Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2013.mssvm04

Vol. **6**, No. 1, pp. 72-94
February 2013

3D Anisotropic Diffusion on GPUs by Closed-Form Local Tensor Computations

Arjan Kuijper[1,2,*], Andreas Schwarzkopf[2], Thomas Kalbe[2], Chandrajit Bajaj[3], Stefan Roth[2] and Michael Goesele[2]

[1] *Fraunhofer IGD, 64283 Darmstadt, Germany.*
[2] *Department of Computer Science, TU Darmstadt, D-64289 Darmstadt, Germany.*
[3] *ICES-CVC, University of Texas at Austin, Austin, Texas 78712, USA.*

Received 6 December 2011; Accepted (in revised version) 15 May 2012

Available online 11 January 2013

Abstract. We present an efficient implementation of volumetric anisotropic image diffusion filters on modern programmable graphics processing units (GPUs), where the mathematics behind volumetric diffusion is effectively reduced to the diffusion in 2D images. We hereby avoid the computational bottleneck of a time consuming eigenvalue decomposition in $\mathbb{R}^3$. Instead, we use a projection of the Hessian matrix along the surface normal onto the tangent plane of the local isodensity surface and solve for the remaining two tangent space eigenvectors. We derive closed formulas to achieve this and prevent the GPU code from branching. We show that our most complex volumetric anisotropic diffusion filters gain a speed up of more than 600 compared to a CPU solution.

AMS subject classifications: 68U10

Key words: Image processing, enhancement, anisotropic diffusion, tensors, 3D filtering.

1. Introduction

Diffusion equations can be considered as physically motivated iterative filters applying a diffusion process on (mostly) noisy image data. They smooth out noise effectively and deliver a framework providing a scale space representation of the image, when time is considered as a natural, continuous scale space parameter [20, 21, 24, 25, 37]. They are well known in the field of image processing and have been subject to many enhancements during the last decades, see e.g. the monographs by Weickert [38, 40] for a complete and comprehensive introduction. Especially the ability to steer the direction and amount of

*Corresponding author. *Email address:* arjan.kuijper@gris.informatik.tu-darmstadt.de (A. Kuijper)

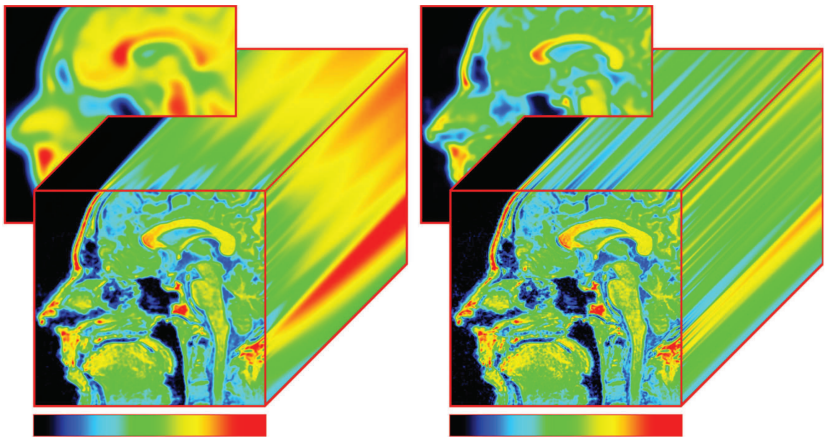

Figure 1: Scale-space representation of homogeneous (left) and inhomogeneous diffusion (right) for a 2D image. 200 iterations, $\Delta t = 0.05$.

diffusion in pre-described directions based on local image structure increased the popularity of these processes. Fig. 1 gives a visual example of the effect of two different types of diffusion on a 2D image extended with a time axes.

Recent publications in both computer graphics and computer vision apply this diffusion for smoothing of normal maps [33], fairing of surfaces and functions on surfaces and meshes [4,5,27], and image compression and inpainting [11], to mention only some possibilities. One should note the essential difference between approaches on the full voxel set and subsets – e.g. meshes – describing shapes. Anisotropic diffusion of whole volume images or general meshes [1, 26] and smoothing vector valued volume images [41] are also common tasks arising in medical applications. It has been reported in 3D imaging as "the most favorable approach regarding the efficiency of noise reduction, signal preservation and computing effort" [10], but the computation time is a bottleneck on traditional CPU implementations for (close to) real-time applications. For off-line denoising and enhancing of 2D images, impressive and fast results are obtained by Weickert's (nD) AOS implementation [38]. It can be parallelized for many types of diffusion by exploiting intrinsic parallelism. Weickert et al. showed this using a regularized Perona-Malik diffusion filter on a $138 \times 208 \times 138$ 3D ultrasound data set [36]. Alternatively, fast explicit schemes can be used [13].

Anisotropic diffusion requires to solve second order partial differential equations (PDEs) numerically. As the amount of discretized data is rapidly increasing, especially in the volumetric 3D case, it is a perfect application for modern graphic cards. Current Graphics Processing Units (GPUs) can easily handle large medical data sets, for example CT or MRI images beyond the traditional 512^3 voxels boundary [2, 18]. The current GPU SIMD (Single instruction, multiple data) architecture allows to solve each iteration in a few milliseconds due to massively parallel processing [7]. The big "however" is that this holds only for equations that lead to an *efficient* parallelization. This is at least difficult for most interesting, non-linear, PDEs. This is due to the local structure in each voxel that determines