DENOISING VOLUMETRIC DATA ON GPU

Jan Horáček, Jan Kolomazník, Josef Pelikán
Faculty of Mathematics and Physics, Charles University in Prague, Ke Karlovu 3, Prague, Czech Republic

Martin Horák
Radiology Department, Na Homolce Hospital, Prague, Czech Republic

Keywords: Denoising, Volumetric data, GPU, OpenCL, Nonlocal means.

Abstract: Volumetric data is currently gradually being used more and more in everyday aspect of our lives. Processing such data is computationally expensive and until now more sophisticated algorithms could not be used. The possibilities of processing such data have considerably widened since the increase of parallel computational power in modern GPUs.
We present a novel scheme for running a nonlocal means denoising algorithm on a commodity-grade GPU. The speedup is considerable, shortening the time needed for denoise one abdominal CT scan in minutes instead of hours without compromising the result quality. Such approach allows for example lowering the radiation doses for patients being examined with a CT scan.

1 INTRODUCTION

Many current medical imaging methods produce volumetric data, such as computed tomography (CT), magnetic resonance (MRI) and others. These data provide good insight into the workings of the human body, but also are quite large and it is difficult to process them with more advanced processing techniques. One example of such examination is CT Enterography - a method for diagnosis Crohn's disease of small intestine (Federle, 2007) (Paulsen et al., 2006). It is a noninvasive method of *postcontrast* small intestine CT (computed tomography) examination. It combines the speed and resolution of multidetector CT scanners with enhancing properties of both ingested and intravenous contrast agent. The visualization of the intestinal wall and lumen is much better than by performing a normal CT scan or other techniques and clearly shows small intestine inflammation by displaying the thickening of the intestinal wall.

For the examination of complicated structure and small details we need thin slices and also low radiation doses to avoid harming the patient. But thin slices bring a lot of noise with it and are thus very hard to follow and segment with automatic or semiautomatic methods. Many current segmentation algorithms need a robust edge detection. In case of an average CT enterography scan, the standard deviation of the *homogeneous* inner parts of the intestine (lumen filled with negative contrast agent) is bigger than the difference between the mean value of the lumen and the contrast enhanced intestinal wall. We are talking only about the random noise present in the data and discarding the effects of acquisition artifacts. Therefore we need a robust denoising approach to apply some proven and efficient segmentation algorithm.

We will only focus on random noise, which is very apparent in the intestinal part of the body. Acquisition artifacts, such as *star* artifacts resulting from dense objects being present in the body are not so apparent, because unless the patient has some sort of metal implant this part of the body usually contains only the spine and upper part of pelvis and no other dense bones or objects. For example (Gu et al., 2006) discusses a method of star artifact removal.

2 DENOISING APPROACHES

A thorough description of existing denoising techniques is not given due to the space constraints. We only refer to currently used algorithms, such as Gaussian filter, median filter in (Gallagher and Wise, 1981), total variation minimization in (Rudin and Osher, 1981) (Rudin et al., 1992) and nonlocal means

Horáček J., Kolomazník J., Pelikán J. and Horák M..
DENOISING VOLUMETRIC DATA ON GPU.
DOI: 10.5220/0003355001430146
In *Proceedings of the International Conference on Imaging Theory and Applications and International Conference on Information Visualization Theory and Applications* (IMAGAPP-2011), pages 143-146
ISBN: 978-989-8425-46-1

filter and a survey given in (Buades et al., 2005) with improvements in (Coupe et al., 2008).

3 NONLOCAL-MEANS ALGORITHM

First introduced by (Buades et al., 2005) this algorithm is nowadays considered one of the best approaches quality-wise. It has very good properties in respect to detail preservation and very successfully removes white noise. However the computational complexity is very high, especially for 3D data.

The definition is as follows:

$$NL(u)(x_i) = \sum_{x_j \in \Omega^3} w(x_i, x_j) u(x_j) \quad (1)$$

$$w(x_i, x_j) = \frac{1}{Z_i} e^{-\frac{||u(N_i) - u(N_j)||^2_{2,a}}{h^2}} \quad (2)$$

where u is the original noisy image, Ω^3 is the definition range of the image, w is the weight computed from the similarity of the local neighbourhoods of two voxels and N_i, N_j are local neighbourhoods around given voxels, h is a filtering parameter and Z_i is a normalization constant.

This can be explained as follows: Each voxel is reconstructed by the weighted averaging of the most similar voxels in its vicinity. The similarity of two voxels is computed from the L2 norm of their neighbourhoods - voxelwise.

It has been proven in (Buades et al., 2005) that NL-Means is a very efficient algorithm (quality-wise) and performs an optimal denoising. But the computational complexity ($O(n * m * k)$, where n = number of voxels, m = size of the search space and k = size of local neighbourhoods for computing similarity) is nowadays too large for a daily medical praxis.

3.1 NL Means Optimizations

NLM has been tried on GPU, mostly for photos and other 2D or 1D datasets, e.g. (Kharlamov and Podlozhnyuk, 2007). The optimization was to use the block approach without overlaps, which brings strong block artifacts and the whole implementation is done only for 2D images.

Good optimizations for 3D data are given in (Coupe et al., 2008). The original aim was denoising of the MRI datasets of head and brain, but they are good for CT images as well. The basis is a selection of relevant voxels based on local mean and variance values before the L2 norm is computed.

Combining this voxel selection with blockwise approach from (Buades et al., 2005) brings a very significant speedup, the authors claim as much as 40x to 66x. No implementation details are given though. We did not manage to achieve such performance without drastically reducing the quality of result.

As a result of these optimizations, the algorithm is able to run in merely minutes instead of hours. Our implementation has achieved approximately 20-30 minutes on a quad core i7 processor for a volume of size 512x512x548.

4 OpenCL CAPABLE GPU

Current generation of GPUs supporting OpenCL interface (Khronos OpenCL Working Group, 2009) is designed to process very efficiently large tasks that are parallelizable. The sheer computational power is one order higher than a commodity-grade CPU. However, efficient use needs some practice and optimizations that are unnecessary on CPU (NVIDIA Corporation, 2008) (NVIDIA Corporation, 2009).

4.1 Basic HW Implementation of Optimized Version

After implementing optimized NL means for the OpenCL interface, we have found out that on a GeForce GTX 275 it is only about 6 times faster than on a single 2.6GHz CPU. However, current processors have usually more than one core and thus are already able to perform parallel computations. Running this version on a Core i7 with 4 cores and hyperthreading, the speed difference was even after careful optimization of the GPU version negligible, even though a GeForce was running many more threads concurrently.

This is caused by a very slow memory access and no local cache on this version of GeForce. About 90% of the computational time was spent on two memory reads, thus resulting 45% each. Also the process of choosing which voxels will be computed breaks the continuity of the thread warps and slows down the computation instead of speeding it up. This algorithm is thus unsuitable for direct implementaton on GPU and needs to be changed.

4.2 HW Implementation of the Original NL Means

We tried reimplementing the original NL-means, this time to maximally utilize the capabilities of the GPU

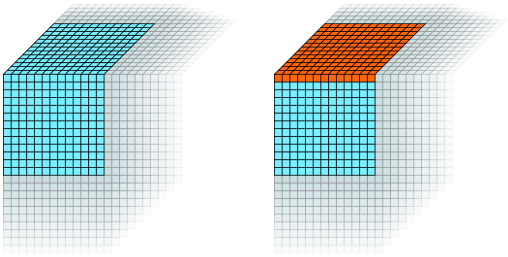

Figure 1: Blue voxels are the search space, orange voxels are highlighted for explanation, currently processed voxel is in the middle of the blue voxels: (left) Volume needed for one voxel. (right) One thread is executed for each column.

with minimized branching and global memory access. That means preloading the data to the fast on-chip memory and no voxel selection.

The parameters of NL-means filter for best quality are taken from (Coupe et al., 2008): local neighbourhood radius 2 voxels, search radius 4 or 5 voxels (only small difference in quality) and automatic smoothing parameter computed from pseudoresiduals.

Thus memory consumption reconstruction of one voxel for search radius 4 would be: number of voxels $= (2 \cdot (4+2)+1)^3 = 2197$, so source data size $= 2197 \cdot 4$ bytes per float = 8788 bytes. To that we add temporary memory for weights $= (2 \cdot 4+1)^3 = 729$ floats $= 2916$ bytes and additional memory for summing weights = 360 bytes.

This fits completely into the local memory and thus we will use these parameters. Search radius of 5 voxels would not fit into the on-chip memory, but the quality difference between 4 and 5 voxels in means of signal to noise ratio is shown in (Coupe et al., 2008) to be small enough.

The algorithm itself:

1. Each thread reads the first $2 \cdot (4+2)+1$ voxels in the Z direction from global memory to local memory (Figure 1(a, b)).
2. For each relevant voxel, compute the weight function with central voxel (Figure 1(c)).
3. Compute the sum of weigths.
4. Compute the sum of weight multiplied by the value for each relevant voxel.
5. Normalize with weight sum.
6. Store result into global memory.
7. In each thread move all voxels in local memory by 1 and read one new voxel (Figure 1(d)).
8. Continue with step (Figure 2).

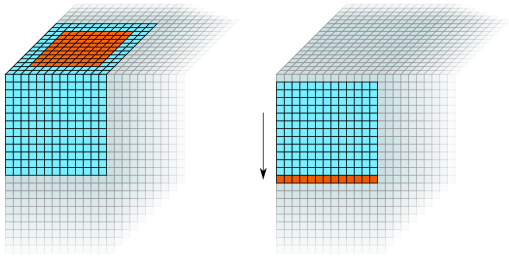

Figure 2: (left) Threads actually computing weights and sum. (right) Loading next slice.

Algorithm	Processor	Threads	Time
NL-Means	i7 3.07GHz	8	5:38:15
Block NLM	i7 3.07GHz	8	0:26:35
NL-Means	C2Q 2.4GHz	4	$>$ 10 hours
Block NLM	C2Q 2.4GHz	4	0:55:57
NL-Means	GF 8800GT	13^2	0:13:41
NL-Means	GF 275GTX	13^2	0:06:44
NL-Means	GF 460GTX	13^2	0:04:22

Figure 3: Measured on dataset of size 512x512x548.

4.3 New HW Implementation of the Optimized Version

We also tried optimized selection of computed voxels by using this presented algorithm with good utilization of the local memory. However, the results were much worse than on the CPU. The algorithm has been running about 2x slower than unoptimized version in 4.2. It was running about the same speed as the unoptimized version only with drastic reduction in quality. The speed was strongly dependent on the actual data (how much of the data was discarded).

It turned out that the original brute-force version without voxel selection is faster than the optimized one.

5 RESULTS

The times measured for given algorithms are shown in figure 3. As you can see, the simple and computationally very expensive algorithm runs much faster on current GPU than the most optimized CPU version on 8-threads on a good current CPU. It should be stressed, that the results on CPU are already parallelized and the comparison is done with the *full CPU power* and not a single-threaded version.

The application performance has been validated with a profiler. The *instruction throughput* on a GT200 was 0.921813. The *retired instruction per cycle* on a GF104 was 1.86737.

6 CONCLUSIONS

GPU version on a OpenCL-enabled graphics card allows denoising of the whole patient dataset in merely minutes instead of hours and does not bring any artifacts created with aggressive optimizations needed for CPU. Thus we consider this a good approach for processing such intensive problems.

We have also validated that the optimized version (with selection of voxels), which brings about one order speedup on CPU, is actually slower on GPU due to the breaking of the thread coherency with conditional jumps and the limited size of the on-chip memory which results in lenghty memory fetches.

A full evaluation of quality is not given in this paper, it can be found in (Coupe et al., 2008)

The profiling has shown that we have successfully eliminated the memory bandwidth problem, on a GT200 architecture is room only for cca 8% improvement onto the peak theoretical performance.

However, on the new GF104 the performance is about half of the theoretical peak. This is caused by heavy usage of the on-chip memory - two warps do not fit on a single SM and this prevents the HW from executing 4 instructions per cycle. We have not found a way to reach this limit without drastically increasing the bandwith dependency or quality of denoising.

Future improvements may be in implementing the blockwise approach for GPU (will very probably bring substantial performance improvement at the cost of result quality) and overcoming the one-warp-per-multiprocessor limit when running on a GF104 and newer architectures. Another improvement may be trying to implement a GPU version of (Darbon et al., 2008), which has even lower computational complexity than the original NLM.

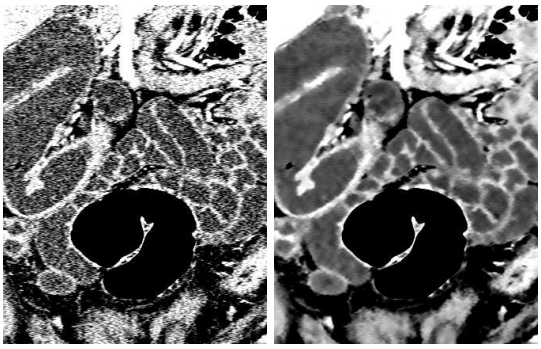

Figure 4: Example of denoised result: (left) Original data. (right) Denoised data with OpenCL GPU implementation of the original algorithm.

ACKNOWLEDGEMENTS

This work was supported by the Grant Agency of Charles University, Prague (project number 121409).

REFERENCES

Buades, A., Coll, B., and Morel, J. (2005). A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530.

Coupe, P., Yger, P., Prima, S., Hellier, P., Kervrann, C., and Barillot, C. (2008). An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images. *Medical Imaging, IEEE Transactions on*, 22.

Darbon, J., Cunha, A., Chan, T. F., Osher, S., and Jensen, G. J. (2008). Fast nonlocal filtering applied to electron cryomicroscopy. In *ISBI'08*, pages 1331–1334.

Federle, M. (2007). CT of the small intestine: Enterography and angiography. *Applied Radiology*.

Gallagher, N. and Wise, G. (1981). A theoretical analysis of the properties of median filters. *IEEE Transactions on Acoustic, Speech and Signal Processing*, ASSP-29(6).

Gu, J., Zhang, L., Yu, G., Xing, Y., and Chen, Z. (2006). X-ray CT metal artifacts reduction through curvature based sinogram inpainting. *Journal of X-Ray Science and Technology*, 14(2):73–82.

Kharlamov, A. and Podlozhnyuk, V. (2007). Image denoising. Technical report, nVidia Corporation.

Khronos OpenCL Working Group (2009). The OpenCL specification. Technical report, Khronos OpenCL Working Group.

NVIDIA Corporation (2008). OpenCL Programming guide. Technical report, NVIDIA Corporation.

NVIDIA Corporation (2009). OpenCL Best practices guide. Technical report, NVIDIA Corporation.

Paulsen, S., Huprich, J., Fletcher, J., Booya, F., Young, B., Fidler, J., Johnson, C., Barlow, J., and Earnest IV, F. (2006). CT enterography as a diagnostic tool in evaluating small bowel disorders: Review of clinical experience with over 700 cases. *RadioGraphics*, 26(3):641–657.

Rudin, L. and Osher, S. (1981). Total variation based image restoration with free local constraints. *IEEE Transactions on Image Processing*, 1:31–35.

Rudin, L., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60.