTION

The analysis of temporal changes in anatomic structures in image assisted diagnostics and surgery planning strongly depends on robust correlation of images taken at different times. This problem called image registration is therefore one of the fundamental tasks in medical imaging. The aim is to correlate two images via a usually non-rigid deformation. This deformation may reflect temporal changes in the image source or can compensate unknown deformation effects from the image acquisition technology.

The optimal correlation between two images depends on the definition of a coherence measure. However, there may be many minimizers to such a measure. Therefore many regularizations of the registration problem have been discussed in the literature [1, 2, 3, 4, 5]. The graphics hardware algorithm in this paper follows in its implementation the gradient flow registration presented in [6]. It incorporates many of the ideas of iterative Tikhonov regularization methods [7], fast multi-grid smoothing [8] and multi-scale use for large displacements [9]. The model will be summarized in the next section. The implementation in this paper focuses on a basic intensity based model. Morphological image matching is to be considered in future.

Modern graphics hardware can be used for very complex procedural texturing and shading [10, 11] allowing an enormous range of visual effects. But the optimization of graphics hardware for the processing of large data volumes made them also attractive for many other problems as diverse as robot motion planning [12], computation of Voronoi diagrams [13], flow visualization [14], morphological operations [15], segmentation [16] and many others ([17] contains a good overview). Although schemes for general computation and especially discrete solvers for partial differential equations in graphics hardware have been previously described [18, 19, 20], the researchers always emphasized the problem of the low number precision and incomplete set of operations which restricted the application area.

The new DX9 graphics hardware overcomes these problems by introducing a floating point number format and a set of the most common mathematical operations. Together with an access from high level languages to these features, most of the code running on common micro-processors could be coded for the graphics hardware. But this does not mean that any problem could be accelerated in this way, rather the analysis of the problem structure must determine the choice of the hardware architecture. So while in the past the challenge of graphics hardware accelerated implementations lay in the translation of the problem solver into the restricted functionality, today the challenge lies in the construction of a problem solver whose data-flow structure ideally fits the streaming architecture of graphics hardware. The choice of the gradient flow registration algorithm has been guided by this idea.

2. GRADIENT FLOW REGISTRATION

2.1. Continuous Model

Given two images $T, R : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^2$ we look for a deformation $\phi : \Omega \to \Omega$ which maps the intensities of T via ϕ to the intensities of R such that $T \circ \phi \approx R$. Since ϕ will be small in comparison to $|\Omega|$ it can be suitably expressed as

$\phi = \mathbb{1} + u$, with a displacement function u. The displacement u is sought as the minimum of the energy

$$E[u] = \frac{1}{2}\int_\Omega |T \circ (\mathbb{1} + u) - R|^2 \,.$$

A minimizer u in some Banach space $\mathcal{V}$ is characterized by the condition

$$\int_\Omega E'[u] \cdot \theta = 0 \,,$$

for all $\theta \in [C_0^\infty(\Omega)]^2$, with the L^2-representation of E'

$$E'[u] = (T \circ (\mathbb{1} + u) - R)\nabla T \circ (\mathbb{1} + u) \,.$$

This gradient may be used as the descent direction towards a minimum in a gradient descent method. But there may be many minima since any displacements within a level-set of T do not change the energy. Therefore the descent along the gradient will be regularized by $A(\sigma)^{-1}E'[u]$, with $A(\sigma) = \mathbb{1} - \frac{\sigma^2}{2}\Delta, \sigma \in \mathbb{R}^+$. Then the regularized gradient flow

$$\begin{aligned} \partial_t u &= -A(\sigma)^{-1}E'[u] \,, \\ u(0) &= u_0 \end{aligned}$$

has a unique solution u with $u(t) \in \mathcal{V}$ for some function space $\mathcal{V}$ (Theorem 3.1 [6]).

But since the energy E is non-convex the gradient descent path may easily get trapped in local minima instead of finding the global minimum of E. Therefore a continuous annealing method is used by defining a multi-scale of image pairs $T_\varepsilon := S(\varepsilon)T$, $R_\varepsilon := S(\varepsilon)R$ for $\varepsilon \geq 0$ with a filter operator $S(\cdot)$. The choice $S(\varepsilon) = A(\varepsilon)^{-1}$ corresponds again to a Gaussian filtering. The energy

$$E_\varepsilon[u] = \frac{1}{2}\int_\Omega |T_\varepsilon \circ (\mathbb{1} + u) - R_\varepsilon|^2$$

induces the corresponding gradient flow which has the solution $u_\varepsilon(\cdot)$ on scale ε.

2.2. Discretization

Time is discretized by the explicit Euler scheme

$$\frac{u_\varepsilon^{n+1} - u_\varepsilon^n}{\tau_\varepsilon^n} = -A(\sigma)^{-1}E'_\varepsilon[u^n] \,,$$

where τ_ε^n is determined by Armijo's rule. Finite-Elements are used for the discretization in space. Let $\{\Psi^i\}_{i \in I_h}$ be the canonical nodal basis of the linear finite element space $\mathcal{V}^h$. Suppose $\bar{U}^n$ is the nodal vector at the n-th time step. Then we obtain the fully discrete scheme

$$\bar{U}_\varepsilon^{n+1} = \bar{U}_\varepsilon^n - \tau_\varepsilon^n A_h(\sigma)^{-1}\bar{E}'_\varepsilon[\bar{U}_\varepsilon^n] \,,$$

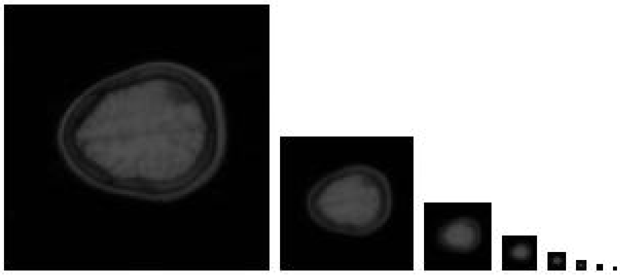

Fig. 1. Textures at different spatial resolution corresponding to the multi-scale of images.

where the matrix $A_h(\sigma)$ is the discrete counterpart of the operator $A(\sigma)$ in $\mathcal{V}^h$ [6]. We apply this formula to compute an approximate solution $\bar{U}_\varepsilon^{N_\varepsilon}$ on scale ε by iterating it N_ε times until the update $\tau_\varepsilon^n A_h(\sigma)^{-1}\bar{E}'_\varepsilon[\bar{U}_\varepsilon^{N_\varepsilon}]$ is sufficiently small in the L^2 norm.

Since multi-grid solvers are the most efficient tools in solving linear systems of equations, the gradient smoothing $A_h(\sigma)^{-1}\bar{E}'_\varepsilon[\bar{U}_\varepsilon^n]$ is performed as a multi-grid V-cycle with Jacobi iterations as smoother and standard prolongation and restriction operators. Indeed to ensure an appropriate regularization it turns out to be sufficient to consider only a single multi-grid cycle $\mathrm{MGM}(\sigma) \approx A_h(\sigma)^{-1}$. The same is considered in the computation of the multi-scale of images.

The image scales are chosen exponentially, i. e. we consider scales $(\varepsilon_k)_{k=0,\dots,K}, K \in \mathbb{N}$ and couple them with the spatial resolution. We reduce the number of necessary computations for large scales by defining a pyramid of grids $(\mathcal{M}_{h_l})_{l=0,\dots,L}$, $h_l = 2^{l-L}$ and resolving the images T_{ε_k} and R_{ε_k} on the coarsest grid $\mathcal{M}_{h_l}$ for which $\varepsilon_k \geq h_l$ still holds (cf. Figure 1). For further details we refer to [6].

3. HARDWARE IMPLEMENTATION

3.1. Data-flow

The two dimensional input images T and R are represented as 2D textures on the finest grid $\mathcal{M}_{h_0}$. The multi-grid hierarchy $(\mathcal{M}_{h_l})_{l=0,\dots,L}$ corresponds to textures of successively smaller size (Figure 1). Several such hierarchies are reserved in graphics memory to store any intermediate results, because once the two images T and R are stored in graphics memory all operations are performed on the graphics hardware from where the final result is displayed.

All textures are implemented as floating point pbuffers, such that one can consecutively write and read from them. Computations are performed by loading an operational kernel to the programmable fragment pipeline and streaming the texture operands through that kernel into a target pbuffer. The target pbuffer can then be used as a texture operand in the succeeding operation. Such a streaming operation is extremely fast if only local information is accessed, i.e. to compute the result for any texel in the target pbuffer information from only small neighborhoods of the corresponding texels in the texture operands are necessary. The ingredients of the gradient flow registration fulfill this condition very well.

3.2. Algorithm

The algorithm starts by setting the initial displacement $\bar{U}^0_{\varepsilon_K}$ on the coarsest scale ε_K to zero. Then the gradient flow at this scale computes from this vector the approximate solution $\bar{U}^{N_{\varepsilon_K}}_{\varepsilon_K}$. This solution is used as the initial displacement $\bar{U}^0_{\varepsilon_{K-1}}$ at the next finer scale ε_{K-1}. This process $\bar{U}^0_{\varepsilon_K} := \bar{0}$, $\bar{U}^0_{\varepsilon_{i-1}} := \bar{U}^{N_{\varepsilon_i}}_{\varepsilon_i}$ continues for $i = K-1, \ldots, 1$ until the final solution $U^{N_{\varepsilon_0}}_{\varepsilon_0}$ on the finest scale ε_0 is obtained.

The main computational part, the solution to the gradient flow problem at scale ε is given in pseudo-code notation:

gradient flow at scale ε {
compute new image scales $\mathrm{MGM}(\varepsilon)\bar{T}$,$\mathrm{MGM}(\varepsilon)\bar{R}$;
for each n {
evaluate energy gradient $\bar{E}'_\varepsilon[\bar{U}^n_\varepsilon]$;
perform smoothing multi-grid V-cycle $\mathrm{MGM}(\sigma)\bar{E}'_\varepsilon[\bar{U}^n_\varepsilon]$;
evaluate Armijo's rule;
compute new solution $U^{n+1}_\varepsilon = \bar{U}^n_\varepsilon - \tau^n_\varepsilon \mathrm{MGM}(\sigma)\bar{E}'_\varepsilon[\bar{U}^n_\varepsilon]$;
stop if $\|\tau^n_\varepsilon \mathrm{MGM}(\sigma)\bar{E}'_\varepsilon[\bar{U}^n_\varepsilon]\|_2^2 < \delta$;
}
}

The smoothing with the multi-grid V-cycle involves as operational kernels the operations of prolongation, restriction, and the Jacobi iterations with A_h. Armijo's rule on the other hand requires a kernel for the error computation $\bar{T}_\varepsilon \circ (\bar{1} + \bar{U}^n_\varepsilon) - \bar{R}_\varepsilon$ and a L^2 scalar product. The energy $\bar{E}_\varepsilon[\bar{U}^n_\varepsilon]$ is namely evaluated as the L^2 scalar product of the error with itself.

All this kernels have been programmed in Cg [21] and perform the operations in one pass. Only the L^2 scalar product must be evaluated by a component-wise multiplication and an iterative addition of local texels because it involves a global access to all texels of a texture. The result of the iterative addition is retrieved from the coarsest level.

3.3. Results and Performance

In Figure 2 and 3 four 256^2 images are arranged in the following way. On the upper left we see the template with a possible acquisition artefact. On the upper right the original is displayed. On the lower left we see the computed deformation applied to a uniform grid and on the right the matching result. Obviously both correlations can eliminate the introduced artefacts very well.

Currently the implementation runs on a emulator of the DX9 graphics unit GeForceFX which should be on sale in February. Therefore performance numbers can only be estimated. From the experience with PDE implementations on DX8 graphics hardware one can expect the matching of 256^2 images to be completed in less than a second.

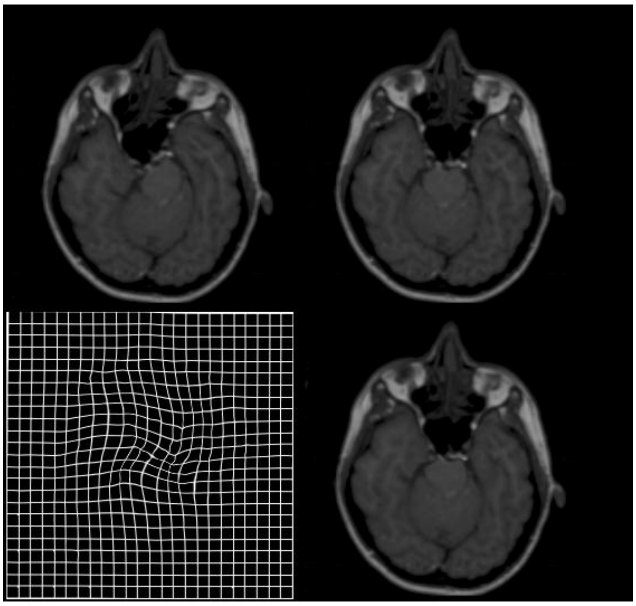

Fig. 2. The elimination of a whirl artefact.

4. CONCLUSIONS

A graphics hardware implementation of the gradient flow registration has been presented. The algorithm has been suitably divided into tasks which can be quickly performed by streaming the data through programmable kernels, an operation which best fits the streaming architecture of DX9 graphics hardware. The use of floating point pbuffers eliminates previous precision problems. The expected performance of less than one second for matching of 256^2 images would allow the use of this algorithm in interactive image assisted diagnostics. Further research will concentrate on fast morphological matching, which could also register images of different modalities.

5. REFERENCES

[1] G. E. Christensen, S. C. Joshi, and M. I. Miller, "Volumetric transformations of brain anatomy," *IEEE Trans. Medical Imaging*, vol. 16, no. 6, pp. 864–877, 1997.

[2] C. A. Davatzikos, R. N. Bryan, and J. L. Prince, "Image registration based on boundary mapping," *IEEE Trans. Medical Imaging*, vol. 15, no. 1, pp. 112–115, 1996.

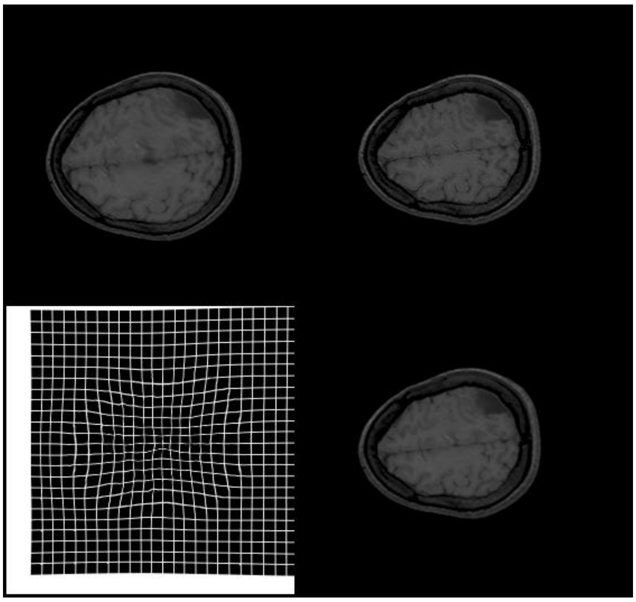

Fig. 3. The elimination of a lens artefact.

[3] U. Grenander and M. I. Miller, "Computational anatomy: An emerging discipline," *Quarterly Appl. Math.*, vol. LVI, no. 4, pp. 617–694, 1998.

[4] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multi–modal volume registration by maximization of mutual information," *IEEE Trans. Medical Imaging*, vol. 16, no. 7, pp. 187–198, 1997.

[5] J. P. Thirion, "Image matching as a diffusion process: An analogy with maxwell's demon," *Medical Imag. Analysis*, vol. 2, pp. 243–260, 1998.

[6] U. Clarenz, M. Droske, and M. Rumpf, "Towards fast non–rigid registration," in *Proceedings of the AMS*, 2002.

[7] M. Hanke and C.W. Groetsch, "Nonstationary iterated tikhonov regularization," *J. Optim. Theory and Applications*, vol. 98, pp. 37–53, 1998.

[8] S. Henn and K. Witsch, "Iterative multigrid regularization techniques for image matching," *SIAM J. Sci. Comput. (SISC)*, vol. Vol. 23 no. 4, pp. pp. 1077–1093, 2001.

[9] L. Alvarez, J. Weickert, and J. Sánchez, "Reliable estimation of dense optical flow fields with large displacements," *International Journal of Computer Vision*, vol. 39, pp. 41–56, 2000.

[10] Mark S. Peercy, Marc Olano, John Airey, and P. Jeffrey Ungar, "Interactive multi-pass programmable shading," in *Siggraph 2000, Computer Graphics Proceedings,*, Kurt Akeley, Ed. 2000, Annual Conference Series, pp. 425–432, ACM Press / ACM SIGGRAPH / Addison Wesley Longman.

[11] Kekoa Proudfoot, William R. Mark, Svetoslav Tzvetkov, and Pat Hanrahan, "A real-time procedural shading system for programmable graphics," in *SIGGRAPH 2001, Computer Graphics Proceedings*, Eugene Fiume, Ed. 2001, Annual Conference Series, pp. 159–170, ACM Press / ACM SIGGRAPH.

[12] J. Lengyel, M. Reichert, B.R. Donald, and D.P. Greenberg, "Real-time robot motion planning using rasterizing computer graphics hardware," in *Proceedings of SIGGRAPH 1990*, 1990, pp. 327–335.

[13] Kenneth E. Hoff III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver, "Fast computation of generalized Voronoi diagrams using graphics hardware," *Computer Graphics*, vol. 33, no. Annual Conference Series, pp. 277–286, 1999.

[14] D. Weiskopf, M. Hopf, and T. Ertl, "Hardware-accelerated visualization of time-varying 2d and 3d vector fields by texture advection via programmable per-pixel operations," in *Proceedings of VMV'01*, 2001, pp. 439–446.

[15] M. Hopf and T. Ertl, "Accelerating Morphological Analysis with Graphics Hardware," in *Workshop on Vision, Modelling, and Visualization VMV '00*, 2000, pp. 337–345.

[16] M. Rumpf and R. Strzodka, "Level set segmentation in graphics hardware," in *Proceedings ICIP'01*, 2001, vol. 3, pp. 1103–1106.

[17] Mark J. Harris, "General purpose computation using graphics hardware," http://www.cs.unc.edu/˜harrism/gpgpu.

[18] Chris Trendall and A. James Stewart, "General calculations using graphics hardware, with applications to interactive caustics," in *Eurographics Workshop on Rendering*, June 2000.

[19] M. Rumpf and R. Strzodka, "Using graphics cards for quantized FEM computations," in *Proceedings VIIP'01*, 2001, pp. 193–202.

[20] Mark J. Harris, Greg Coombe, Thorsten Scheuermann, and Anselmo Lastra, "Physically-based visual simulation on graphics hardware," in *Proceedings of Graphics Hardware 2002*, 2002, pp. 109–118.

[21] NVIDIA Corporation, "Cg programming language," http://developer.nvidia.com/view.asp?PAGE=cg_main, 2002.