Medical Image Analysis 17 (2013) 1073–1094

Contents lists available at SciVerse ScienceDirect

Medical Image Analysis

journal homepage: www.elsevier.com/locate/media

Survey Paper

Medical image processing on the GPU – Past, present and future

Anders Eklund [a,*], Paul Dufort [b], Daniel Forsberg [c,d,e], Stephen M. LaConte [a,f]

[a] Virginia Tech Carilion Research Institute, Virginia Tech, Roanoke, USA
[b] Department of Medical Imaging, University of Toronto, Toronto, Canada
[c] Division of Medical Informatics, Department of Biomedical Engineering, Linköping University, Linköping, Sweden
[d] Center for Medical Image Science and Visualization (CMIV), Linköping University, Linköping, Sweden
[e] Sectra, Linköping, Sweden
[f] School of Biomedical Engineering & Sciences, Virginia Tech-Wake Forest University, Blacksburg, USA

ARTICLE INFO

Article history:
Received 12 October 2012
Received in revised form 7 May 2013
Accepted 22 May 2013
Available online 5 June 2013

Keywords:
Medical imaging
Image processing
Image reconstruction
Graphics processing unit (GPU)
CUDA

ABSTRACT

Graphics processing units (GPUs) are used today in a wide range of applications, mainly because they can dramatically accelerate parallel computing, are affordable and energy efficient. In the field of medical imaging, GPUs are in some cases crucial for enabling practical use of computationally demanding algorithms. This review presents the past and present work on GPU accelerated medical image processing, and is meant to serve as an overview and introduction to existing GPU implementations. The review covers GPU acceleration of basic image processing operations (filtering, interpolation, histogram estimation and distance transforms), the most commonly used algorithms in medical imaging (image registration, image segmentation and image denoising) and algorithms that are specific to individual modalities (CT, PET, SPECT, MRI, fMRI, DTI, ultrasound, optical imaging and microscopy). The review ends by highlighting some future possibilities and challenges.

1. Introduction

Medical image processing on the graphics processing unit (GPU) has become quite popular recently, since this technology makes it possible to apply more advanced algorithms and to perform computationally demanding tasks quickly in a clinical context. Despite this fact, survey papers on GPU-based medical imaging have been rare. Notable exceptions include reviews focused on specific medical imaging algorithms like image registration (Shams et al., 2010b; Fluck et al., 2011), and a recent overview on GPU computing in medical physics (Pratx and Xing, 2011), focused on image reconstruction and treatment plan optimization. More generally, many excellent introductory resources are available on GPUs and GPU programming (Kirk and Hwu, 2010; Sanders and Kandrot, 2011; Farber, 2011).

Here, we broadly consider a range of medical imaging modalities and types of algorithms. We have specifically included some topics that have not been previously reviewed such as functional magnetic resonance imaging (fMRI) and diffusion tensor imaging (DTI), as well as image denoising algorithms and basic functions such as filtering, interpolation, histogram estimation and distance transforms. We have also included papers on GPU-based image reconstruction, both for completeness and to highlight the recent explosion of work in this area. Readers who are particularly interested in image reconstruction are also encouraged to read the review by Pratx and Xing (2011) on this topic. The aim of the review is to give the reader a broad overview of medical image processing on the GPU, and to introduce parallel approaches to many algorithms. Our hope is that by including most types of algorithms and modalities in a single review, a reader focused on a specific field can become inspired by solutions and implementations used in other fields. The review was conducted in a semi-systematic fashion, by using a combination of PubMed, Google Scholar and reference lists of other papers. The inclusion criterion was that the paper should be about an algorithm that is or can be used in medical imaging, and that a GPU implementation is used or described.

1.1. GPUs for general purpose parallel computing

The main reason for the evolution of powerful GPUs is the constant demand for greater realism in computer games. During the past few decades, the computational performance of GPUs has increased much more quickly than that of central processing units (CPUs). Currently, the difference in theoretical performance can differ by a factor ten in favor of the GPU. Computer graphics often require fast rendering of complex scenes. For a good gaming experience, it is normally recommended to use a framerate of at

* Corresponding author. Address: Virginia Tech Carilion Research Institute, 2 Riverside Circle, Roanoke, 240 16 VA, USA. Tel.: +1 540 589 2067.
E-mail address: andek034@gmail.com (A. Eklund).

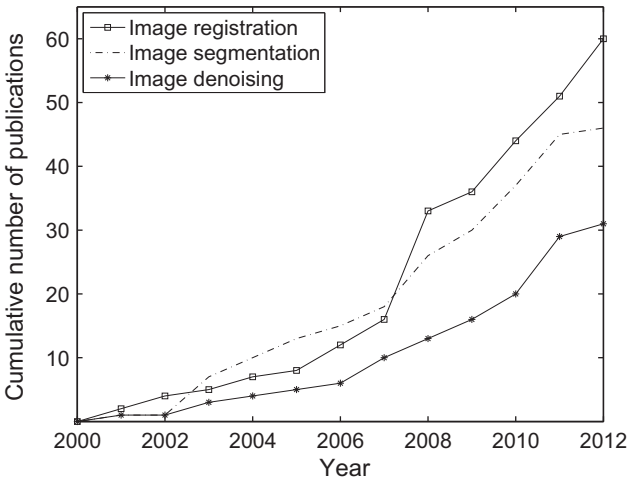
1361-8415/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.media.2013.05.008

least 40 Hz, which means that the scene has to be rendered in less than 25 ms. The rendering is typically performed in exactly the same way for each pixel, but with different data, which explains the single instruction multiple data (SIMD) parallel design of GPUs. While both CPUs and GPUs can manage thousands of threads concurrently via time-slicing, a modern CPU can actually make progress on only 4–12 in parallel. Today's GPU, running a suitable SIMD algorithm, can make progress on over a thousand threads at a time. Intel's streaming SIMD extensions (SSE) can be used to speedup parallel operations on their CPUs, but they often require assembler programming, which is time consuming and error prone.

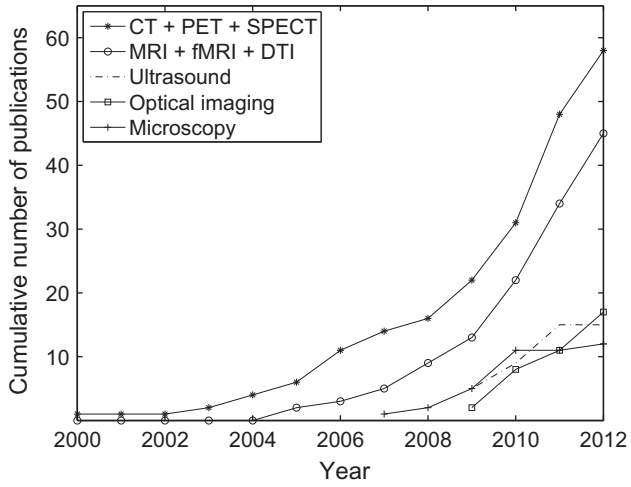
Due to their SIMD parallelism, the performance of GPU implementations greatly depend on how easy it is to make a parallel adaptation of a given algorithm. Image processing algorithms are often massively parallel by nature, which simplifies their implementation on a GPU. But not all algorithms are suited for a parallel implementation. In many cases a hybrid CPU-GPU implementation yields the best performance. A good example is image registration algorithms, where the GPU can be used to calculate a chosen similarity measure in parallel, while the CPU can run a serial optimization algorithm. The evolution of GPU programming models and available tools have greatly simplified the adaptation of key algorithms to the GPU's massive parallelism. Initially, true to their origins, GPUs could only be programmed through computer graphics APIs, such as the open graphics language (OpenGL) or DirectX. The use of GPUs for general purpose computing (GPGPU) was, despite this fact, already popular five years ago (Owens et al., 2007). The release of the compute unified device architecture (CUDA) programming language made using Nvidia GPUs for more general purpose calculations much easier (Nickolls et al., 2008; Garland et al., 2008; Che et al., 2008). The combination of a higher theoretical performance and an easy programming language has led to many reports of large computational gains compared to optimized CPU implementations, though some of these speedups have been questioned (Lee et al., 2010). CUDA can, however, only be used with Nvidia GPUs. The open computing language (OpenCL) is another alternative, which can be used for programming of any hardware, such as AMD GPUs, CPUs, and field programmable gate arrays (FPGAs).

1.2. GPUs in medical imaging

Fig. 1 illustrates how prevalent GPUs are in the field of medical imaging. The first plot presents the number of publications describing GPU acceleration of algorithms often used in medical imaging, while the second plot considers publications for a specific imaging modality. It is clear that GPU accelerated image registration became much more common in 2008, after the release of CUDA in 2007. The CT modality took advantage of graphics hardware much earlier than the MRI modality. This is possibly explained by the fact that MR images often can be reconstructed with an inverse fast Fourier transform, while CT data normally requires more demanding algorithms (e.g. an inverse Radon transform). GPU acceleration of MRI reconstruction algorithms has become much more common during the last three years. One application is to increase the spatial and temporal resolution with techniques such as compressed sensing (Donoho, 2006). During recent years, GPUs have also been used for quite a few ultrasound applications. Since ultrasound systems can generate data with a high temporal resolution (e.g. 20–30 Hz), both in 2D and in 3D, GPUs can aid real-time processing and visualization. These systems are also portable, making it is especially important that the computer hardware is energy efficient and compact. Optical imaging and microscopy have also started to take advantage of GPUs, mainly for acceleration of demanding reconstruction algorithms.



(a) Algorithms



(b) Modalities

Fig. 1. These two plots show the cumulative number of publications for GPU acceleration of algorithms and modalities.

1.3. The GPU architecture and GPU programming

To ease the understanding of the paper, a short introduction to the GPU architecture is presented. Interested readers are referred to books about GPU programming (Kirk and Hwu, 2010; Sanders and Kandrot, 2011; Farber, 2011) and the CUDA programming guide for further details.

Modern GPUs consist of a number of multi-processors, each containing a number of processor cores and memory chips with very high bandwidth. The Nvidia GTX 680 consumer graphics card has 8 multi-processors with 192 processor cores each. Calculations are performed by launching a large number of threads, for example 10,000, that each process a small part of the data (e.g. one pixel or voxel). The threads are divided into a number of thread blocks and each multi-processor can concurrently manage a number of these blocks. The data is first copied from the CPU to the GPU's global memory, which is extensive (several GB) but has a relatively low bandwidth. Each multi-processor has a large number of registers, currently 16,384–65,536, which are used to store thread specific variables. All threads start by reading data from global memory into the local registers, and end by finally writing the result back to global memory when the calculations have been completed.

To facilitate cooperation between threads in a thread block, a small shared memory (16–48 KB) at each multi-processor is available for exchanging data efficiently. A cached small constant memory (64 KB) is also shared between all multi-processors, for example useful if all the threads need to access some constant values. Texture memory is a special way of using the global memory and is also cached, in a format that can be used to speedup irregular reads from global memory that are spatially local. Through the texture memory, linear interpolation in 1, 2 and 3 dimensions can also be performed very efficiently. Additionally, modern GPUs have a general L1 and L2 cache, to further speedup global reads. Rather than achieving coverage of the image domain using nested for loops, GPU code is typically written for one thread, then launched as a 1, 2 or 3 dimensional grid of threads that cover the extent of the image or volume.

To simplify fast development of optimized code, Nvidia has released a number of CUDA libraries for different purposes. Some examples are CUBLAS (basic linear algebra operations, dense matrix operations), CUFFT (fast Fourier transforms in 1, 2, and 3 dimensions), CURAND (random number generation), NPP (Nvidia performance primitives, functions for image processing), thrust (sorts, searches, reductions) and CUSPARSE (sparse matrix operations). The thriving CUDA community has also produced several non-Nvidia libraries as well, including MAGMA[1] and CULA,[2] which both contain functions for matrix algebra. ArrayFire[3] contains a broad variety of functions and also includes support for OpenCL. See the Nvidia homepage[4] for documentation and a complete list of libraries.

1.4. Structure of the paper

The remainder of this review is divided into four sections. The first section considers basic image processing operations, such as filtering and interpolation. The second section is about the most commonly used algorithms in medical imaging; image registration, image segmentation and image denoising. The third section treats algorithms and implementations that are specific to different modalities, e.g., CT, MRI and ultrasound. The last section contains a discussion about future challenges and possibilities of GPU accelerated medical imaging.

2. Basic operations

2.1. Filtering

Filtering is an important processing step for many image processing algorithms. The easiest case involves applying a separable Gaussian low-pass filter, e.g. prior to a downsampling. More advanced cases include edge detection or applying several non-separable filters to estimate a local structure tensor (Knutsson, 1989; Knutsson et al., 2011). For different applications, filtering might be more appropriate in the spatial domain (using convolution) or in the frequency domain (using multiplication). The processing time depends on several factors, the most important being the size and separability of the filter, whether the data dimensions are a power of 2 (which is required for optimal performance with the fast Fourier transform (FFT)), and whether one or several filters are to be applied. In general, convolution is faster for separable filters and small non-separable filters. A major advantage of filtering in the frequency domain is that the processing time is independent of the filter size.

[1] http://icl.cs.utk.edu/magma/.
[2] http://www.culatools.com/.
[3] http://www.accelereyes.com/products/arrayfire.
[4] http://developer.nvidia.com/gpu-accelerated-libraries.

Two-dimensional convolution between a signal s and a filter f can be written for position $[x,y]$ as

$$(s*f)[x,y] = \sum_{f_x=-N/2}^{f_x=N/2} \sum_{f_y=-N/2}^{f_y=N/2} s[x-f_x, y-f_y] \cdot f[f_x, f_y], \quad (1)$$

where N is the filter size. By studying this formula, it is clear that convolution is well suited for parallel processing. The result of each element (e.g. pixel or voxel) can be calculated independently and there is substantial data re-use, since the results for neighboring elements use mainly the same values. A modern GPU implementation would put the filter kernel in the constant memory, as it is used by all threads, and take advantage of the data re-use through texture or shared memory.

Two-dimensional convolution on graphics hardware has been possible through OpenGL for a number of years (Rost, 1996), but the first GPUs did not have features such as constant and shared memory, making it harder to obtain high performance. Hopf and Ertl (1999) were the first to use the GPU for 3D convolution, but only for separable filters. One year later they instead accelerated wavelet filtering (Hopf and Ertl, 2000). James (2001) performed two dimensional filtering in graphics hardware and used it for texture animation (e.g. dynamic bump maps and special effects), but this implementation worked only for small filter kernels. General convolution in 1, 2 and 3 dimensions was eventually also reported (Hadwiger et al., 2001; Hadwiger et al., 2002), mainly for image reconstruction by higher order interpolation (e.g. cubic and spline interpolation). Most of the early papers used OpenGL for the implementation, but the DirectX programming language is another possible alternative, as shown by Mitchell et al. (2003) and in similar work by Sugita et al. (2003). A general problem for the early implementations was the GPU's lack of support for floating point calculations, which resulted in some quality issues (Hadwiger et al., 2003). Two years later, Payne et al. (2005) implemented 2D convolution using Nvidia's C for graphics (Cg) programming language, convolving a 2048×2048 pixel image with a 3×3 filter at 60 Hz. During the same year, a convolution approach to fast cubic interpolation was used (Sigg and Hadwiger, 2005). The cubic interpolation was decomposed into several linear interpolations, to reduce the number of times memory was accessed.

The first examples of GPU accelerated filtering relied heavily on texture memory due to the presence of a fast texture cache. The drawback of this approach is that the convolution operation is then limited by the texture memory bandwidth. With the introduction of CUDA and the GT80 architecture, it became possible to take advantage of the additional shared memory local to each multiprocessor. While being small (16–48 KB), the shared memory make it possible for all threads in a thread block to share data, which is extremely beneficial for convolution operations. In 2007, Nvidia released a white paper (Podlozhnyuk, 2007b) for convolution with CUDA using both texture and shared memory, but it only considered separable convolutions in 2D. The NPP library (Nvidia performance primitives) contains several functions for convolution, but they again only support two dimensional signals and filters stored as integers. In 2008, CUDA was used to accelerate the popular Canny algorithm for edge detection (Yuancheng and Duraiswami, 2008) and in 2010 for convolution with Gabor filters (Wang and Shi, 2010). Eklund et al. (2012c) implemented non-separable 3D filtering with CUDA for volume registration, where both spatial convolution and FFT based filtering was tested. Similar work was presented by Forsberg et al. (2011). Some work on 4D convolution with CUDA has recently also been presented (Eklund et al., 2011b). Eleven non-separable filters of size $11 \times 11 \times 11 \times 11$ were applied to a 4D dataset of resolution $512 \times 512 \times 446 \times 20$, requiring a total of 375,000 billion multiply-add operations. Two dimensional convolution was performed on the GPU and the results were

then accumulated by looping over the last two dimensions on the CPU. Broxvall et al. (2012) also used 4D convolution for a denoising algorithm.

Normalized convolution (Knutsson and Westin, 1993) makes it possible to attach a certainty value to each pixel or voxel. This can be used for filtering of uncertain data, or to properly handle image border effects. A simplified version, which only works for simple filters such as Gaussian lowpass filters, is also known as normalized averaging. Full normalized convolution requires a matrix inverse and several matrix multiplications in each pixel or voxel, which is computationally demanding. A CUDA implementation of normalized convolution was recently presented by Lindholm and Kronander (2011). This implementation makes it possible to estimate gradients from noisy data at interactive frame rates, for example useful for calculation of normal vectors in direct volume rendering.

Multi-dimensional FFTs can be efficiently accelerated by GPUs, as each GPU thread can independently process data along each individual row of a 2D or 3D image. In contrast, GPU-based one dimensional FFTs often require a very large number of samples for the speedup to be significant. Two of the earliest examples of FFT based filtering on a GPU were presented in 2003 (Moreland and Angel, 2003; Mitchell et al., 2003). Since the release of the CUDA programming language in 2007, FFTs in 1, 2 and 3 dimensions can be performed through the CUFFT library. However, due to the rapid development of GPUs and GPU programming languages, GPU libraries are often not as optimized as CPU libraries. For example, Nukada et al. (2008) were able to create a GPU implementation of a 3D FFT that outperformed the CUFFT library. Recently, FFT based filtering on a GPU was used for large volumes in optical microscopy (Karas and Svoboda, 2011). For non-separable filters of the size $100 \times 100 \times 100$ voxels, FFT based filtering is much faster than convolution. A GPU implementation of a 4D FFT is straightforward to achieve by applying two consecutive 2D FFTs (Eklund et al., 2011b), but is still limited by the large memory requirements.

A comparison between spatial and frequency based filtering on the GPU was made by Fialka and Cadik (2006). The conclusion was that FFT-based filtering was faster for non-separable filters, large separable filters and when several filters are applied. The comparison, however, only considered images with dimensions being powers of 2. A similar comparison was presented by Wang and Shi (2010). A general drawback of comparisons between spatial convolution and FFT-based filtering is that a third alternative to perform filtering is often neglected. Filter networks (Andersson et al., 1998; Andersson et al., 1999; Svensson et al., 2005) can speed up convolution by applying and combining many small separable filters, instead of one large non-separable filter. This can, theoretically, reduce the number of multiplications required by a factor of 5 in 2D, 25 in 3D and 300 in 4D (Andersson et al., 1998). Filter networks can thereby also outperform FFT-based filtering for non-separable filters. We are not aware of any GPU implementation of filter networks. A thorough comparison between convolution, FFT-based filtering and filter networks would be a worthwhile addition to the literature. Another unexplored approach is to perform the convolution as a matrix product between a very large matrix and the image, represented as a long vector. This could be done with the CUSPARSE library, for example. It would also be interesting to use the GPU for acceleration of filter optimization (Knutsson et al., 1999), which involves solving a large linear system. For example, for 4D filters of the size $11 \times 11 \times 11 \times 11$, the resulting system is of the size $14{,}641 \times 14{,}641$. Optimizing a filter network is usually even more demanding.

To summarize, a great deal of work has been done on GPU-based filtering. However, there still exists no freely available library for separable and non-separable convolution in 2, 3 and 4 dimensions. In our opinion, such a library would be beneficial for many that work with medical image processing and GPUs. Such a library ideally would go beyond the NPP library, by supporting floating point convolution and providing support for applying several filters simultaneously.

2.2. Interpolation

Interpolation is an application where the GPU achieves maximum advantage. Since computer graphics rely heavily on interpolation calculations, the GPU has been given special dedicated hardware support (texture memory) to accelerate these operations. In medical imaging, interpolation is an important processing step for algorithms ranging from image registration to reconstruction of CT data. Initially the GPU hardware only supported 2D textures, which made it necessary to perform additional calculations in software to do 3D interpolation. Now that 3D textures are supported, both 2D and 3D linear interpolation are no more costly than nearest neighbor interpolation. With the CUDA programming language, linear interpolation in 3D for the position (x,y,z) is easily performed with a single instruction as

$$\texttt{value} = \texttt{tex3D}(\texttt{mytex}, \texttt{x} + 0.5\texttt{f}, \texttt{y} + 0.5\texttt{f}, \texttt{z} + 0.5\texttt{f}),$$

where mytex is a reference to a three-dimensional texture object mapped to global memory. The addition of 0.5 is due to the fact that the original pixel values are for textures actually located between the integer coordinates. Higher order interpolation, such as cubic or spline interpolation, can be achieved with a small amount of additional processing.

As previously mentioned, a convolution approach to performing higher order interpolation was proposed over a decade ago (Hadwiger et al., 2001; Hadwiger et al., 2002), recognizing that convolution and interpolation involve very similar operations. Cubic B-spline interpolation on the GPU was implemented some years later (Sigg and Hadwiger, 2005) by combining several linear interpolations. In direct volume rendering, a multi-resolution approach can be used to lower the memory consumption, but requires the combination of data from different resolution blocks. Ljung et al. (2006) therefore performed interblock interpolation on the GPU to improve the quality of the visualization. Kraus et al. (2007) implemented edge-directed image interpolation with GPU acceleration, to achieve upsampling in real-time without ringing artefacts. Similar work was presented by Cao et al. (2009). Ruijters et al. (2008) implemented interpolation with uniform B-splines by using texture memory. This work was recently extended to achieve improved accuracy and speed (Ruijters and Thevenaz, 2012). By pre-filtering the data to be interpolated, an impressive speedup of 38 times was measured compared to SSE accelerated multi-threaded CPU code. When compared to a straightforward single core CPU implementation, the GPU was 481 times faster.

There are few publications that specifically discuss how to perform interpolation in graphics hardware, but many of the publications on GPU accelerated image registration mention that the texture memory is used for fast interpolation. Nonetheless, since the exact sampling position used for linear interpolation with texture hardware is currently only resolved to 8–9 bits of precision, many have still chosen to implement their own interpolation in software. Just as for convolution, this can be performed efficiently through shared memory, which also makes it easy to apply higher order interpolation. As several medical imaging modalities can generate 4D data (e.g. 3D + time), it would be beneficial to be able to perform 4D interpolation in graphics hardware. We are not aware of any work yet done on this subject.

2.3. Histogram estimation

Histograms are needed for a number of algorithms in image processing. A common example is histogram equalization (Pizer et al., 1987), which can be used for contrast enhancement. Histograms are also a critical component of mutual information-based image registration algorithms (Viola and Wells, 1997; Pluim et al., 2003; Mellor and Brady, 2005), where a joint image histogram needs to be estimated at each iteration. In the field of visualization, histograms are employed to set transfer functions for direct volume rendering.

The basic idea for estimation of a histogram consists of counting the number of values that fall into bins of a certain size. This was difficult to perform using GPU hardware until recently, since naive accumulation of bin counts in parallel would involve several threads trying to modify the same global bin counter simultaneously. One of the earliest examples of histogram estimation on a GPU is the work by Fluck et al. (2006). They used the texture memory and first estimated many small histograms in parallel, then combined them into one histogram in a gather step. Scheuermann and Hensley (2007) instead used a scattering approach, which consists of bin selection for each pixel followed by accumulation of bin contents as in a straightforward CPU implementation. In 2007, Nvidia released a histogram example in the CUDA SDK (Podlozhnyuk, 2007a). A drawback of this example is that it only worked for 64 bins, while a joint image histogram for mutual information based image registration may require as many as 10,000 bins for a 100×100 2D histogram. During the same year (Shams and Barnes, 2007; Shams and Kennedy, 2007) presented two general algorithms for histogram estimation with CUDA. The first algorithm simulated a fence mechanism, such that several threads were blocked from modifying the same bin counter simultaneously. The second algorithm used N bins per GPU thread, for collision free memory accesses, followed by parallel reductions to form the final histogram. Saxena et al. (2010) used a similar idea, but with an algorithm that is severely limited by the lower bandwidth of global memory. Other solutions that have been proposed to decrease the number of memory access collisions include presorting (Chen et al., 2009) and calculating an optimal number of bins (Brosch and Tam, 2009).

More recent GPU hardware now supports so-called atomic functions, which greatly simplifies the development of code for histogram estimation. When a thread executes an atomic function, all the other threads are automatically halted, so that the current thread can read and modify a value in global memory without interference. However, atomic functions can be significantly slower than optimized algorithms that steer clear of collisions altogether. Shams et al. (2010a) therefore developed a new algorithm consisting of a sort and a count step. This algorithm was shown to outperform the older methods when the number of bins is over 100 per image. The comparison was made with the Nvidia GTX 280 graphics card. For the more modern Nvidia GTX 680, the atomic approach may now be faster as the efficiency of atomic functions has improved greatly in recent years. Recently, Gomez-Luna et al. (2012) instead presented a replication-based approach to histogram estimation. By storing several sub-histograms instead of one large one, the number of memory conflicts can be reduced significantly. Vetter and Westermann (2011) combined a presorting step with careful trade-off optimization strategies and achieved a speedup by a factor of 4, compared to the count and sort approach by Shams et al. (2010a).

2.4. Distance transforms

Distance transforms (Rosenfeld and Pfaltz, 1968; Borgefors, 1986) have a long history in the image processing domain and can be used for a wide range of applications (Jones et al., 2006), including the creation of object skeletons and the efficient application of morphological operations. A 2D distance transform converts a binary image to a gray level image, where each pixel has a value corresponding to the distance to the nearest feature pixel. In 3D, the distance transform can also be used to calculate distances to vector representations (e.g. a triangle mesh). Several different distance metrics can be calculated, the most common being the Manhattan distance, chamfer metrics, octagonal metrics and the Euclidean distance.

The distance calculation can be performed independently for each element, and is thereby well suited for GPU implementations. A naive way to solve the problem is to separately calculate the distance for each pixel or voxel. More sophisticated algorithms can be divided into two categories, based on either propagation or Voronoi diagrams. Jump flooding is one approach well suited for parallel calculation of distance maps and Voronoi diagrams on GPUs (Rong and Tan, 2006). Inspired by the simple flood fill algorithm, where a seed is propagated one pixel at a time by looking at its neighbors, jump flooding instead propagates the distance map several pixels at a time. A GPU can accelerate this algorithm significantly, by running many seeds in parallel.

Hoff et al. (1999) were one of the first to use graphics hardware to accelerate the computation of Voronoi diagrams. Sigg et al. (2003) implemented the signed distance transform, using OpenGL's ARB fragment program. This work was improved and extended by Sud et al. (2004), who introduced culling to decrease the number of distance calculations, and by Hsieh and Tai (2005), who focused on the construction of Voronoi diagrams. Strzodka and Telea (2004) instead calculated 2D skeletons. Fischer and Gotsman (2006) presented an implementation based on the tangent-plane algorithm, which also could handle higher order Voronoi diagrams and distance functions (e.g. the distance between each pixel and its k'th nearest neighbor). Sud et al. (2006) extended their work during the same year, by using linear factorization and calculations in the texture memory, to achieve interactive distance field computations in 3D. van Dortmont et al. (2006) instead focused on how to generate 3D skeletons very efficiently, and applied it to both CT and MRI volumes. Cao et al. (2010) used CUDA to implement an algorithm that yields the exact Euclidean distance. Recently, Man et al. (2011) used CUDA and an optimal parallel algorithm to calculate distance maps for high resolution images (80 megapixels).

3. Algorithms

3.1. Image registration

Image registration is one of the most common algorithms in medical imaging and the one with the most GPU implementations (see Fig. 1). One reason for this is the GPU's hardware support for linear interpolation, which makes it possible to transform images and volumes very efficiently. Hastreiter and Ertl (1998) were one of the first to take advantage of the GPU for image registration, mainly for its ability to perform fast interpolation in 3D. A common approach is to let the GPU calculate a similarity measure, often mutual information (Viola and Wells, 1997; Pluim et al., 2003; Mellor and Brady, 2005), over the images in parallel while the CPU runs a serial optimization algorithm to find the parameters (e.g. translations and rotations) that give the best match between the two images. The mutual information between two discrete variables a and b is defined as

$$I(a,b) = \sum_{a\in A}\sum_{b\in B} p(a,b)\log\left(\frac{p(a,b)}{p(a)p(b)}\right). \quad (2)$$

Estimation of the joint probabilities $p(a,b)$ can be performed through efficient algorithms, as described in the section about histogram estimation. The marginal probabilities $p(a)$ and $p(b)$ are then simply calculated by summation of the 2D histogram along each direction. The final summations can, for example, be performed through the thrust library. To calculate the normalized cross-correlation between two images or volumes can also be done in parallel. One solution is to see each image or volume as a long vector and use the CUBLAS library for matrix operations to calculate the required scalar products. Non-rigid registration algorithms can easily modify the motion vector in each pixel or voxel in parallel, according to some update rule. Due to the popularity of using GPUs for image registration, two surveys (Shams et al., 2010b; Fluck et al., 2011) have recently appeared. Rather than duplicating these resources, we have summarized recent publications that have appeared after 2009.

Several recent reports have taken advantage of the fact that the GPU was originally designed for computer graphics and visualization. A popular algorithm for registration between a volume and an image involves the generation of 2D projections, called digitally reconstructed radiographs (DRRs), from the volume. The projections are compared to the image using an appropriate similarity metric, and the registration is performed by finding the rotation and the translation of the volume that maximizes the similarity. The projections can be generated with algorithms that are similar to those used in direct volume rendering, e.g. ray casting. Gao et al. (2012) used this approach to perform registration between 3D trans-esophageal echocardiography and X-ray fluoroscopy images. Spoerk et al. (2012) made a comparison between ray casting and wobbled splat rendering for registration between CT volumes and X-ray images. Dorgham et al. (2012) proposed faster rendering by sparse sampling. With a satisfactory image quality, a DRR of a CT volume comprised of $256 \times 256 \times 133$ voxels could be generated in 2 ms. Steininger et al. (2012) compared three similarity metrics and found that rank correlation performed best. A comparison between different frameworks for GPU acceleration was conducted by Membarth et al. (2011). Five different frameworks (RapidMind, PGI Accelerator, HMPP Workbench, OpenCL and CUDA) were applied to 2D/3D image registration. Code manually written in CUDA resulted in the best performance. The HMPP workbench was the best alternative, in terms of performance, for automatic generation of a GPU implementation from C or Fortran code.

A long term goal in the field of image registration is to perform the required calculations in real-time. Yuen et al. (2008) presented a real-time motion compensation system for ultrasound volumes, such that mitral valve repair can be performed while the heart is beating. By utilizing the fact that the motion of some intra cardiac structures is largely constrained to translation along one axis, the motion could be estimated at 28 Hz. Real-time registration is especially required when combining different modalities during surgery. Brounstein et al. (2011) focused on the problem of registration between pre-operatively acquired CT scans and time resolved ultrasound data, by using an approach based on Gaussian mixture models and local phase. The registration could be performed in about 2 s, while several ultrasound systems can generate volume data at a frame rate of about 25 Hz. Ruijters et al. (2011) reported a fast implementation for non-rigid registration between pre- and intra-operative cone-beam CT volumes. By first finding the large displacements on a coarse scale, volumes of the size $256 \times 256 \times 256$ voxels could be registered in about 7 s.

Other recent examples include GPU acceleration for registration of MRI volumes (Ha et al., 2010; Oh et al., 2011; Huang et al., 2011), diffeomorphic registration algorithms (Han et al., 2010; Huang et al., 2010), free-form deformation (Modat et al., 2010), motion tracking of video microscopy (Liu et al., 2012) and mass-conserving image registration (Castillo et al., 2012). Lee et al. (2012) described how to optimize GPU implementations that are compute- or memory-bound and applied it to image registration. Their optimization strategies resulted in a speedup of a factor 6, compared to a naive GPU implementation.

Except for the work by Brounstein et al. (2011), the previous citations have used image registration algorithms that are based on the image intensity. A less common approach is to look at the neighborhood of a pixel or voxel, e.g. by using a quadrature filter (Granlund and Knutsson, 1995), and instead use phase-based image registration. The main advantage of the local phase is that it is invariant to the image intensity. Phase-based optical flow was introduced in the computer vision field by Fleet and Jepson (1990) and eventually propagated into the medical imaging domain (Hemmendorff et al., 2002). Mutual information-based image registration (Viola and Wells, 1997; Pluim et al., 2003) is often acknowledged for its ability to perform multi-modality registration, but phase mutual information can in some cases perform better (Mellor and Brady, 2004; Mellor and Brady, 2005; Eklund et al., 2011c). However, a general drawback of phase-based image registration is the increase in computational complexity; several non-separable filters have to be applied at each iteration. Pauwels and Hulle (2008) therefore made a GPU implementation of phase-based optical flow in 2D, using Gabor filters. Eklund et al. (2010a) instead used quadrature filters for phase-based affine volume registration and Forsberg et al. (2011) used the GPU to accelerate the Morphon (Knutsson and Andersson, 2005), which is a phase-based non-rigid registration algorithm. In each iteration, a local structure tensor (Knutsson, 1989; Knutsson et al., 2011) is used to improve the registration. Three dimensional convolution with six complex valued (i.e. twelve real valued) non-separable quadrature filters is required to estimate a phase invariant tensor. These works, although few, demonstrate that the GPU also can be used to enable more advanced registration algorithms, which might otherwise be dismissed as being too computationally demanding.

3.2. Image segmentation

Image segmentation in medical imaging is often used to segment brain structures, blood vessels, organs, tumors and bones. Some of the most common algorithms are simple thresholding (global or spatially varying), clustering, level sets (Osher and Sethian, 1988), active contours (snakes) (Kass et al., 1988), region growing algorithms (Adams and Bischof, 1994), the watershed transform (Digabel and Lantuejoul, 1977; Roerdink and Meijster, 2000), classification-based algorithms, graph cuts (Shi and Malik, 2000), segmentation by registration to templates or atlases and segmentation based on (statistical) shape models (Cootes et al., 1995; Heimann and Meinzer, 2009). Segmentation is still an active area of research, and there is no single segmentation algorithm yet found that can solve all problems.

GPU-based image segmentation can be used for three purposes. First, to rapidly compare multiple candidate segmentation algorithms. Second, once a working segmentation algorithm has been established, a GPU can accelerate automatic segmentation of large datasets. Third, a GPU can also perform interactive segmentation and visualization, where the user can help the algorithm to provide a satisfactory result. Combined interactive segmentation and visualization are perfectly suited to the GPU, as data already in GPU memory can be rendered very efficiently.

Level set-based segmentation works by defining a scalar function with positive values inside the current segmentation and negative values outside. The segmentation boundary is thus defined implicitly by the function's zero level set. The function is evolved according to equations depending on many different characteristics of the image and the zero level set itself. A GPU can perform

a parallel update of the level set function over the entire spatial domain of the image at each iteration, requiring many interpolation operations through texture memory. However, since only the zero level set is of interest, a better approach is to only process elements in a narrow band around the zero level set. The difficulty lies in how to represent and process the constantly changing narrow band, requiring irregular memory accesses. Rumpf and Strzodka (2001a) were one of the first to use graphics hardware for level set segmentation in 2D, one application was brain segmentation. Similar work was presented by Hong and Wang (2004), who used their implementation for segmentation of cancer cells and brain structures. Hagan and Zhao (2009) used a Lattice Boltzmann model (LBM) approach to solve level set-based segmentation, a GPU cluster was used to handle large medical volumes. More recent examples include combining level sets and CUDA for segmentation of CT and MRI data (Roberts et al., 2010; Sharma et al., 2010).

Segmentation by active contours is similar to segmentation by level sets, the main difference being that the contour is represented explicitly by a large number of nodes rather than a mathematical function. A GPU can in parallel update the state and location of each node. A first example of GPU-based active contours was presented by He and Kuester (2006), who applied their implementation to MR images. Wang et al. (2011b) instead combined LBM with active contours and applied it to ultrasound and MR images. Domas et al. (2011) especially focused on fast segmentation of large images (15–150 megapixels) and achieved a mean speedup of about 7.

The watershed transform is based upon the fact that any image can be seen as a topographic surface, by introducing an equivalence between image intensity and height. Water falling from above will flow down the sloped surface and reach a local minimum or watershed. Water gathered in different basins, for example separated by ridges of high image gradient, represents segmented regions. A GPU can efficiently search for pixel neighbors with the lowest gray level, which defines the downstream direction of the water, and then quickly propagate the water along these directions. Pixels that are classified as local minima are then merged into regions, an operation which may require communication between different thread blocks. Kauffmann and Piche (2008) instead used a cellular automaton (a collection of simple processing cells arranged on a regular lattice) approach to perform watershed segmentation. A volume of the size $512 \times 512 \times 512$ voxels could be processed in about 17 s. Pan et al. (2008) used a more regular approach, described in the beginning of this subsection, and tested it through segmentation of different organs. Körbes et al. (2011) recently presented an overview of how GPUs have been used for the watershed transform, in which CUDA-based implementations were studied in detail.

Region growing is a simple segmentation algorithm in which each region starts as a single seed element. At each iteration, the neighbors of all pixels in the region are scanned to determine whether the region should be expanded to include them. The criterion for inclusion can be based on many different local properties of the image and can also evolve as the region grows and changes shape. Region growing can be accelerated on the GPU by running many seeds in parallel with shared memory used to avoid reading the same value from global memory several times. A general difficulty involves the need to ensure that different regions do not attempt to absorb the same neighbor at the same time. Erdt et al. (2008) were one of the first to combine region growing with GPU acceleration, and applied it to segmentation of liver vessels. Pan et al. (2008) instead used fast region growing for segmentation of different human organs. Region growing is ideal for interactive segmentation, as the user can easily select a starting point and see how the region grows. Several examples are given in the subsection about interactive and real-time segmentation.

Strzodka et al. (2003) instead used a GPU accelerated version of the Hough transform for object recognition and pose detection. Schoenemann and Cremers (2007) took advantage of a GPU for globally optimal segmentation based on shape priors while Kauffmann and Piche (2010) accelerated a graph-cut based segmentation algorithm, which they applied to segmentation of organs in medical datasets. During recent years, many reports on CUDA implementations of a large variety of segmentation algorithms have been published. Some examples are GPU acceleration of graph cuts (Vineet and Narayanan, 2008), expectation maximization and k-means clustering for analysis of histopathological images of neuroblastoma (Ruiz et al., 2008), registration-based segmentation of MRI volumes (Han et al., 2009), liver segmentation based on Markov random fields (Walters et al., 2009), shape models for segmentation of vertebra in X-ray images (Mahmoudi et al., 2010), random walks (Collins et al., 2012), fuzzy connected image segmentation of CT and MRI volumes (Zhuge et al., 2011) and a hybrid approach to segmentation of vessel laminae from confocal microscope images (Narayanaswamy et al., 2010).

Interactive or real-time segmentation has been the focus of many papers. To accomplish this, most of the early papers used simpler segmentation methods. Yang and Welch (2003) used a global thresholding approach, combined with erosion (shrinking) and dilation (growing) to improve the results. Viola et al. (2003) applied edge preserving smoothing, followed by an interactive global thresholding for segmentation of liver vessels. Also in 2003, Sherbondy et al. (2003) reported a slightly more advanced approach, namely region growing combined with anisotropic diffusion (Perona and Malik, 1990) and also applied it to vessel segmentation. During the same year, Lefohn et al. (2003a), Lefohn et al. (2003b) were able to perform interactive segmentation with level sets in 3D. Similar work was presented by Cates et al. (2004), who applied their implementation to brain tumor segmentation. Already by 2004, an extensive review about interactive segmentation on graphics hardware had appeared (Hadwiger et al., 2004). A large number of segmentation algorithms were evaluated from a parallel processing perspective and implementation issues were also discussed. One year later, Schenke et al. (2005) proposed interactive volume segmentation by region growing and applied it to MRI volumes, while Grady et al. (2005) instead used an algorithm based on the random walker algorithm for organ segmentation and Eidheim et al. (2005) performed segmentation of ultrasound images at 18 Hz using active contours. Chen et al. (2006) combined fast region growing and real-time visualization for interactive segmentation of brains. Similar work was presented by Beyer et al. (2007). Novotny et al. (2007) described a GPU accelerated system for real-time instrument tracking with 3D ultrasound. Instruments often need to be detected and tracked in ultrasound volumes to assist surgeons performing beating heart intra cardiac surgery. To accomplish this, a Radon transform was applied to each volume to detect the instrument. By using a GPU the detection could be performed in 31 ms, which is sufficient for a volume rate of 25 Hz. Similar work was presented by Neshat and Patel (2008). Unger et al. (2008) used active contours, total variation minimization and a GPU implementation for interactive object extraction. Chen et al. (2008) made an interactive framework for volume sculpting and segmentation of medical volumes, which for example can be used for medical education. Just as with conventional segmentation, the CUDA programming language has also recently been used for several interactive implementations. Some examples are interactive visualization and segmentation of neural processes in electron microscopy datasets (Jeong et al., 2009), real-time segmentation by clustering (Abramov et al., 2011; Fulkerson and Soatto, 2010), interactive multi-label segmentation (Santner et al., 2010) and interactive segmentation of bones by using

discrete deformable models (Schmid et al., 2011) or the random walker algorithm (Top et al., 2011).

To summarize, there has been tremendous activity related to 3D image segmentation using the GPU, since there is a huge variety of segmentation algorithms. Given the progress in interactive segmentation, an interesting challenge for the future would be the development of interactive image segmentation approaches in 4D data.

3.3. Image denoising

Image denoising has been a mainstay of medical image processing since its inception (Hunt, 1973; Lee, 1980; Knutsson et al., 1983). The most common algorithms include anisotropic diffusion (Perona and Malik, 1990; Weickert, 1998), bilateral filtering (Tomasi and Manduchi, 1998; Elad, 2002), adaptive filtering (Knutsson et al., 1983; Granlund and Knutsson, 1995; Westin et al., 2001) and non-local means (Buades et al., 2005; Coupe et al., 2008). Besides improving overall image quality, a frequent application of image denoising in medical imaging is to counteract an increased noise level caused by lowering the amount of ionizing radiation in CT. Image registration and segmentation algorithms can also benefit from a decreased noise level.

Anisotropic diffusion is based on partial differential equations (PDEs), which need to be solved for each pixel or voxel. An edge detector is normally used to attenuate diffusion close to edges. The solutions to the PDEs cannot be generated directly, but a large number of iterations (timesteps) are often required. Here the GPU can be used to efficiently update the solution for each element in parallel. Rumpf and Strzodka (2001b) described one of the earliest uses of graphics hardware for anisotropic diffusion. Similar work was presented by Colantoni et al. (2003). The Lattice Boltzmann model (LBM) is one way to solve PDEs for anisotropic diffusion. GPU acceleration of LBM was proposed by Zhao (2008), who also applied it to image denoising. Zhu et al. (2011) used an anisotropic diffusion approach to denoise ultrasound data, showing that a volume could be processed in less than a second. Recently, Schwarzkopf et al. (2012) used CUDA to accelerate nonlinear anisotropic diffusion-based image denoising in 3D. The GPU implementation was 170–600 times faster than an un-optimized CPU implementation.

Bilateral filtering is very similar to ordinary filtering, with the exception that a value range function needs to be evaluated in addition to each multiplication between filter coefficients and signal values. The value range function can easily be evaluated in parallel for each element, and controls how properties other than the Euclidean distance (e.g. the image intensity) between the elements should be taken into consideration during the filtering. Viola et al. (2003) implemented different nonlinear filters, such as median filters and bilateral filters, using the high-level shading language (HLSL), while Xu and Pattanaik (2005) used the GPU to accelerate a Monte Carlo approach to noise reduction based on bilateral filtering. Instead of denoising a video sequence frame by frame, Langs and Biedermann (2007) performed 3D bilateral filtering with the OpenGL shading language (GLSL). A GPU accelerated Kd-tree framework for general nonlinear filtering, such as bilateral filtering and non-local means, was derived by Adams et al. (2009). Zheng et al. (2011) discussed performance tuning for denoising filters and achieved a speedup of 20-40% for bilateral filtering. Howison (2010) made a comparison of CUDA accelerated bilateral filtering and anisotropic diffusion for 3D denoising of biomedical datasets. For similar denoising results, the bilateral approach was about five times faster. This is mainly explained by the fact that anisotropic diffusion is an iterative algorithm, while bilateral filtering is a direct approach.

Adaptive filtering is performed by weighting filter responses of filters sensitive to structures in different directions. The weights are calculated through a local structure tensor, to avoid perpendicular smoothing of edges and lines. The tensor itself is estimated by combining filter responses from another set of filters (Knutsson, 1989; Knutsson et al., 2011). Efficient implementations of both adaptive and bilateral filtering thus mainly require an optimized implementation of filtering, which has been covered in Section 2.1. Just as Langs and Biedermann (2007), Malm et al. (2007) applied 3D denoising to a video sequence, but instead used an adaptive filtering approach. Beyond 2D and 3D image denoising, true 4D denoising (where the algorithm simultaneously considers several volumes over time) on the GPU has recently been achieved by Eklund et al. (2011b), who applied it to CT data, and Broxvall et al. (2012), who applied it to echocardiography (ultrasound imaging of the heart). Both reports used an adaptive filtering approach for its computational efficiency.

The main idea of non-local means is to average information that is close in a feature space, regardless of the spatial distance. The most demanding operation is to calculate the similarity between the current neighborhood and many others. The current neighborhood can be put into the cached constant memory and a search area can be copied into shared memory, to greatly reduce the number of global memory accesses. Due to the computational complexity of the non-local means algorithm, several GPU implementations have been proposed. A first example is the CUDA SDK implementation by Nvidia (Kharlamov and Podlozhnyuk, 2007), followed by denoising of sensor range information (Huhle et al., 2010), ultrasound volumes (Hu and Hou, 2011) and CT data (Huang et al., 2009; Wu et al., 2011a; Zheng et al., 2011; Yu et al., 2011b).

Total variation models (Rudin et al., 1992) are another alternative to image denoising. Pock et al. (2008) developed a CUDA implementation in order to speed up the solution of such models, both in 2D and 3D. Wavelet based image denoising is also a popular choice, and has been accelerated by Su and Xu (2010). Gomersall et al. (2011) instead performed deconvolution of ultrasound data with a spatially varying blur function, also requiring GPU acceleration.

Real-time image denoising can often be achieved using a GPU and has consequently been the focus of several papers. Bertalmio et al. (2004) used a GPU implementation of anisotropic diffusion to produce depth of field effects in real-time. Another example is the work by Winnemöller et al. (2006), who applied anisotropic diffusion in real-time for video editing. Felsberg (2008) also performed real-time image denoising by anisotropic diffusion. Chen et al. (2007) used a GPU to perform bilateral filtering on images of the resolution 1920×1080 at 30 Hz. Similar work has been presented by Yang et al. (2009), who developed a parallel implementation of O(1) bilateral filtering (for which the processing time is invariant to the filter kernel size). Jiang et al. (2011) used a bilateral filter for speckle reduction of ultrasound images and achieved a framerate of 30 Hz. Goossens et al. (2010) reported a real-time implementation of the computationally demanding non-local means algorithm, for denoising of video sequences. de Fontes et al. (2011) used a similar approach for real-time denoising of ultrasound data, showing that images of resolution 1080×864 pixels could be denoised in 60 ms. Bruce and Butte (2013) recently used a FFT approach for deconvolution of 3D confocal microscopy data in real-time, to reduce blurring, such that the user can adjust experimental parameters on the fly.

4. Modalities

4.1. CT

Computed tomography (CT) has a long history in the medical imaging domain, originating with the seminal work on X-ray

imaging by Röntgen. The major advantages of CT are its high spatial resolution and short scanning time. A modern CT scanner can obtain a high resolution volume, e.g. voxels of the size $0.75 \times 0.75 \times 0.75$ mm, of the thorax in about 0.5 s. High resolution 4D imaging of a beating heart is therefore possible. Dual-energy CT scanners enable multi-variate imaging, but can also be used to further accelerate scanning. The major disadvantage of CT is the ionizing radiation to which each subject is exposed. Efforts to mitigate this are hobbled by the increase in noise that accompanies any reduction in the amount of radiation used. It is therefore common to apply image denoising techniques or more advanced reconstruction algorithms to maintain an adequate image quality.

While data from an MR scanner can often be reconstructed by applying an inverse FFT, reconstruction of CT data is much more complex. The detectors of a CT scanner measure a line integral of the X-ray attenuation through the scanned object. To generate an image from these data requires backprojection, which is equivalent to an inverse Radon transform. Filtered backprojection (FBP) (Feldkamp et al., 1984) is the most popular algorithm for reconstructing CT data. Iterative methods require both forward and backprojection operations, and are therefore much more time consuming. Significant effort has therefore been expended to investigate how GPUs can accelerate CT reconstruction.

GPU-based filtered back projection is today normally conducted by mapping filtered projections to voxels in parallel, using the interpolation properties of the texture memory. Iterative methods also require a forward projection to be performed on the GPU. This can be achieved by for each projection line accumulating the contribution of each intersecting voxel in the volume, again using the texture memory (this operation is very similar to ray casting used for direct volume rendering). The same basic ideas can be used for reconstruction of PET and SPECT data, as long as the samples are collected as sinograms and not in list-mode format. If a linear approximation of the imaging system is sufficient, the reconstruction can more generally be performed by inverting a linear transform, which can be represented as a product between a large matrix and a long vector with all the samples. As the transformation matrix is often sparse, the CUSPARSE library can be applied for such operations. Another alternative is the library CUSP,[5] which can easily solve a sparse linear system $Ax = b$ with the conjugate gradient method on a GPU, according to

```
cusp :: krylov :: cg(A, x, b).
```

One of the earliest examples of GPU accelerated CT reconstruction is the work by Cabral et al. (1994), who used a Silicon Graphics (SGI) workstation to speedup FBP reconstruction of cone-beam CT data. Mueller and Yagel (2000) used the OpenGL programming language for faster reconstruction with the simultaneous algebraic reconstruction technique (SART), which is an iterative method (Andersen and Kak, 1984). In 2004, cheap commodity PC graphics hardware could be used instead of an expensive SGI system (Xu and Mueller, 2004). One year later, the work was extended to three popular reconstruction algorithms; FBP, SART, and expectation maximization (Xu and Mueller, 2005). Kole and Beekman (2006) used an ordered subset convex reconstruction algorithm. In 2007, the work by Xu and colleagues was further extended to real-time reconstruction (Xu and Mueller, 2007), such that interactive 3D image generation could be performed for flat-panel X-ray detectors and C-arm gantries. During the same year, Scherl et al. (2007) were one of the first to use the CUDA programming language for FBP, while Yan et al. (2008) still used OpenGL and Sharp et al. (2007) used the Brook programming environment. Zhao et al. (2009) focused on the problem of reconstruction of large volumes (e.g. 1024^3 voxels) on GPU hardware, a problem also mentioned by Mueller and Xu (2006). Both solved the problem by dividing the data into smaller blocks. Since these early works, the field of GPU-accelerated CT reconstruction has exploded (Noel et al., 2010; Okitsu et al., 2010; Xu et al., 2010a; Vintache et al., 2010). Recent examples include several papers on using multiple GPUs to further accelerate reconstruction (Jang et al., 2009; Liria et al., 2012; Zhang et al., 2012; Zhu et al., 2012), including the incorporation of scatter correction into the reconstruction process (Sisniega et al., 2011), SART based reconstruction combined with motion compensation (Pang et al., 2011), CT reconstruction with OpenCL (Zhang et al., 2009) and an up-to-date and thorough comparison of different hardware implementations (CPU, GPU, FPGA and the Cell Broadband Architecture) of FBP (Scherl et al., 2012).

To decrease the amount of radiation further than what is presently attainable with iterative reconstruction methods, more advanced reconstruction algorithms and sampling patterns are necessary. The field of compressed sensing (Donoho, 2006) provides several new solutions for this purpose. These approaches typically represent the data as a long vector and minimize an L0 or L1 norm through iterative algorithms involving many matrix operations. The CUDA libraries CUBLAS and CUSPARSE can speedup these dense and sparse matrix–matrix multiplications and matrix–vector multiplications, but a general problem for CT data is how to fit the huge system matrix in the relatively small global memory. Jia et al. (2010) implemented the total variation (TV) method (Rudin et al., 1992) on the GPU in order to more efficiently complete the reconstruction. Compared to the ordinary FBP algorithm, the TV method can handle undersampled and noisy projection data, which can be employed to lower the radiation dose. The reconstruction was performed by minimizing an energy functional, which can be written in terms of matrix algebra. To reduce the memory requirements, Jia et al. (2010) reformulated the functional such that it could be evaluated without the need of large matrix operations. The work was extended by Tian et al. (2011b), who developed a GPU-accelerated version of edge-preserving TV in order to minimize unwanted smoothing of edges. The memory problem with large matrices was solved by instead using approaches normally used for FBP. Stsepankou et al. (2012) took advantage of a GPU for TV regularization-based reconstruction to investigate the robustness of such methods, and also used the FBP approach for forward and backward projection. Tight frame regularization (Daubechies et al., 2003), based on wavelets, is another alternative to TV but is also computationally demanding (Jia et al., 2011). Compared to the TV approach (Jia et al., 2010), the TF regularization yielded sharper edges and a slightly shorter processing time. CUBLAS was used for simple operations involving vectors while FBP methods were used for the matrix operations. Another alternative that works with a lower imaging dose is to use exact algorithms instead of approximate ones. Yan et al. (2010) made a GPU implementation of the exact Katsevich algorithm for this purpose.

Four-dimensional CT is much more computationally demanding than 3D CT, for two reasons. First, the reconstruction needs to be applied to several volumes (e.g. 10–20). Second, the amount of radiation is much higher than for 3D CT, thus increasing the need for low-dose techniques. Tian et al. (2011a) combined GPU acceleration with regularization based on temporal non-local means, inspired by the non-local means denoising algorithm (Buades et al., 2005), to take advantage of the temporal correlation between the volumes. Badea et al. (2011) used a GPU to accelerate the reconstruction of 4D data from a micro-CT scanner, for cardiac imaging of mice. Similar work was presented by Johnston et al. (2012), who also applied five-dimensional bilateral filtering to further suppress noise levels.

A great deal of effort has clearly been expended on the acceleration of iterative reconstruction algorithms, but FBP is still used with most commercial CT scanners (Pan et al., 2009). We believe

[5] http://cusplibrary.github.com.

that GPUs are one key to practically enable clinical use of iterative reconstruction algorithms in CT, but significant challenges remain (Beister et al., 2012). A major obstacle is that radiologists are accustomed to the appearance of images generated by filtered back projection, and learning how to interpret images from other reconstruction algorithms can be prohibitively time-consuming.

For references on GPU accelerated dose calculation and treatment plan optimization, see the review by Pratx and Xing (2011).

4.2. PET

CT is normally used for imaging anatomical structure, while positron emission tomography (PET) can be used to study organ function using radioactive tracers. CT is based on transmission, while PET is based on emission. Prior to the introduction of functional magnetic resonance imaging (fMRI), PET was the preferred modality for studying whole-brain function in humans. Compared to CT, PET has a much lower spatial resolution, but the reconstruction can still be costly as the main processing steps are again forward and backward projection.

Reconstruction of PET and SPECT sinograms can be accelerated by using techniques presented for CT data. Measurements acquired in list-mode format, however, cannot take advantage of the fast texture memory as the projection lines are not ordered. List-mode reconstruction must be performed in a line-driven manner, in contrast to a voxel-driven manner for sinograms. This can be accomplished by using each GPU thread to add the contribution of a projection to all voxels it intersects. The problem with this approach is that it is based on write instead of read operations, and write operations cannot be cached or efficiently interpolated through texture memory. In contrast, filtering in the spatial domain can be achieved through convolution or impulse summation (the difference being the order of the for-loops for elements and filter coefficients). Convolution requires many read operations (one per filter coefficient and image element) and a single write operation per element, while impulse summation instead requires many write operations per element. Just as for CT, most GPU implementations for PET and SPECT reconstruction do not utilize libraries such as CUBLAS or CUSPARSE, due to the large memory requirements of a matrix formulation.

Chidlow and Möller (2003) rather early implemented two expectation maximization (EM) algorithms, which are iterative, for emission tomography reconstruction on graphics hardware. Similar work was later done by Bai and Smith (2006) and Pratx et al., 2006. As with CT, the number of papers describing GPU accelerated PET reconstruction methods has seen tremendous growth in recent years. Pratx et al. (2009) focused on reconstruction of sparse projection data stored in list-mode format, commonly used for high-resolution, dynamic and time-of-flight (ToF) PET. The OpenGL implementation resulted in a speedup factor of 50. Two years later, an extended CUDA implementation for ToF PET increased the speedup to a factor 300 compared to a single-threaded CPU implementation (Cui et al., 2011a). Fast reconstruction of list-mode format data for OpenPET scanners was presented by Kinouchi et al. (2010). Barker et al. (2009) accelerated the most computationally demanding steps of MOLAR PET, which includes motion compensation to improve the image quality. Herraiz et al. (2011) made a CUDA implementation for iterative reconstruction of sinogram PET, which is much more common in commercial scanners than list-mode PET.

Several papers have focused on new algorithms for image reconstruction, including a sparse matrix approach (Zhou and Qi, 2011), a particle filter approach (Yu et al., 2011a) and a spherical basis approach (Cabello et al., 2012). Image quality can be further improved by including scatter correction in the reconstruction algorithm (Wirth et al., 2009; Barker et al., 2009; Kim and Ye, 2011; Magdics et al., 2011), which adds an additional computational load. Other examples illustrating the use of GPUs with PET are online modeling of the detector response for high-resolution PET (Pratx and Levin, 2011) and the use of a spatially varying point spread function (Cui et al., 2011b) or system response (Ha et al., 2012) in place of a constant one.

4.3. SPECT

Single photon emission computed tomography (SPECT) can also be used to study the function of organs. The primary difference in terms of reconstruction comes from the fact that SPECT tracers emit individual gamma photons, compared to PET's use of tracers that (after positron annihilation) emit two photons traveling in the opposite direction. Wen et al. (2004) were the first to use a GPU to speedup filtered back projection reconstruction of SPECT images. This work was then extended to an iterative EM algorithm by Wang et al. (2005), similar to work done by Beenhouwer et al. (2006), Vetter and Westermann (2008) and Pedemonte et al. (2010). As is the case for CT and PET, GPU accelerated SPECT reconstruction has recently experienced a substantial increase in popularity. Beenhouwer and Pieters (2011) used the GPU for Monte Carlo simulation of fan beam geometry in order to improve the reconstruction quality. Other examples include multipinhole SPECT (Alhassen et al., 2011), content adaptive mesh models (for reconstruction based on non-uniform sampling) (Massanes and Brankov, 2012) and high resolution SPECT (Deprez et al., 2010; Miller et al., 2011).

4.4. MRI

Magnetic resonance imaging (MRI) makes it possible to create images with high spatial resolution, without the hazard of ionizing radiation. The object to be scanned is placed in a strong magnetic field, which causes the spin of its nuclei to align either parallel or anti-parallel to the field. Radio frequency (RF) pulses are then applied to excite the nuclei, which emit energy when they return to their original state. The most common nucleus used in MRI is the hydrogen proton. The key in MRI is that it is sensitive to many physical parameters. Different tissue types have different relaxation rates and densities of protons, and the measured MRI signal can be made sensitive to temperature, diffusion of water, blood flow, etc. Thus MRI, itself, can be sub-categorized into multiple imaging modalities. Here we discuss MRI generally, and then in two subsequent sections consider diffusion-weighted imaging (DWI) and functional magnetic resonance imaging (fMRI).

If a standard scanning protocol is used, where data is collected in k-space (the frequency domain) on a regular Cartesian grid, the images can be reconstructed by applying an inverse FFT. Acceleration of such reconstructions using the GPU was proposed early on (Sumanaweera and Liu, 2005), and can now be easily performed using the CUFFT library. Alternately, the sampling of data along a radial or a spiral pattern has some benefits over the Cartesian pattern, including faster acquisitions and a lower sensitivity to artefacts. However, this has the disadvantage that the straightforward application of the FFT for reconstruction is no longer possible. Reconstruction by gridding, i.e. interpolation in k-space from a non-Cartesian grid to a Cartesian one followed by the standard inverse FFT, was proposed using the GPU by Schiwietz et al. (2006) and further explored by Sorensen et al. (2008). Interpolation of the k-space samples can be performed in parallel through a radial point approach or a Cartesian point approach, both having advantages and disadvantages for a GPU implementation. The radial point approach uses few read operations, but many write operations (often through slow atomic functions to avoid write hazards). The Cartesian point approach uses few write

operations, but has an uneven workload distribution between threads. Guo et al. (2009) used the GPU for acceleration of the popular propeller MRI sequence, which is very robust to motion artefacts. Similar work was recently presented by Yang et al. (2012), who also proposed a reverse gridding approach in order to reduce the number of write conflicts. The reverse gridding approach resulted in a speedup of a factor 7.5, compared to conventional gridding.

From a signal processing perspective, interpolation in k-space is often considered to be a sub-optimal approach, as a small error in the frequency domain will affect the entire image. A better and more general solution to reconstruct non-Cartesian sampled data is to instead define a reconstruction error in terms of matrix algebra. This solution also enables the use of additional constraints, e.g. inserted as Lagrangian multipliers. If an L2 norm is used, the best solution can be found by solving a large linear system. However, for a single image of 512×512 pixels, the size of the system becomes $262{,}144 \times 262{,}144$, which may prohibit direct solutions. Fortunately the system is often sparse, allowing the CUSPARSE library to be leveraged to develop efficient solutions. For example, iterative Krylov subspace solvers like the conjugate gradient (CG) method can be used. Stone et al. (2008) made a GPU implementation of a CG algorithm to enable practical use of the anatomically constrained reconstruction algorithm presented by Haldar et al. (2008). Two years later, this work was extended to include correction for field inhomogeneities (Zhuo et al., 2010b), multi-GPU support (Zhuo et al., 2010c) and a regularization term for spatial smoothness (Zhuo et al., 2010a). The work was recently released as a freely available toolkit (Wu et al., 2011b). Johnsson et al. (2010) used the GPU for improved fat–water reconstruction, and also employed an iterative reconstruction algorithm.

The major drawback with MRI compared to CT is that the acquisitions take significantly more time. The basic physics of MRI makes this difficult to circumvent. To increase the sampling rate beyond non-Cartesian sampling patterns, parallel imaging and compressed sensing can be applied. For these more advanced scanning protocols, the reconstruction becomes even more complex and time consuming, which prohibits clinical use. In parallel imaging, several receiver coils collect data simultaneously, but during a shorter time frame compared to a single coil. To reconstruct data from the multiple measurements, image domain approaches (SENSE) (Pruessmann et al., 1999) use a matrix inversion specific to each set of aliased pixels (as opposed to the entire image). Solving the linear system can be performed by combining Cholesky factorization with forward and backward substitution. The factorization for each set of pixels can be achieved by cooperation between a number of threads, while the substitution step needs to be performed by a single thread. Hansen et al. (2008) described reconstruction of regular SENSE on the GPU. This work was then extended to data collected with temporal SENSE (Roujul et al., 2009) and radial sampling with SENSE (Sorensen et al., 2009). In compressed sensing (Donoho, 2006; Lustig et al., 2007), the idea is to sample the data in a sparse basis. The reconstruction error is normally defined with an L1 or L0 norm, which prohibits direct solutions through matrix inversion. Instead, numerical optimization algorithms must be applied. Just as for iterative reconstruction algorithms with L2 norm minimization, reconstruction algorithms for compressed sensing utilize the GPU through operations such as matrix–matrix multiplications, matrix–vector multiplications and multidimensional FFTs. As previously mentioned, there exist highly optimized libraries for all these operations. Kim et al. (2011) were one of the first to use a GPU to speedup reconstruction of MRI data collected through compressed sensing. The implementation utilized the CUFFT library and was used for contrast-enhanced MR angiography, which requires high spatial and temporal resolution. Knoll et al. (2011) reconstructed radially sampled data by combining total generalized variation regularization with a GPU implementation. The work was recently released as an open source library (Freiberger et al., 2013) which internally uses CUBLAS. Similar work has been presented by Nam et al. (2013), who used both CUFFT and CUBLAS and focused on whole-heart imaging with 3D radial sampling. The compressed sensing approach was shown to outperform the simpler gridding method. Smith et al. (2012) instead used a Split Bregman solver and were able to reconstruct breast images of the size 4096×4096 in about 8 s. Murphy et al. (2012) combined compressed sensing with parallel imaging, for 32 channels the multi-GPU implementation was about 50 times faster compared to a multi-threaded C++ implementation.

In general, it can take a long time before new algorithms, or scanning protocols, are employed in a clinical context. Vasanawala et al. (2011) recently reported a successful combination of parallel imaging, compressed sensing and a GPU implementation in this realm. Kowalik et al. (2012) also used GPU accelerated reconstruction of parallel imaging in order to monitor the flow of cardiac output in real-time. These examples show that a GPU can shorten the time from an initial idea to real world application.

The GPU has not only been used for faster reconstruction. Yang et al. (2011) used a GPU to accelerate correction of geometric distortions in echo planar imaging (EPI) data, while Knoll et al. (2010) instead used total variation based regularization to reduce undersampling artefacts in radial MR angiography. Deng et al. (2011) accelerated the design of multidimensional radio frequency pulses, which can be used to improve high-field imaging. Another important issue for high-field imaging is shimming, i.e. the removal of inhomogeneities in the magnetic field, which can also be computationally costly (Chi et al., 2010).

4.5. fMRI

In addition to high resolution anatomical scans, magnetic resonance imaging enables non-invasive functional studies of the human brain in action. Since its inception (Belliveau et al., 1991; Ogawa et al., 1992), functional magnetic resonance imaging (fMRI) has had a tremendous impact on brain research. fMRI data is 4D, as volumes of the brain repeatedly are collected during an experiment. A single fMRI dataset can for example be of the size $64 \times 64 \times 30 \times 300$ time voxels. Stronger magnetic fields and more advanced sampling techniques will improve the spatial and temporal resolution of fMRI in the near future (Feinberg and Yacoub, 2012), further increasing the computational load.

GPUs can easily be used to process fMRI data, as most existing algorithms for brain activity detection perform exactly the same calculations for each voxel. Each GPU thread can thus independently process the time series of one voxel (e.g. for voxel-wise *t*-tests). Motion correction of fMRI data can be accelerated by applying GPU-based registration to each volume. Registration is also required for multi-subject fMRI, where each subject is aligned to a brain template. Spatial smoothing of each fMRI volume is normally performed to reduce noise and to use information from neighboring voxels. This procedure can take advantage of GPU-based filtering.

Goddard et al. (1997) were among the first to propose analysis of fMRI data on parallel platforms while Bagarinao et al. (2003) used a PC cluster for real-time fMRI analysis. One of the first applications of GPUs for fMRI was presented by Gembris et al. (2011), who used the GPU to speedup calculation of correlation coefficients. To investigate functional connectivity in subjects at rest (Biswal et al., 1995), a common approach is to calculate the correlation between the time series of a reference voxel and all other voxels in the volume. If the fMRI dataset contains 20,000 brain voxels, this requires the calculation of 20,000 correlations for just one

reference voxel. The complete correlation matrix is thus of the size 20,000 × 20,000. Several other papers have used the GPU to accelerate functional connectivity algorithms. Some examples are spatial regularization through high-dimensional Markov random fields (Liu et al., 2010), non-local fMRI analysis (Eklund et al., 2012), brain network analysis (Wu et al., 2010), interactive connectivity analysis (Eklund et al., 2011d), and a multi-platform comparison of functional network calculation (Rao et al., 2011).

Application of the GPU to conventional task-based fMRI analysis is less common. To the best of our knowledge, the first work addressing GPU acceleration of the general linear model (GLM) for fMRI analysis (Friston et al., 1995) was presented by (Eklund et al., 2012c). The paper also describes how to perform the various preprocessing steps, such as motion correction, on the GPU. Compared to the commonly used statistical parametric mapping (SPM) software,[6] the GPU implementation achieved a speedup of about 290 for the preprocessing and 87,000 for the statistical analysis. These large speedups are not only due to a highly optimized GPU implementation, but also reflect the fact that SPM often stores intermediate results to file.

A general drawback of parametric statistics, such as GLM based fMRI analysis (Friston et al., 1995), is that it relies on several assumptions. One example is that the errors of the GLM are assumed to be normally distributed and independent. Another is that the test statistics are assumed to follow a parametric null distribution. In contrast, the non-parametric random permutation test only relies on the assumption that the samples can be exchanged under the null hypothesis. To empirically estimate the null distribution of the test statistics, the data is resampled, or permuted, and analyzed a large number of times, normally 1,000–10,000. These large numbers of permutations have tended to discourage routine use of non-parametric fMRI analysis (Nichols and Holmes, 2001). Random permutations can be hard to conduct on a GPU, due to the irregular memory access patterns. For 4D data, however, it is possible to take advantage of the fact that the same permutation normally is applied to each voxel time series (to keep the spatial correlation structure). GPUs can thus rather easily accelerate these non-parametric measures for fMRI, which has been suggested to be more reliable than commonly used parametric approaches (Eklund et al., 2011a). A random permutation test can also be used to derive the null distribution of more complicated multi-variate test statistics (Eklund et al., 2013). Bayesian methods are another alternative to parametric ones, and are quite popular for fMRI analysis (Woolrich, 2012). As with non-parametric approaches, Bayesian fMRI analysis can also be costly, with fully Bayesian modeling of a single fMRI dataset taking up to several days (Woolrich et al., 2004). Bayesian statistics often require random draws from different distributions, especially for inference through Markov chain Monte Carlo (MCMC) techniques, which can be performed directly on the GPU with random number generation through the CURAND library. Ferreira da Silva (2011) developed a CUDA implementation of a Bayesian multilevel model for fMRI analysis, and applied MCMC in parallel for each voxel time series.

GPU acceleration is also helping to spur a growing number of studies focused on large-scale fMRI datasets. An example is the recent work of Eklund et al. (2012a), who used their previous GPU implementations (Eklund et al., 2011a; Eklund et al., 2012c) to analyze 1484 freely available rest datasets from the functional connectomes project (Biswal et al., 2010) to contrast the validity of parametric versus non-parametric fMRI analyses. Compared to the SPM software, the processing time was reduced from about 100 years to 10 days. There are several other large fMRI studies that could benefit from using the computational power of the GPU. A first example is the recent work by Thyreau et al. (2012), where activity data from 1326 subjects were analyzed. Another is the work by Biswal et al. (2010), who described the collection and functional connectivity analysis of rest data from 1414 subjects. Recently, (Boubela et al., 2012) made an interface between the statistical program R and three popular programs for fMRI analysis (SPM, AFNI,[7] FSL[8]). Through this interface, the preprocessing could be performed with standard tools and further calculations could be accelerated with CUDA. As a demonstration, independent component analysis (ICA) was applied to 300 datasets from the functional connectomes project (Biswal et al., 2010).

GPUs have not yet, to our knowledge, been used for real-time fMRI (Cox et al., 1995; Weiskopf et al., 2003; deCharms, 2008; LaConte, 2011), which involves analyzing the data as it is collected from the subject in the MR scanner. Real-time fMRI is enabling a number of novel applications such as brain computer interfaces (BCI) (Eklund et al., 2010b) and interactive brain mapping. It is also possible for the subject to monitor his or her own brain activity, for example to learn how to suppress chronic pain (deCharms et al., 2005). A challenge in this context is that fMRI data has to be analyzed in real-time, and here the GPU may be used to open up groundbreaking new applications.

4.6. DTI

Diffusion-weighted imaging enables the measurement of water diffusion rates in different directions. Through diffusion tensor imaging (DTI), the measurements can be combined into a tensor to represent the orientation of the flow in each voxel. The diffusion tensor is hence analogous to the local structure tensor (Knutsson, 1989; Knutsson et al., 2011), often used in image processing to represent the orientation of signal energy. Like fMRI data, DTI data is also 4D, but in this case the fourth dimension represents direction instead of time. While it is sufficient to measure diffusion in 6 directions to obtain an estimate of the diffusion tensor, more directions, e.g. 16–128, generally yield more refined estimates. A single DTI dataset collected with an isotropic voxel size of 2 mm and 128 directions contains in the order of 128 × 128 × 60 × 128 samples. The interested reader is referred to the excellent recent review by Bihan and Johansen-Berg (2012) for an extensive overview of diffusion MRI.

The resulting diffusion tensor can be used for example to track fiber bundles in the brain, since water preferentially diffuses along the axons. A fiber trajectory is started as a seed, which propagates along the main direction of the diffusion tensor in each voxel. A GPU can perform fiber tracking in parallel, as each thread can independently process one seed. Before the tracking is started, a GPU can also speedup the decomposition of the diffusion tensor in each voxel into eigenvalues and eigenvectors. Just as for fMRI, GPUs can additionally perform motion correction of DTI data in a short amount of time.

To the best of our knowledge, the use of GPU acceleration for the processing of DTI data is not widespread. In contrast, it has become commonplace to use the GPU for visualization of DTI data, due to the difficulty associated with visualizing a tensor field. Kondratieva et al. (2005) therefore used the GPU for particle tracing, which can be used to generate stream lines or tubes representing white matter tracts. McGraw and Nadar (2007) took advantage of a GPU for stochastic brain connectivity mapping, by using a Bayesian approach and the GLSL programming language. During the same year, Jeong et al. (2007) made a GPU implementation of a Hamilton–Jacobi solver, to compute measures of white matter

[6] http://www.fil.ion.ucl.ac.uk/spm/.

[7] http://afni.nimh.nih.gov/afni/.

[8] http://fsl.fmrib.ox.ac.uk/fsl/.

connectivity from DTI data. Mittmann et al. (2008) used the GPU for fast tracking of fiber bundles in the human brain, with similar work described by Köhn et al. (2009) and van Aart et al., 2011. Real-time interactive tracking through CUDA was some years later presented by Mittmann et al. (2011). A framerate of about 20 Hz was achieved for 1,000 concurrent seeds. Recently, Hernandez et al. (2012) accelerated a Bayesian approach to estimation of fiber orientations and uncertainties, used in the FMRIB software library (FSL). In their implementation, each GPU thread performs a Levenberg–Marquardt optimization and a posterior estimation through MCMC. As mentioned for Bayesian fMRI analysis, the random number generation for MCMC was performed directly on the GPU through CURAND. The processing time was reduced from 24 h to about 17 min, making it possible to experiment with different parameter settings much more efficiently.

4.7. Ultrasound

Ultrasound is unique among medical imaging modalities for its affordability (Bierig and Jones, 2009), portability (Thomenius, 2009) and high temporal resolution (e.g. 20–30 Hz). To perform real-time processing of ultrasound data, however, places high demands on the computer hardware. GPUs play a key role in enabling the clinical use of many algorithms, but a limiting factor for real-time applications is the relatively low memory bandwidth between the CPU and the GPU. Fortunately, modern GPUs can concurrently perform calculations and transfer data. Elnokrashy et al. (2009) described a low-cost system for reconstruction and display of 4D ultrasound data. Similar frameworks were proposed by Kim et al. (2010) and Brattain and Howe (2011). Recently, So et al. (2011) made a comparison between CPUs and GPUs for ultrasound systems, in terms of power efficiency and cost effectiveness. The conclusion was that a hybrid CPU-GPU system performed best.

Chang et al. (2009) used a GPU for acceleration of color doppler ultrasound, to provide information about blood flow in real-time. Another clinical application was presented by Shi et al. (2010). In their work, the GPU was used for automatic time gain compensation (TGC) in real-time. Rather than manually adjusting the TGC, the focus can be on the diagnosis and increasing patient throughput. A strain tensor estimate of the left ventricular motion can be beneficial for diagnostic purposes, but the required calculations are often too demanding for clinical purposes. Kiss et al. (2009) therefore developed a GPU implementation of block matching for 3D echocardiography. Similar work was presented by Rosenzweig et al. (2011), who accelerated acoustic radiation force impulse imaging, which can be used to estimate small displacements of soft tissue. Another clinical application of ultrasound is minimally invasive thermal therapies, such as high intensity focused ultrasound. In order to monitor the temperature change in real-time, a GPU accelerated system, based on speckle tracking, was proposed by Liu and Ebbini (2010). Hlawitschka et al. (2011) instead used a GPU to accelerate calculations of how the beam energy is spread in the tissue.

As a final processing step before the images are displayed, ultrasound data need to be converted from polar to Cartesian coordinates in a process known as scan conversion. Wang et al. (2011a) described how to perform this operation on the GPU. To further increase the temporal resolution, more advanced scanning protocols such as plane wave imaging (PWI) and synthetic aperture imaging (SAI) can also be used. The drawback of these methods is an increase in processing time for image formation. Yiu et al. (2010) therefore proposed the use of GPUs to practically enable PWI and SAI. The GPU can also be used to accelerate the simulation of ultrasound data (Reichi et al., 2009; Shams et al., 2011), e.g. by using CT data (Kutter et al., 2009). These simulations can be used, for example, for navigation and examination training.

While GPU implementations for CT and MRI have focused primarily on image reconstruction, the ultrasound modality has taken advantage of the GPU for a wide range of applications. Two major reasons for this are probably the portability of ultrasound systems and their high temporal resolution. As mentioned in the previous sections, GPUs have also been used to perform denoising and segmentation of ultrasound data in real-time.

4.8. Optical imaging

Optical imaging involves techniques such as optical tomography (OT), diffuse optical tomography (DOT) and optical coherence tomography (OCT), which for example can be used to study the human eye. The main advantage of OCT compared to other modalities in medical imaging is its micrometer resolution, combined with millimeter penetration depth. Micro-CT can also yield a very high resolution, but in contrast to OCT requires ionizing radiation. Frequency-domain OCT (FDOCT) enables faster scanning, due to its superior signal to noise ratio. However, FDOCT significantly increases the computational burden of the image reconstruction, as an inverse Fourier transform needs to be applied to each line sampled. Like ultrasound, optical imaging can output the acquired data directly, but real-time signal processing requires sophisticated computer hardware. Optical imaging can take advantage of GPUs for more advanced imaging, real-time reconstruction and visualization, and also for additional processing such as image denoising.

The first example of GPU-based optical imaging is the work by Watanabe and Itagaki (2009), who used a GPU to perform the inverse one-dimensional FFT required for FDOCT in real-time. The result was an imaging system with a frame rate of 28 Hz, for a resolution of 2048×1000 pixels. During the same year, Vinegoni et al. (2009) instead used a GPU to accelerate reconstruction of optical projection tomography (OPT) data. Just as for CT data, filtered back projection was used to create a 3D volume from the acquired images. Zhang and Kang (2010b) extended the work by Watanabe and Itagaki (2009), by also performing spectral resampling on the GPU, making it possible to use a GPU together with a regular nonlinear-k FDOCT system. The processing could be performed in real-time, such that volumes could be reconstructed and visualized with ray-casting at a frame rate of 10 Hz. Work reported by van der Jeught et al. (2010) and Rasakanthan et al. (2011) also include a GPU for resampling and inverse FFTs of OCT data in real-time. The resampling was eventually improved by using zero-padding and FFTs (Watanabe et al., 2010), instead of simple linear interpolation. The work by Zhang and Kang (2010b) was extended from uniform to non-uniform FFTs (Zhang and Kang, 2010a) and finally to full range FDOCT (Zhang and Kang, 2011), to reduce ghosting from long surgical instruments. Real-time reconstruction of dispersion encoded full-range FDOCT, which has an increased depth range, was achieved by Wang et al. (2012). Recently, GPUs have also been used to enable high-dynamic range OPT (Fei et al., 2012) and Doppler OCT (Huang et al., 2012).

Belanger et al. (2010) managed to reconstruct DOT data in real-time, by using a pseudo-inverse approach. Prakash et al. (2010) instead reconstructed DOT data by an iterative approach based on Levenberg–Marquardt optimization, by taking advantage of the CULA library for matrix algebra. Modeling of the light transportation in DOT is computationally demanding, and was therefore accelerated by Schweiger (2011) who used a finite element method. This modeling may be incorporated as a component in an iterative nonlinear reconstruction algorithm.

Other examples of GPU accelerated optical imaging include artifact removal for FDOCT (Huang and Kang, 2012; Watanabe, 2012) and automated recovery of the center of rotation (Dong et al.,

2012), which can be used instead of reference phantoms or a manual centering procedure.

4.9. Microscopy

Microscopy enables high resolution imaging of objects that cannot be seen with the naked eye, and can broadly be divided into optical microscopy, electron microscopy and scanning probe microscopy. A major challenge of high resolution microscopy is that the images acquired can be huge, often exceeding resolutions of $10{,}000 \times 10{,}000$ pixels. This makes it cumbersome to apply standard image processing operations. GPUs can be used for microscopy to speedup reconstruction algorithms and other common operations such as image denoising and segmentation. A general difficulty is how to fit the large images or volumes into the relatively small global memory.

Castano-Diez et al. (2007) used the Cg programming language to implement the iterative reconstruction algorithms SIRT and SART (also used for CT) for reconstruction of electron microscopy data. This work was extended by Xu et al. (2010b), who improved the performance significantly to enable 3D reconstruction. Compensation for long-objects was also added to reduce reconstruction errors. Several papers have focused on reconstruction of data collected with electron cryomicroscopy, used for example to study biological macromolecules. See the work by Schmeisser et al. (2009) for an overview of computing technologies in this realm. Tan et al. (2009) used CUDA for an iterative reconstruction approach, which in each iteration requires generating projections from the 3D model, classification and realignment of the particles and building of a 3D model. Tagare et al. (2010) instead used an adaptive expectation–maximization algorithm, which is more robust to noise, while Zhang et al. (2010) used an approach based on direct Fourier space reconstruction. The implementation was for example applied to a dataset consisting of 44,250 images of the resolution 880×880 pixels. FREALIGN is a popular software package for reconstruction and refinement of single particle structures, but suffers from long processing times. Li et al. (2010) made a GPU implementation of this package and achieved a 240 fold speedup with a system consisting of 8 GPUs.

Shimobaba et al. (2008) were one of the first to take advantage of a GPU for real-time reconstruction of digital holographic microscopy data. The hologram can be reconstructed through Fresnel diffraction, which can be performed through two forward FFTs and one inverse FFT. The system was able to reconstruct images of 512×512 pixels at 24 Hz. Orzo et al. (2010) presented similar work, but instead reconstructed volume data. Other examples of GPU accelerated microscopy include phase unwrapping and affine transformations for optical quadrature microscopy (Mistry et al., 2009), image analysis for localization-based super resolution microscopy for imaging of live cells (Quan et al., 2010) and reconstruction of volumes in tomographic diffractive microscopy (Bailleul et al., 2012).

5. Discussion

Thus far, we have covered the majority of past and the present works on GPU accelerated medical image processing. In this closing section, we shift our focus to the future and highlight some challenges and possibilities for this rapidly developing field.

5.1. Memory size and speed

The size of the global memory currently ranges from 1 GB (e.g. the Nvidia GTX 285) to 6 GB (e.g. the Nvidia GTX Titan) for consumer graphics cards and 4–8 GB for professional graphics cards. For a workstation, the amount of CPU memory can easily range from 4 GB to 128 GB. Six GB of global memory is normally sufficient for 2D image processing, but it can be too little for 3D image processing. Micro-CT scanners can today, for example, generate volumes of $2048 \times 2048 \times 2048$ voxels (Tommasini et al., 2012), which translates to a storage requirement of 34 GB using 32-bit floats. For 4D image processing, the memory requirements can be even more dire. In the recent work by Eklund et al. (2011b); Eklund et al., 2012b, a 4D CT heart dataset of the size $512 \times 512 \times 446 \times 20$ was denoised using a GPU. With the storage of 14 floating point filter responses for a dataset of this size, estimation of a 4D local structure tensor (Knutsson et al., 2011) theoretically requires 131 GB of memory. To perform the calculations with only 1.5 GB of global memory, it was necessary to process only small parts of the dataset at a time. Five dimensional MRI (Sigfridsson et al., 2007), where there are two time dimensions (one for the heart rhythm and one for the breathing rhythm), poses an even greater challenge.

Even if a modern GPU theoretically is much faster than a modern CPU, the performance difference can be erased by the small amount of global memory. For future GPU applications, it is therefore extremely important that the amount of global memory increases faster than the spatial and temporal resolution of medical imaging modalities. It is also crucial that (global) memory bandwidth improves significantly, as many contemporary algorithms are limited by this bottleneck as opposed to computational performance. This is especially true for CT reconstruction algorithms, which rely heavily on the interpolation capabilities of texture memory. Since several medical imaging modalities (CT, MRI, ultrasound) can generate 4D data, it would also be beneficial for future GPUs to support 4D interpolation in texture memory as well. Additionally, the number of registers per GPU thread must increase significantly, such that it becomes possible to, for example, process a 10×10 matrix in each pixel (e.g. for normalized convolution (Knutsson and Westin, 1993) or to estimate voxel specific auto regressive models in fMRI (Eklund et al., 2011a)).

5.2. Programming languages and libraries

The CUDA programming language has clearly made it much easier to program the GPU for general-purpose calculations. The main drawback of CUDA is that it only supports Nvidia GPUs. Here we believe that OpenCL (or any similar programming language) will play an important role for further GPGPU proliferation and development. Code written in CUDA and OpenCL can, however, yield different performance. Initially, CUDA enjoyed a significant performance advantage over OpenCL (Kong et al., 2010; Karimi et al., 2011), but this is diminishing as OpenCL compilers improve and now typically stands at no more than 20–30% (Fang et al., 2011).

The larger advantage offered by CUDA at present is Nvidia's commitment to supplying highly optimized libraries for use with their proprietary platform. However, these libraries have yet to match the breadth and refinement of existing CPU libraries. It is therefore important that new libraries continue to be produced and that existing ones are continuously improved. For example, the medical imaging community would greatly benefit from support for 4D FFT's in the CUFFT library and, as previously mentioned, support for (floating point) convolution in 2, 3 and 4 dimensions in the NPP library. CUBLAS includes support for large matrix inversions (more specifically, support for solving linear systems) but to our knowledge, there is no freely available library with support for small matrix inversions in each GPU thread. In contrast, such functions are commonplace in Intel's Performance Primitives library for the CPU.

For researchers that do not want to learn all the details of GPU programming, a very important development is the encapsulation of GPU accelerated versions of commonly used algorithms and the release of high-level programming interfaces. Mathworks recently added GPU support to the parallel computing toolbox in Matlab. Another option is to use the Matlab GPU interfaces Jacket[9] (Pryor et al., 2011; Larsen et al., 2011) or GPUmat.[10] A comparison between these three options was recently made by Zhang et al. (2011). For the C and Fortran programming languages, it is possible to use the PGI accelerator model (Wolfe, 2010) or the HMPP workbench compiler (Dolbeau et al., 2007) for the acceleration of existing code. The programmer adds directives, which tell the compiler which parts of the code to parallelize. As previously mentioned, Membarth et al. (2011) made a comparison between several of these frameworks, while Malik et al. (2012) compared CUDA, OpenCL, PGI and Jacket.

5.3. Debugging and optimization

In the early days of GPU programming, code debugging and optimization were extremely challenging. Today it is possible to both debug and optimize GPU code as a routine part of development. For Windows users, debugging and optimization can be performed through Nvidia Nsight and Microsoft Visual Studio. For Linux users, debugging is generally harder since at least two GPUs are required if one wishes to run the display manager during the debugging. With the release of CUDA 5.0, Nsight has been ported to the Eclipse IDE, simplifying development and profiling of GPU code on Linux and Mac platforms.

Even as optimization tools for GPU code have evolved, it can still be difficult to sustain optimal performance over generations of hardware. Code optimized for a certain graphics card (e.g. Nvidia GTX 480), in general has to be rewritten for each new graphics card from a later architecture (e.g. Nvidia GTX 680). The need to re-optimize existing code is a limiting factor which reduces the utilization of new hardware capabilities. This is especially true for researchers, who also have to write papers about their implementations and applications. A good approach can be to use functions from standard libraries as often as possible, and at each new hardware release simply wait for these libraries to be re-optimized. Another approach can be to use PGI or HMPP, instead of writing code in CUDA or OpenCL.

5.4. Possibilities

In terms of possibilities, GPUs can be used to solve a wide array of problems in the field of medical imaging. Many algorithms can become interactive, such that it is possible to experiment with many different parameter settings or initialization points. Alternately, the analysis of datasets of a magnitude previously considered unwieldy may now be tackled as a matter of routine. The most promising concept is perhaps that more advanced algorithms can be applied in a clinical context. CT scanners can for example use iterative reconstruction algorithms, instead of the old filtered back projection approach (Pan et al., 2009). Another trend is to combine different modalities during surgery. A high resolution MRI or CT volume can be collected in advance, and during surgery be combined with time-resolved ultrasound. The major obstacle is that real-time registration between the two modalities is required, which is very computationally demanding. As the size of datasets produced in the healthcare sector continues to surge, GPUs have in many cases become a de facto necessity for the processing and visualization of these datasets (Scholl et al., 2011). It remains an open question the extent to which entirely new categories of medical image processing algorithms may emerge, as investigators become accustomed to the greater processing power now commonly at their disposal on the desktop and in the cloud.

5.5. What about Intel?

Finally, we note some recent industry developments that could have a significant impact on the future of GPU-based medical image computing. The vast majority of the works described above were developed using CUDA for Nvidia GPUs. Since releasing CUDA in 2007, Nvidia has moved aggressively into the scientific computing domain with courses, conferences, documentation, development tools, and performance libraries intended to ease the transition from CPU to GPU programming. Despite the radically different architecture of the GPU and the host of additional programming constraints that come with it, academic scientists have risen to the challenge of completely redeveloping their algorithms and code bases to take maximum advantage of this new platform, as evidenced by the volume of work referenced here. Nonetheless, the substantial time investment required to port and optimize existing code bases for maximum GPU performance has remained beyond the means of many commercial high-performance software enterprises that simply cannot afford the associated costs. This cohort enjoyed progressive enhancements to the performance of their code bases for several decades with very little effort, simply by waiting for industry giant Intel to release the next in a long line of faster and more sophisticated CPUs. Yet Intel has remained largely absent from the new low-cost, massively parallel coprocessor market, allowing Nvidia to completely dominate this niche.

Intel recently released its first massively parallel coprocessor, the Xeon Phi 5110P, which contains 60 processor cores running at 1.0 GHz. Each core can execute four simultaneous hardware threads via hyper-threading and has a 512-bit vector unit capable of performing sixteen 32-bit or eight 64-bit floating point operations per clock cycle, resulting in a theoretical peak performance of one teraflops. The coprocessor comes with 8 GB of memory and plugs into the PCI-e bus of modern workstations like a GPU, with similar power requirements. While at the time of writing the device is too new for systematic comparisons with GPUs to have been published, initial commentary predicts its performance will fall just shy of Nvidia's latest Kepler series GPUs. However, in an apparent appeal to organizations that have not yet invested in GPU computing, Intels marketing material suggests that Phi users will be able to leverage its acceleration with minimal programming effort, by using the same suite of development tools and performance libraries already widely deployed for use with its multicore CPUs. It is not known how many in the community of medical image computing researchers and practitioners may belong to this group.

While software and hardware trends are not the primary focus of medical image computing, the ability to efficiently employ more sophisticated algorithms as faster technology emerges is a significant driving force in this field, largely precluding any kind of convergence in techniques or algorithms. With the hardware ecosystem likely to remain dynamic and heterogeneous for the foreseeable future, it is expected that research into new algorithms custom-designed for new or evolving platforms will remain a continuing area of focus.

Acknowledgement

We would like to acknowledge the helpful comments and suggestions of Cyrus Eierud. Daniel Forsberg was funded by the Swedish Research Council (Grant 2007-4786).

[9] http://www.accelereyes.com/.
[10] http://sourceforge.net/projects/gpumat/.

References

Abramov, A., Kulvicius, T., Wörgötter, F., Dellen, B., 2011. Real-time image segmentation on a GPU. Lecture Notes in Computer Science, Facing the Multi-core Challenge 6310, 131–142.

Adams, A., Gelfand, N., Dolson, J., Levoy, M., 2009. Gaussian KD-trees for fast high-dimensional filtering. In: Proceedings of ACM SIGGRAPH. (Article no. 21).

Adams, R., Bischof, L., 1994. Seeded region growing. IEEE Transactions on Pattern Analysis and Machine Intelligence 16, 641–647.

Alhassen, F., Sangtaek, K., Sayre, G., Bowen, J., Gould, R., Seo, Y., Kudrolli, H., Singh, B., Nagarkar, V., 2011. Ultrafast multipinhole single photon emission computed tomography iterative reconstruction using CUDA. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, pp. 2558–2559.

Andersen, A., Kak, A., 1984. Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm. Ultrasonic Imaging 6, 81–94.

Andersson, M., Wiklund, J., Knutsson, H., 1998. Sequential Filter Trees for Efficient 2D, 3D and 4D Orientation Estimation. Report LiTH-ISY-R-2070, Linköping University, Sweden.

Andersson, M., Wiklund, J., Knutsson, H., 1999. Filter networks. In: Proceedings of Signal and Image Processing (SIP), pp. 213–217.

Badea, C., Johnston, S., Qi, Y., Johnson, A., 2011. 4D micro-CT for cardiac and perfusion applications with view under sampling. Physics in Medicine and Biology 56, 3351–3369.

Bagarinao, E., Matsuo, K., Nakai, T., 2003. Real-time functional MRI using a PC cluster. Concepts in Magnetic Resonance 19B, 14–25.

Bai, B., Smith, A., 2006. Fast 3D iterative reconstruction of PET images using PC graphics hardware. In: IEEE Nuclear Science Symposium Conference Record, pp. 2787–2790.

Bailleul, J., Simon, B., Debailleul, M., Liu, H., Haeberle, O., 2012. GPU acceleration towards real-time image reconstruction in 3D tomographic diffractive microscopy. In: Proceedings of SPIE, Real-Time Image and Video Processing, pp. 843707.

Barker, W., Thada, S., Dieckmann, W., 2009. A GPU-accelerated implementation of the MOLAR PET reconstruction package. In: IEEE Nuclear Science Symposium Conference Record, pp. 4114–4119.

Beenhouwer, J.D., Pieters, B., R. Van de Walle, 2011. Fast GATE fan beam SPECT projector. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, pp. 4188–4191.

Beenhouwer, J.D., R. Van Holen, Vandenberghe, S., Staelens, S., D'Asseler, Y., Lemahieu, I., 2006. Graphics hardware accelerated reconstruction of SPECT with a slat collimated strip detector. In: International Conference on Image Processing, Computer Vision and Pattern Recognition (IPCV), pp. 451–457.

Beister, M., Kolditz, D., Kalender, W., 2012. Iterative reconstruction methods in X-ray CT. Physica Medica 28, 94–108.

Belanger, S., Abran, M., Intes, X., Casanova, C., Lesage, F., 2010. Real-time diffuse optical tomography based on structured illumination. Journal of Biomedical Optics 15, 016006.

Belliveau, J., Kennedy, D., McKinstry, R., Buchbinder, B., Weisskoff, R., Cohen, M., Vevea, J., Brady, T., Rosen, B., 1991. Functional mapping of the human visual cortex by magnetic resonance imaging. Science 254, 716–719.

Bertalmio, M., Fort, P., Sanchez-Crespo, D., 2004. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In: International Symposium on 3D Data Processing, Visualization and Transmission, pp. 767–773.

Beyer, J., Langer, C., Fritz, L., Hadwiger, M., Wolfsberger, S., Buhler, K., 2007. Interactive diffusion-based smoothing and segmentation of volumetric datasets on graphics hardware. Methods of Information in Medicine 46, 270–274.

Bierig, S., Jones, A., 2009. Accuracy and cost comparison of ultrasound versus alternative imaging modalities, including CT, MR, PET, and angiography. Journal of Diagnostic Medical Sonography 25, 138–144.

Bihan, D.L., Johansen-Berg, H., 2012. Diffusion MRI at 25: exploring brain tissue structure and function. NeuroImage 61, 324–341.

Biswal, B., Mennes, M., Zuo, X.N., Gohel, S., Kelly, C., Smith, S.M., Beckmann, C.F., Adelstein, J.S., Buckner, R.L., Colcombe, S., Dogonowski, A.M., Ernst, M., Fair, D., Hampson, M., Hoptman, M.J., Hyde, J.S., Kiviniemi, V.J., Kötter, R., Li, S.J., Lin, C.P., Lowe, M.J., Mackay, C., Madden, D.J., Madsen, K.H., Margulies, D.S., Mayberg, H.S., McMahon, K., Monk, C.S., Mostofsky, S.H., Nagel, B.J., Pekar, J.J., Peltier, S.J., Petersen, S.E., Riedl, V., Rombouts, S.A., Rypma, B., Schlaggar, B.L., Schmidt, S., Seidler, R.D., Siegle, G.J., Sorg, C., Teng, G.J., Veijola, J., Villringer, A., Walter, M., Wang, L., Weng, X.C., Whitfield-Gabrieli, S., Williamson, P., Windischberger, C., Zang, Y.F., Zhang, H.Y., Castellanos, F.X., Milham, M.P., 2010. Toward discovery science of human brain function. PNAS 107, 4734–4739.

Biswal, B., Yetkin, F., Haughton, V., Hyde, J., 1995. Functional connectivity in the motor cortex of resting state human brain using echo-planar MRI. Magnetic Resonance in Medicine 34, 537–541.

Borgefors, G., 1986. Distance transforms in digital images. Computer Vision, Graphics and Image Processing 34, 344–371.

Boubela, R., Huf, W., Kalcher, K., Sladky, R., Filzmoser, P., Pezawas, L., Kasper, S., Windischberger, C., Moser, E., 2012. A highly parallelized framework for computationally intensive MR data analysis. Magnetic Resonance Materials in Physics, Biology and Medicine 25, 313–320.

Brattain, L., Howe, R., 2011. Real-time 4D ultrasound mosaicing and visualization. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI) 6891, 105–112.

Brosch, T., Tam, R., 2009. A self-optimizing histogram algorithm for graphics card accelerated image registration. In: Medical Image Computing and Computer Assisted Intervention (MICCAI) Grid Workshop, pp. 35–44.

Brounstein, A., Hacihaliloglu, I., Guy, P., Hodgson, A., Abugharbieh, R., 2011. Towards real-time 3D US to CT bone image registration using phase and curvature feature based GMM matching. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI) 6891, 235–242.

Broxvall, M., Emilsson, K., Thunberg, P., 2012. Fast GPU based adaptive filtering of 4D echocardiography. IEEE Transactions on Medical Imaging 31, 1165–1172.

Bruce, M., Butte, M., 2013. Real-time GPU-based 3D deconvolution. Optics Express 21, 4766–4773.

Buades, A., Coll, B., Morel, J., 2005. A non-local algorithm for image denoising. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 60–65.

Cabello, J., Gillam, J., Rafecas, M., 2012. High performance 3D PET reconstruction using spherical basis functions on a polar grid. International Journal on Biomedical Imaging (Article ID 452910).

Cabral, B., Cam, N., Foran, J., 1994. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In: Symposium on Volume Visualization, pp. 91–98.

Cao, J., Che, M.C., Wu, X., Liang, J., 2009. GPU-aided directional image/video interpolation for real-time resolution upconversion. In: IEEE International workshop on Multimedia signal processing (MMSP), pp. 1–6.

Cao, T., Tang, K., Mohamed, A., Tan, T., 2010. Parallel banding algorithm to compute exact distance transform with the GPU. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 83–90.

Castano-Diez, D., Mueller, H., Frangakis, A., 2007. Implementation and performance evaluation of reconstruction algorithms on graphics processors. Journal of Structural Biology 157, 288–295.

Castillo, E., Castillo, R., White, B., Rojo, J., Guerrero, T., 2012. Least median of squares filtering of locally optimal point matches for compressible flow image registration. Physics in Medicine and Biology 57, 4827–4833.

Cates, J., Lefohn, A., Whitaker, R., 2004. GIST: an interactive, GPU-based level set segmentation tool for 3D medical images. Medical Image Analysis 8, 217–231.

Chang, L.W., Hsu, K.H., Li, P.C., 2009. Graphics processing unit-based high-frame-rate color doppler ultrasound processing. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 56, 1856–1860.

Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J., Skadron, K., 2008. A performance study of general-purpose applications on graphics processors using CUDA. Journal of Parallel and Distributed Computing 68, 1370–1380.

Chen, H., Samavati, F., Sosa, M., Mitchell, J., 2006. Sketch-based volumetric seeded region growing. In: Proceedings of Eurographics, pp. 123–129.

Chen, H., Samavati, F., Sousa, M., 2008. GPU-Based point radiation for interactive volume sculpting and segmentation. The Visual Computer 24, 689–698.

Chen, J., Paris, S., Durand, F., 2007. Real-time edge-aware image processing with the bilateral grid. In: ACM Transactions on Graphics, Proceedings of the SIGGRAPH Conference, 9p. (Article 103).

Chen, S., Qin, J., Xie, Y., Pang, W., Heng, P., 2009. CUDA-based acceleration and algorithm refinement for volume image registration. In: International Conference on Future BioMedical Information Engineering (FBIE), pp. 544–547.

Chi, J., Liu, F., Jin, J., Mason, D., Crozier, S., 2010. GPU accelerated FDTD solver and its application in MRI. In: IEEE International Conference on Engineering in Medicine and Biology Society (EMBS), pp. 3305–3308.

Chidlow, K., Möller, T., 2003. Rapid emission tomography reconstruction. In: Proceedings of the Eurographics/IEEE TVCG Workshop on Volume Graphics, pp. 15–26.

Colantoni, P., Boukala, N., Da-Rugna, J., 2003. Fast and accurate color image processing using 3D graphics cards. In: Vision Modeling and Visualization, pp. 383–390.

Collins, M., Xu, J., Grady, L., Singh, V., 2012. Random walks based multi-image segmentation: quasiconvexity results and GPU-based solutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1656–1663.

Cootes, T., Taylor, C., Cooper, D., Graham, J., 1995. Active shape models - their training and application. Computer Vision and Image Understanding 61, 38–59.

Coupe, P., Yger, P., Prima, S., Hellier, P., Kervrann, C., Barillot, C., 2008. An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images. IEEE Transactions on Medical Imaging 27, 425–441.

Cox, R.W., Jesmanowicz, A., Hyde, J.S., 1995. Real-time functional magnetic resonance imaging. Magnetic Resonance in Medicine 33, 230–236.

Cui, J., Pratx, G., Prevrhal, S., Levin, C., 2011a. Fully 3D list-mode time-of-flight PET image reconstruction on GPUs using CUDA. Medical Physics 38, 6775–6786.

Cui, J., Pratx, G., Prevrhal, S., Zhang, B., Shao, L., Levin, C., 2011b. Measurement-based spatially-varying point spread function for list-mode PET reconstruction on GPU. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, pp. 2593–2596.

Daubechies, I., Han, B., Ron, A., Shen, Z., 2003. Framelets: MRA-based constructions of wavelet frames. Applied and Computational Harmonic Analysis 14, 1–46.

deCharms, R.C., 2008. Applications of real-time fMRI. Nature Reviews Neuroscience 9, 720–729.

deCharms, R.C., Maeda, F., Glover, G.H., Ludlow, D., Pauly, J.M., Soneji, D., Gabrieli, J.D., Mackey, S.C., 2005. Control over brain activation and pain learned by using real-time functional MRI. PNAS 102, 18626–18631.

Deng, W., Yang, C., Stenger, V.A., 2011. Accelerated multidimensional radiofrequency pulse design for parallel transmission using concurrent

computation on multiple graphics processing units. Magnetic Resonance in Medicine 65, 363–369.
Deprez, K., R. Van Holen, Staelens, S., Vandenberghe, S., 2010. A high resolution scintillator based SPECT detector with digital pulse processing (SPECTatress). In: IEEE Nuclear Science Symposium Conference Record, pp. 3100–3104.
Digabel, H., Lantuejoul, C., 1977. Iterative algorithms. In: Actes du Second Symposium Europeen d ' Analyse Quantitative des Microstructures en Sciences des Materiaux, Biologie et Medecine, pp. 85–99.
Dolbeau, R., Bihan, S., Bodin, F., 2007. HMPP: A hybrid multi-core parallel programming environment. In: Workshop on General Purpose Processing on Graphics Processing Units.
Domas, G.P.S., Couturier, R., Bertaux, N., 2011. GPU implementation of a region based algorithm for large images segmentation. In: IEEE International Conference on Computer and Information Technology (CIT), pp. 291–298.
Dong, D., Zhu, S., Qin, C., Kumar, V., Stein, J., Oehler, S., Savakis, C., Tian, J., Ripoll, J., 2012. Automated recovery of the center of rotation in optical projection tomography in the presence of scattering. IEEE Transactions on Information Technology in Biomedicine. http://dx.doi.org/10.1109/TITB.2012.2219588.
Donoho, D., 2006. Compressed sensing. IEEE Transactions on Information Theory 52, 1289–1306.
Dorgham, O., Laycock, S., Fisher, M., 2012. GPU accelerated generation of digitally reconstructed radiographs for 2-D/3-D image registration. IEEE Transactions on Biomedical Engineering 59, 2594–2603.
van Dortmont, M., van de Wetering, H., Telea, A., 2006. Skeletonization and distance transforms of 3D volumes using graphics hardware. Lecture Notes in Computer Science, Discrete Geometry for Computer Imagery 4245, 617–629.
van Aart, E., Sepasian, N., Jalba, A., Vilanova, A., 2011. CUDA-accelerated geodesic ray-tracing for fiber tracking. International Journal of Biomedical Imaging. Article ID 698908.
Eidheim, O., Skjermo, J., Aurdal, L., 2005. Real-time analysis of ultrasound images using GPU. Proceedings of the 19th International Congress and Exhibition on Computer Assisted Radiology and Surgery 1281, 284–289.
Eklund, A., Andersson, M., Josephson, C., Johannesson, M., Knutsson, H., 2012a. Does parametric fMRI analysis with SPM yield valid results? - An empirical study of 1484 rest datasets. NeuroImage 61, 565–578.
Eklund, A., Andersson, M., Knutsson, H., 2010a. Phase based volume registration using CUDA. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 658–661.
Eklund, A., Andersson, M., Knutsson, H., 2011a. Fast random permutation tests enable objective evaluation of methods for single subject fMRI analysis. International Journal of Biomedical Imaging (Article ID 627947).
Eklund, A., Andersson, M., Knutsson, H., 2011b. True 4D image denoising on the GPU. International Journal of Biomedical Imaging (Article ID 952819).
Eklund, A., Andersson, M., Knutsson, H., 2012b. 4D medical image processing with CUDA. In: Nvidia GPU Technology Conference. <http://nvidia.fullviewmedia.com/gtc2012/0516-A8-S0017.html>.
Eklund, A., Andersson, M., Knutsson, H., 2012c. fMRI analysis on the GPU - possibilities and challenges. Computer Methods and Programs in Biomedicine 105, 145–161.
Eklund, A., Andersson, M., Knutsson, H., 2012. A functional connectivity inspired approach to non-local fMRI analysis. In: IEEE International Conference on Image Processing (ICIP). pp. 1245–1248. http://dx.doi.org/10.1109/ICIP.2012.6467092.
Eklund, A., Andersson, M., Ohlsson, H., Ynnerman, A., Knutsson, H., 2010b. A brain computer interface for communication using real-time fMRI. In: Proceedings of International Conference on Pattern Recognition (ICPR), pp. 3665–3669.
Eklund, A., Björnsdotter, M., Stelzer, J., LaConte, S., 2013. Searchlight goes GPU – fast multi-voxel pattern analysis of fMRI data. In: International Society for Magnetic Resonance in Medicine (ISMRM).
Eklund, A., Forsberg, D., Andersson, M., Knutsson, H., 2011c. Using the local phase of the magnitude of the local structure tensor for image registration. Proceedings of the Scandinavian Conference on Image Analysis SCIA, vol. 6688. Lecture Notes in Computer Science, pp. 414–423.
Eklund, A., Friman, O., Andersson, M., Knutsson, H., 2011d. A GPU accelerated interactive interface for exploratory functional connectivity analysis of fMRI data. In: IEEE International Conference on Image Processing (ICIP), pp. 1621–1624.
Elad, M., 2002. On the origin of the bilateral filter and ways to improve it. IEEE Transactions on Image Processing 11, 1141–1151.
Elnokrashy, A., Elmalky, A., Hosny, T., Ellah, M., Megawer, A., Elsebai, A., Youssef, A.B., Kadah, Y.M., 2009. GPU-based reconstruction and display for 4D ultrasound data. In: IEEE International Ultrasonics Symposium, pp. 189–192.
Erdt, M., Raspe, M., Suehling, M., 2008. Automatic hepatic vessel segmentation using graphics hardware. Lecture Notes in Computer Science, Medical Imaging and Augmented Reality 5128, 403–412.
Fang, J., Varbanescu, A., Sips, H., 2011. A comprehensive performance comparison of CUDA and OpenCL. In: International Conference on Parallel Processing (ICPP), pp. 216–225.
Farber, R., 2011. CUDA Application Design and Development. Morgan Kaufmann, ISBN:978-0-12-388426-8.
Fei, P., Yu, Z., Wang, X., Lu, P., Fu, Y., He, Z., Xiong, J., Huang, Y., 2012. High dynamic range optical projection tomography (HDR-OPT). Optics Express 20, 8824–8836.
Feinberg, D., Yacoub, E., 2012. The rapid development of high speed, resolution and precision in fMRI. NeuroImage 62, 720–725.
Feldkamp, L., Davis, L., Kress, J., 1984. Practical cone-beam algorithm. Journal of the Optical Society of America A 1, 612–619.
Felsberg, M., 2008. On the relation between anisotropic diffusion and iterated adaptive filtering. In: Lecture notes in Computer Science, DAGM Symposium Mustererkennung, pp. 436–445.
Ferreira da Silva, A., 2011. A Bayesian multilevel model for fMRI data analysis. Computer Methods and Programs in Biomedicine 102, 238–252.
Fialka, O., Cadik, M., 2006. FFT and convolution performance in image filtering on GPU. In: Tenth International Conference on Information Visualization, pp. 609–614.
Fischer, I., Gotsman, C., 2006. Fast approximations of high-order Voronoi diagrams and distance transforms on the GPU. Journal of Graphics, GPU and Game Tools 11, 39–60.
Fleet, D., Jepson, A., 1990. Computation of component image velocity from local phase information. International Journal of Computer Vision 5, 77–104.
Fluck, O., Aharon, S., Cremers, D., Rousson, M., 2006. GPU histogram computation. In: International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH.
Fluck, O., Vetter, C., Wein, W., Kamen, A., Preim, B., Westermann, R., 2011. A survey of medical image registration on graphics hardware. Computer Methods and Programs in Biomedicine 104, e45–e57.
de Fontes, F.P.X., Barroso, G.A., Hellier, P., 2011. Real-time ultrasound image denoising. Journal of Real-Time Image Processing 6, 15–22.
Forsberg, D., Eklund, A., Andersson, M., Knutsson, H., 2011. Phase-based non-rigid 3D image registration - from minutes to seconds using CUDA. In: Joint MICCAI Workshop on High Performance and Distributed Computing for Medical Imaging.
Freiberger, M., Knoll, F., Bredies, K., Scharfetter, H., Stollberger, R., 2013. The AGILE library for image reconstruction in biomedical sciences using graphics card hardware acceleration. Computing in Science and Engineering 15, 34–44.
Friston, K., Holmes, A., Worsley, K., Poline, J., Frith, C., Frackowiak, R., 1995. Statistical parametric maps in functional imaging: a general linear approach. Human Brain Mapping 2, 189–210.
Fulkerson, B., Soatto, S., 2010. Really quick shift: image segmentation on a GPU. In: Proceedings of the Workshop on Computer Vision Using GPUs.
Gao, G., Penney, G., Ma, Y., Gogin, N., Cathier, P., Arujuna, A., Morton, G., Caulfield, D., Gill, J., Rinaldi, C., Hancock, J., Redwood, S., Thomas, M., Razavi, R., Gijsbers, G., Rhode, K., 2012. Registration of 3D trans-esophageal echocardiography to X-ray fluoroscopy using image-based probe tracking. Medical Image Analysis 16, 38–49.
Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V., 2008. Parallel computing experiences with CUDA. IEEE Micro 28, 13–27.
Gembris, D., Neeb, M., Gipp, M., Kugel, A., Männer, R., 2011. Correlation analysis on GPU systems using NVIDIA's CUDA. Journal of Real-Time Image Processing 6, 275–280.
Goddard, N., Hood, G., Cohen, J., Eddy, W., Genovese, C., Noll, D., Nystrom, L., 1997. Online analysis of functional MRI datasets on parallel platforms. Journal of Supercomputing 11, 295–318.
Gomersall, H., Hodgson, D., Prager, R., Kingsbury, N., Treece, G., Gee, A., 2011. Efficient implementation of spatially-varying 3-D ultrasound deconvolution. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control 58, 234–238.
Gomez-Luna, J., Gonzalez-Linares, J., Benavides, J., Guil, N., 2012. An optimized approach to histogram computation on GPU. Machine Vision and Applications. http://dx.doi.org/10.1007/s00138-012-0443-3.
Goossens, B., Luong, H., Aelterman, J., Pizurica, A., Philips, W., 2010. A GPU-Accelerated real-time NLMeans algorithm for denoising color video sequences. In: Proceedings of 12th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), vol. 6475, Lecture Notes in Computer Science, pp. 46–57.
Grady, L., Schiwietz, T., Aharon, S., Westermann, R., 2005. Random walks for interactive organ segmentation in two and three dimensions: implementation and validation. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI) 3750, 773–780.
Granlund, G., Knutsson, H., 1995. Signal Processing for Computer Vision. Kluwer Academic Publishers, ISBN:0-7923-9530-1.
Guo, H., Dai, J., He, Y., 2009. GPU acceleration of propeller MRI using CUDA. In: International Conference on Bioinformatics and Biomedical Engineering (ICBBE), pp. 1–4.
Ha, L., Prastawa, M., Gerig, G., Gilmore, J., Silva, C., Joshi, S., 2010. Image registration driven by combined probabilistic and geometric descriptors. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI) 6362, 602–609.
Ha, S., Ispiryan, M., Matej, S., Mueller, K., 2012. GPU-based spatially variant SR kernel modeling and projections in 3D DIRECT TOF PET reconstruction. In: IEEE Medical Imaging Conference.
Hadwiger, M., Hauser, H., Möller, T., 2003. Quality issues of hardware-accelerated high-quality filtering on PC graphics hardware. In: Proceedings of WSCG, pp. 213–220.
Hadwiger, M., Langer, C., Scharsach, H., Buhler, K., 2004. State of the art report 2004 on GPU-based segmentation. Report TR-VRVis-2004-017. VRVis Research Center, Vienna, Austria.
Hadwiger, M., Theussl, T., Hauser, H., Gröller, E., 2001. Hardware-accelerated high-quality filtering on PC hardware. In: Workshop on Vision, Modelling, and Visualization (VMV), pp. 105–112.
Hadwiger, M., Viola, I., Hauser, H., 2002. Fast and flexible high-quality texture filtering with tiled high-resolution filters. In: Workshop on Vision, Modelling, and Visualization (VMV), pp. 155–162.

Hagan, A., Zhao, Y., 2009. Parallel 3D image segmentation of large datasets on a GPU cluster. Lecture Notes in Computer Science, Advances in Visual Computing 5876, 960–969.
Haldar, J., Hernando, D., Song, S.K., Liang, Z., 2008. Anatomically constrained reconstruction from noisy data. Magnetic Resonance in Medicine 59, 810–818.
Han, X., Hibbard, L., Willcut, V., 2009. GPU-accelerated, gradient-free MI deformable registration for atlas-based MR brain image segmentation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR), pp. 141–148.
Han, X., Hibbard, L., Willcut, V., 2010. An efficient inverse-consistent diffeomorphic image registration method for prostate adaptive radiotherapy. Lecture Notes in Computer Science, Prostate Cancer Imaging. Computer-Aided Diagnosis, Prognosis and Intervention 6367, 34–41.
Hansen, M., Atkinson, D., Sörensen, T., 2008. Cartesian SENSE and $k-t$ SENSE reconstruction using commodity graphics hardware. Magnetic Resonance in Medicine 59, 463–468.
Hastreiter, P., Ertl, T., 1998. Integrated registration and visualization of medical image data. In: Computer Graphics International, pp. 78–85.
He, Z., Kuester, F., 2006. GPU-Based active contour segmentation using gradient vector flow. Lecture Notes in Computer Science, Advances in Visual Computing 4291, 191–201.
Heimann, T., Meinzer, H.P., 2009. Statistical shape models for 3D medical image segmentation: a review. Medical Image Analysis 13, 543–563.
Hemmendorff, M., Andersson, M., Kronander, T., Knutsson, H., 2002. Phase-based multidimensional volume registration. IEEE Transactions on Medical Imaging 21, 1536–1543.
Hernandez, M., Guerrero, G., Cecilia, J., Garcia, J., 2012. Accelerating fibre orientation estimation from diffusion weighted magnetic resonance imaging using GPUs. In: Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 622–626.
Herraiz, J., Espana, S., Cabido, R., Montemayor, A., Desco, M., Vaquero, J., Udias, J., 2011. GPU-Based fast iterative reconstruction of fully 3-D PET sinograms. IEEE Transactions Nuclear Science 58, 2257–2263.
Hlawitschka, M., McGough, R., Ferrara, K., Kruse, D., 2011. Fast ultrasound beam prediction for linear and regular two-dimensional arrays. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control 58, 2001–2012.
Hoff, K., Culver, T., Keyser, J., Lin, M., Manocha, D., 1999. Fast computation of generalized Voronoi diagrams using graphics hardware. In: SIGGRAPH, pp. 277–285.
Hong, J., Wang, M., 2004. High speed processing of biomedical images using programmable GPU. In: IEEE International Conference on Image Processing (ICIP), pp. 2455–2458.
Hopf, M., Ertl, T., 1999. Accelerating 3D convolution using graphics hardware. In: IEEE Visualization Conference, pp. 471–475.
Hopf, M., Ertl, T., 2000. Hardware accelerated wavelet transformations. In: Proceedings EG/IEEE TCVG Symposium on Visualization, pp. 93–103.
Howison, M., 2010. Comparing GPU Implementations of Bilateral and Anisotropic Diffusion Filters for 3D Biomedical Datasets. Report LBNL-3425E. Lawrence Berkeley National Laboratory, Berkeley, CA, USA.
Hsieh, H., Tai, W., 2005. A simple GPU based approach for 3D Voronoi diagram construction and visualization. Simulation Modelling Practice and Theory 13, 681–692.
Hu, S., Hou, W., 2011. Denosing 3D ultrasound images by non-local means accelerated by GPU. In: Proceedings of International Conference on Intelligent Computation and Bio-Medical Instrumentation, pp. 43–45.
Huang, K., Zhang, D., Wang, K., 2009. Non-local means denoising algorithm accelerated by GPU. Proceedings of SPIE, 7497.
Huang, T., Tang, Y., Ju, S., 2011. Accelerating image registration of MRI by GPU-based parallel computation. Magnetic Resonance Imaging 29, 712–716.
Huang, Y., Kang, J., 2012. Real-time reference A-line subtraction and saturation artifact removal using graphics processing unit for high-frame-rate Fourier-domain optical coherence tomography video imaging. Optical Engineering 51, 073203.
Huang, Y., Liu, X., Kang, J., 2012. Real-time 3D and 4D Fourier domain Doppler optical coherence tomography based on dual graphics processing units. Biomedical Optics Express 3, 2162–2174.
Huang, Y., Tong, T., Liu, W., Fan, Y., Feng, H., Li, C., 2010. Accelerated diffeomorphic non-rigid image registration with CUDA based on demons algorithm. In: International Conference on Bioinformatics and Biomedical Engineering (ICBBE), pp. 1–4.
Huhle, B., Schairer, T., Jenke, P., Strasser, W., 2010. Fusion of range and color images for denoising and resolution enhancement with a non-local filter. Computer Vision and Image Understanding 114, 1336–1345.
Hunt, B., 1973. The application of constrained least squares estimation to image restoration by digital computer. IEEE Transactions on Computers C-22, 805–812.
James, G., 2001. Operations for hardware-accelerated procedural texture animation. In: Game Programming Gems 2, Charles River Media, pp. 497–509.
Jang, B., Kaeli, D., Do, S., Pien, H., 2009. Multi GPU implementation of iterative tomographic reconstruction algorithms. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 185–188. http://dx.doi.org/10.1109/ISBI.2009.5193014.
Jeong, W., Fletcher, P., Tao, R., Whitaker, R., 2007. Interactive visualization of volumetric white matter connectivity in DTMRI using a parallel-hardware Hamilton–Jacobi solver. IEEE Transactions on Visualization and Computer Graphics 13, 1480–1487, http://dx.doi.org/10.1109/TVCG.2007.70571.
Jeong, W.K., Beyer, J., Hadwiger, M., Vazquez, A., Pfister, H., Whitaker, R., 2009. Scalable and interactive segmentation and visualization of neural processes in EM datasets. IEEE Transactions on Visualization and Computer Graphics 15, 1505–1514.
Jia, X., Dong, B., Lou, Y., Jiang, S., 2011. GPU-based iterative cone-beam CT reconstruction using tight frame regularization. Physics in Medicine and Biology 56, 3787–3807.
Jia, X., Li, R., Song, W., Jiang, S., 2010. GPU-based fast cone-beam CT reconstruction from undersampled and noisy projection data via total variation. Medical Physics 37, 1757–1760.
Jiang, F., Shi, D., Liu, D., 2011. Fast adaptive ultrasound speckle reduction with bilateral filter on CUDA. In: International Conference on Bioinformatics and Biomedical Engineering (ICBBE), pp. 1–4.
Johnsson, D., Narayan, S., Flask, C., Wilson, D., 2010. Improved fat-water reconstruction algorithm with graphics hardware acceleration. Journal of Magnetic Resonance Imaging 31, 457–465.
Johnston, S., Johnson, A., Badea, C., 2012. Temporal and spectral imaging with micro-CT. Medical Physics 39, 4943–4958.
Jones, M., Barentsen, J., Sramek, M., 2006. 3D distance fields: a survey of techniques and applications. IEEE Transactions on Visualization and Computer Graphics 12, 581–599.
Karas, P., Svoboda, D., 2011. Convolution of large 3D images on GPU and its decomposition. EURASIP Journal on Advances in Signal Processing 2011, 120.
Karimi, K., Dickson, N., Hamze, F., 2011. A performance comparison of CUDA and OpenCL. (arXiv:1005.2581v3).
Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. International Journal of Computer Vision 1, 321–331.
Kauffmann, C., Piche, N., 2008. Cellular automaton for ultra-fast watershed transform on GPU. In: IEEE International Conference on Pattern Recognition (ICPR), pp. 1–4.
Kauffmann, C., Piche, N., 2010. Seeded ND medical image segmentation by cellular automaton on GPU. International Journal of Computer Assisted Radiology and Surgery 5, 251–262.
Kharlamov, A., Podlozhnyuk, V., 2007. Image denoising technical report. In: Nvidia.
Kim, D., Trzasko, J., Smelyanskiy, M., Haider, C., Dubey, P., Manduca, A., 2011. High-performance 3D compressive sensing MRI reconstruction using many-core architectures. International Journal of Biomedical Imaging (Article ID 473128).
Kim, K.S., Ye, J.C., 2011. Fully 3D iterative scatter-corrected OSEM for HRRT PET using a GPU. Physics in Medicine and Biology 56, 4991–5009.
Kim, S., Sohn, H., Chang, J., Song, T., Yoo, Y., 2010. A PC-based fully-programmable medical ultrasound imaging system using a graphics processing unit. In: IEEE Ultrasonics Symposium, pp. 314–317.
Kinouchi, S., Yamaya, T., Yoshida, E., Tashima, H., Kudo, H., Suga, M., 2010. GPU implementation of list-mode DRAMA for real-time OpenPET image reconstruction. In: IEEE Nuclear Science Symposium Conference Record, pp. 2273–2276.
Kirk, D., Hwu, W., 2010. Programming Massively Parallel Processors, A Hands-on Approach. Morgan Kaufmann (ISBN:978-0-12-381472-2).
Kiss, G., Nielsen, E., Orderud, F., Torp, H., 2009. Performance optimization of block matching in 3D echocardiography. In: IEEE Ultrasonics Symposium, pp. 1403–1406.
Knoll, F., Bredies, K., Pock, T., Stollberger, R., 2011. Second order total generalized variation (TGV) for MRI. Magnetic Resonance in Medicine 65, 480–491.
Knoll, F., Unger, M., Diwoky, C., Clason, C., Pock, T., Stollberger, R., 2010. Fast reduction of undersampling artifacts in radial MR angiography with 3D total variation on graphics hardware. Magnetic Resonance Materials in Physics, Biology and Medicine 23, 103–114.
Knutsson, H., 1989. Representing local structure using tensors. In: Scandinavian Conference on Image Analysis (SCIA), pp. 244–251.
Knutsson, H., Andersson, M., 2005. Morphons: segmentation using elastic canvas and paint on priors. In: IEEE International Conference on Image Processing (ICIP), pp. 1226–1229.
Knutsson, H., Andersson, M., Wiklund, J., 1999. Advanced filter design. In: Scandinavian Conference on Image Analysis (SCIA), pp. 185–193.
Knutsson, H., Westin, C.F., 1993. Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), pp. 515–523.
Knutsson, H., Westin, C.F., Andersson, M., 2011. Representing local structure using tensors II. In: Proceedings of the Scandinavian Conference on Image Analysis (SCIA), vol. 6688, Lecture Notes in Computer Science, pp. 545–556.
Knutsson, H., Wilson, R., Granlund, G.H., 1983. Anisotropic non-stationary image estimation and its applications - Part I: Restoration of noisy images. IEEE Transactions on Communications 31, 388–397.
Köhn, A., Klein, J., Weiler, F., Peitgen, H.O., 2009. A GPU-based fiber tracking framework using geometry shaders. In: Proceedings of SPIE Medical Imaging, pp. 72611J-7–2611J-10.
Kole, J., Beekman, F., 2006. Evaluation of accelerated iterative X-ray CT image reconstruction using floating point graphics hardware. Physics in Medicine and Biology 51, 875–889.
Kondratieva, P., Kruger, J., Westermann, R., 2005. The application of GPU particle tracing to diffusion tensor field visualization. In: IEEE Visualization, pp. 73–78.
Kong, J., Dimitrov, M., Yang, Y., Liynage, J., Cao, L., Staples, J., Mantor, M., Zhou, H., 2010. Accelerating Matlab image processing toolbox functions on GPUs. In: 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU), pp. 75–85.

Körbes, A., Vitor, G., Lotufo, R., Ferreira, J., 2011. Advances on watershed processing on GPU architecture. Lecture Notes in Computer Science, Mathematical Morphology and its Applications to Image and Signal Processing 6671, 260–271.

Kowalik, G.T., Steeden, J.A., Pandya, B., Odille, F., Atkinson, D., Taylor, A., Muthurangu, V., 2012. Real-time flow with fast GPU reconstruction for continuous assessment of cardiac output. Journal of Magnetic Resonance Imaging 14, W63.

Kraus, M., Eissele, M., Strengert, M., 2007. GPU-Based edge-directed image interpolation. Lecture Notes in Computer Science, Image Analysis 4522, 532–541.

Kutter, O., Shams, R., Navab, N., 2009. Visualization and GPU-accelerated simulation of medical ultrasound from CT images. Computer Methods and Programs in Biomedicine 94, 250–266.

LaConte, S.M., 2011. Decoding fMRI brain states in real-time. NeuroImage 56, 440–454.

Langs, A., Biedermann, M., 2007. Filtering video volumes using the graphics hardware. In: Proceedings of Scandinavian Conference on Image Analysis (SCIA), vol. 4522, Lecture Notes in Computer Science, pp. 878–887.

Larsen, T., Pryor, G., Malcolm, J., 2011. Jacket: GPU powered Matlab acceleration. In: GPU Computing Gems Jade Edition. Morgan Kauffman (Chapter 28).

Lee, D., Dinov, I., Dong, B., Gutman, B., Yanovsky, I., Toga, A., 2012. CUDA optimization strategies for compute- and memory-bound neuroimaging algorithms. Computer Methods and Programs in Biomedicine 106, 175–187.

Lee, J.S., 1980. Digital image enhancement and noise filtering by use of local statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence 2, 165–168.

Lee, V.W., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A.D., Satish, N., Smelyanskiy, M., Chennupaty, S., Hammarlund, P., Singhal, R., Dubey, P., 2010. Debunking the 100x GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU. In: Proceedings of the 37th Annual International Symposium on Computer Architectures (ICSA), pp. 451–460.

Lefohn, A., Cates, J., Whitaker, R., 2003. Interactive, GPU-based level sets for 3D segmentation. In: Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI), vol. 2878, Lecture Notes in Computer Science, pp. 564–572.

Lefohn, A., Kniss, J., Hansen, C., Whitaker, R., 2003b. Interactive deformation and visualization of level set surfaces using graphics hardware. In: IEEE Visualization, pp. 75–82.

Li, X., Grigorieff, N., Cheng, Y., 2010. GPU-enabled FREALIGN: accelerating single particle 3D reconstruction and refinement in Fourier space on graphics processors. Journal of Structural Biology 172, 407–412.

Lindholm, S., Kronander, J., 2011. Accounting for uncertainty in medical data: A CUDA implementation of normalized convolution. In: SIGRAD.

Liria, E., Higuero, D., Abella, M., de Molina, C., Desco, M., 2012. Exploiting parallelism in a X-ray tomography reconstruction algorithm on hybrid multi-GPU and multi-core platforms. In: IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), pp. 867–868.

Liu, D., Ebbini, E.S., 2010. Real-time 2-D temperature imaging using ultrasound. IEEE Transactions on Biomedical Engineering 57, 12–16.

Liu, W., Zhu, P., Anderson, J., Yurgelun-Todd, D., Fletcher, P., 2010. Spatial regularization of functional connectivity using high-dimensional markov random fields. In: Proceedings of the 13th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), vol. 6362, Lecture Notes in Computer Science, pp. 363–370.

Liu, X., Iyengar, S., Rittscher, J., 2012. Monitoring cardiomyocyte motion in real time through image registration and time series analysis. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 1308–1311.

Ljung, P., Lundström, C., Ynnerman, A., 2006. Multiresolution interblock interpolation in direct volume rendering. In: Proceedings of Eurographics/IEEE Symposium on Visualization, pp. 259–266.

Lustig, M., Donoho, D., Pauly, J., 2007. Sparse MRI: the application of compressed sensing for rapid MR imaging. Magnetic Resonance in Medicine 58, 1182–1195.

Magdics, M., Szirmay-Kalos, L., Toth, B., Legrady, D., Cserkaszky, A., Balkay, L., Domonkos, B., Volgyes, D., Patay, G., Major, P., Lantos, J., Bukki, T., 2011. Performance evaluation of scatter modeling of the GPU-based Tera-Tomo 3D PET reconstruction. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, pp. 4086–4088.

Mahmoudi, S., Lecron, F., Manneback, P., Benjelloun, M., Mahmoudi, S., 2010. GPU-Based segmentation of cervical vertebra in X-Ray images. In: IEEE International Conference on Cluster Computing Workshops and Posters, pp. 1–8.

Malik, M., Li, T., Sharif, U., Shahid, R., El-Ghazawi, T., Newby, G., 2012. Productivity of GPUs under different programming paradigms. Concurrency and Computation: Practice and Experience 24, 179–191.

Malm, H., Oskarsson, M., Warrant, E., Clarberg, P., Hasselgren, J., Lejdfors, C., 2007. Adaptive enhancement and noise reduction in very low light-level video. In: IEEE International Conference on Computer Vision (ICCV), pp. 1–8.

Man, D., Uda, K., Ueyama, H., Ito, Y., Nakano, K., 2011. Implementations of a parallel algorithm for computing Euclidean distance map in multicore processors and GPUs. International Journal of Networking and Computing 1, 260–276.

Massanes, F., Brankov, J., 2012. Parallel computation of a SPECT projection operator for a content adaptative mesh model. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 58–61.

McGraw, T., Nadar, M., 2007. Stochastic DT-MRI connectivity mapping on the GPU. IEEE Transactions on Visualization and Computer Graphics 13, 1504–1511.

Mellor, M., Brady, M., 2004. Non-rigid multimodal image registration using local phase. Lecture Notes in Computer Science, Medical Image Computing and Computer Assisted Intervention (MICCAI) 32, 789–796.

Mellor, M., Brady, M., 2005. Phase mutual information as similarity measure for registration. Medical Image Analysis 9, 330–343.

Membarth, R., Hannig, F., Teich, J., Korner, M., Eckert, W., 2011. Frameworks for GPU accelerators: a comprehensive evaluation using 2D/3D image registration. In: IEEE Symposium on Application Specific Processors (SASP), pp. 78–81.

Miller, B., Van Holen, R., Barrett, H., Furenlid, L., 2011. A system calibration and fast iterative reconstruction method for next-generation SPECT imagers. In: IEEE Nuclear Science Symposium and Medical Imaging Conference, pp. 3548–3553.

Mistry, P., Braganza, S., Kaeli, D., Leeser, M., 2009. Accelerating phase unwrapping and affine transformations for optical quadrature microscopy using CUDA. In: Proceedings of Workshop on General Purpose Processing on Graphics Processing Units, pp. 28–37.

Mitchell, J., Ansari, M., Hart, E., 2003. Advanced image processing with DirectX 9 pixel shaders. In: Engel, W. (Ed.), ShaderX: Shader Programming Tips and Tricks with DirectX 9.0. Wordware Publishing, Inc..

Mittmann, A., Comunello, E., von Wangenheim, A., 2008. Diffusion tensor fiber tracking on graphics processing units. Computerized Medical Imaging and Graphics 32, 521–530.

Mittmann, A., Nobrega, T., Comunello, E., Pinto, J., Dellani, P., Stoeter, P., von Wangenheim, A., 2011. Performing real-time interactive fiber tracking. Journal of Digital Imaging 24, 339–351.

Modat, M., Ridgway, G., Taylor, Z., Lehmann, M., Barnes, J., Hawkes, D., Fox, N., Ourselin, S., 2010. Fast free-form deformation using graphics processing units. Computer Methods and Programs in Biomedicine 98, 278–284.

Moreland, K., Angel, E., 2003. The FFT on a GPU. In: Proceedings of the ACM Siggraph/Eurographics Conference on Graphics Hardware, pp. 112–119.

Mueller, K., Xu, F., 2006. Practical considerations for GPU-accelerated CT. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 1184–1187.

Mueller, K., Yagel, R., 2000. Rapid 3-D cone-beam reconstruction with the simultaneous algebraic reconstruction technique (SART) using 2-D texture mapping hardware. IEEE Transactions on Medical Imaging 19, 1227–1237.

Murphy, M., Alley, M., Demmel, J., Keutzer, K., Vasanawala, S., Lustig, M., 2012. Fast l1-SPIRiT compressed sensing parallel imaging MRI: scalable parallel implementation and clinically feasible runtime. IEEE Transactions on Medical Imaging 31, 1250–1262.

Nam, S., Akcakaya, M., Basha, T., Stehning, C., Manning, W., Tarokh, V., Nezafat, R., 2013. Compressed sensing reconstruction for whole-heart imaging with 3D radial trajectories: A graphics processing unit implementation. Magnetic Resonance in Medicine 69, 91–102.

Narayanaswamy, A., Dwarakapuram, S., Bjornsson, C., Cutler, B., Shain, W., Roysam, B., 2010. Robust adaptive 3-D segmentation of vessel laminae from fluorescence confocal microscope images and parallel GPU implementation. IEEE Transactions on Medical Imaging 29, 583–597.

Neshat, H., Patel, R., 2008. Real-time parametric curved needle segmentation in 3D ultrasound images. In: IEEE International Conference on Biomedical Robotics and Biomechatronics, pp. 670–675.

Nichols, T.E., Holmes, A.P., 2001. Nonparametric permutation tests for functional neuroimaging: a primer with examples. Human Brain Mapping 15, 1–25.

Nickolls, J., Buck, I., Garland, M., Skadron, K., 2008. Scalable parallel programming with CUDA. Queue - GPU Computing 6, 40–53.

Noel, P., Walczak, A., Xu, J., Corso, J., Hoffmann, K., Schafer, S., 2010. GPU-based cone beam computed tomography. Computer Methods and Programs in Biomedicine 98, 271–277.

Novotny, P.M., Stoll, J.A., Vasilyev, N.V., del Nido, P.J., Dupont, P.E., Zickler, T.E., Howe, R.D., 2007. GPU based real-time instrument tracking with three-dimensional ultrasound. Medical Image Analysis 11, 458–464.

Nukada, A., Ogata, Y., Endo, T., Matsuoka, S., 2008. Bandwidth intensive 3-D FFT kernel for GPUs using CUDA. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, pp. 1–11.

Ogawa, S., Tank, D., Menon, R., Ellermann, J., Kim, S., Merkle, H., Ugurbil, K., 1992. Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. PNAS 89, 5951–5955.

Oh, J., Martin, D., Skrinjar, O., 2011. GPU-based motion correction of contrast-enhanced liver MRI scans: An OpenCL implementation. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 783–786.

Okitsu, Y., Ino, F., Hagihara, K., 2010. High-performance cone beam reconstruction using CUDA compatible GPUs. Parallel Computing 36, 129–141.

Orzo, L., Göröcs, Z., Szatmari, I., Tokes, S., 2010. GPU implementation of volume reconstruction and object detection in digital holographic microscopy. In: International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), pp. 1–4.

Osher, S., Sethian, J., 1988. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. Journal of Computational Physics 79, 12–49.

Owens, J., Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A., Purcell, T., 2007. A survey of general-purpose computation on graphics hardware. Computer Graphics Forum 26, 80–113.

Pan, L., Gu, L., Xu, J., 2008. Implementation of medical image segmentation in CUDA. In: International Conference on Information Technology and Applications in Biomedicine (ITAB), pp. 82–85.

Pan, X., Sidky, E., Vannier, M., 2009. Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction? Inverse Problems, 25.
Pang, W., Qin, J., Lu, Y., Xie, Y., Chui, C., Heng, P., 2011. Accelerating simultaneous algebraic reconstruction technique with motion compensation using CUDA-enabled GPU. International Journal of Computer Assisted Radiology and Surgery 6, 187–199.
Pauwels, K., Hulle, M.V., 2008. Realtime phase-based optical flow on the GPU. In: Computer Vision and Pattern Recognition Workshops, pp. 1–8.
Payne, B., Belkasim, S., Owen, S., Weeks, M., Zhu, Y., 2005. Accelerated 2D image processing on GPUs. In: Proceedings of ICCS, vol. 3515, Lecture notes in Computer Science, pp. 256–264.
Pedemonte, S., Bousse, A., Erlandsson, K., Modat, M., Arridge, S., Hutton, B., Ourselin, S., 2010. GPU accelerated rotation-based emission tomography reconstruction. In: IEEE Nuclear Science Symposium Conference Record, pp. 2657–2661.
Perona, P., Malik, J., 1990. Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, 629–639.
Pizer, S., Amburn, E., Austin, J., Cromartie, R., Geselowitz, A., Greer, T., Romeny, B., Zimmerman, J., Zuiderveld, K., 1987. Adaptive histogram equalization and its variations. Computer Vision, Graphics and Image Processing 39, 355–368.
Pluim, J., Maintz, A., Viergever, M., 2003. Mutual information based registration of medical images: a survey. IEEE Transactions on Medical Imaging 22, 986–1004.
Pock, T., Unger, M., Cremers, D., Bischof, H., 2008. Fast and exact solution of total variation models on the GPU. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1–8.
Podlozhnyuk, V., 2007a. 64-Bin Histogram, Nvidia Technical Report.
Podlozhnyuk, V., 2007b. Image Convolution with CUDA, Nvidia White Paper.
Prakash, J., Chandrasekharan, V., Upendra, V., Yalavarthy, P., 2010. Accelerating frequency-domain diffuse optical tomographic image reconstruction using graphics processing units. Journal of Biomedical Optics 15, 066009.
Pratx, G., Chinn, G., Habte, F., Olcott, P., Levin, C., 2006. Fully 3-D list-mode OSEM accelerated by graphics processing units. In: IEEE Nuclear Science Symposium Conference Record, pp. 2196–2202.
Pratx, G., Levin, C., 2011. Online detector response calculations for high-resolution PET image reconstruction. Physics in Medicine and Biology 56, 4023–4040.
Pratx, G., Olcott, P., Chinn, G., Levin, C., 2009. Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU. IEEE Transactions on Medical Imaging 28, 435–445.
Pratx, G., Xing, L., 2011. GPU computing in medical physics: a review. Medical Physics 38, 2685–2697.
Pruessmann, K., Weiger, M., Scheidegger, M., Boesiger, P., 1999. SENSE: Sensitivity encoding for fast MRI. Magnetic Resonance in Medicine 42, 952–962.
Pryor, G., Lucey, B., Maddipatla, S., McClanahan, C., Melonakos, J., Venugopalakrishnan, V., Patel, K., Yalamanchili, P., Malcolm, J., 2011. High-level GPU computing with Jacket for Matlab and C/C++. In: Proceedings of SPIE, Modeling and Simulation for Defense Systems and Applications.
Quan, T., Li, P., Long, F., Zeng, S., Luo, Q., Hedde, P., Nienhaus, G., Huang, Z.L., 2010. Ultra-fast, high-precision image analysis for localization-based super resolution microscopy. Optics Express 18, 11867–11876.
Rao, A., Bordawekar, R., Cecchi, G., 2011. Fast computation of functional networks from fMRI activity: a multi-platform comparison. In: Proceedings of SPIE Medical Imaging, vol. 7962.
Rasakanthan, J., Sugden, K., Tomlins, P., 2011. Processing and rendering of Fourier domain optical coherence tomography images at a line rate over 524 kHz using a graphics processing unit. Journal of Biomedical Optics 16, 020505.
Reichi, T., Passenger, J., Tamayo, O., Salvado, O., 2009. Ultrasound goes GPU: real-time simulation using CUDA. Progress in Biomedical Optics and Imaging, 10.
Roberts, M., Packer, J., Sousa, M., Mitchell, J., 2010. A work-efficient GPU algorithm for level set segmentation. In: Proceedings of the Conference on High Performance Graphics (HPG), pp. 123–132.
Roerdink, J., Meijster, A., 2000. The watershed transform: definitions, algorithms and parallelization strategies. Fundamenta Informaticae 41, 187–288.
Rong, G., Tan, T., 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In: Proceedings of the Symposium on Interactive 3D Graphics and Games, pp. 109–116.
Rosenfeld, A., Pfaltz, J., 1968. Distance functions on digital pictures. Pattern Recognition 1, 33–61.
Rosenzweig, S., Palmeri, M., Nightingale, K., 2011. GPU-based real-time small displacement estimation with ultrasound. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control 58, 399–405.
Rost, R., 1996. Using OpenGL for imaging. SPIE Medical Imaging, Image display Conference 2707, 473–484.
Roujul, S., de Senneville, B.D., Vahala, E., Sorensen, T.S., Moonen, C., Ries, M., 2009. Online real-time reconstruction of adaptive TSENSE with commodity CPU/GPU hardware. Magnetic Resonance in Medicine 62, 1658–1664.
Rudin, L., Osher, S., Fatemi, E., 1992. Nonlinear total variation based noise removal algorithms. Physica D 60, 259–268.
Ruijters, D., ter Haar Romeny, Bart M., Suetens, P., 2011. GPU-accelerated elastic 3D image registration for intra-surgical applications. Computer Methods and Programs in Biomedicine 103, 104–112.
Ruijters, D., ter Haar Romeny, B.M., Suetens, P., 2008. Efficient GPU-based texture interpolation using uniform b-splines. Journal of Graphics, GPU & Game Tools 13, 61–69.
Ruijters, D., Thevenaz, P., 2012. GPU prefilter for accurate cubic b-spline interpolation. The Computer Journal 55, 15–20.
Ruiz, A., Kong, J., Ujaldon, M., Boyer, K., Saltz, J., Gurcan, M., 2008. Pathological image segmentation for neuroblastoma using the GPU. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 296–299.
Rumpf, M., Strzodka, R., 2001a. Level set segmentation in graphics hardware. In: IEEE International Conference on Image Processing (ICIP), pp. 1103–1106.
Rumpf, M., Strzodka, R., 2001b. Nonlinear diffusion in graphics hardware. In: Proceedings of EG/IEEE TCVG Symposium on Visualization, 2001, pp. 75–84.
Sanders, J., Kandrot, E., 2011. CUDA by Example – An Introduction to General-Purpose GPU Programming. Addison-Wesley (ISBN:978-0-13-138768-3).
Santner, J., Pock, T., Bischof, H., 2010. Interactive multi-label segmentation. Lecture Notes in Computer Science, Computer Vision (ACCV) 6492, 397–410.
Saxena, V., Rohrer, J., Gong, L., 2010. A parallel GPU algorithm for mutual information based 3D nonrigid image registration. Lecture Notes in Computer Science, Euro-Par 6272, 223–234.
Schenke, S., Wuensche, B., Denzler, J., 2005. GPU-based volume segmentation. In: Proceedings of Image and Vision Computing, pp. 171–176.
Scherl, H., Keck, B., Kowarschik, M., Hornegger, J., 2007. Fast GPU-Based CT reconstruction using the common unified device architecture (CUDA). In: IEEE Nuclear Science Symposium Conference Record, pp. 4464–4466.
Scherl, H., Kowarschik, M., Hofmann, H., Keck, B., Hornegger, J., 2012. Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction. Parallel Computing 38, 111–124.
Scheuermann, T., Hensley, J., 2007. Efficient histogram generation using scattering on GPUs. In: Proceedings of the Symposium on Interactive 3D Graphics and Games, pp. 33–37.
Schiwietz, T., Chang, T., Speier, P., Westermann, R., 2006. MR image reconstruction using the GPU. Proceedings of SPIE, Advanced optical and Quantum Memories and Computing III 6142, 1279–1290.
Schmeisser, M., Heisen, B., Luettich, M., Busche, B., Hauer, F., Koske, T., Knauber, K.H., Stark, H., 2009. Parallel, distributed and GPU computing technologies in single-particle electron microscopy. Acta Crystallographica Section D Biological Crystallography 65, 659–671.
Schmid, J., Guitian, J., Gobbetti, E., Magnenat-Thalmann, N., 2011. A GPU framework for parallel segmentation of volumetric images using discrete deformable models. The Visual Computer 27, 85–95.
Schoenemann, T., Cremers, D., 2007. Globally optimal image segmentation with an elastic shape prior. In: IEEE International Conference on Computer Vision (ICCV), pp. 1–6.
Scholl, I., Aach, T., Deserno, T., Kuhlen, T., 2011. Challenges of medical image processing. Computer Science - Research and Development 26, 5–13.
Schwarzkopf, A., Kalbe, T., Bajaj, C., Kuijper, A., Goesele, M., 2012. Volumetric nonlinear anisotropic diffusion on GPUs. Lecture Notes in Computer Science, Scale Space and Variational Methods in Computer Vision 6667, 62–73.
Schweiger, M., 2011. GPU-accelerated finite element method for modelling light transport in diffuse optical tomography. International Journal of Biomedical Imaging, 403892 (Article ID:403892).
Shams, R., Barnes, N., 2007. Speeding up mutual information computation using NVIDIA CUDA hardware. In: Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications, pp. 555–560.
Shams, R., Kennedy, R., 2007. Efficient histogram algorithms for NVIDIA CUDA compatible devices. In: Proceedings of International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 418–422.
Shams, R., Luna, F., Hartley, R., 2011. An algorithm for efficient computation of spatial impulse response on the GPU with application in ultrasound simulation. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 45–51.
Shams, R., Sadeghi, P., Kennedy, R., Hartley, R., 2010a. Parallel computation of mutual information on the GPU with application to real-time registration of 3D medical images. Computer Methods and Programs in Biomedicine 99, 133–146.
Shams, R., Sadeghi, P., Kennedy, R.A., Hartley, R.I., 2010b. A survey of medical image registration on multicore and the GPU. IEEE Signal Processing Magazine 27, 50–60.
Sharma, O., Zhang, Q., Anton, F., Bajaj, C., 2010. Multi-domain, higher order level set scheme for 3D image segmentation on the GPU. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2211–2216.
Sharp, G., Kandasamy, N., Singh, H., Folkert, M., 2007. GPU-based streaming architectures for fast cone-beam CT image reconstruction and demons deformable registration. Physics in Medicine and Biology 52, 5771–5783.
Sherbondy, A., Houston, M., Napel, S., 2003. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In: IEEE Visualization, pp. 171–176.
Shi, D., Fan, Z., Yin, H., Liu, D., 2010. Fast GPU-based automatic time gain compensation for ultrasound imaging. In: International Conference on Bioinformatics and Biomedical Engineering (ICBBE), pp. 1–4.
Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 888–905.
Shimobaba, T., Sato, Y., Miura, J., Takenouchi, M., Ito, T., 2008. Real-time digital holographic microscopy using the graphic processing unit. Optics Express 16, 11776–11781.
Sigfridsson, A., Kvitting, J.P.E., Knutsson, H., Wigström, L., 2007. Five-dimensional MRI incorporating simultaneous resolution of cardiac and respiratory phases for volumetric imaging. Journal of Magnetic Resonance Imaging 25, 113–121.
Sigg, C., Hadwiger, M., 2005. Fast third-order texture filtering. In: Pharr, Matt (Ed.), GPU Gems 2. Addison-Wesley, pp. 307–329.
Sigg, C., Peikert, R., Gross, M., 2003. Signed distance transform using graphics hardware. In: IEEE Visualization, pp. 83–90.

Sisniega, A., Abella, M., Lage, E., Desco, M., Vaquero, J., 2011. Automatic Monte-Carlo based scatter correction for X-ray cone-beam CT using general purpose graphic processing units (GP-GPU): a feasibility study. In: IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), pp. 3705–3709.
Smith, D., Gore, J., Yankeelov, T., Welch, B., 2012. Real-time compressive sensing MRI reconstruction using GPU computing and split Bregman methods. International Journal of Biomedical Imaging (Article ID 864827).
So, H.K.H., Chen, J., Yiu, B.Y., Yu, A.C., 2011. Medical ultrasound imaging: to GPU or not to GPU? IEEE Micro 31, 54–65.
Sorensen, T., Atkinson, D., Schaeffter, T., Hansen, M., 2009. Real-time reconstruction of sensitivity encoded radial magnetic resonance imaging using a graphics processing unit. IEEE Transactions on Medical Imaging 28, 1974–1985.
Sorensen, T., Schaeffter, T., Noe, K.O., Hansen, M., 2008. Accelerating the nonequispaced fast Fourier transform on commodity graphics hardware. IEEE Transactions on Medical Imaging 27, 538–547.
Spoerk, J., Gendrin, C., Weber, C., Figl, M., Pawiro, S., Furtado, H., Fabri, D., Bloch, C., Bergmann, H., Gröller, E., Birkfellner, W., 2012. High-performance GPU based rendering for real-time rigid 2D/3D image registration and motion prediction in radiation oncology. Zeitschrift fur Medizinische Physik 22, 13–20.
Steininger, P., Neuner, M., Weichenberger, H., Sharp, G., Winey, B., Kametriser, G., Sedlmayer, F., Deutschmann, H., 2012. Auto-masked 2D/3D image registration and its validation with clinical cone-beam computed tomography. Physics in Medicine and Biology 57, 4277–4292.
Stone, S., Haldar, J., Tsao, S., Hwu, W., Sutton, B., Liang, Z., 2008. Accelerating advanced MRI reconstructions on GPUs. Journal of Parallel and Distributed Computing 68, 1307–1318.
Strzodka, R., Ihrke, I., Magnor, M., 2003. A graphics hardware implementation of the generalized Hough transform for fast object recognition, scale, and 3D pose detection. In: International Conference on Image Analysis and Processing, pp. 188–193.
Strzodka, R., Telea, A., 2004. Generalized distance transforms and skeletons in graphics hardware. In: Proceedings of VisSym, pp. 221–230.
Stsepankou, D., Arns, A., Ng, S., Zygmanski, P., Hesser, J., 2012. Evaluation of robustness of maximum likelihood cone-beam CT reconstruction with total variation regularization. Physics in Medicine and Biology 57, 5955–5970.
Su, Y., Xu, Z., 2010. Parallel implementation of wavelet-based image denoising on programmable PC-grade graphics hardware. Signal Processing 90, 2396–2411.
Sud, A., Govindaraju, N., Gayle, R., Manocha, D., 2006. Interactive 3D distance field computation using linear factorization. In: Proceedings of the Symposium on Interactive 3D Graphics and Games, pp. 117–124.
Sud, A., Otaduy, M., Manocha, D., 2004. DiFi: Fast 3D distance field computation using graphics hardware. Computer Graphics Forum 23, 557–566.
Sugita, K., Naemura, T., Harashima, H., 2003. Performance evaluation of programmable graphics hardware for image filtering and stereo matching. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST), pp. 176–183.
Sumanaweera, T., Liu, D., 2005. Medical image reconstruction with the FFT. In: Pharr, M. (Ed.), GPU Gems 2. Addison Wesley, pp. 765–784 (Chapter 48).
Svensson, B., Andersson, M., Knutsson, H., 2005. Filter networks for efficient estimation of local 3D structure. In: IEEE International Conference on Image Processing (ICIP), pp. 573–576.
Tagare, H., Barthel, A., Sigworth, F., 2010. An adaptive expectation maximization algorithm with GPU implementation for electron cryomicroscopy. Journal of Structural Biology 171, 256–265.
Tan, G., Guo, Z., Chen, M., Meng, D., 2009. Single-particle 3D reconstruction from cryo-electron microscopy images on GPU. In: Proceedings of the International Conference on Supercomputing (ICS), pp. 380–389.
Thomenius, K., 2009. Miniaturization of ultrasound scanners. Ultrasound Clinics 4, 385–389.
Thyreau, B., Schwartz, Y., Thirion, B., Frouin, V., Loth, E., Vollstädt-Klein, S., Paus, T., Artiges, E., Conrod, P., Schumann, G., Whelan, R., Poline, J.B., 2012. Very large fMRI study using the IMAGEN database: sensitivity–specificity and population effect modeling in relation to the underlying anatomy. NeuroImage 61, 295–303.
Tian, Z., Jia, X., Dong, B., Lou, Y., Jiang, S., 2011a. Low-dose 4DCT reconstruction via temporal nonlocal means. Medical Physics 38, 1359–1365.
Tian, Z., Jia, X., Yuan, K., Pan, T., Jiang, S., 2011b. Low-dose CT reconstruction via edge-preserving total variation regularization. Physics in Medicine and Biology 56, 5949–5967.
Tomasi, C., Manduchi, R., 1998. Bilateral filtering for gray and color images. In: Proceedings International Conference on Computer Vision, pp. 839–846.
Tommasini, S., Trinward, A., Acerbo, A., Carlo, F., Miller, L., Judex, S., 2012. Changes in intracortical microporosities induced by pharmaceutical treatment of osteoporosis as detected by high-resolution micro-CT. Bone 50, 596–604.
Top, A., Hamarneh, G., Abugharbieh, R., 2011. Active learning for interactive 3D image segmentation. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI) 6893, 603–610.
Unger, M., Pock, T., Trobin, W., Cremers, D., Bischof, H., 2008. TVSeg - interactive total variation based image segmentation. In: British Machine Vision Conference (BMVC).
van der Jeught, S., Bradu, A., Podoleanu, A., 2010. Real-time resampling in Fourier domain optical coherence tomography using a graphics processing unit. Journal of Biomedical Optics 15, 030511.
Vasanawala, S., Murphy, M., Alley, M., Lai, P., Keutzer, K., Pauly, J., Lustig, M., 2011. Practical parallel imaging compressed sensing MRI: Summary of two years of experience in accelerating body MRI of pediatric patients. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 1039–1043.
Vetter, C., Westermann, R., 2008. SPECT reconstruction on the GPU. Proceedings of SPIE Medical Imaging, 6913.
Vetter, C., Westermann, R., 2011. Optimized GPU histograms for multi-modal registration. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 1227–1230.
Vineet, V., Narayanan, P., 2008. CUDA cuts: fast graph cuts on the GPU. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1–8.
Vinegoni, C., Fexon, L., Feruglio, P., Pivovarov, M., Figueiredo, J.L., Nahrendorf, M., Pozzo, A., Sbarbati, A., Weissleder, R., 2009. High throughput transmission optical projection tomography using low cost graphics processing unit. Optics Express 17, 22320–22332.
Vintache, D., Humbert, B., Brasse, D., 2010. Iterative reconstruction for transmission tomography on GPU using Nvidia CUDA. Tsinghua Science and Technology 15, 11–16.
Viola, I., Kanitsar, A., Gröller, M.E., 2003. Hardware-based nonlinear filtering and segmentation using high-level shading language. In: IEEE Visualization Conference, pp. 309–316.
Viola, P., Wells, W., 1997. Alignment by maximization of mutual information. International Journal of Computer Vision 24, 137–154.
Walters, J., Balu, V., Kompalli, S., Chaudhary, V., 2009. Evaluating the use of GPUs in liver image segmentation and HMMER database searches. In: IEEE International Symposium on Parallel Distributed Processing (IPDPS), pp. 1–12.
Wang, L., Hofer, B., Guggenheim, J., Povazay, B., 2012. Graphics processing unit-based dispersion encoded full-range frequency-domain optical coherence tomography. Journal of Biomedical Optics 17, 077007.
Wang, L., Shi, D., Zhao, A., Tan, C., Liu, D., 2011a. Real-time scan conversion for ultrasound imaging based on CUDA with Direct3D display. In: International Conference on Bioinformatics and Biomedical Engineering (ICBBE), pp. 1–4.
Wang, X., Shi, B., 2010. GPU implementation of fast Gabor filters. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 373–376.
Wang, Z., Hand, G., Li, T., Liang, Z., 2005. Speedup OS-EM image reconstruction by PC graphics card technologies for quantitative SPECT with varying focal-length fan-beam collimation. IEEE Transactions on Nuclear Science 52, 1274–1280.
Wang, Z., Yan, Z., Chen, G., 2011b. Lattice Boltzmann method of active contour for image segmentation. In: Sixth International Conference on Image and Graphics (ICIG), pp. 338–343.
Watanabe, Y., 2012. Real time processing of Fourier domain optical coherence tomography with fixed-pattern noise removal by partial median subtraction using a graphics processing unit. Journal of Biomedical Optics 17, 050503.
Watanabe, Y., Itagaki, T., 2009. Real-time display on Fourier domain optical coherence tomography system using a graphics processing unit. Journal of Biomedical Optics 14, 060506.
Watanabe, Y., Maeno, S., Aoshima, K., Hasegawa, H., Koseki, H., 2010. Real-time processing for full-range Fourier-domain optical-coherence tomography with zero-filling interpolation using multiple graphic processing units. Applied Optics 49, 4756–4762.
Weickert, J., 1998. Anisotropic Diffusion in Image Processing. Teubner-Verlag, Stuttgart.
Weiskopf, N., Veit, R., Erb, M., Mathiak, K., Grodd, W., Goebel, R., Birbaumer, N., 2003. Physiological self-regulation of regional brain activity using real-time functional magnetic resonance imaging (fMRI): methodology and exemplary data. NeuroImage 19, 577–586.
Wen, J., Wang, Z., Li, B., Li, T., Liang, Z., 2004. Speed up of an analytical algorithm for nonuniform attenuation correction by using PC video/graphics card architecture. IEEE Transactions on Nuclear Science 51, 726–732.
Westin, C.F., Wigström, L., Loock, T., Sjöqvist, L., Kikinis, R., Knutsson, H., 2001. Three-dimensional adaptive filtering in magnetic resonance angiography. Journal of Magnetic Resonance Imaging 14, 63–71.
Winnemöller, H., Olsen, S., Gooch, B., 2006. Real-time video abstraction. ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH, 1221–1226.
Wirth, A., Cserkaszky, A., Kari, B., Legrady, D., Feher, S., Czifrus, S., Domonkos, B., 2009. Implementation of 3D Monte Carlo PET reconstruction algorithm on GPU. In: IEEE Nuclear Science Symposium Conference Record, pp. 4106–4109.
Wolfe, M., 2010. Implementing the PGI accelerator model. In: Workshop on General Purpose Computation on Graphics Processing Units. ACM, pp. 43–50.
Woolrich, M., 2012. Bayesian inference in fMRI. NeuroImage 15, 801–810.
Woolrich, M.W., Jenkinson, M., Brady, M., Smith, S.M., 2004. Fully bayesian spatio-temporal modeling of fMRI data. IEEE Transactions on Medical Imaging 23, 213–231.
Wu, D., Wu, T., Shan, Y., Wang, Y., He, Y., Xu, N., Yang, H., 2010. Making human connectome faster: GPU acceleration of brain network analysis. In: IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 593–600.
Wu, H., Zhang, W., Gao, D., Yin, X., Chen, Y., 2011a. Fast CT image processing using parallelized non-local means. Journal of Medical and Biological Engineering 31, 437–441.
Wu, X., Gai, J., Lam, F., Fu, M., Haldar, J., Zhuo, Y., Liang, Z., Hwu, W., Sutton, B., 2011b. Impatient MRI: Illinois massively parallel acceleration toolkit for image reconstruction with enhanced throughput in MRI. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 69–72.
Xu, F., Mueller, K., 2004. Ultra-fast 3D filtered backprojection on commodity graphics hardware. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 571–574.

Xu, F., Mueller, K., 2005. Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware. IEEE Transactions on Nuclear Science 52, 654–663.
Xu, F., Mueller, K., 2007. Real-time 3D computed tomographic reconstruction using commodity graphics hardware. Physics in Medicine and Biology 52, 3405–3419.
Xu, F., Xu, W., Jones, M., Keszthelyi, B., Sedat, J., Agard, D., Mueller, K., 2010a. On the efficiency of iterative ordered subset reconstruction algorithms for acceleration on GPUs. Computer Methods and Programs in Biomedicine 98, 261–270.
Xu, R., Pattanaik, S., 2005. Non-iterative, robust Monte Carlo noise reduction. IEEE Computer Graphics and Applications 25, 31–35.
Xu, W., Xu, F., Jones, M., Keszthelyi, B., Sedat, J., Agard, D., Mueller, K., 2010b. High-performance iterative electron tomography reconstruction with long-object compensation using graphics processing units (GPUs). Journal of Structural Biology 171, 142–153.
Yan, G., Tian, J., Zhu, S., Dai, Y., Qin, C., 2008. Fast cone-beam CT image reconstruction using GPU hardware. Journal of X-Ray Science and Technology 16, 225–234.
Yan, G., Tian, J., Zhu, S., Qin, C., Dai, Y., Yang, F., Dong, D., Wu, P., 2010. Fast Katsevich algorithm based on GPU for helical cone-beam computed tomography. IEEE Transactions on Information Technology in Biomedicine 14, 1053–1061.
Yang, J., Feng, C., Zhao, D., 2012. A CUDA-based reverse gridding algorithm for MR reconstruction. Magnetic Resonance Imaging 31, 313–323.
Yang, Q., Tan, K.H., Ahuja, N., 2009. Real-time O(1) bilateral filtering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 557–564.
Yang, R., Welch, G., 2003. Fast image segmentation and smoothing using commodity graphics hardware. Journal of Graphics Tools 7, 91–100.
Yang, Y.H., Huang, T.Y., Wang, F.N., Chuang, T.C., Che, N.K., 2011. Accelerating EPI distortion correction by utilizing a modern GPU-based parallel computation. Journal of Neuroimaging. http://dx.doi.org/10.1111/j.1552-6569.2011.00654.x.
Yiu, B., Tsang, I., Yu, A., 2010. Real-time GPU-based software beamformer designed for advanced imaging methods research. In: IEEE Ultrasonics Symposium, pp. 1920–1923.
Yu, F., Liu, H., Shi, P., 2011a. PET image reconstruction: GPU-accelerated particle filter framework. In: IEEE International Conference on Image Processing (ICIP), pp. 417–420.
Yu, W., Chen, Y., Luo, L., 2011b. De-noising of low-dose CT images using space-time nonlocal means over large-scale neighborhoods. In: IEEE/ICME International Conference on Complex Medical Engineering (CME), pp. 455–459.
Yuancheng, L., Duraiswami, R., 2008. Canny edge detection on Nvidia CUDA. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1–8.
Yuen, S., Kesner, S., Vasilyev, N., del Nido, P., Howe, R., 2008. 3D ultrasound-guided motion compensation system for beating heart mitral valve repair. Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI) 5241, 711–719.
Zhang, B., Xu, S., Zhang, F., Bi, Y., Huang, L., 2011. Accelerating Matlab code using GPU: a review of tools and strategies. In: International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), pp. 1875–1878.
Zhang, H., Yan, B., Lu, L., Li, L., Liu, Y., 2012. High performance parallel backprojection on multi-GPU. In: International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 2693–2696.
Zhang, K., Kang, J., 2010a. Graphics processing unit accelerated non-uniform fast Fourier transform for ultrahigh-speed, real-time Fourier-domain OCT. Optics Express 18, 23472–23487.
Zhang, K., Kang, J., 2010b. Real-time 4D signal processing and visualization using graphics processing unit on a regular nonlinear-k Fourier-domain OCT system. Optics Express 18, 11772–11784.
Zhang, K., Kang, J., 2011. Real-time intraoperative 4D full-range FD-OCT based on the dual graphics processing units architecture for microsurgery guidance. Biomedical Optics Express 2, 764–770.
Zhang, W., Zhang, L., Sun, S., Xing, Y., Wang, Y., Zheng, J., 2009. A preliminary study of OpenCL for accelerating CT reconstruction and image recognition. In: IEEE Nuclear Science Symposium Conference Record, pp. 4059–4063.
Zhang, X., Zhang, X., Zhou, Z., 2010. Low cost, high performance GPU computing solution for atomic resolution cryoEM single-particle reconstruction. Journal of Structural Biology 172, 400–406.
Zhao, X., Hu, J.J., Zhang, P., 2009. GPU-based 3D cone-beam CT image reconstruction for large data volume. International Journal of Biomedical Imaging (Article ID 149079).
Zhao, Y., 2008. Lattice Boltzmann based PDE solver on the GPU. The Visual Computer 24, 323–333.
Zheng, Z., Xu, W., Mueller, K., 2011. Performance tuning for CUDA-accelerated neighborhood denoising filters. In: Workshop on High Performance Image Reconstruction (HPIR).
Zhou, J., Qi, J., 2011. Fast and efficient fully 3D PET image reconstruction using sparse system matrix factorization with GPU acceleration. Physics in Medicine and Biology 56, 6739–6757.
Zhu, H., Chen, Y., Wu, J., Gu, J., Eguchi, K., 2011. Implementation of 3D SRAD algorithm on CUDA. In: International Conference on Intelligent Networks and Intelligent Systems, pp. 97–100.
Zhu, Y., Zhao, Y., Zhao, X., 2012. A multi-thread scheduling method for 3D CT image reconstruction using multi-GPU. Journal of X-ray Science and Technology 20, 187–197.
Zhuge, Y., Cao, Y., Udupa, J., Miller, R., 2011. Parallel fuzzy connected image segmentation on GPU. Medical Physics 38, 4365–4371.
Zhuo, Y., Sutton, B., Wu, X., Haldar, J., Hwu, W., Liang, Z., 2010a. Sparse regularization in MRI iterative reconstruction using GPUs. In: International Conference on Biomedical Engineering and Informatics (BMEI), pp. 578–582.
Zhuo, Y., Wu, X., Haldar, J., Hwu, W., Liang, Z., Sutton, B., 2010b. Accelerating iterative field-compensated MR image reconstruction on GPUs. In: IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, pp. 820–823.
Zhuo, Y., Wu, X., Haldar, J., Hwu, W., Liang, Z., Sutton, B., 2010c. Multi-GPU implementation for iterative MR image reconstruction with field correction. In: International Society for Magnetic Resonance in Medicine (ISMRM), p. 2942.