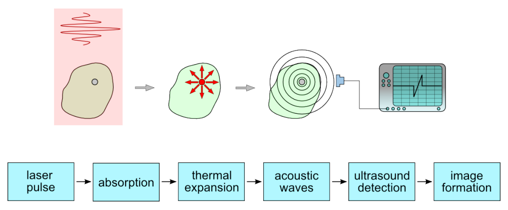
http://excel.fit.vutbr.cz

Fast Reconstruction of Photoacoustic Images

Filip Kuklis*



Abstract
The ability of reconstruction of photoacoustic images is important requirement to study soft tissues or vascular and lymphatic systems in high resulution but in small space. Today solution needs extensive computing power and it is noticeably time-consuming. In this study we would like to introduce a new solution which would be a way much faster and easy to use. My solution is up to twenty times faster and needs forty percent less memory than existing solution in Matlab. This solution may be a better alternative for scietnist who study soft tissues by photoacoustic imaging.

Keywords: Photoacoustic — Ultrasonic — Imaging — HPC — Parallel computing — Vectorisation

Supplementary Material: Downloadable Code

*xkukli03@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

[Motivation] This work is a part of international project k-Wave which is designed for time domain acoustic and ultrasound simulations in complex and tissue-realistic media. The main goal is to optimize and accelerate current solution of photoacoustic imaging which is written in Matlab. Ultrasonic detection of thermal expansion created by laser today takes approximately fifteen to twenty minutes. Photoacoustic reconstruction in good resolution takes another fifteen minutes. We want to decrease reconstruction time to minimum in order to see result as soon as posible.

[Problem definition] Time reversal image reconstruction
Photoacoustic images are reconstructed from the detected photoacoustic signal using a time-reversal image reconstruction algorithm, which compensates for the frequency dependent acoustic attenuation exhibited by soft tissues. The algorithm uses a pseudo-spectral (k-space) acoustic propagation model to simulate the retransmission of the measured photoacoustic signals into the domain in time-reversed order. The photoacoustic waves then refocus to yield an image of the initial pressure distribution.[1]

[Existing solutions] Complex but slow
The current solution is written in Matlab. This solution is described by mathematic formulas and mathematic operations and works with ultrasonic signal. The advantage of this solution is its universiality. It is general and any type of input can be used. The main disadvantage is that this solution is too slow and uses too much primary storage of computer memory. Other disadvatages are that at higher resolution Matlab takes so much memory that some of the calculations can not be done, or parallelization of computing is not balanced on architectures with more CPUs. The main operations taking the most time are the Fourier transform, inverse Fourier transform and a trilinear interpolation.

[Our solution] C++ plus paralelism
The proposed solution is based on a hardware friendly code which is parallelized. It is written in C++, and OpenMP is used to parallelize the code. The code is not so complex and universal as the Matlab solu-

tion, but it is specialized for one thing, photoacoustic imaging. It is as simple as possible with many optimalisations. Main disadvantage is that input files have to be saved from Matlab, then open in program and then the output have to be saved and open in Matlab to be plotted.

[Contributions] Quick and fluffy
My solution takes about 40% less memory than the existing Matlab solution. However what is more important, it is ten to twenty times faster (depends on used resolution). Now the scientists can see results in a very good resolution in less than one minute.

2. Theoretical background

2.1 Photoacoustic imaging

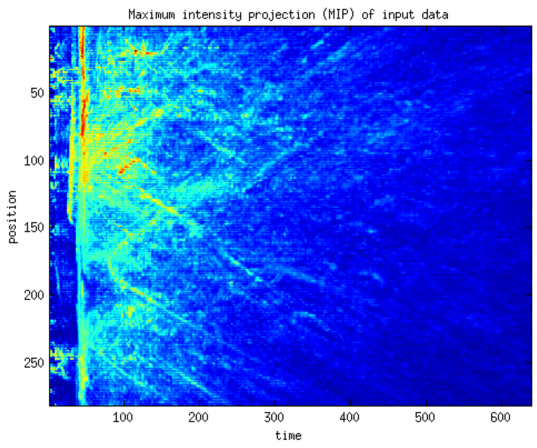


Figure 1. Input signal data recorded outside the tissue.

A photoacoustic imaging is an emerging technique which can provide label-free non-invasive three- dimensional image of the vasculature to the depths of several cm with a spatial resolution ranging from tens to hundreds of microns (depending on the depth). It is based upon the generation of ultrasound waves through the absorption of nanosecond laser pulses by light absorbing tissue chromophores. The acoustic waves travel to the tissue surface where they are detected by an ultrasound receiver array. From the detected signals, the three dimensional (3-D) images (which are proportional to the absorbed optical energy distribution) can be reconstructed. An image reconstruction is based on the acoustic time reversal algorithm.[1]

3. Methods

3.1 Stand-alone program

The C++ solution is implemented as a stand-alone program. An alternative was a mex function in Matlab (built-in function). Its main advantage is passing arguments without saving them to hard drive. A standalone program was choosen because we wanted independence from Matlab. The problem with passing arguments by hard disk can be eliminated by using ssd (solid state drive) or RAMDISK.

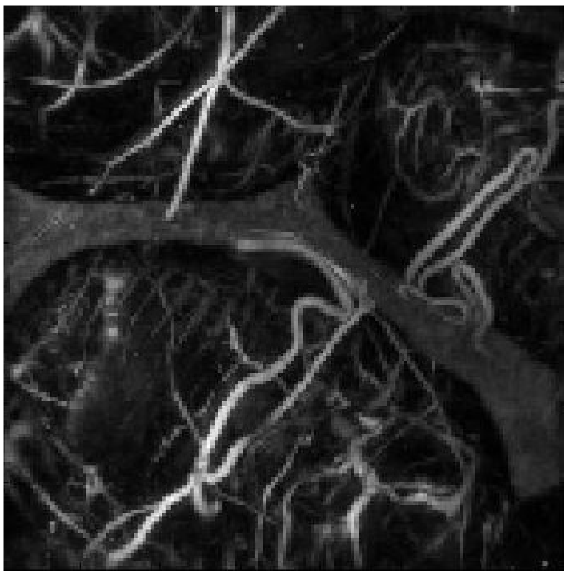


Figure 2. Result, reconstructed image of a mouse embryos in vivo. Original input data.

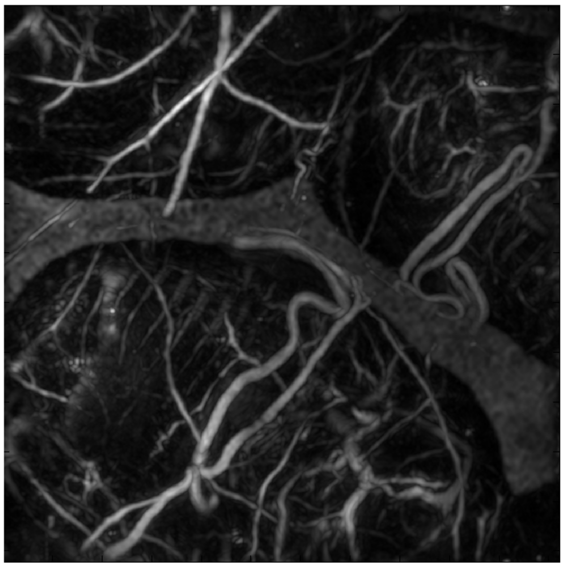


Figure 3. Result, reconstructed image of a mouse embryos in vivo. 64x upsampled input data.

3.2 HDF5

HDF5 is a file format designed to store and organize large amounts of numerical data. It is used for passing arguments from Matlab to a stand-alone C++ program. It is supported by Matlab and there is a library for C++.

3.3 Parallelization and vectorisation

OpenMP was used to parallelize and vectorize the code. OpenMP is a specification for a set of compiler directives, library routines and environment variables which can be used to specify high-level parallelism in Fortran and C/C++ programs.[2]

3.4 High performance fascilities

All work was done on Anselm, a supercomputer cluster in Ostrava, Czech Republic. The Anselm cluster consists of 209 compute nodes, totaling 3 344 compute cores with 15TB RAM and giving over 94 Tflop/s theoretical peak performance. Each node is a powerful x86-64 computer, equipped with 16 cores, at least 64GB RAM and 500GB hard drive.[3]

The compute node used for testing consists of two eight-core Intel Sandy Bridge E5-2665 processor and 64GB memory. Processors support Advanced Vector Extensions (AVX) 256-bit instruction set.

Intel Sandy Bridge E5-2665 Processor:

- eight-core
- speed: 2.4 GHz, up to 3.1 GHz using Turbo Boost Technology
- peak performance: 19.2 Gflop/s per core

3.5 Testing

PAPI (Performance Application Programming Interface) was used to measure various performance indicators such as time, FLOPS(FLoating-point Operations Per Second) or distribution of load on CPU cores. Rusage(Resource usage) was used to measure memory use.

4. Implementation

4.1 Analysis and implementation of Matlab code

Matlab Profiler was used for analysis of individual lines of the code. At first, the data was saved before and after line that was implemened in C++ on one core and outputs are compared. After that the C++ code was parallelized and optimalized for the hardware. At the end, the solution was tested and measured. The lines which take the most time in the code were selected at first.

Table 1. Lines which take most time

Line	
Vq = F(Xq,Yq,Zq);	(1)
p = sf.*fftshift(fftn(fftshift(p)));	(2)
p = real(ifftshift(ifftn(ifftshift(p))));	(3)

In the Table 1 the first line is part of trilinear interpolation of three-dimensional data. Data are interpolated only in one direction, in one coordinate axis. The second and third lines are shifts and forward or inverse fourier transforms.

Trilinear interpolation. Trilinear interpolation was implemented to interpolate only in one direction. It was paralelised and searching for nearby points was optimized for input data.

Shifts. Data is swaped between two parts of memory in this part. This part is also paralelised. Data is read in sequence from memory to improve performance. Speed of shifts depends on memory speed, too.

Fourier transform. The FFTW3 library was used for forward and inverse fourier transform. FFTW3 plans can be saved to a file and reused for another data input. FFTW3 uses SIMD vectorisation by SSE or AVX.

5. Testing

All tests were done on one node(2 CPUs, 16 cores). 64x upsampled data was used for testing. In Figure 4 you can see comparison even with 8x and 1x upsampled data. In the Table 2 you can see comparation between Matlab and C++ solution. In Figure 4 can be see that on sixteen cores the code is approximately eleven times faster than on one core. In the Table 3 we can see the flat profile of the C++ solution and performance of single functions in FLOPS. Forward and inverse Fourier transform from FFTW3 library provide around 33,000 MFLOPS which is real peak performance you can get with this size of dimensions. Interpolation gives almost maximum performance without vectorisation. The theorethical peak performance you can get with LINPACK but no with real complex code.

Table 2. Comparison between of the C++ and Matlab solution. Acceleration and decrease of memory needs on one node(2 CPUs, 16 cores).

Existing solution		
Input data scale	Computing time existing solution	Memory allocation existing solution
1x	6 s	2.7 GB
8x	64 s	20 GB
64x	999 s	120 GB
New solution		
Input data scale	Computing time new solution	Memory allocation new solution
1x	0.45 s	1.1 GB
8x	4.7 s	8.5 GB
64x	46.4 s	67.5 GB

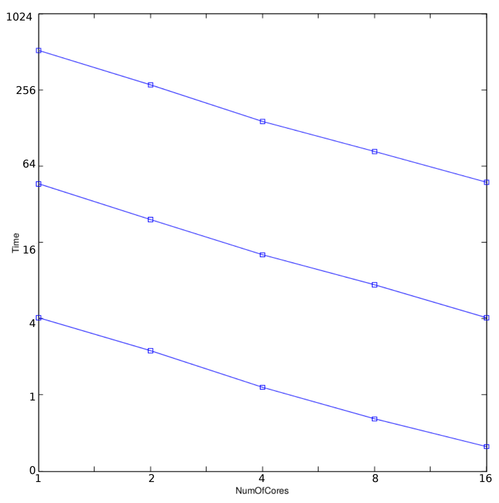


Figure 4. Concurrency and scalability of threads(1, 2, 4, 8, 16) for 64x, 8x and 1x upsampled data. Time decrease almost linearly with more threads.

Table 3. The flat profile of computing, 64x upsampled input data. Some of the most important functions.

Function	Time	MFLOPS
Fourier transform	5.09 s	33,224.3
inv. Fourier transform	5.17 s	32,614.5
Interpolation	27.22 s	20,661.3
Shift	1.26 s	0.000258
Inverse shift	1.25 s	0.000447
Whole computing	46.40 s	19,384.4

In the Table 3 can be seen the most important functions, their computing time and MFLOPS. Shifts work with memory hence FLOPS should be zero, there is a measurement error. Other functions uses 7 to 10 percents of theorethical peak performance of the CPUs. In the Table 2 you can see that C++ solution is many times faster than Matlab one.

6. Conclusions

[Paper Summary] This study has shown that photoacoustic imaging implemented in Matlab can be accelerated and optimised by using hardware-friendly code.

[Highlights of Results] C++ solution can be up to twenty times faster than Matlab solution. It reaches seven to ten percent of theorethical performance of CPU, which is nearly maximum for real application without vectorisation.

[Paper Contributions] The ability to obtain high resolution images of the vasculature or soft tissues many times faster than reference solution in Matlab.

[Future Work] The work will be used as a part k-Wave and can be even more optimized by vectorisation.

7. Acknowledgment

I would like to acknowledge the support of Jiri Jaros.

This work was supported by the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), funded by the European Regional Development Fund and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme, as well as Czech Ministry of Education, Youth and Sports via the project Large Research, Development and Innovations Infrastructures (LM2011033).

References

[1] Jan Laufer Francesca Norris Jon Cleary Edward Zhang Bradley Treeby Ben Cox Peter Johnson Pete Scambler Mark Lythgoe and Paul Beard. In vivo photoacoustic imaging of mouse embryos, 2012.

[2] OpenMP ARB Corporation. What is openmp, 1997-2013. http://openmp.org/openmp-faq.html.

[3] It4Innovations. Anselm cluster documentation, 2014. https://docs.it4i.cz/anselm-cluster-documentation.