R. STRZÓDKA [*], M. DROSKE, M. RUMPF [**]

FAST IMAGE REGISTRATION IN DIRECTX9 GRAPHICS HARDWARE

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 obtains registration results at very high performance, registering two 256^2 in less than 2 seconds, such that it could be used as an interactive tool in medical image analysis.

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 maybe 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 optimisation 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

[*]   Research Center Caesar, D-53044 Bonn, strzodka@caesar.de
[**]   Duisburg University, D-47048 Duisburg {droske,rumpf}@math.uni-duisburg.de

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 correspondence of the problem and hardware structure must determine the choices for an appropriate algorithm and hardware architecture. The choice of the gradient flow method for fast image registration in graphics hardware has been guided by this idea.

2. GRADIENT FLOW REGISTRATION

Given two images $T, R : \Omega \rightarrow \Re, \Omega \subset \Re^2$ we look for a deformation $\phi : \Omega \rightarrow \Omega$ 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 $\phi = \amalg + u$, with a displacement function u. The displacement u is sought as the minimum of the energy

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

A minimizer u in some Banach space V is characterized by the condition

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

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

$$E'[u] = (T \circ (\amalg + u) - R) \nabla T \circ (\amalg + u)$$

This gradient may be used as the descent direction towards a minimum in a gradient descent method. But there maybe 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 $A(\sigma) = \amalg - \frac{\sigma^2}{2} \Delta, \sigma \in \Re^+$. Then the regularized gradient flow

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

has a unique solution u with $u(t) \in V$ for some function space 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

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

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

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 V^h. Suppose $\overline{U}^n$ is the nodal vector at the n-th time step. Then we obtain the fully discrete scheme

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

where the matrix $A_h(\sigma)$ is the discrete counterpart of the operator $A(\sigma)$ in V^h [6]. We apply this formula to compute an approximate solution $\overline{U}_\varepsilon^{N_\varepsilon}$ on scale ε by iterating it N_ε times until the update $\tau_\varepsilon^n A_h(\sigma)^{-1} \overline{E}_\varepsilon'[\overline{U}_\varepsilon^{N_\varepsilon}]$ 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 $A_h(\sigma)^{-1} \overline{E}_\varepsilon'[\overline{U}_\varepsilon^n]$ 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 $MGM(\sigma) \approx A_h(\sigma)^{-1}$. The same is considered in the computation of the multi-scale of images.
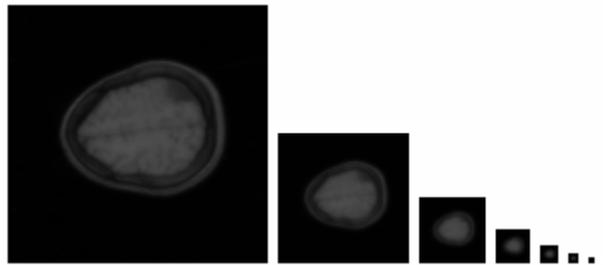

Fig.1. Textures at different spatial resolution corresponding to the multi-scale of images.

The image scales are chosen exponentially, i. e. we consider scales $(\varepsilon_k)_{k=0,\dots,K}, K \in \aleph$ and couple them with the spatial resolution. We reduce the number of necessary computations for large scales by defining a pyramid of grids $(M_{h_l})_{l=0,\dots,L}, h_l = 2^{l-L}$ and resolving the images T_{ε_k} and R_{ε_k}

on the coarsest grid 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

The two dimensional input images T and R are represented as 2D textures on the finest grid M_{h_0}. The multi-grid hierarchy $(M_{h_l})_{l=0,..,L}$ 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 streaming operations are 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 neighbourhoods of the corresponding texels in the texture operands are necessary. The ingredients of the gradient flow registration fulfill this condition very well.

3.1. ALGORITHM

The algorithm starts by setting the initial displacement $\overline{U}_{\varepsilon_K}^0$ on the coarsest scale ε_K to zero. Then the gradient flow at this scale computes from this vector the approximate solution $\overline{U}_{\varepsilon_K}^{N\varepsilon_K}$. This solution is used as the initial displacement $\overline{U}_{\varepsilon_{K-1}}^0$ at the next finer scale ε_{K-1}. This process $\overline{U}_{\varepsilon_K}^0 := \overline{0}, \overline{U}_{\varepsilon_{i-1}}^0 := \overline{U}_{\varepsilon_i}^{N\varepsilon_i}$ continues for i=K-1,..,1 until the final solution $U_{\varepsilon_0}^{N\varepsilon_0}$ 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 MGM(ε)T̄, MGM(ε)R̄;
      for each n {
            evaluate energy gradient Ē'ε[Ū ε n];
            perform smoothing multi-grid V-cycle MGM(σ)Ē'ε[Ū ε n];
            evaluate Armijo's rule;
            compute new solution U ε n+1 = Ū ε n − τ ε n MGM(σ)Ē'ε[Ū ε n]
            stop if ‖τ ε n MGM(σ)Ē'ε[Ū ε n]‖²₂ < δ;
      }
}
```

The smoothing with the multi-grid V-cycle involves as operational kernels the operations of prolongation, restriction, and the Jacobi iterations with Ah. Armijo's rule on the other hand requires

a kernel for the error computation $T_\varepsilon \circ (1 + \overline{U}_\varepsilon^n) - \overline{R}_\varepsilon$ and a L^2 scalar product. The energy $\overline{E}_\varepsilon\left[\overline{U}_\varepsilon^n\right]$ is namely evaluated as the L^2 scalar product of the error with itself. All this kernels have been programmed in Cg [21], a high level graphics programming language, 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.

3.2. RESULTS AND PERFORMANCE

We have tested the algorithm by imposing a non-linear deformation on a source image and then trying to eliminate this artefact by applying the gradient flow registration. Figures 2 and 3 present two such examples with the four 256^2 images 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 the algorithm can eliminate the introduced artefacts very well.

The implementation depends on the programmability of the kernels from the previous subsection and thus requires aDX9 compatible graphics card. We have used a card powered by the GeForceFX 5800 Ultra chip from NVIDIA. The computation were performed in the half float format, which requires 16 bit for each colour component of a texture. This precision turns out to be sufficient for our registration purposes concerning stability of the algorithm and quality of the results. The registration results were obtained in less than 2 seconds. The complete initialisation of the program, including the reading of the images from the hard disk and their transfer to the graphics hardware, does not need more than 2 seconds either. The speed of the algorithm will also further increase in the future, because current methods of pbuffer handling impose unnecessary delays.
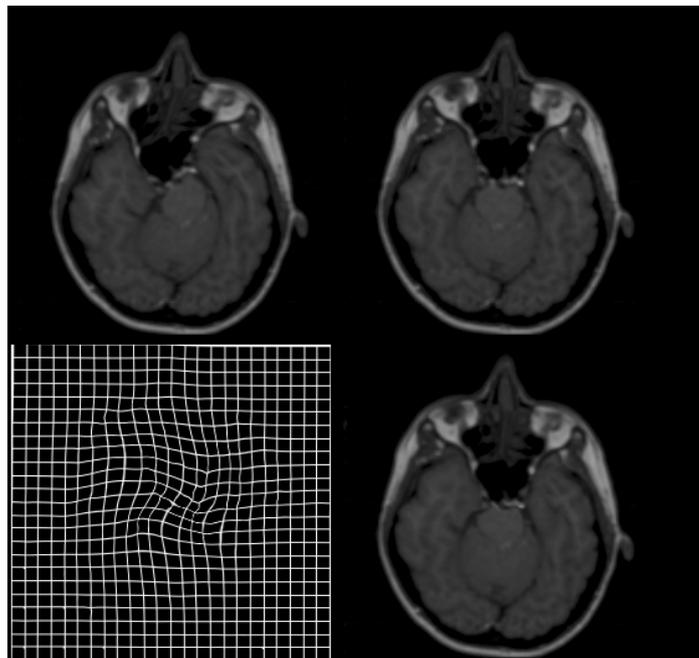

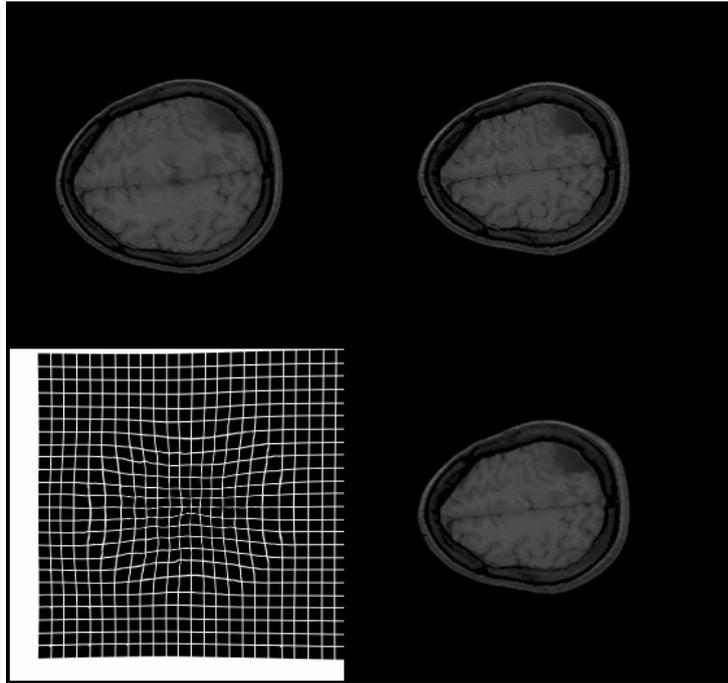Fig. 2. The elimination of a whirl artefact.

Fig. 3. The elimination of a lens 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 of graphics hardware accelerated image processing. The performance of less than 2 seconds for matching of 2562 images will allow the use of this algorithm in interactive image assisted diagnostics. Future work will concentrate on the morphological image matching which could also register images of different modalities.

BIBLIOGRAPHY

[1] G. E. CHRISTENSEN, S. C. JOSHI, 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, J. L. PRINCE, "Image registration based on boundary mapping," *IEEE Trans. Medical Imaging*, vol. 15, no. 1, pp. 112–115,1996.

[3] U. GRENANDER, 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,, 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, M. RUMPF, "Towards fast non-rigid registration," in *Contemporary Mathematics, Special Issue on Inverse Problems and Image Analysis*, Z. Nashed and O Scherzer, Eds. 2002, AMS.

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

[8]   S. HENN, 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, J. S´ANCHEZ, "Reliableestimation 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, 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, 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, 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, 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, T. ERTL, "Hardware accelerated visualization of time-varying 2d and 3dvector fields by texture advection via programmable per-pixel operations," in *Proceedings of VMV'01*,2001, pp. 439–446.

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

[16] M. RUMPF, R. STRZODKA, "Level set segmentationin 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, A. JAMES STEWART, "General calculations using graphics hardware, with applications to interactive caustics," in *Eurographics Workshop on Rendering*, June 2000.

[19] M. RUMPF, R. STRZODKA, "Using graphics cards for quantized FEM computations," in *ProceedingsVIIP'01*, 2001, pp. 193–202.

[20] MARK J. HARRIS, GREG COOMBE, THORSTEN SCHEUERMANN, ANSELMO LASTRA, "Physically-based visual simulation on graphics hardware," in *Proceedings of Graphic sHardware 2002*, 2002, pp. 109–118.

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