PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

IMPLEMENTATION OF AN ERROR-ROBUST BUCKET-METHOD ALGORITHM FOR ELABORATION OF WHITE LIGHT INTERFEROMETRY DATA ON GPGPUS

Max Schneider, Dietmar Fey

Chair of Computer Science 3 (Computer Architecture)
Friedrich-Alexander-University Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen

ABSTRACT

3D surface analysis of objects presents one of the most challenging tasks in optical metrology concerning the required computing power. In particular white light interferometry causes a sophisticated evaluation due to the high data volume occurring during the measurement. In principle, the scanning process can be accelerated by actual cameras. We show that the elaboration of the incoming data can even not satisfactorily be fulfilled by utilization of current multi-core technology if the scanning process is integrated in manufacturing processes. However, by appointing and assigning computational heavy tasks to co-processor devices like GPGPUs (General-Purpose Graphics Processing Unit) a rapid evaluation can be realized. We demonstrate that for the so-called five-bucket preprocessing method about 2 GB measured data can be processed in ms range using NVIDIA's CUDA (Compute Unified Device Architecture) technology. Due to the efficient utilization of CUDA-capable GPU devices a large-scale speed-up in comparison to a parallel implementation with OpenMP and SSE is achieved.

Index Terms— White light interferometry, Five-Bucket-Algorithm, GPGPUs, CUDA, OpenMP, SSE

1. INTRODUCTION

The scanning device used in white light interferometry is usually an adapted Michelson interferometer equipped with a broadband light source and a video camera or digital imaging system (CCD array) for capturing interferometry images (see figure 1) [1]. The white light beam is split in two paths. One path is projected onto the reference mirror, the other one to a in z-direction automatic moveable translation arm carrying the object to measure. If the optical path length difference of both beams is less than the coherence length, interferences arise [2]. For a complete surface inspection it is necessary that the whole interference range is covered in the measurement process. Sampling frames are taken at discrete z-positions, so that for each pixel the autocorrelation function of the original beam is obtained

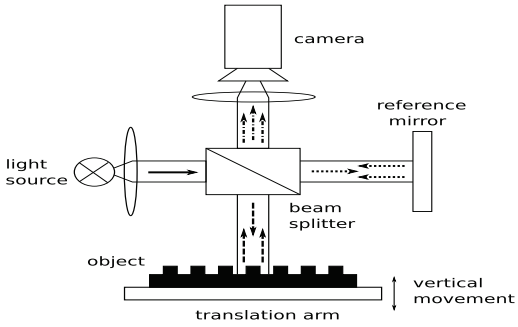


Fig. 1. Michelson Interferometer.

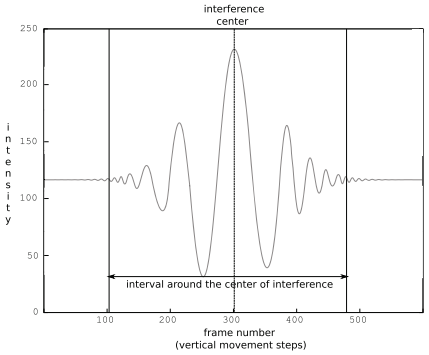


Fig. 2. A synthetic interferogram signal of a pixel.

[3]. The resulting signal, shown in figure 2, is the so called interferogram or correlogram.

A discrete height value is assigned to each frame captured in z-direction. With the knowledge about the scale from the transmission ratio of the translation arm the height of the pixels can be determined [1]. For that, the center of interference in each pixels' interferogram has to be computed. Since the interference is to observe only within a certain number of frames around the interference center a data reduction can be applied in a preprocessing step. This is necessary since a complete measurement process can span thousand of pictures with XGA (1024×768) resolution and above,

in 8-Bit or 16-Bit color depth. This would result in a data volume of 750 MB (XGA/8-Bit/1000 pictures) or 1.5 GB (XGA/16-Bit/1000 pictures), respectively of mostly irrelevant data. For an up-to-date computer system it's not a problem to hold this amount of data, but if the measurement process needs several thousands images such systems would also run out of memory. Furthermore, the calculation of the height map is a computational intensive process, so that the elaboration of irrelevant data should be avoided.

2. GENERAL-PURPOSE GRAPHICS PROCESSING UNITS AND CUDA

Since their introduction GPUs have been consistently optimized to do fast calculations, e.g. on 3D image data for computer games or video processing. Meanwhile GPUs evolved to very high performance massively parallel architectures, that are capable of executing many of hundreds threads in parallel, to do SIMD-(Single Instruction Multiple Data)-like processing of incoming data. Furthermore, processing pipelines of modern GPUs are programmable, allowing the usage for general-purpose computations in applications beyond their intended purpose (e.g. scientific algorithms, economy simulations).

In 2007 NVIDIA introduced a new parallel computing architecture called CUDA [4]. With that, a simple programming access to device resources through C/C++ libraries is given, hiding the complex graphics related interface. CUDA-capable devices possess a number of SIMT (Single Instruction Multiple Thread) multiprocessors. While SIMT is an analogue concept to SIMD, for the realization of data-parallel computations, CUDA-device multiprocessors consist (in contrast to conventional vector processors) of a set of scalar processors. When a data-parallel program is executed, a number of threads, e.g. one thread per data element, is instantiated and each thread is assigned to a scalar processor, executing the same program on different data in a SIMD-fashion. Since thread assignment is nondeterministic, non-adjacent data can be assigned to scalar processors of the same multiprocessor. Due to the implementation of GPUs as an aggregation of several multiprocessors, consisting of multiple scalar processing elements, a corresponding memory hierarchy is used. So called global or device memory is accessible by all processors of the device. Further, each multiprocessor has its own shared memory and read-only caches for constants and textures. Those are accessible from all scalar processors within the multiprocessor [5].

In contrast to a thread scheduling on operating system level, thread scheduling on a GPU requires much lower switching time. Allocated threads are grouped in thread blocks and mapped onto SIMT multiprocessors. Thread blocks are furthermore divided in pieces called thread warps which are set active and piecewise executed on scalar processors of corresponding multiprocessors. If a thread of an active warp waits for data during its execution, this thread is suspended while the memory management unit (MMU) gets data from global memory and another thread is scheduled onto that processor. By that, data transfers latencies can be completely hidden [5]. Threads of the same block communicate with each other through the fast shared memory. Communication between threads of different blocks is accomplished through the global memory, causing much slower access compared to the shared memory inside a multiprocessor.

By coalescing additional performance is attainable during accesses to the global memory. If neighboring threads of a multiprocessor make requests for data located consecutively in global memory, those requests are packed and executed together, resulting in up to 16 times higher transfer performance. The base address must yet satisfy the alignment requirements for coalesced transfers (base address must be multiple of 16 times the size of requested values type size) [6].

3. ERROR-ROBUST FIVE-BUCKET-METHOD

To cover the whole interference range, needed for a complete surface inspection, hundreds or thousands of images have to be captured and vertical movement steps of the translation arm are necessary. In order to meet strict timing constraints data amount should be reduced, if possible. This can be achieved by identifying, for the height map calculation, relevant fragment (primary height map) of each pixels' correlogram [7]. This primary height map corresponds to the correlogram interval where the optical path length difference of both reflected beams is shorter than the coherence length of the used light beam. It is obtained through calculation of the interference pattern center in each pixels' correlogram and extracting of intensities in a specified range around that center. Furthermore, incoming data has to be processed simultaneously to the scanning of the object, to get results as soon as possible.

The calculation of height maps for an object consists of a preprocessing step, the extracting of the relevant correlogram part, and a postprocessing step, where by denoising and correction of determined correlogram segments the actually height is obtained. Depending on the surface characteristics and the signal-to-noise ratio of the generated signals, different approaches for primary height maps calculation have to be considered [1, 7]. Since scanning of rough surfaces is naturally error-prone an error-robust algorithm is required to determine the correct fragments out of resulting correlogram signals [8, 9].

An elementary way to determine the center of a correlogram is based on simple maximum search [10], where only a pixel-wise comparison is required. Unfortunately this simple approach can lead to missclassified

correlogram intervals, which even the following post-processing step cannot compensate. Therefore, error-robust procedures like the five-bucket-method are required [7, 9]. This algorithm computes so called bucket values, by applying equations (1) to (4) to each pixels' correlogram intensities. When a new bucket is calculated, it is compared to the current maximum bucket. In case that, the new value is greater, the maximum bucket and value of the center of interference (set to frame index z) are updated.

$$DF02(z) = I(z+0) - I(z+2) \quad (1)$$
$$DF13(z) = I(z+1) - I(z+3) \quad (2)$$
$$DF24(z) = I(z+2) - I(z+4) \quad (3)$$
$$B(z) = \frac{1}{2}\sqrt{DF13(z)^2 - DF02(z) \cdot DF24(z)} \quad (4)$$

where z is the number of the current translation step, B(z) bucket value of the current frame, I($z+k$) with k $\in$ $\{0...4\}$ pixels intensity in corresponding frame. Based on the calculation method in equation (4), B(z_i) can be computed by using the contrast values DF13(z_{i-1}) and DF24(z_{i-1}) of B(z_{i-1}) as values for DF02(z) and DF13(z) accordingly, leading to equations (5) to (8).

$$DF02(z_i) = DF13(z_{i-1}) \quad (5)$$
$$DF13(z_i) = DF24(z_{i-1}) \quad (6)$$
$$DF24(z_i) = I(z_i+2) - I(z_i+4) \quad (7)$$
$$B(z_i) = \frac{1}{2}\sqrt{DF24(z_{i-1})^2 - DF13(z_{i-1}) \cdot DF24(z_i)} \quad (8)$$

with $i \in \{1...levels-1\}$. Therefore in the next step only the contrast value DF24(z_i) has to be calculated, reducing the complexity of the maxima calculation. Since the difference operations corresponds to a high-pass characteristic, the five-bucket-approach shows an increased error-robustness [1]. Derived from given equations, at first images of all translation steps are captured and each pixels' correlogram is elaborated after another.

The only dependency in above equations exists between corresponding pixels of successive images. Thus, the parallel calculation of each pixels' relevant correlogram interval is a crucial option to gain better performance and drastically reduces run times for the preprocessing phase. In the following two parallel implementations of the five-bucket-algorithm are presented. The first one integrates OpenMP and SSE (Intels SIMD-solution) and was tested on a Nehalem-Quad-Core i7 (920) 2.66 GHz system with 12 GB DDR3 RAM. The second version was build upon NVIDIAs CUDA-framework and tested on a Tesla C1060 with 4 GB GDDR3 RAM and a GeForce GTX 480 Fermi with 1.5 GB GDDR5 RAM. Although parallelization by OpenMP and SSE gives several times speed-up against sequential approach, much higher performance boost can be achieved by harvesting GPGPUs features (separate memory, higher memory transfer rates, several hundred scalar processors).

3.1. Using OpenMP and SSE Technics for Optimization on Multi-Core Systems

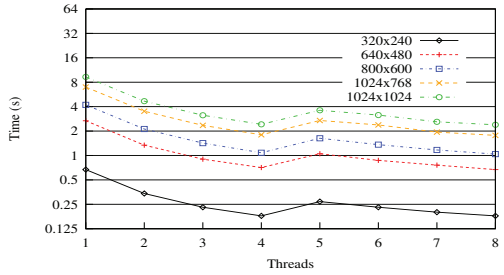
OpenMP is an Application Program Interface (API) for multi-platform shared-memory parallel programming in C/C++ and Fortran on all conventional architectures. By applying OpenMP pragmas on specific code sections, e.g. loops, sequential programs can be automatically parallelized onto underlying multi-core architectures, without a huge programming effort [11].

To get any performance boost by OpenMP not a pixel-wise but a frame-wise processing is to prefer, since the measurement data of all translation stages is stored frame by frame in main memory, i.e that intensities belonging to one pixel are not stored contiguous. Therefore, pixel-wise processing results in accesses to not continuous main memory addresses and therefore in a high cache miss count. In contrast, in a frame-wise processing each image is elaborated completely, before intensities of the next image are touched. At first five translation steps are executed and the corresponding intensity data stored in system memory. These values are used in an initialization step, where for each pixel the first bucket is calculated and the center of interference value initialized. While the values are computed, the next frame is captured. After the initialization procedure for each pixel is finished, stored data of levels two to six are used to calculate the second bucket and updating maximum bucket and height values. During this step captured intensity image is stored in the memory space of level one, which data is not needed anymore. This procedure is repeated until all images are processed and the primary height maps generated. By reusing the allocated memory only space for six successive levels is needed, in contrast to the pixel-wise processing, which has required the complete loading of all frames.

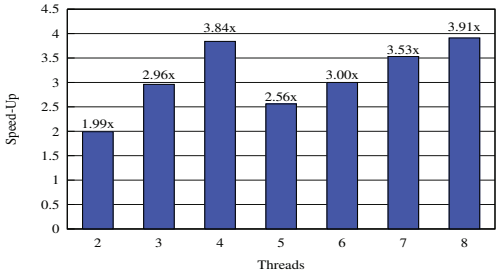
Figure 3a illustrates on the Nehalem-Quad-Core system achieved run times for the OpenMP implementation of the center of interference computation. It is obvious, that due to the independency in the elaboration of each pixels' interferogram, the more threads are used the better performance is attainable. In figure 3b shown speed-ups approve this statement, as up to four threads a nearly linear speed-up is gained. However, with enabled HyperThreading similar but not better results were achieved, which is explainable by the overhead arose by switching between threads on the same processor core.

For white light interferometry used cameras usually operate in 8-bit or 16-bit color depth. During the maximum analysis procedure, floating-point intermediate results with single or even double precision are required. Hence, intensities values must be converted. For the 64-bit Core i7 architecture this circumstance doesn't have a significant effect on the performance of the OpenMP implementation, because all scalar op-

erations are executed internally on 64-bit wide registers. Thus, independent of the used color depth and the precision of the computations analogue run times and speed-ups were achieved. However, in the case of SSE-support this conversion has an enormous impact on processing time and programmer's effort, too.



(a) Runtimes of the OpenMP implementation.

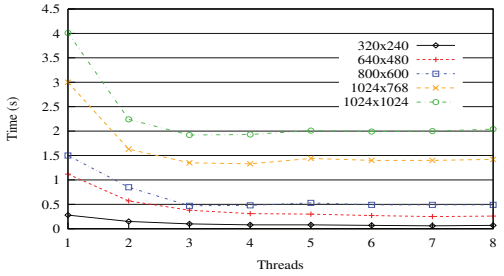


(b) Average speed-ups of the OpenMP implementation

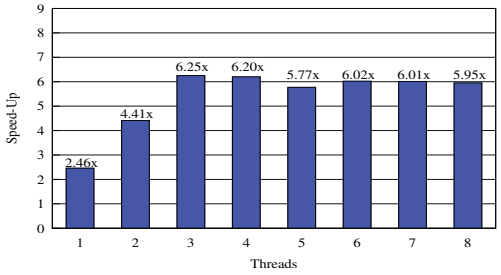
Fig. 3. Runtimes and corresponding speed-ups of the OpenMP implementation in comparison to sequential implementation (1000 levels / single precision).

Since introduction of the Multi Media Extension (MMX) by Intel in 1997 [12] and its successor Streaming SIMD Extensions (SSE) in 1999 [13] almost all CPU vendors decided to include SIMD instruction extensions in their processor-architectures to accelerate data-parallel processing [14]. Intel's enhancement to an SSE instruction set x86 architecture leaded to additional eight directly addressable 128-bit SIMD general-purpose registers and new instructions that work on packed floating-point and integer data. The current implementation (SSE4) provides 16-way, 8-way, 4-way and 2-way integer and 4-way or 2-way floating-point vector instructions, so multiple data elements are processed in a single cycle, increasing the performance of data-parallel applications several times over that of the sequential counterpart-algorithms [15].

By harvesting SSE-technics the five-bucket-method can gain an additionally performance boost. However, during primary height map elaboration, needed data types conversion has negative effects. For a proper processing of vector elements, SIMD-instructions require, that all in the same operation used vector values contain the same number of scalar units [15]. If the complete analysis procedure could be accomplished without any usage of floating-point values in each run of the inner loop 16 or 8 pixels would be processed in parallel, yielding a huge increase in the achievable performance. This is not possible yet, because even without the square root, storing of intermediate results requires at least 32-bit integer values. Thus, the 16 or 8 elements of a vector, accordingly to the used color depth, have to be unpacked to new vector values, each holding only four or two elements (dependent on the required precision) and processed after another.



(a) Runtimes of the OpenMP implementation with enabled SSE-processing.



(b) Average speed-ups of the OpenMP implementation with enabled SSE-processing.

Fig. 4. Runtimes and corresponding speed-ups of the OpenMP implementation with SSE-support in comparison to sequential implementation (1000 levels / single precision).

The measured run times shown in figure 4a show obviously that even though the additional effort of unpacking vector elements to new vectors a further performance increase compared to pure OpenMP solutions without SSE is feasible. In single precision mode already three OpenMP threads with enabled SSE elaborate data faster than the memory bus is capable of providing the data (see figure 4b). Hence, adding further threads don't give any significant speed-up. On the other side, vectorized computations in double precision are not worthwhile, because the unpacking process (e.g. with 8-bit intensity data, eight extracting steps have to be executed) and the following two-way processing of pixel data needs more time, than a sequential elaboration of the same data amount requires. Thus, if double precision is not necessarily needed, combination of OpenMP with single precision SIMD-operations should be preferred, otherwise only the OpenMP prag-

mas should be applied.

3.2. Accelerating by GPGPU utilization

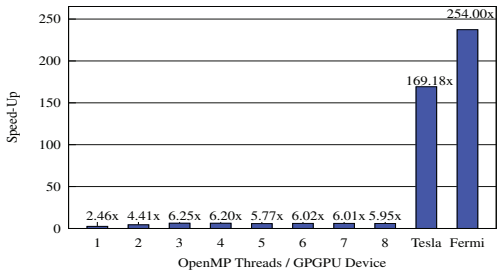
The scanning process can span thousands of translation steps with images in XGA resolution and above. Based on the minimal run times (1.93 seconds for 1000 images with megapixel resolution and 1.33 seconds for images with XGA resolution) achieved by the multithreaded and vectorized implementation, alone the computation of the interference center would require several seconds. However, the postprocessing of the primary height maps is even more complex, thus the whole surface analysis process is not feasible in real-time. Furthermore, the center of interference for each pixels correlogram is determined by a simple calculation, but a huge data volume has to be processed and a lot of intermediate results have to be stored. Because GPUs possess their own memory with even higher bandwidth than conventional memory offers and provide higher compute capability compared to conventional CPUs, it is reasonable to assign the task of computing each pixels interference center to GPUs. By that relocation, host system resources are freed, increasing other tasks execution performance.

To get results as soon as possible, processing of incoming picture data has to be done parallel to the actual scanning process. Due to the fact, that functions (kernels) executed on GPUs are called from the CPU program space but executed asynchronous to other function calls, overlapping of routine execution on GPUs and CPUs is allowed. Thus, required parallel processing of incoming data to the scanning process is achievable. Before data can be elaborated yet, it must be first transfered from the system memory to that of the GPU, because they have no direct access to main memory. Since transfers of small data packages don't utilize the PCIe bus efficiently, data of a predefined image number is stored linearly in host's memory first. During the next translation step is performed, it is transfered at once to GPUs memory. As soon as the data transfer is complete, the bucket-algorithm is applied on that data. A further obvious advantage of separate memory spaces is that all intermediate results (these are not anymore needed after the preprocessing step) are also stored in the same memory. Therefore the host system performance is not affected.

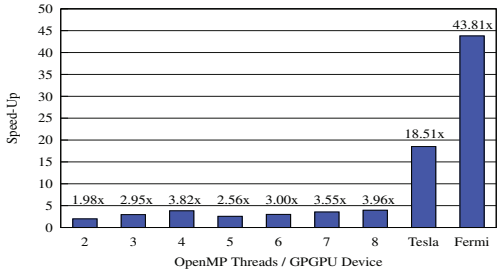
Considering the OpenMP implementation approach discussed above, based on the focus on architecture design of NVIDIAs Tesla C1060, the processing manner on GPGPUs has to be reconsidered. The Tesla devices provide read-only constant and texture caches, allowing fast access to measurement data. However due to the indispensable usage of intermediate values, also write access is required. Without adjustments the program execution would results in high memory traffic, induced by data transfers between registers of the scalar processors and global memory. Because each pixels interferogram is processed independently from those of other pixels, the simplest, but also to the best of our knowledge the most efficient way is to instantiate for each pixel it's own thread.

Threads grouped in the same thread block have access to 16 KB register set (32-bit each) [5]. By specifying the thread block size accordingly to this constraint and holding the needed values inside these registers during computation, all intermediate data can be kept inside registers without being flushed out. Thus, number of accesses to global memory is reduced. Furthermore, by assigning horizontally adjacent pixels to threads with successive ids, due to linear storing of picture data, a coalesced data access is achieved.

Through utilization of GPGPUs as co-processor units and outsourcing the described preprocessing method from conventional CPUs, an enormous speed-up in comparison to the sequential or even multi-threaded and vectorized algorithms was achieved (see figure 5a.)



(a) Speed-up comparison between single precision OpenMP implementation with enabled SSE-processing to CUDA version.



(b) Speed-up comparison between double precision OpenMP implementation to CUDA version.

Fig. 5. Achieved average speed-ups through utilization of GPGPUs (Tesla C1060/GeForce 480GTX) compared to those of OpenMP implementations (1000 levels).

Although the Tesla C1060 devices are specialized on operation execution in single precision mode and offer just a peak double precision performance lesser than one tenth of the peak single precision performance [4], their assignment for double precision procedures is also worthwhile due to the marginal programming effort offered by CUDA-framework. As shown in figure 5b, through utilization of the Tesla C1060 resources al-

most five times higher speed-up in comparison to the best performing double precision OpenMP implementation was gained.

With the new graphic chip generation "Fermi" (e.g. GeForce 480 GTX) NVIDIA introduced cached global memory accesses. This allows better usage of communication paradigmas. Additional scalar processors, higher performance capability for single as well as for double precision computations are also common [4]. As shown in figure 5, on the GeForce 480 GTX overall performance of the five-bucket-algorithm increased on a large scale. E.g. for elaboration of 1000 images, each with a resolution of 1024×1024 and 16-bit color depth (ca. 2 GB measured data), run times in single precision mode about 0.038 seconds are achieved (on Tesla C1060 0.065 seconds). The CUDA implementation benefits from new features offered by Fermi GPUs even though no further program adjustements were induced and no recompiling conducted.

4. CONCLUSION AND FUTURE WORK

In this work, we presented two approaches which enable efficient execution of the data-parallel five-bucket-preprocessing algorithm of the white light interferometry process. On the one side it was shown that through utilization of multi-core systems the analysis phase can achieve a several times speed-up compared to a conventional implementation. On the other side system resources (in particularly main memory) needed for other tasks are restrained, reducing the overall system performance. In the second implementation with NVIDIAs CUDA-framework on graphic devices provided features were harvested to relaxe the system demands of the five-bucket-method. Attained performance of the CUDA implementation reveals the high potential of outsourcing computational tasks in industrial image processing to co-processor devices like GPGPUs.

In the future, a Multi-GPU and a heterogeneous (cooperational work of GPUs with conventional CPUs or other multi-core architectures) solution of the presented algorithm will be examined, providing further details needed for efficient utilization of parallel architectures. Also the postprocessing step algorithms will be implemented using NVIDIAs CUDA-framework, allowing encapsulation of the complete height map elaboration process within GPUs.

5. REFERENCES

[1] M. Hißmann, *Bayesian Estimation for White Light Interferometry*, Ph.D. thesis, Combined Faculties for the Natural Sciences and for Mathematics of the Ruperto-Carola University of Heidelberg, Germany, 2005.

[2] E. Hering and R. Martin, *Photonik - Grundlagen, Technologie und Anwendung*, Springer Berlin Heidelberg, 2006.

[3] D. Kapusi and T. Machleidt, "White Light Interferometry in Combination with a Nanopositioning- and Nanomeasuring Machine (NPMM)," Proceedings of the International Society for Optical Engineering, June 2007.

[4] NVIDIA, "CUDA ZONE," online.

[5] NVIDIA, "NVIDIA CUDA C Programming Guide 3.1.1," Tech. Rep., NVIDIA, 2010.

[6] A. Cevahir, A. Nukada, and S. Matsuoka, *Computational Science – ICCS 2009*, vol. Fast Conjugate Gradients with Multiple GPUs of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2009.

[7] W. Sang, "Entwicklung und Implementierung eines Verfahrens zur Auswertung von Weißlichtinterferogrammen zur Bestimmung der dreidimensionalen Oberflächentopographie von Mikro- und Nanostrukturen als Anwendung fur eine Nanopositionier- und Nanomessmaschine," M.S. thesis, Technische Universität Ilmenau Fakultät für Elektrotechnik und Informationstechnik, 2006.

[8] D. W. Robinson, *Interferogram Analysis: Digital Fringe Pattern Measurement Techniques*, Institute of Physics Publishing, 1993.

[9] K. G. Larkin, *Topics in Multi-dimensional Signal Demodulation*, Ph.D. thesis, The Faculty of Science in the University of Sydney, 2000.

[10] Z. Sarac, R. Groß, C. Richter, B. Wiesner, and G. Häusler, "Optimization of White Light Interferometry on rough Surfaces based on Error Analysis," *Optik - International Journal for Light and Electron Optics*, vol. 121, pp. 351–357, 2010.

[11] Barbara Chapman, Gabriele Jost, and Ruud van der Pas, *Using OpenMP - Portable Shared Memory Parallel Programming*, The MIT Press, 2007.

[12] A. O. Ramirez, "An Overview of Intel's MMX Technology," *LINUX JOURNAL*, vol. 61, 1999.

[13] M. Srivastav, "Vectorization: Writing C/C++ Code in VECTOR Format," Tech. Rep., Intel Software Network, 2009.

[14] J. Stewart, "An Investigation of SIMD Instruction Sets," Research Project on the University of Ballarat School of Information Technology and Mathematical Sciences, November 2005.

[15] Intel Corporation, *Intel 64 and IA-32 Architectures Optimization Reference Manual*, 2009.