Blind image deconvolution algorithm on NVIDIA CUDA platform

Tomáš Mazanec and Antonín Heřmánek
Department of Signal Processing
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Email: {mazanec, hermanek}@utia.cas.cz

Jan Kamenický
Department of Image Processing
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Email: kamenik@utia.cas.cz

***Abstract*—Advanced image processing algorithms usually require high computing performance. Today's personal computers (PCs) offer satisfying resources for implementation of image processing tasks. However, as the image processing techniques are becoming more and more complex other implementation possibilities have to be searched.**

Since image processing algorithms usually comply with the Single Instruction Multiple Data (SIMD) model, implementation efforts using such hardware resources are suitable. An example of the SIMD hardware component available nowadays is the graphics processor (GPU) embedded in modern graphics cards manufactured for PCs.

In this paper, the implementation of a blind image deconvolution algorithm using graphics processor as the SIMD computing resource is presented. The resulting performance is compared to the performance achieved on a common general-purpose processor (CPU).

I. Introduction

Advanced image processing algorithms often require high computing performance. The complexity of such algorithms can easily drain available hardware resources. General-purpose processor based PC is usually the initial platform that a developer would use for their implementation. The absence of sufficient computing performance can become a limiting factor. One of the solutions to overcome the performance insufficiency is to extend the hardware platform (PC) with a computing accelerator.

There exist well proven hardware platforms for computing acceleration (DSP and FPGA for example), but another one arose from the area of PC in recent years. The evolution of the PC graphics cards brought new processing units, graphics processors (GPU), with significantly higher computing performance to offer. The advantages of GPUs used as a computing accelerator are seen in their common availability and so in their easier cooperation with the host hardware platform.

Image processing algorithms usually process images in a fashion similar to the SIMD model. Since the graphics processors' architectures are based on the SIMD hardware model, it is convenient to test the GPU as an image processing accelerator.

II. Compute Unified Device Architecture

Parallel computing architecture known as *CUDA - Compute Unified Device Architecture* [1] provides the computing engine in NVIDIA graphics processing units (GPU). The engine features are accessible to software developers through industry standard programming languages (C with NVIDIA extensions, etc.).

Since PC gained the capability of 3D imaging, each graphics card contained a specialized hardware parts for high efficiency computing. Later re-design of graphics cards architectures to a unified structure included a generic computing capabilities and thus became powerful instrument for scientific computations.

Theoretical computing performance of modern graphics cards exceeds general-purpose processors in order of magnitudes. Modern GPUs with unified architecture comply with SIMD model and the GPU benefits from the massive parallelism, provided by a huge number of lightweight threads. All GPU threads execute the same instructions at once, but on different pieces of distributed data.

If the specific algorithm or its part fits the SIMD model, the use of CUDA can lead to substantial increase of computing performance.

III. Problem definition

Given an image (a photo) impaired by the effects of blur and camera movement, the goal of blind deconvolution methods is to restore the original image without previous knowledge of these degrading effects.

Applying stochastic approach and assuming linearity, the image degradation can be modelled as multichannel convolution of an unknown system $h(x)$ and original image $u(x)$ with added noise $n(x)$:

$$z_k(x) = (h_k \star u)(x) + n_k(x)\,,\ k\ = 1, \ldots, K\ ,$$

where k denotes the channel and K is the number of channels.

In [2], authors presented the method of *Multichannel Blind Deconvolution of Spatially Misaligned Images* and they prove theirs approach of image restoration using an alternating minimization (AM) algorithm for the maximum a posteriori probability (MAP) estimator. The algorithm implementation was then derived to a solution of linear problem $\mathbf{Ax} = \mathbf{b}$, where the matrix $\mathbf{A}$ is consistently updated. Note that this update includes image-blur convolutions discussed in this paper.

978-1-4244-6610-8/10/$26.00 ©2010 IEEE

IV. Implementation details

The blind deconvolution algorithm can be split into a set of single tasks if the following assumption is applied: Camera movement (blur and spatial shift of the image) can have only a specific nature and thus there is only a small set of possible blur matrices that describe these degrading effects. In other words, the original image estimates are enumerated by a group of single algorithm tasks, whereas each task processes different presumed estimate of possible blur system matrix.

From the implementation point of view, the algorithm iteratively enumerates key properties of the original image estimate and blur system properties to achieve required energy minima (algorithm optimum). Including an initial estimate of original image, particular steps of the algorithm can be briefly described as: construction of image derivative, enumeration of gradients, enumeration of energy function and update of the image estimate according to previous results.

The study of the deconvolution algorithm and its initial implementation revealed some critical points. Firstly, the algorithm contains sequential parts with strong data dependencies that cannot be effectively parallelized. Secondly, the convolution routines that enumerate 2D convolutions of images and blur systems were identified as the slowest parts of the algorithm decreasing its overall performance. This observation led us to focus on the effective parallel implementation of convolution cores to increase their performance.

A. Prior implementation

The algorithm authors [2] developed an optimized C-language implementation using the *FFTW* program library [3] as the engine for 2D convolution enumeration.

The *FFTW* engine performance optimizations were applied including the pre-computation of FFT plan and the multi-threading feature. The C compiler optimizations, including the Streaming SIMD Extension (SSE), were also applied to achieve the best possible performance.

B. CUDA accelerator

The first version of the accelerator implements the 2D-convolution with a naive approach. This implementation enumerates the result in two loops that control matrices shift in both dimensions. The inner computation is realized with the GPU threads aligned in two-dimensional blocks.

The second approach implements the 2D convolution using Fourier transform method. The convolution function is realized so that both degraded image and blur system matrices are transformed and multiplied on the graphics processor using the CUFFT component and custom arithmetic routines.

Both CUDA accelerators were extended to comprise the inner part of gradient routine, where the convolution routines occurred twice. With this extension the overhead was removed and overall performance increased.

V. Performance measurements

The performance results of all three implemented variants (CPU, accelerated by CUDA using naive approach and accelerated using CUDA CUFFT) are presented in Table I.

TABLE I
Timing and FPS measurements - GeForce GTX260

Image size	CPU with FFTW	CUDA naive approach	CUDA with CUFFT
1 Mpix	2.6 s	4.1 s	1.3 s
2 Mpix	4.6 s	6.3 s	2.0 s
4 Mpix	9.7 s	13.1 s	4.0 s
8 Mpix	55.6 s	26.8 s	13.8 s
1 Mpix	5.1 fps	3.2 fps	10.3 fps
2 Mpix	2.8 fps	2.1 fps	6.5 fps
4 Mpix	1.3 fps	1.0 fps	3.3 fps
8 Mpix	0.2 fps	0.5 fps	0.9 fps

All presented values were acquired as arithmetic average of three consecutive measurements and additional frame per second rate values were computed.

The important parameters were kept the same during all measurements. The floating point arithmetic was set to the single precision number representation. Input image sizes varied from 1 to 8 Megapixel, while the blur system matrices kept of the same size $[11 \times 11]$ as recommended by the algorithm authors.

VI. Conclusion

The performance results achieved for all three realizations of blind image deconvolution method indicate that such a complex image processing method can be accelerated using the CUDA and graphics card even if there is an optimized implementation on modern general-purpose CPU platform.

With an astonishing theoretical performance capabilities of the latest graphics cards, higher increase increase in performance was expected from the CUDA accelerator we implemented. The speed-up was lower than expected, which can be justified by the fact that the deconvolution algorithm contains a significant number of sequential parts that cannot be parallelized.

Nevertheless, the naive implementation approach within the CUDA convolution routine represents a competitive alternative for the case with large input images and powerful graphics card. This observation approves suitability of the SIMD paradigm for image processing. The usage of Fourier transform in the other implementation shows roughly twofold performance increase as it has been expected.

Acknowledgment

This work was supported by the SCALOPES project; project number: Artemis JU 100029, MSMT 7H09005.

References

[1] NVIDIA, "Cuda zone website." [Online]. Available: http://www.nvidia.com/object/cuda_home.html

[2] F. Sroubek and J. Flusser, "Multichannel blind deconvolution of spatially misaligned images," *Image Processing, IEEE Transactions on*, vol. 14, no. 7, pp. 874–883, July 2005.

[3] M. Frigo and S. Johnson, "FFTW project website." [Online]. Available: http://www.fftw.org