Journal of Real-Time Image Processing
The final publication is available at Springer via http://dx.doi.org/10.1007/s11554-014-0436-8

Mark Howison · E. Wes Bethel

GPU-accelerated denoising of 3D magnetic resonance images

Received: 20 January 2013 / Accepted: 29 May 2014

Abstract The raw computational power of GPU accelerators enables fast denoising of 3D MR images using bilateral filtering, anisotropic diffusion, and non-local means. In practice, applying these filtering operations requires setting multiple parameters. This study was designed to provide better guidance to practitioners for choosing the most appropriate parameters by answering two questions: What parameters yield the best denoising results in practice? And what tuning is necessary to achieve optimal performance on a modern GPU? To answer the first question, we use two different metrics, mean-squared error (MSE) and mean structural similarity (MSSIM), to compare denoising quality against a reference image. Surprisingly, the best improvement in structural similarity with the bilateral filter is achieved with a small stencil size that lies within the range of real-time execution on an NVIDIA Tesla M2050 GPU. Moreover, inappropriate choices for parameters, especially scaling parameters, can yield very poor denoising performance. To answer the second question, we perform an autotuning study to empirically determine optimal memory tiling on the GPU. The variation in these results suggest that such tuning is an essential step in achieving real-time performance. These results have important implications for the real-time application of denoising to MR images in clinical settings that require fast turnaround times.

M. Howison
Center for Computation and Visualization
Brown University
180 George Street, Providence, RI 02912, USA
E-mail: mhowison@brown.edu

M. Howison and E. Wes Bethel
Computational Research Division
Lawrence Berkeley National Laboratory
One Cyclotron Road, Berkeley, CA 94720, USA
E-mail: ewbethel@lbl.gov

Disclaimer. This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

1 Introduction

Denoising is a important step in many image processing pipelines for brain magnetic resonance imaging (MRI). We studied three 3D filters – the bilateral, anisotropic diffusion, and non-local means filters – that remove noise and smooth features within MR images while preserving edges. Our implementations of these stencil-based algorithms use NVIDIA's CUDA programming language to take advantage of the high computational throughput of GPU accelerators.

All three of these filters have tunable parameters, and we performed parameter sweeps to identify optimal settings both for runtime and noise reduction. To measure noise reduction, we began with MR images from a simulated brain database called BrainWeb (McConnel Brain Imaging Center 1997) that contains both noiseless reference images and images with added noise. Then, we ran our filtering on the noisy images and compared the results fo the noiseless images using two different metrics,

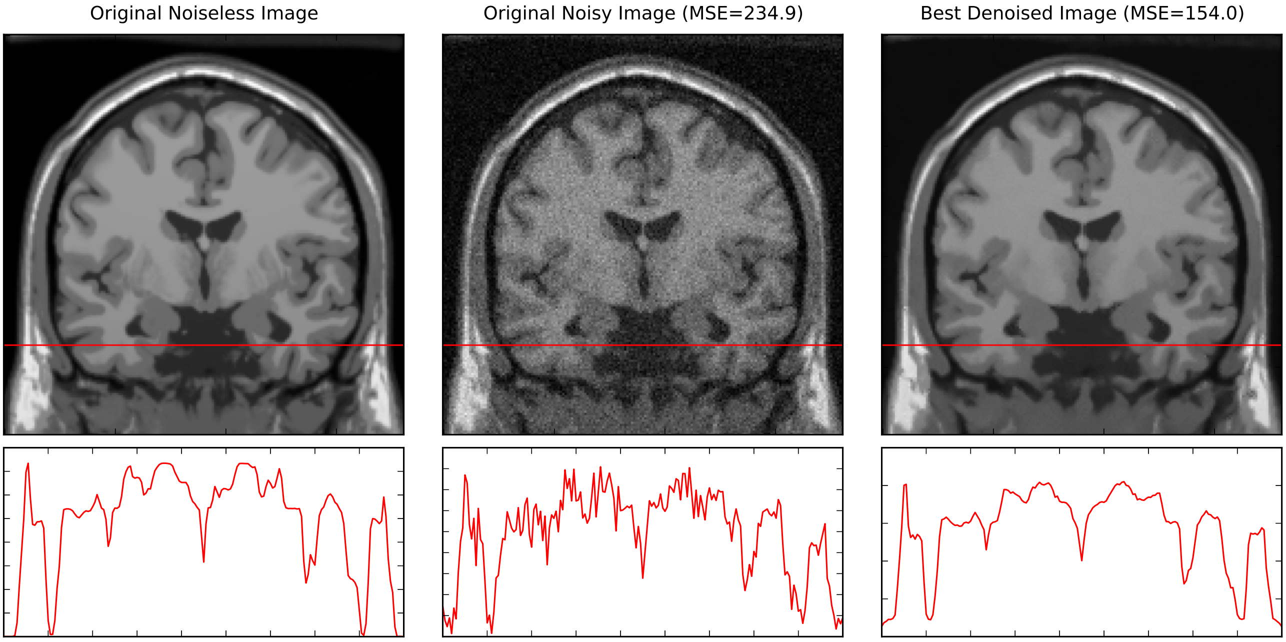

Original Noiseless Image
Original Noisy Image (MSE=234.9)
Best Denoised Image (MSE=154.0)

Fig. 1 Visual comparison of a 2D slice of noiseless and noisy (9%) BrainWeb images along side a denoising result from the non-local means filter at $W = 3$, $P = 1$, and $h = 0.1$. The plots below the images show the variation in signal across a single scan line, which is also shown in red in the images.

the mean squared error (MSE) and the mean structural similarity (MSSIM). Our findings show a wide variaion in both denoising and runtime performance, and that it is important to tune both.

The tunable parameters can be divided into two groups: those that affect runtime (e.g. stencil size) and those that do not (e.g. scaling factors), which we will call *runtime parameters* and *smoothing parameters*, respectively. With current GPU technology, real-time execution is possible for small stencil sizes, which should yield less denoising. That is, we would expect larger stencil sizes to increase runtime, but also to yield more noise reduction.

Yet, surprisingly, we will show that for the bilateral filter, smaller stencil sizes achieive the best denoising. Thus, we are able to, in practice, perform real-time 3D denoising of MR images that would appear to require too much computation, even on modern GPUs. Table 1 summarizes our test data and the best tuned runtimes (including transfer time to and from GPU memory) for each of the filters at their most effective smoothing parameters. Figure 1 shows the original noisy data set, and the resulting denoised image using the non-local means filter.

One potential application of our results is the rapid turn-around of MRI denoising over a range of filtering parameters. For instance, following an MRI scan, practitioners may wish to quickly perform a sweep of denoising filters at different parameters so that they can visually compare them and select the best one. Similarly, real-time denoising could support an interactive GUI for exploring the parameter space.

2 Background

2.1 Denoising

2.1.1 Bilateral Filtering

Bilateral filtering, as originally described by Tomasi and Manduchi (1998), performs anisotropic image smoothing using a low-cost, non-iterative formulation. It computes the influence of nearby pixels in a way that removes noise locally and that does not have the undesirable property of smoothing edge features. This formulation uses a straightforward, tunable estimate for region boundaries: a Gaussian-weighted difference in the range of the signal, or photometric space. Wherever a sharp edge exists, there will be a large difference in signal that will correspond to a small Gaussian weight, thereby attenuating the smoothing. The range estimate is combined with a traditional Gaussian-weighted distance function in the spatial domain to lessen the contribution of more distant pixels.

For a 3D volume $\mathbf{V}$, the output of the bilateral filter at each voxel $\mathbf{V}_i$ is the weighted average of the voxels at nearby locations j in the neighborhood N_i with radius R (e.g., for $R = 1$ the $3 \times 3 \times 3$ cube centered at $\mathbf{V}_i$). The influence of each voxel in N_i is computed as the product

Table 1 Summary of Test Data and Best Tuned Runtimes

Scanner	Modality	Dimensions	M. Voxels	Bilateral Filter R=2	Anisotropic Diffusion N=20	Non-Local Means (W,P)=(3,2)
7-T	T2 (TRA)	(480, 480, 36)	8.3	84 ms	495 ms	4,892 ms
3-T	MPRAGE	(256, 256, 200)	13.1	147 ms	803 ms	8,240 ms
7-T	T2 (2D)	(1024, 1024, 20)	21.0	183 ms	1,236 ms	12,108 ms
7-T	T1	(352, 352, 249)	30.9	324 ms	1,851 ms	18,466 ms
3-T	FLAIR	(576, 576, 300)	99.5	1,132 ms	6,019 ms	58,188 ms

Note: all runtimes include transfer times to and from GPU memory.

of two Gaussian kernels, one in the spatial domain (G_{σ_d}) and one in the photometric range (G_{σ_r}). The Gaussian kernels attenuate the influence of nearby points such that those nearby in geometric or signal space have greater influence, while those further away in geometric or signal space have less influence. The voxels in the filtered image $\mathbf{V}'$ are computed as:

$$\mathbf{V}'_i = \frac{1}{w_i} \sum_{j \in N_i} G_{\sigma_d}(\|i-j\|) G_{\sigma_r}(|\mathbf{V}_i - \mathbf{V}_j|)\mathbf{V}_j \tag{1}$$

where w_i is a normalization factor that sums all of the Gaussian weights,

$$w_i = \sum_{j \in N_i} G_{\sigma_d}(\|i-j\|) G_{\sigma_r}(|\mathbf{V}_i - \mathbf{V}_j|). \tag{2}$$

While it is possible to precompute the weights in w_p contributed by G_{σ_d}, the contributions from G_{σ_r} are not known *a priori* as they depend on the photometric values of a particular image.

In earlier work, we extended the original Tomasi bilateral filtering formulation for use on 3D volumetric data and compared its scalability and performance on shared- and distributed-memory platforms using several different parallel programming models and execution frameworks (Bethel 2009).

The bilateral filter has also been extended by Hamarneh and Hradsky (2007) to smooth diffusion tensor MRI data by replacing the notion of photometric similarity with a metric suitable for measuring the dissimilarity of diffusion tensors.

2.1.2 Anisotropic Diffusion

Anisotropic diffusion is a process similar to isotropic diffusion, or Gaussian smoothing, but includes an additional "conductance" function to regulate how much diffusion takes place at different locations in the image. This is useful for edge-detection and for edge-preserving denoising of images because the conductance function can be chosen to limit the diffusion near sudden changes in value (i.e. large gradients) and these areas typically correspond to edges.

In its continuous and most general form, anisotropic diffusion solves the following PDE that is similar to the heat equation, but with the addition of the conductance function g:

$$\frac{\partial u}{\partial t} = \nabla \cdot (g(|\nabla u|)\nabla u). \tag{3}$$

Several anisotropic filters for 2D edge-preserving denoising have been proposed over the years, starting with Perona and Malik (1990). Extending their formulation to a 3D volume V, we can approximate the solution of the PDE at each voxel as:

$$\mathbf{V}_i^{t+1} = \mathbf{V}_i^t + \Delta t \sum_{j \in N_i} g(\mathbf{V}_j^t - \mathbf{V}_i^t)(\mathbf{V}_j^t - \mathbf{V}_i^t) \tag{4}$$

where Δt is the timestep and N_i is the neighborhood of 6 points that are ±1 lattice point from i in each direction.

Gerig et al (1992) suggested enhancing this approximation by sampling from a larger neighborhood that includes all of the diagonal lattice directions. This is a 26-point stencil in 3D, and therefore incurs more computation. It is a similar formulation to Perona and Malik's, but adds a normalization factor l_j that corrects for the length of the diagonal between voxels $\mathbf{V}_i$ and $\mathbf{V}_j$:

$$\mathbf{V}_i^{t+1} = \mathbf{V}_i^t + \Delta t \sum_{j \in N_i} \frac{g(\mathbf{V}_j^t - \mathbf{V}_i^t)}{l_j}(\mathbf{V}_j^t - \mathbf{V}_i^t). \tag{5}$$

The normalization factor also accomodates datasets with non-uniform elements, such as in 3D MR images where the resolution is lower in the z-dimension. However, some implementors warn that images acquired with excessively skewed ratios between resolutions, such as $1:2$ or greater, contain insufficient information to be used successfully in biomedical settings (see Ibanez et al (2003) pg. 242 for a discussion).

Weickert and Benhamouda (1997) described a filter that precomputes the gradient approximation by central differences in a buffer G, then calculates:

$$\mathbf{V}_i^{t+1} = \mathbf{V}_i^t + \Delta t \sum_{j \in N_i} \frac{(G_j^t - G_i^t)}{l_j}(\mathbf{V}_j^t - \mathbf{V}_i^t). \tag{6}$$

In an earlier study (Howison 2010), we implemented and tested all three of these discrete forms along with four conductance functions, using the same test data as in the present study. We found that the best noise reduction was achieved by the Weickert stencil in combination with the second conductance function described by Perona and Malik (1990), namely

$$g(t) = \frac{1}{1 + (t/\kappa)^2} \tag{7}$$

where κ is a scaling factor. Therefore, we tested only this combination in the present study.

2.1.3 Non-Local Means

Like the bilateral filter and anisotropic diffusion, non-local means denoises an image by averaging values within the image, and was first introduced by Buades et al (2005). Whereas the bilateral filter and anistropic diffusion consider only a local neighborhood around each voxel, non-local means incorporates additional "non-local" information by computing the weights for the voxels in the local neighborhood, or "window," as the average over a second neighborhood, or "patch." This scheme can be expressed as:

$$\mathbf{V}'_i = \frac{1}{Z_i} \sum_{j \in N_i} w(i,j) \mathbf{V}_i \tag{8}$$

where N_i is the window neighborhood of radius W around the voxel $\mathbf{V}_i$. The $w(i,j)$ are weights that are calculated for each voxel $\mathbf{V}_j$ in the window by averaging over a patch neighborhood $\varDelta$ of radius P centered at $\mathbf{V}_j$:

$$w(i,j) = \sum_{\delta \in \varDelta} G_h \left(|\mathbf{V}_{i+\delta} - \mathbf{V}_{j+\delta}| \right) \tag{9}$$

where G_h is a Gaussian kernel with standard deviation h. The normalization factor Z_i for voxel $\mathbf{V}_i$ is the sum of the voxel's window weights:

$$Z_i = \sum_{j \in N_i} w(i,j). \tag{10}$$

Non-local means is by far the most computationally complex denoising approach of the three we tested in this study. The asymptotic runtime of a stencil calculation grows as the product of the size of its nested loops. Therefore, the asymptotic runtime for non-local means is $O(|\mathbf{V}|W^3P^3)$, where $|\mathbf{V}|$ is number of voxels in the 3D volume and W and P are the window and patch sizes. In contrast, the bilateral filter's asymptotic runtime is $O(|\mathbf{V}|R^3)$ for the radius R, and is therefore faster by a cubic factor. The asymptotic runtime for anisotropic diffusion, $O(|\mathbf{V}|N)$, is linear in the number of iterations N, and the least complex of the three.

Several studies have addressed the practical issues of using non-local means on both 2D and 3D data that arise because of its computational complexity. Mahmoudi and Sapiro (2005) applied a filtering technique that precalculates a classification of all the neighborhoods by hashing them by their average gray values and gradient orientations. In practice, this led to a ≈ 10× speed-up for 2D images.

Coupe et al (2008) extended Mahmoudi and Sapiro (2005)'s filtering technique to 3D MR images and combined it with an automated selection of the smoothing parameter h, a blocked approximation of the non-local means stencil, and threaded parallelism to reduce runtimes. However, the fastest runtime they report for the same dataset used in this study is 63s, which is far from real-time.

Darbon et al (2008) described an optimized algorithm for non-local means of 2D images, for applications in electron cryomicroscopy. By precomputing the window weights, this algorithm reduces the asymtotic runtime for 2D images from $O(|\mathbf{V}|W^2P^2)$ to $O(|\mathbf{V}|W^2)$. That is, it is able to calculate the weights in constant time, independent of the patch size P, and is therefore a factor of $O(P^2)$ faster. While this optimization leads to real-time runtimes in a threaded CPU implementation, these are only for small image sizes with an order of magnitude fewer pixels than the number of voxels in our test data.

2.2 GPU Computing

A GPU implementation of bilateral filtering appeared in Chen et al (2007). Using a combination of vertex and fragments shaders, it performs filtering on a data structure called a "bilateral grid" that contains a reduced-size approximation of the original data, thereby achieving good performance and memory utilization. For a two-dimensional image, the three-dimensional bilateral grid contains two spatial dimensions that correspond to pixel location, and the third dimension corresponds to pixel range information, typically pixel intensity. The authors implement bilateral filtering, along with several other edge-aware algorithms, on the GPU using a combination of vertex and fragment shaders. Because the bilateral grid enables aggressive downsampling of the source image, they are able to achieve high performance using relatively small memory footprints on the GPU. In contrast, our work is a direct implementation in CUDA of bilateral filtering in 3D, rather than an approximation, which is more appropriate for clinical medical imaging applications.

Stone et al (2008) presented results from optimizing a MRI reconstruction algorithm on an NVIDIA Quadro FX 5600 GPU. They compute anatomically constrained reconstructions of non-cartesian MRI data. Their CUDA-based algorithm performs a least-squares minimization using conjugate gradient iterations along with FFT, inverse FFT, BLAS and sparse BLAS operations. They experimented with a number of optimizations: register allocation, coalesced memory access, use of constant memory, fast math operations, and finally exhaustive search for optimal tiling, number of threads per block and loop unrolling factors. They reported a speedup of ≈ 13× over a CPU implementation for a 128^3 dataset. While we use similar GPU optimizations, we use a different stencil-based algorithm with a different data access pattern. In an MR imaging pipeline, denoising could be used as a complementary post-processing technique after the data has been acquired and processed using their non-cartesian acquisition method.

Zheng et al (2011) showed that pre-fetching blocks of memory on the GPU accelerates both the bilateral filter and non-local means for 2D images. Their reported speed-ups were as high as 1.5× for the bilateral filter and 4.8× for non-local means. However, they did not conduct a systematic evaluation of block sizes as we do in this study, nor did they consider 3D denoising.

Eklund et al (2011) used GPUs to accelerate true 4D denoising of time-series 3D computed tomography (CT) data sets. Because such data are collected at low dosages, the resulting images are noisy. True 4D denoising applies the denoising filter in both the 3D spatial and 1D temporal domain to improve the denoising results, since some reconstruction artifacts vary with time. However, moving from 3D to 4D images increases computational complexity, and their CPU implementation takes several days to perform adaptive filtering, or 50 minutes when using an FFT-based filter. With GPU acceleration, the runtime can be decreased to 25 minutes and 8 minutes, respectiley. While this is far from real-time, the 4D data sets they process are two orders-of-magnitude larger than our 3D MR images.

2.3 Autotuning

Autotuning is a methodology for finding the combinations of tunable algorithmic parameters that result in the best performance of an algorithm (or implementation) on a particular platform and particular problem configuration. This approach has been used with success to optimize performance of stencil-based codes on multi-core CPUs and many-core GPUs (Datta et al 2009; Williams et al 2010; Hollingsworth and Tiwari 2010; Kamil et al 2010; Magni et al 2013). Because it uses empirical data, autotuning can accommodate a large diversity of parallel computing architectures, along with their complex and rapidly changing performance characteristics. Alternately, some recent research, such as that by de la Cruz and Araya-Polo (2011), focuses on deriving a predictive performance model for an 3D stencil computation code, which has a uniform and predictable memory access pattern.

Autotuning assumes that all possible tunable configurations of an algorithm can be enumerated and tested for a given problem size. Since the number of permutations can be quite large, search strategies have emerged to avoid expensive exhaustive searches. For example, Ryoo et al (2008) presented a set of performance metrics to estimate the performance of a given optimization configuration for CUDA-based code running on a GPU. They computed two metrics, efficiency and utilization, by examining developer-readable assembler and GPU resource utilization maps produced by the CUDA compiler. The basic idea is to estimate values for each of these metrics by examining resource utilization maps. Then, to avoid an exhaustive search of the optimization space, they estimated the relative performance change by altering parameters that contribute to both metrics. They pruned the size of the search space by examining only those configurations that have only high levels in both metrics.

Other approaches to reducing the search parameter space include techniques like genetic algorithms (Garcia et al 2013), simulated annealing, or even a simple random search, which has been shown to be surprisingly effective in studies of dense computational codes on CPUs (Seymour et al 2008). The savings can be dramatic: a study by Ganapathi et al (2009) that used machine learning techniques to reduce a parameter search space yielded a speed-up of 2000× over the full search, while achieving tuning results comparable to a human expert. Our search space is small enough that we do not explore these search space optimizations in this study, but they may be worthwhile to employ in a production system based on our findings.

3 Methods

3.1 Software Implementation

Our GPU implementations are written in the highly-threaded, data-parallel CUDA language (NVIDIA Corporation 2012), which enables access to GPU architectural resources via a set of extensions to C. We implemented a simple memory tiling scheme originally described by Rivera and Tseng (2000) to accelerate 3D stencil computations by preloading a block of neighboring voxels into a GPU's shared memory. This reduces the number of redundant memory loads among neighboring voxels, especially at larger stencil radii, where there is more opportunity for cache reuse.

We also implemented utility routines to transfer 3D datasets to and from CUDA global memory while maintaining the appropriate padding required by the stencil at boundary locations. Finally, we padded out the dataset in the fastest moving dimension to a multiple of the CUDA "warp" size of 32, which is the number of threads that are executed simultaneously on each of the GPU's multi-processors.

We used version 5.0.35 of the CUDA compiler and runtime and version 310.32 of the NVIDIA driver for Linux. ECC support was enabled. During compilation, we targeted CUDA "compute capability" 2.0 (NVIDIA Corporation 2012) to take advantage of additional optimizations available on the Fermi-series architecture.

Our implementation is open-source and available from `https://bitbucket.org/berkeleylab/crd-gd3d`.

3.2 Test Platform and Data

We conducted our tests on the Oscar cluster at the Brown University Center for Computation and Visualization. Each IBM iDataPlex node in the cluster has a dual 2.5 GHz Intel 5630 Nehalem processors and 24 GB of RAM. The GPU accelerator is a NVIDIA Fermi M2050, a many-core GPU with 448 "CUDA cores" grouped as

14 “multiprocessors” (NVIDIA Corporation 2012) sharing 3 GB of GDDR5 memory. The M2050 is installed as a PCI-E x16v2 add-on card.

The BrainWeb (McConnel Brain Imaging Center 1997; Cocosco et al 1997; Kwan et al 1999) database provides reference MR images with corresponding noisy images that have been artificially generated using a mathematical model of real-world noise. From this collection, we chose 180 images spanning two anatomical models (normal vs. multiple sclerosis lesion), three modalities (T1, T2, PD), two slice thicknesses (1mm, 9mm), five noise levels (1%, 3%, 5%, 7%, 9%), and three levels of intensity non-uniformity (0%, 20%, 40%). Each 1mm thickness image has $181 \times 217 \times 181$ voxels, and each 9mm thickness image has $181 \times 217 \times 20$ voxels.

Because the size of the largest BrainWeb image (only 7.1 million voxels) may not be representative of clinical images obtained with modern MR imaging technologies, we have included several larger images from the Multi-Modal MRI Reproducibility Resource (Landman et al 2011). These images were downloaded from the BIRN Data Repository and come from several modalities, as summarized in Table 1.

3.3 Metrics for Evaluating Denoising Quality

Using the noiseless BrainWeb reference images as a baseline, we can calculate metrics to compare the original and denoised image, including the mean-squared error (MSE) and the mean structural similarity (MSSIM) (Wang et al 2004)). For a 3D volume $\mathbf{V}$ with $|\mathbf{V}|$ voxels, we calculated MSE as:

$$\frac{1}{|\mathbf{V}|}\sum_{i=0}^{|\mathbf{V}|}|\mathbf{V}'_i - \mathbf{V}_i|^2 \tag{11}$$

where $\mathbf{V}'$ is the filtered volume and $\mathbf{V}$ is the noiseless reference.

This methodology of measuring MSE against the BrainWeb reference data has been used previously in the literature on MR image denoising. Indeed, the intent of the BrainWeb authors was to provide a reference data set that could be controllably degraded to enable quantitative evaluation of MR image processing techniques. Coupe et al (2008)’s study of approximate non-local means for 3D MR images performs parameter sweeps on the BrainWeb data, using MSE as their metric for comparison. In a similar study, Manjón et al (2008) applied a 2D non-local means implementation to the BrainWeb data and used MSE to determine optimal parameters. The key difference between these studies and ours is that we are considering the exact 3D non-local means solution and are also tuning for optimal runtime parameters, in order to achieve real-time performance.

The mean structural similarity (MSSIM) is a more sophisticated metric that is based on known properties of visual perception. It aims to more accurately quantify how different two images appear to a human observer and has been validated against real subjective ratings of image similarity. We have implemented MSSIM as described by Wang et al (2004) in their Equations 13–17. Briefly, we calculate the SSIM around each voxel within a neighborhood of diameter 11 weighted by a Gaussian filter w with standard deviation 1.5:

$$\mu_x = \sum_{i=1}^{11^3} w_i x_i \tag{12}$$

$$\sigma_x = \left(\sum_{i=1}^{11^3} w_i (x_i - \mu_x)^2\right)^{\frac{1}{2}} \tag{13}$$

$$\sigma_{xy} = \sum_{i=1}^{11^3} w_i (x_i - \mu_x)(y_i - \mu_y) \tag{14}$$

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{15}$$

where $\mathbf{x}_j$ and $\mathbf{y}_j$ are the neighborhoods around the voxels $\mathbf{V}_j$ and $\mathbf{V}'_j$ respectfully. We used the values $C_1 = (0.01)^2$ and $C_2 = (0.03)^2$, as suggested by Wang et al (2004). Then, MSSIM is the average of these SSIM values across all voxels in the volume:

$$\mathrm{MSSIM}(\mathbf{V}, \mathbf{V}') = \frac{1}{|\mathbf{V}|}\sum_{j=1}^{|\mathbf{V}|}\mathrm{SSIM}(\mathbf{x}_j, \mathbf{y}_j). \tag{16}$$

3.4 Filter Parameters

Each of our filters has several tunable parameters that we divide into two groups: those that affect runtime, or *runtime* parameters, and those that do not, or *smoothing* parameters. We conducted a parameter sweep study to find the combinations of both runtime and smoothing parameters that yielded the best denoising quality.

The runtime parameters for the bilateral and non-local means filters are the spatial supports for the stencil: the half-radius

$R \in (1, 2, 3, 5, 7)$

for the bilateral filter and the window/patch sizes

$(W, P) \in ((1, 1), (2, 1), (3, 1), (3, 2), (5, 3))$

for non-local means. For anisotropic smoothing, the runtime parameter is the number of iterations

$N \in (1, 5, 10, 20, 50)$.

The smoothing parameters for the bilateral filter are the standard deviation

$\sigma_d \in (0.25, 0.5, 1, 2, 3, 4, 8, 16)$

in the spatial domain and the standard deviation

$\sigma_r \in (2, 4, 8, 16, 32, 64)$

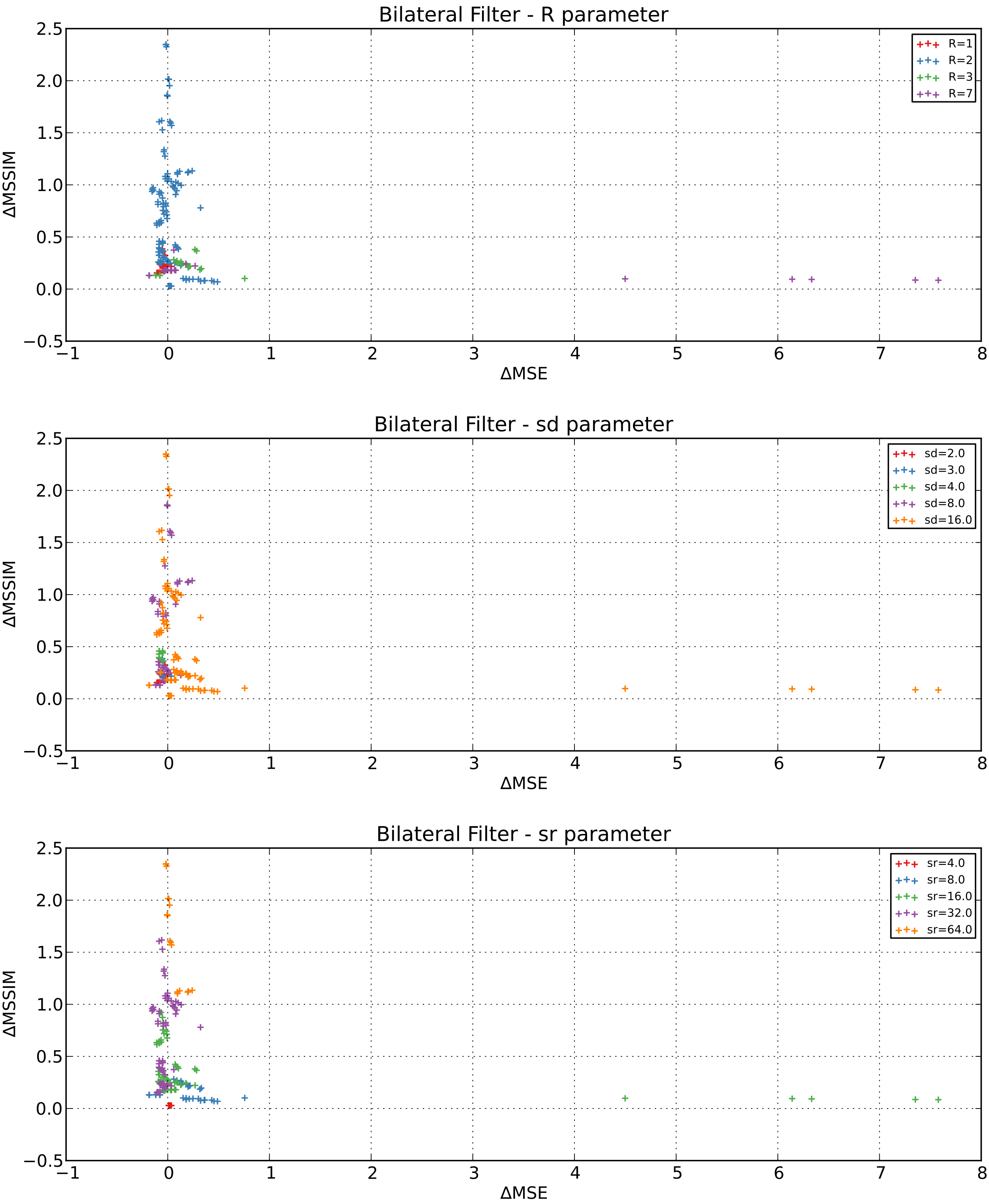


Fig. 2 The relative change in MSSIM and MSE after denoising with the bilateral filter over a range of standard deviations σ_d and σ_r and filter radii R. For each of the 180 BrainWeb images, the parameter configuration exhibiting the best increase in MSSIM is plotted against its correspoding change in MSE.

in the image range. For anisotropic smoothing, they are the size of discrete timesteps

$\varDelta t \in (1/128, 1/64, 1/32, 1/16, 1/8, 1/4)$

and the scaling factor

$\kappa \in (10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1)$

in the conductance function. For non-local means, there is a single scaling factor

$h \in (3.2, 1.6, 0.8, 0.4, 0.2, 0.1, 0.05, 0.025)$.

We also conducted an autotuning study to determine the optimal memory tiling across each filter's runtime parameters. Our 2D memory tiling optimization is highly sensitive to the dimensions of the CUDA thread blocks: each thread block loads a cache block of the same dimension (plus appropriate padding for the stencil calculation) into local cache. For the purposes of this study, we explored all combinations of power-of-two dimensions

$X \in (1, 2, 4, 8, 16, 32, 64, 128, 256, 512)$,

$Y \in (1, 2, 4, 8, 16, 32, 64, 128, 256, 512)$

for the thread blocks to get a thorough sense of the parameter space. We excluded (X, Y) pairs with fewer than 32 elements (too small to fill a warp of CUDA threads) and which would require more than 48KB of memory (larger than amount of local cache). In a production system, however, it may be sufficient to search a smaller parameter space using techniques like those discussed in Section 2.3.

4 Results

4.1 Denoising Quality

For each of the 180 BrainWeb images, we performed a sweep of 240 parameters configurations for the bilateral filter, 180 for anisotropic diffusion, and 40 for non-local means, as detailed in Section 3.4. Because the images have varying levels of added noise, we calculated the *relative change* of the MSE and MSSIM between the noisy image $\mathbf{V}'$ and the denoised image $\mathbf{V}''$, as

$$\varDelta \mathrm{MSE}(\mathbf{V}', \mathbf{V}'') = \frac{\mathrm{MSE}(\mathbf{V}'', \mathbf{V}) - \mathrm{MSE}(\mathbf{V}', \mathbf{V})}{\mathrm{MSE}(\mathbf{V}', \mathbf{V})} \qquad (17)$$

where $\mathbf{V}$ is the corresponding noiseless image (and similarly for $\varDelta$MSSIM). For each image, we identified the configuration that maximized $\varDelta$MSSIM, i.e. yielded the best improvement in structural similarity. Figures 2, 3 and 4 show this maximal value for each image, plotted against the corresponded value of $\varDelta$MSE.

Many of the best configurations for a given BrainWeb image actually *increased* the MSE of the denoised image relative to the noisy image. These configurations are seen as the positive relative change in MSE in Figures 2, 3 and 4. Yet, all of the best configurations improved structural similarity regardless of their effect on MSE, and in general there was poor correlation between these two error metrics. In contrast to these best configurations, many of the hundreds of other configurations that are not shown in the figures fared poorly, causing decreases in structural similarity and in some cases order-of-magnitude increases in MSE. This variation demonstrates the need to carefully choose appropriate parameters for these filters.

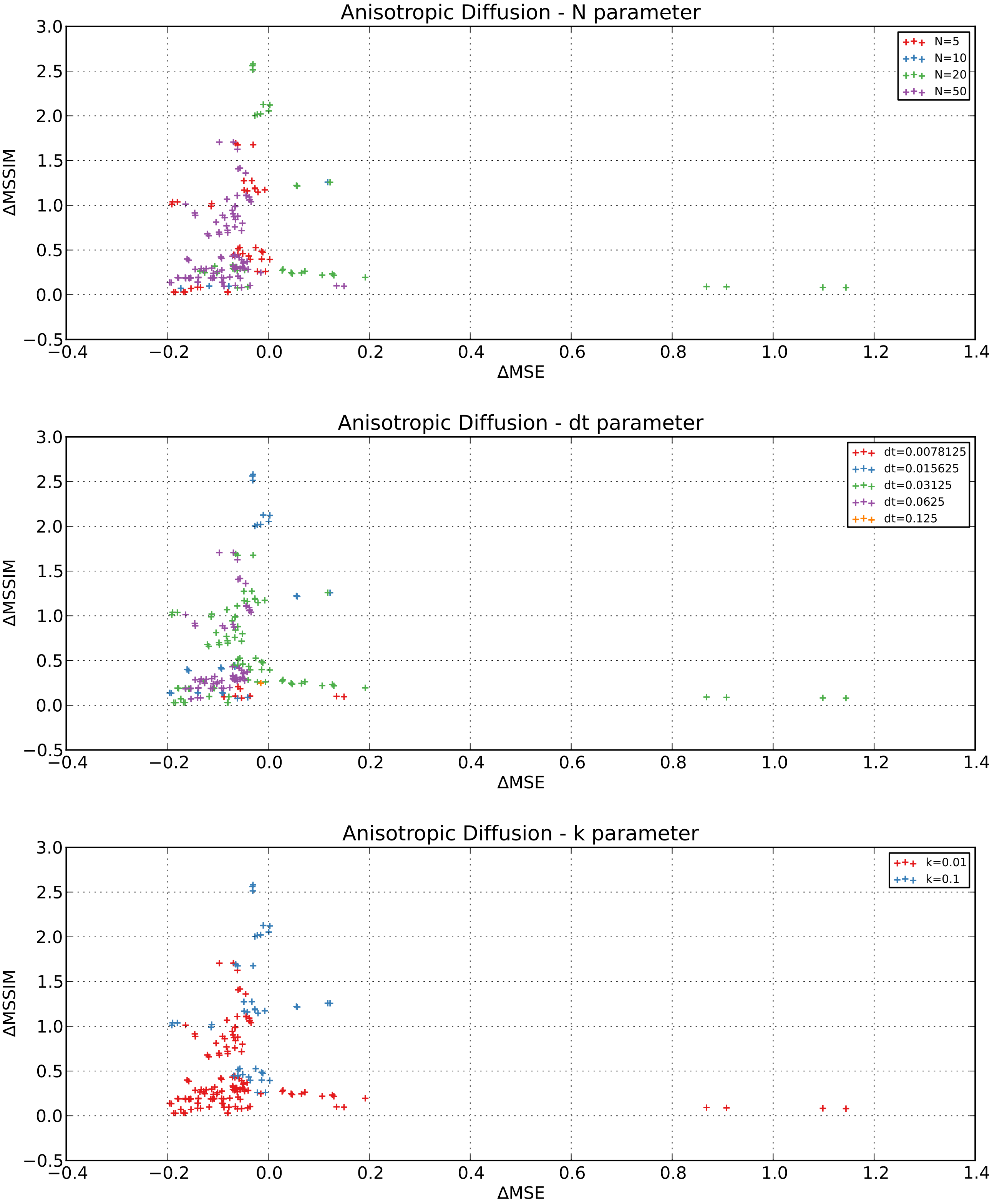
Surprisingly, the bilateral filter yielded the best improvement in structural similarity for most of the BrainWeb images at a stencil size of only $R = 2$. In contrast, anisotropic diffusion and non-local means achieved the best improvement in structural similarity at relatively greater iterations ($N = 20$) or larger stencil size ($(W, P) = (3, 2)$). All filters, however, achieved similar improvements in structural similarity, peaking around 2.5×.

4.2 Runtime

We chose the runtime parameters that exhibited the maximal $\varDelta$SSIM for further analysis: $R = 2$, $N = 20$, and $(W, P) = (3, 2)$. In Figure 5, we show an autotuning study of cache dimensions that varies the size of the input data (across the five MMRR test images) to address the question of whether this autotuning strategy scales to larger images. The best runtime scaled linearly with image size, except for a slight super-linear speed-up between the second and third image for the bilateral filter. This is likely because the Z-dimension changes so drastically between those two images, and memory access is slowest in the Z direction.

The spread of the scatter points for each image in Figure 5 shows the variance in runtime between best and worst performing cache dimensions. Autotuning yielded significant speed-ups over the worst-case performance, as the overall variation in runtime across all the tested dimensions varied by as much 4.8× for bilateral filtering, 3.4× for anisotropic diffusion and 4.5× for non-local means. If we define real-time execution as within 1/8 of a second, then the bilateral filter at $R = 2$ achieved real-time execution for the two smallest MMRR images.

One disadvantage of using a GPU accelerater is the cost of transferring data between main memory on the host system and the GPU's memory. This takes place over the PCI Express x16 Gen2 bus on our test system, which has a theoretical bandwidth of 8GB/s, a factor of 18.5× less than the theoretical memory bandwidth of the onboard GPU memory for the M2050 (148GB/s). We measured the median time across all of our autotuning tests to copy the MMRR datasets to and from GPU memory as 4-byte float values (see Table 2). For the largest dataset, FLAIR, real-time performance is not possible because the transfer alone takes more than 125ms.



Anisotropic Diffusion - N parameter

Anisotropic Diffusion - dt parameter

Anisotropic Diffusion - k parameter

Fig. 3 The relative change in MSSIM and MSE after denoising with the 26-point anisotropic diffusion filter with PM2 conductance function over a range of iterations N, conductance parameters κ, and time steps dt. For each of the 180 BrainWeb images, the parameter configuration exhibiting the best increase in MSSIM is plotted against its correspoding change in MSE.

Table 2 Median Host/GPU Transfer Time and Bandwidth

Dataset	M. Voxels	Time To	Bandwidth To	Time From	Bandwidth From
T2/TRA	8.3	8 ms	4039 MB/s	8 ms	3811 MB/s
MPRAGE	13.1	13 ms	4026 MB/s	13 ms	3823 MB/s
T2/2D	21.0	20 ms	4048 MB/s	21 ms	3868 MB/s
T1	30.9	30 ms	4075 MB/s	32 ms	3859 MB/s
FLAIR	99.5	130 ms	3057 MB/s	129 ms	3085 MB/s

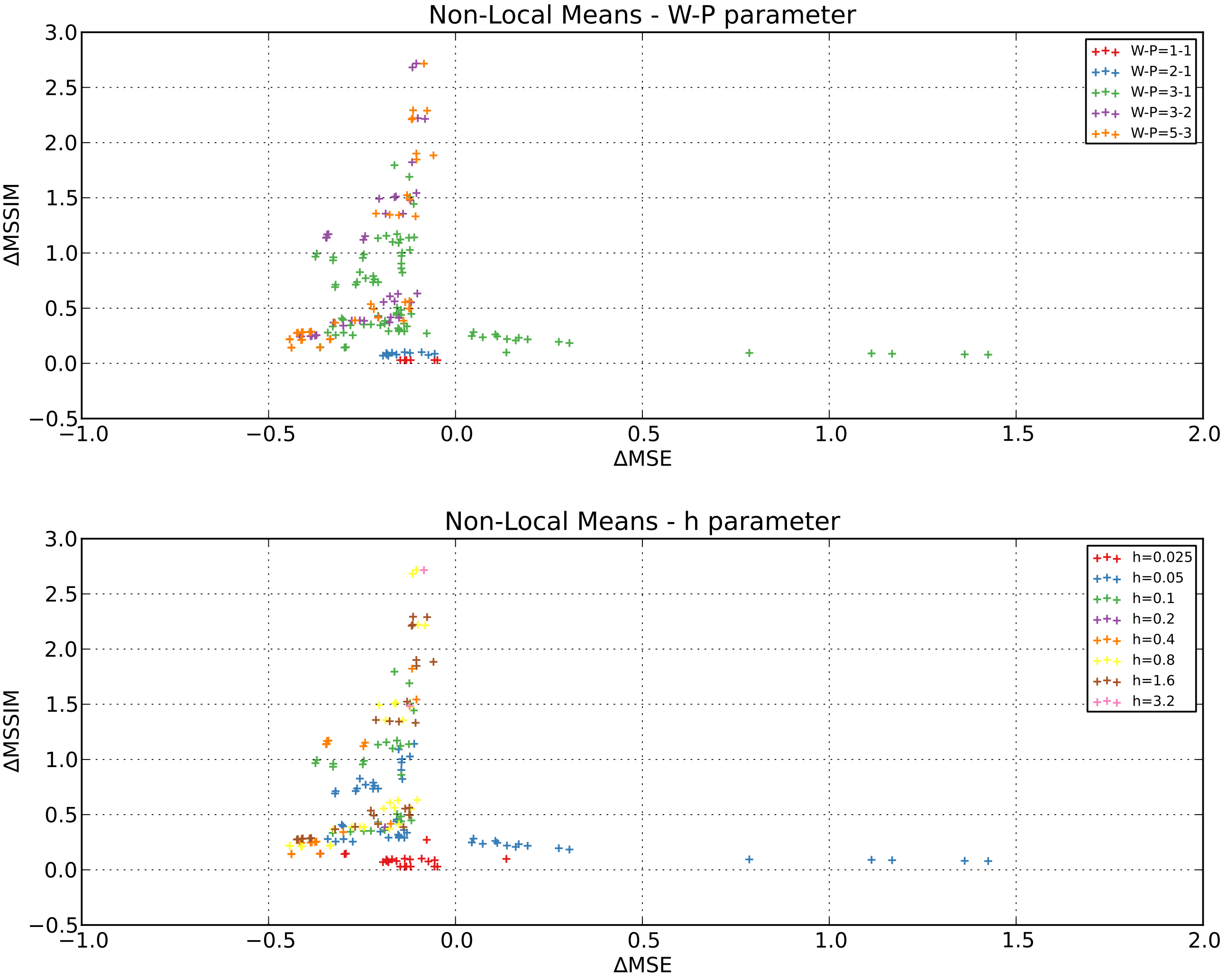


Fig. 4 The relative change in MSSIM and MSE after denoising with the non-local means filter over a range of window sizes W, patch sizes P, and scaling parameters h. For each of the 180 BrainWeb images, the parameter configuration exhibiting the best increase in MSSIM is plotted against its correspoding change in MSE.

5 Conclusions and Future Work

Our analysis of noise reduction for these three denoising filters, as measured by MSE and structural similarity, suggest that it is easy to choose parameters that either oversmooth or provide diminishing returns for their cost in runtime. The best denoising achieved by the bilateral filter was for parameter combinations that also achieved real-time execution on an NVIDIA M2050 GPU for the smaller MMRR test images.

Three promising directions for future work could help achieve real-time execution for the larger MMRR images and with the best anisotropic diffusion and non-local means configurations. First, NVIDIA is now shipping the Tesla K10 (Kepler architecture). It has a memory bandwidth of 320GB/s and supports 4.58 teraflops of single precision arithmetic. These are 2.2× and 4.5× greater

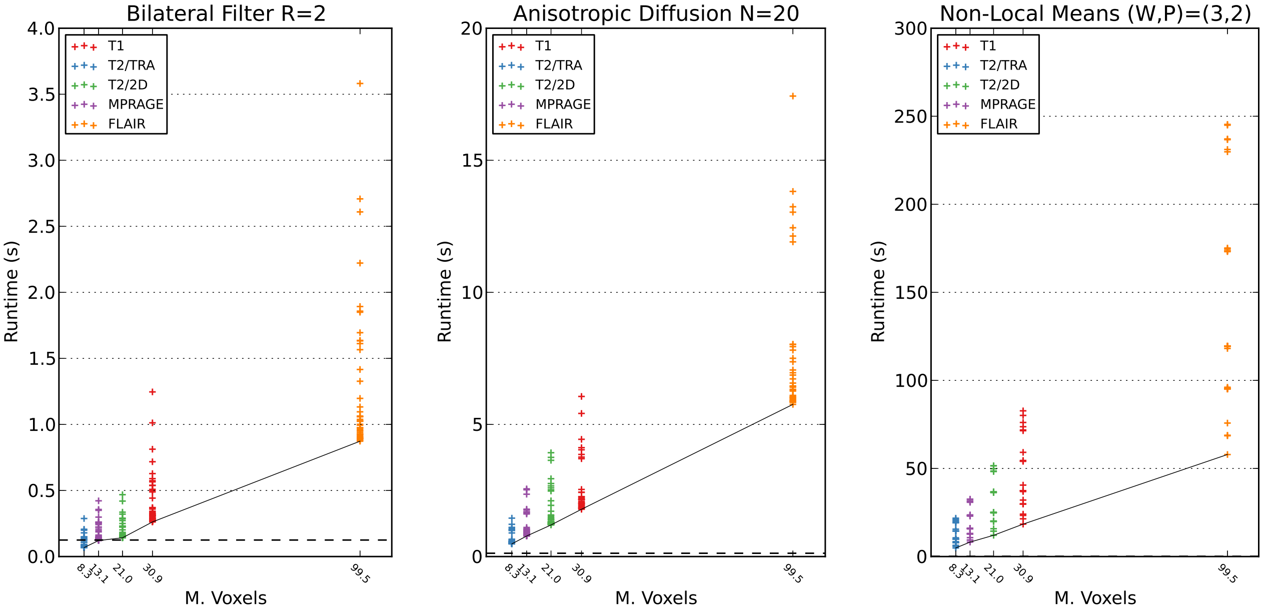


Fig. 5 Scatter plots summarize the distribution of runtimes (including transfer times) for the memory block configurations we autotuned across. Runtimes increased linearly with the number of voxels in the MMRR images, except for a slight super-linear increase for the bilateral filter. A few configurations for the bilateral filter achieve real-time results, indicated by the dotted line at $y = 0.125$.

than the bandwidth and peak arithmetic of the M2050, respectively. Therefore, we would expect at least a 2× reduction in runtime simply by moving to this newer architecture.

Second, we could extend the optimized algorithm for non-local means described by Darbon et al (2008) to 3D images and implement it for GPUs. So far, it has only been implemented and tested on CPUs and it requires storing several temporary buffers during the precomputation, which may overrun the relatively small 48KB local cache on the M2050. Even the Tesla K10 only has 64KB of local cache. While the factor of $O(P^3)$ savings becomes significant for large values of P (for instance, Darbon et al (2008) use $(W, P) = (7, 3)$ on a 2D image), our parameter sweep suggests that non-local means works best around $(W, P) = (3, 2)$, and so the savings may not be significant.

A third solution would be to use domain decomposition to parallelize the denoising computation across multiple GPUs, using for instance an MPI+CUDA framework. Based on a previous study we conducted (Bethel 2009), we expect this would scale close to linearly for a small number of MPI tasks, but this speed-up would be offset by the added latency of performing the reduction necesssary to output the final denoised image. As the size of data sets that can be generated from MR instruments increases, we suspect that this approach – the use of domain decomposition and distributed-memory parallelism – will be the most promising way to meet the real-time processing constraints for 3D MR image denoising.

Acknowledgements Thanks to Dani Ushizima, Alex Cunha, and Owen Carmichael for their comments and input on earlier versions of this study, and to D. Louis Collins for following up with us on problems accessing the BrainWeb database.

This work was supported by the Director, Office of Science, Office and Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was conducted using computational resources and services at the Center for Computation and Visualization, Brown University.

Data used for this study were downloaded from the Biomedical Informatics Research Network (BIRN) Data Repository (http://www.nbirn.net/bdr), supported by grants to the BIRN Coordinating Center (U24-RR019701), Function BIRN (U24-RR021992), Morphometry BIRN (U24-RR021382), and Mouse BIRN (U24-RR021760) Testbeds funded by the National Center for Research Resources at the National Institutes of Health, U.S.A.

References

Bethel EW (2009) High Performance, Three-Dimensional Bilateral Filtering. Tech. Rep. LBNL-1601E, Lawrence Berkeley National Laboratory

Buades A, Coll B, Morel JM (2005) A non-local algorithm for image denoising. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2, IEEE Computer Society, Washington, DC, USA, CVPR '05, p 6065, DOI 10.1109/CVPR.2005.38, URL `http://dx.doi.org/10.1109/CVPR.2005.38`

Chen J, Paris S, Durand F (2007) Real-time Edge-aware Image Processing with the Bilateral Grid. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, ACM, New

York, NY, USA, p 103, DOI http://doi.acm.org/10.1145/1275808.1276506

Cocosco CA, Kollokian V, Kwan RKS, Pike GB, Evans AC (1997) BrainWeb: Online Interface to a 3D MRI Simulated Brain Database. NeuroImage 5:425

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(4):425 –441, DOI 10.1109/TMI.2007.906087

de la Cruz R, Araya-Polo M (2011) Towards a Multi-level Cache Performance Model for 3D Stencil Computation. In: Procedia Computer Science, Proceedings of the International Conference on Computational Sciences, ICCS, vol 4, pp 2145–2155

Darbon J, Cunha A, Chan T, Osher S, Jensen G (2008) Fast nonlocal filtering applied to electron cryomicroscopy. In: Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'08), pp 1331 –1334, DOI 10.1109/ISBI.2008.4541250

Datta K, Williams S, Volkov V, Carter J, Oliker L, Shalf J, Yelick K (2009) Auto-tuning the 27-point Stencil for Multicore. In: 4th International Workshop on Automatic Performance Tuning (iWAPT)

Eklund A, Andersson M, Knutsson H (2011) True 4D image denoising on the GPU. International Journal of Biomedical Imaging 2011, DOI 10.1155/2011/952819, URL http://dx.doi.org/10.1155/2011/952819

Ganapathi A, Datta K, Fox A, Patterson D (2009) A case for machine learning to optimize multicore performance. In: Proceedings of the 1st USENIX Conference on Hot Topics in Parallelism (HotPar '09), USENIX Association, Berkeley, CA, USA, HotPar'09, p 11, URL http://dl.acm.org/citation.cfm?id=1855591.1855592

Garcia C, Botella G, Ayuso F, Prieto M, Tirado F (2013) Multi-gpu based on multicriteria optimization for motion estimation system. EURASIP Journal on Advances in Signal Processing 2013(1):23

Gerig G, Kübler O, Kikinis R, Jolesz FA (1992) Nonlinear Anisotropic Filtering of MRI Data. IEE Transactions on Medical Imaging 11(2):221–232

Hamarneh G, Hradsky J (2007) Bilateral Filtering of Diffusion Tensor Magnetic Resonance Images. IEEE Transactions on Image Processing 16(10):2463–2475, DOI 10.1109/TIP.2007.904964

Hollingsworth J, Tiwari A (2010) End-to-end Auto-tuning with Active Harmony. In: Bailey DH, Lucas RF, Williams SW (eds) Performance Tuning of Scientific Applications, CRC Press

Howison M (2010) Comparing GPU implementations of bilateral and anisotropic diffusion filters for 3D biomedical datasets. Chicago, IL, USA, URL http://vis.lbl.gov/Publications/2010/LBNL-3425E.pdf

Ibanez L, Schroeder W, Ng L, Cates J (2003) The ITK Software Guide: The Insight Segmentation and Registration Toolkit. Kitware

Kamil S, Chan C, Oliker L, Shalf J, Williams S (2010) An Auto-tuning framework for Parallel Multicore Stencil Computations. In: International Parallel & Distributed Processing Symposium (IPDPS)

Kwan R, Evans A, Pike G (1999) MRI simulation-based evaluation of image-processing and classification methods. IEEE Transactions on Medical Imaging 18(11):1085 –1097, DOI 10.1109/42.816072

Landman BA, Huang AJ, Gifford A, Vikram DS, Lim IAL, Farrell JA, Bogovic JA, Hua J, Chen M, Jarso S, et al (2011) Multi-parametric neuroimaging reproducibility: A 3-t resource study. NeuroImage 54(4):28542866

Magni A, Grewe D, Johnson N (2013) Input-aware autotuning for directive-based gpu programming. In: Proceedings of the 6th Workshop on General Purpose Processor Using Graphics Processing Units, ACM, GPGPU-6, p 6675

Mahmoudi M, Sapiro G (2005) Fast image and video denoising via nonlocal means of similar neighborhoods. IEEE Signal Processing Letters 12(12):839 – 842, DOI 10.1109/LSP.2005.859509

Manjón JV, Carbonell-Caballero J, Lull JJ, Garca-Mart G, Mart-Bonmat L, Robles M (2008) MRI denoising using non-local means. Medical Image Analysis 12(4):514–523, DOI 10.1016/j.media.2008.02.004, URL http://www.sciencedirect.com/science/article/pii/S1361841508000248

McConnel Brain Imaging Center MNI (1997) http://www.bic.mni.mcgill.ca/brainweb/

NVIDIA Corporation (2012) CUDA C Programming Guide. URL http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Perona P, Malik J (1990) Scale-Space and Edge Detection Using Anisotropic Diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(7):629–639

Rivera G, Tseng C (2000) Tiling optimizations for 3D scientific computations. In: SC '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing

Ryoo S, Rodrigues CI, Stone SS, Baghsorkhi SS, Ueng SZ, Stratton JA, mei W Hwu W (2008) Program optimization space pruning for a multithreaded GPU. In: CGO '08: Proceedings of the Sixth Annual IEEE/ACM International Symposium on Code Generation and Optimization, pp 195–204

Seymour K, You H, Dongarra J (2008) A comparison of search heuristics for empirical code optimization. In: 2008 IEEE International Conference on Cluster Computing, pp 421 –429, DOI 10.1109/CLUSTR.2008.4663803

Stone SS, Haldar JP, Tsao SC, Hwu WmW, Sutton BP, Liang ZP (2008) Accelerating advanced mri reconstructions on gpus. J Parallel Distrib Comput 68(10):1307–1318, DOI http://dx.doi.org/10.1016/j.jpdc.2008.05.013

Tomasi C, Manduchi R (1998) Bilateral Filtering for Gray and Color Images. In: ICCV '98: Proceedings of the Sixth International Conference on Computer Vision, IEEE Computer Society, Washington, DC, USA, p 839

Wang Z, Bovik A, Sheikh H, Simoncelli E (2004) Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13(4):600612

Weickert J, Benhamouda B (1997) Why the Perona-Malik filter works. Tech. Rep. DKIU-TR-97/22, Department of Computer Science, University of Copenhagen

Williams S, Datta K, Oliker L, Carter J, Shalf J, Yelick K (2010) Auto-tuning Memory-Intensive Kernels for Multicore. In: Bailey DH, Lucas RF, Williams SW (eds) Performance Tuning of Scientific Applications, CRC Press

Zheng Z, Xu W, Mueller K (2011) Performance tuning for CUDA-accelerated neighborhood denoising filters. In: 3rd Workshop on High-Performance Image Reconstruction (HPIR), Potsdam, Germany, p 5255