Portability of the fragmented l1-norm transform for massively parallel processing

Scholz, T., Rosenberger, M., Notni, G.

T. Scholz, M. Rosenberger, G. Notni, "Portability of the fragmented l1-norm transform for massively parallel processing," Proc. SPIE 11144, Photonics and Education in Measurement Science 2019, 111441A (17 September 2019); doi: 10.1117/12.2531945

Event: Joint TC1 - TC2 International Symposium on Photonics and Education in Measurement Science 2019, 2019, Jena, Germany

Portability of the fragmented l1-norm transform for massively parallel processing

T. Scholz[a], M. Rosenberger[a], and G. Notni[a]

[a]TU-Ilmenau, Gustav-Kirchhoff-Platz 2, 98693 Ilmenau, Germany

ABSTRACT

White light interferometry is a major optical non-contact and therefore nondestructive testing method for nanostructures and surface reconstruction. By the reason of scanning, the data throughput is high and the resulting data stack can exceed gigabytes of raw data. Effective data compression was realized in an FPGA early in the signaling cascade. On the one hand this can significantly boost the achievable data throughput from the sensor, on the other hand the compression results in fragmented raw data with non-equidistant sampling steps and is therefore incompatible with FFT based reconstruction algorithms. In order to face this issue the fragmented l1-norm transform (flot) was developed. The flot reconstruction algorithm is a symbiosis of the l1-norm known from compressive sensing and additionally the wavelet-transform. In contrast to the traditional wavelet-transform the flot algorithm has no dependence on FFT and can quickly handle non-equidistant sampled data. Raw data is heavily independent between pixels in white light interferometry by design. Therefore, implementing the reconstruction algorithm on massively parallel hardware is promising. In the last decade this usually meant data processing on GPUs. Nowadays alternatives in the form of affordable CPU clusters or easy to program FPGAs gain importance. OpenCL is a framework to accelerate highly parallel problems on all of the three platforms. In this paper the implementation of the flot algorithm in OpenCL will be explained, compared by speed and power consumption and categorized for suitable use-cases.

Keywords: white light interferometry, FPGA, Vivado HLS, OpenCL, GPGPU

1. INTRODUCTION

White light interferometry (WLI) is a key technology in precise surface reconstruction in the nanometer scale. Major benefits of this approach are the non-destructive and non-contact behavior. In ideal situations the sub-nanometer height resolution is achievable.[1] Unlike laser interferometry WLI allows for the measurement of the absolute distance because of the small coherence length of the light source.[2]

A measurement setup for the acquisition of raw data is comparable to a microscope. For best results Mirau objectives are most commonly applied. The beam splitter is moved to different positions during a measurement cycle by means of a piezoelectric actuator. This movement allows for the acquisition of a sequence of images with different path length differences between reference mirror and surface. The depth information is stored in the height of the stack for each pixel. It is not uncommon for the raw data to amount to multiple gigabytes. Since each pixel stack can be analyzed separately parallel computation is possible and can lead to a massive speed-up in combination with suitable hardware.

Additionally, some knowledge on the surface is necessary. Dispersive specimen cause an phase shift which must be faced. The reconstruction algorithm must be suitable to gain this phase information.

Further author information: (Send correspondence to T. Scholz)
T. Scholz: E-mail: tobias.scholz@tu-ilmenau.de, Telephone: +493677 - 693804

Photonics and Education in Measurement Science 2019, edited by Maik Rosenberger,
Paul-Gerald Dittrich, Bernhard Zagar, Proc. of SPIE Vol. 11144, 111441A
© 2019 SPIE · CCC code: 0277-786X/19/$21 · doi: 10.1117/12.2531945

2. FRAGMENTED L1-NORM TRANSFORM

The fragmented l1-norm transform (flot) is a symbiosis of the wavelet transform and compressed sensing. Therefore it is only suitable on sparse data even possible is violating the Shannon-Nyquist sampling theorem.[345] In contrast to compressed sensing the reconstruction matrix is not filled with random orthogonal functions.[6] Both share the reconstruction using the $l1$ minimization for solving the under determined linear system.[7] Compression and reconstruction can be separated operations without sharing the same matrix or even the same compression-reconstruction algorithm. To achieve this, suitable orthogonal base functions must be known as a priori knowledge. During the reconstruction a solution matrix of multidimensional l1 values occur which is a perfect task for massively parallel hardware.

2.1 flot for white light interferometry

A wide variety of white light interferometry algorithms are in practical applications. Most of them rely on fast Fourier transform directly or indirectly. This leads to a strict requirement on the raw data to be achieved in equidistant sampling steps. Perfectly true for classic wavelet transform which is a harsh restriction on the sampled data. The degrees of possible solutions can be shrinked by a huge factor. Tested and implemented on several platforms the reconstruction abilities are on par with classical wavelet transform.[89] Overall tested implementation for white light interferometry contains several steps. Achieved data will be compressed preferably near the sensor. For faster reconstruction the compressed raw data should be sorted. As last step the flot-algorithm is used to reconstruct the compressed raw data to a surface in sub-nanometre resolution.

2.1.1 compression

Most simple to implement but hard to reconstruct way to compress non-relevant raw data in white light interferometry is subtracting pixel-wise constant offset and define a suitable threshold in the amplitude of the noise. Two major impacts are caused by this simple compression. While the pixel data can neither be determined nor contiguous the pixel data expand by a factor of 4 when data is outside of the threshold. This is necessary to save piezo position in stack and the pixel position in image. Even if the maximum amount of compressed data can be estimated the resulting compressed data is not equidistantly sampled anymore and must be faced by using a matching algorithm.

2.1.2 sorting

Processing systems can speed-up their processing time when all necessary data are aligned. In White light interferometry the data is pixelwise in the depth of the stack. For that reason it is advisable to copy the scattered raw data to pixelwise contiguous memory. Afterwards whole memory blocks containing the compressed raw data of every pixel can be loaded to the reconstruction hardware.

2.1.3 reconstruction

Using the flot-algorithm for white light interferometry creates a three dimensional data cube. An $l1$-minimum corresponding to the piezo-actuator position, the signal phase and the amplitude of the interferogram has to be found. In figure 1 an exemplary data slice at an constant phase is shown. Here you can see several valleys in constant distance. The distance depends on the white light illumination source bandwidth. Additionally, the orientation of the valleys is parallel to the amplitude aberration. Height reconstruction is only relevant to the piezo actuator position and the probably the phase. Therefore, the best matching amplitude is not relevant if it is around the optimum. After finding a coarse minimum the absolute minimum can be found by brute force or a gradient method in possible areas. Real measurements showed that it is necessary to search for a minimum in the nearby data valleys. Unlike the amplitude of the interferogram the exact phase can not be estimated using flot. For each pixel an estimated amplitude has to be chosen and the data slice has to be found. If using a gradient method a starting point of the main valley and its nearby valleys is perfectly doable. Otherwise the complete data slice must be processed.

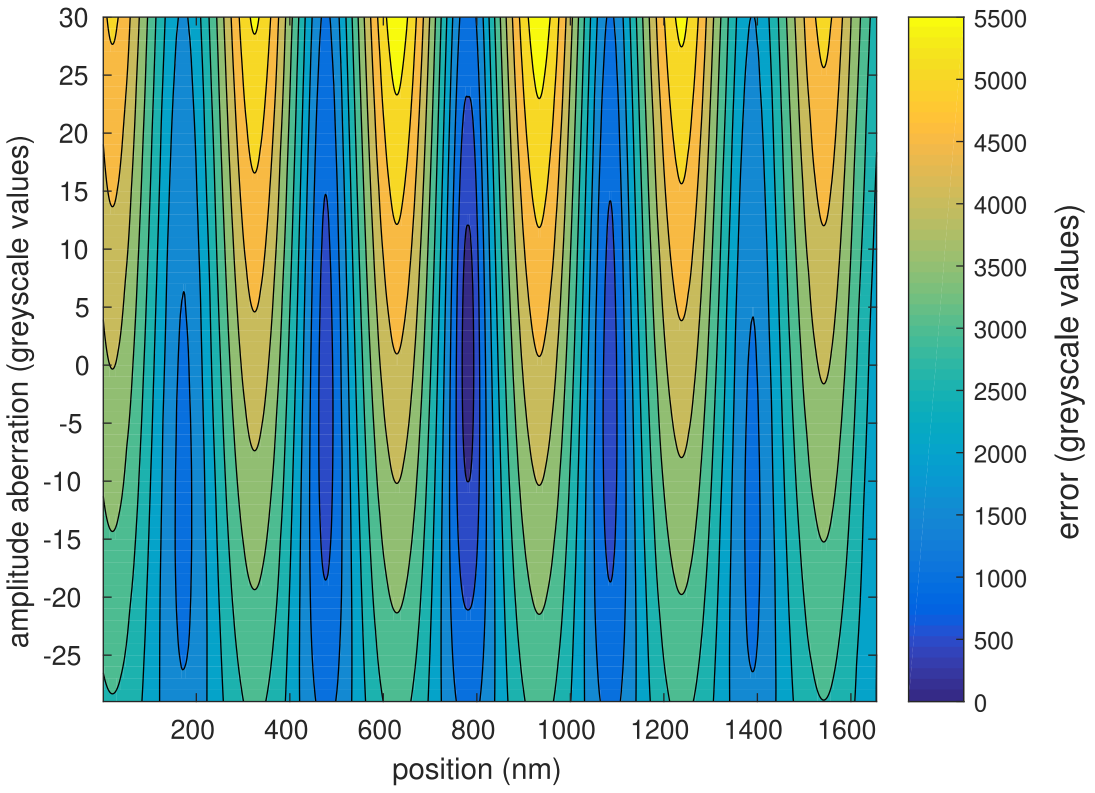

Figure 1. data slice of the l1 values resulting from the mother wavelet in the axis of piezo actuator position and the amplitude of the interferogram at an constant phase

3. CPU IMPLEMENTATION

As a starting point the white light interferometry using fast lossy compression and flot-reconstruction was implemented on a CPU. Applying data optimizations the processing time shrinked a lot. Unfortunately the fastest data alignment highly depends on the hardware architecture. While the x86 architecture is good-natured, more effort is needed to find the highest data throughput on the ARM-architecture. On the ARM-architecture reserving and initializing more memory than needed is better than increasing the data vector.[8]

4. FPGA IMPLEMENTATION

Lowering the power consumption as main goal the use of an Zynq SoC was highly efficient. While sorting the data on the FPGA was extremely slow and area consuming the data compression and selection outperformed the much bigger x86 processor. On the Zynq-architecture the RAM throughput is the main bottleneck. Equipped with LPDDR3-RAM clocked at 666MHz a netto RAM bandwidth of 2.4 GB/s was achieved. Data reconstruction on the FPGA was implemented fully pipelined and massively parallel.[9]

5. GPU IMPLEMENTATION

Creating a highly optimized GPU implementation is out of scope for arbitrary hardware. Common graphics card support different GPGPU languages in different versions. The minimum supported by all investigated is OpenCL in version 1.2. For that reason the possibility to directly port the C++ algorithm to OpenCL was examined. As OpenCL wrapper the library Boost compute is used as basis. Comparing the computation time for the single steps was mainly done on a Intel i7-6700k and a Nvidia Geforce GTX 970. The primary test to calculate the operational time to create a pixelwise difference between two images results in a 10 times speed-up on the

GPU. On the other hand copying the data from and to took more time than the whole CPU computation time. Added to this function the same compression already implemented on the FPGA resulted in an overall two times faster CPU implementation for a full data stack. This behavior is not a big surprise as the algorithm is highly memory intense. Using global memory instead of the much faster local memory and missing data alignment optimizations slows this step down. Further work should show a change to a speed-up of around 4 times faster GPU implementation. Sorting the data to the preferred data alignment is 25% faster compared to CPU. Included is to data transfer which took half the time. The reconstruction itself perfectly fits the ability of GPGPU. Only a few data operations and heavy load of independent calculations. Most easy implementation is to use a for loop and unroll it followed by finding the minimum $l1$ value. Comparing all single steps the straight forward OpenCL implementation was at least on par but usually several times faster than the CPU implementation. Biggest impact is still the data transfer between host and the device. As the steps can be implemented serialized there is an effort in porting the raw data to GPU and work there. Especially the compression and reconstruction should be implemented in local memory rather than the current version using global memory. Porting the CPU solution was not very time intense but is far away from a good solution which needs much more data thought before implementing. On the other hand the best solutions much differ between real hardware even if using the same OpenCL version.

6. CONCLUSIONS

In modern computational architectures massively parallel hardware gains more and more importance. Most common hardware architectures include GPU, FPGA and processor arrays. They differ in ease of programming techniques, available IP-cores and libraries. CPUs are the fastest in the term of time to market but are less efficient in power per operations. GPUs and FPGAs are much more intense in the manner of algorithm design. Especially the data alignment must be faced to achieve a speed-up compared to general purpose processing systems. While CPUs are much more versatile and often equipped with larger easy to access memory it is necessary to hold memory in a local area for an efficient design. Throughput depending data operations like sorting are easy tasks on general purpose processors but slow or extremely complex algorithms in specialized hardware. Raw data streaming the near sensor raw data preprocessing was implemented in a highly efficient FPGA design. Also an optimized data transmission was implemented using a simple compression algorithm. For touching all the raw data and sorting them to logical contiguous data in memory neither the FPGA nor the GPU showed a good behavior. The high language implementation on the general purpose processors showed not only the easiest but also the fastest solution. If there is no overhead by copying the data between the hardware architectures this should be preferred. Processing the data for reconstruction is the most important part. All the three architectures are possible to implement an efficient solution. In embedded low power solutions the FPGA low end CPU combination are preferable. The fastest time to market but much more power consuming solution is using a fast easy to program general purpose processing systems, preferably containing several processing cores. GPUs are a good combination of the fast up-gradable hardware and software and high throughput between CPU and GPU. Compared to FPGAs they offer much more local memory but lack in power consumption. All examined processing hardware have their unique strengths and weaknesses. The most efficient design is the combination of all the three hardware types to overcome their special weaknesses.

White light interferometry is a perfect problem that can be solved on massively parallel hardware. Every pixel contains its own data stack without sharing data between neighbored pixels. Efficient solutions still need some data management. Usually it is necessary to sample much more data than the containing interferograms. The specimen is neither flat nor orthogonal to the optical axis. To achieve the whole interferogram data for all pixels there is an overhead in the piezo actuator range. Especially for specimen containing high differences in their height the amount of raw data can easily reach several gigabytes. This massively slows down the overall reconstruction speed. Swapping the raw data must be avoided. Also the sorting and reconstruction slows down because the uncompressed data must be processed. So lossy compression of the raw data massively speeds-up the reconstruction time. The flot algorithm was designed for non-equidistant massively parallel processing. Compared to the classic wavelet transform it is on par in reconstruction abilities. Less harsh restrictions on the raw data offers an even more robust algorithm.

Even if it was possible to port the flot algorithm and its corresponding first steps for white light interferometry the results need optimizations. FPGAs can compress very good and can reconstruct the surface in a good time sorting is out of scope for this hardware architecture. GPUs are able to handle all the necessary steps but good results need much manual scaling and data handling. Porting the algorithm directly to OpenCL result in bad runtime. Copying data between the calculation steps are even worse. On the other hand local memory promises to speed-up the process a lot. The vector data types and supported OpenCL versions as well as optional standard functions may not exist for the current hardware. Data handling highly depends on the underlying graphics hardware.

ACKNOWLEDGMENTS

The approaches presented in this paper are sponsored by the federal ministry of education and research (BMBF, FKZ: 03ZZ0427G).

REFERENCES

[1] Deck, L. and de Groot, P., "High-speed non-contact profiler based on scanning white light interferometry," *International Journal of Machine Tools and Manufacture* **35**, 147–150 (nov 1995).

[2] Wyant, J. C., "White light interferometry," in [*Proc.SPIE*], Caulfield, H. J., ed., **4737**, 98–107 (jul 2002).

[3] Do, T. T., Gan, L., Nguyen, N. H., and Tran, T. D., "Fast and efficient compressive sensing using structurally random matrices," *IEEE Transactions on Signal Processing* **60**, 139–154 (jan 2012).

[4] Candes, E. J. and Wakin, M. B., "An introduction to compressive sampling: A sensing/sampling paradigm that goes against the common knowledge in data acquisition," *IEEE Signal Processing Magazine* **25**, 21–30 (mar 2008).

[5] Donoho, D. L., "Compressed sensing," *IEEE Transactions on Information Theory* **52**, 1289–1306 (apr 2006).

[6] Figueiredo, M. A., Nowak, R. D., and Wright, S. J., "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal on Selected Topics in Signal Processing* **1**, 586–597 (dec 2007).

[7] Abdelmalek, N. N., "On the discrete linear L1 approximation and L1 solutions of overdetermined linear equations," *Journal of Approximation Theory* **11**, 38–53 (may 1974).

[8] Scholz, T., Hänsel, M., Rosenberger, M., and Notni, G., "Resource efficient white light interferometry algorithm for non-equidistant sampling steps," *Journal of Physics: Conference Series* **1065**, 32007 (aug 2018).

[9] Scholz, T., Rosenberger, M., and Notni, G., "Massively Parallel Implementation of a Fast Resource Efficient White Light Interferometry Algorithm," in [*2018 Digital Image Computing: Techniques and Applications (DICTA)*], 1–6 (2018).