**A Comparative Analysis of Implementing a Deconvolution of a Diffused Image in Real-Time**

Ray Duran  
Department of Electrical Engineering and Computer Science  
Univertisy of North Dakotaray.duran@und.edu

*Abstract*- **There have been many advancements in miniaturized microscopes that produce in-vivo calcium extraction imaging in the study of neuroscience. Typically these small form factor cameras use small lenses but are restricted by the normal physical tradeoffs of resolution and field of view. In place of small lenses there has also been a lot of development for on-chip fluorescence microscopy using diffuser lenses that can allow for better tradeoffs. While diffuser lenses provide a low-cost and simple alternative to traditional lenses, they do require the solution of an ill-posed inverse problem that for in-vivo calcium extraction require real-time processing. We aim to provide in this paper a comprehensive comparison between the latency computation time using a gradient descent optimization minimizing a convex function for three processing devices, a Graphical Processing Unit (GPU), Central Processing Unit (CPU), and Field Programmable Gate Array (FPGA).**

I.Background

At its’ core calcium imaging requires the processing of four major areas: background filtering, motion correction, neural enhancement and finally calcium trace extraction for processing.

While calcium imaging defines the approach technique with which we intend to study neural activity, it remains how best to gather the light field, fluorescence light, considering the maximum field of view, resolution, and efficiency in collecting light. Here, we investigate the use of diffusers as our front-end system for a calcium imaging

Diffusers provide a light field representation of 4-D information into a 2-D format, and in the process capture spatial and angle information. While in this paper diffusers provide our principal optical component, they have been successfully used in many other application (3D neural activity [3], compressive radar imaging [4], synthetic aperture [5], and visual odometry [6].). As a highly diffractive medium, the speckles (caustic patterns) from interference in our diffuser create a type of signature that can encode plenoptic information about our illuminated object and with proper assumptions allows us linearity in reconstruction that lends itself to well-established inverse problem definition and optimization techniques. To aid us in solving diffuser inverse problems with less than full rank we make use of the randomness they provide is to see if this fits within the theoretical framework of compressive sensing. Mathematically, if the combination of the measurement and representation matrix satisfy the property of mutual coherence then full reconstruction of the sparse system is possible.

The literature of using diffusers for lens less applications has been explored by many(/diffusers). In addition to their devicway( amplitude mask Flatscope Paper by Rice univ and Jesse K Adams found in folder (/fluor\_micro) Finally there are other ways of approaching the reconstruction problem tht do not entail, one example is the study of phase space itself, of light as it passes through a scattering medium, [ Optical Phases Space meas, Liu from UC Berkeley}

Having briefly mentioned the overall system modality, calcium imaging, and our optical component choice, diffusers, we now outline the three technologies we intend to compare for the solving of our inverse problem: Graphical Processing Unit (GPU), Central Processing Unit (CPU), and Field Programmable Gate Array (FPGA). Here, it is best to investigate these three computing paradigms by studying three aspects of computing: 1. The programming model: As the language that these computing engines use highlights their internal architecture and will allow us to gain insight into how latency can be computed. 2. How memory is partitioned and used in the device. And 3., how we achieve concurrency in our computation.

Of the three technologies, Graphical…. (See pic 1a)We will focus on the GPU by Nvidia, that uses a language called CUDA, that stands for Compute Unified Device Architecture( CUDA) . ( see reference 24 by Lancouture in folder solving\_inverse/non\_real\_time), which is really just an extension of the C/C++ language, with key-words used by the NVIDIA processor. The CUDA code breaks down the programming workflow into three steps: i) Allocating and internalizing data. This means since a CPU is usually the starting point of a program that the CPU copies data to be used by the GPU appropriate memory. ii) Calls the GPU kernel and launches threads. Iii) Upon completion of computing task, copies data back into the CPU memory. Memory usage in the NVIDIA GPU consist of ….. Concurrency in GPUs is achieved by the architecture and a programming construct. GPUs group threads in a hierarchy called warps . Here threads are controlled intrinsically to make sure that there are no data hazards to computing, A programming construct called barrier synchronization allows synchronization across warps that pauses all threads in a block when needed.

For the technology of CPUs we choose to compare the Intel i7 multicore processor which is Talk about CPUs…( pic 1b). List some references for CPU here ref( <https://www.intel.com/content/www/us/en/products/sku/227853/intel-core-i71265ue-processor-12m-cache-up-to-4-70-ghz/specifications.html> CPUs are programmed by C/C++ and often can include librarties that are needed for the math of linear algebra such as Blaze, GNU scientific library , and librsb to name a few. See Referencr for a longer list of math libraries for linear algebra.

<https://en.wikipedia.org/wiki/Comparison_of_linear_algebra_libraries>

Memory in CPUs ….. Concurrency in CPUs is achieved with the use of multicores, of which in our comparison the i7 Intel processor has four.

Field Programmable Gate Arrays (FPGAs) are dominated by two companies, AMD( Xilinx FPGAs) and Intel(Altera FPGAs). Both families of FPGAs are ubiquitous in the embedded comm are widely used in embedded products that require speed (MHz), latency(time) and allow for the flexibility of being able to re-program an algorithm even when the product is deployed in the field. Here is a small sampling of the use of FPGAs in…..

<https://www.sciencedirect.com/topics/engineering/field-programmable-gate-arrays#:~:text=FPGA%20is%20an%20integrated%20circuit,switches%20in%20the%20interconnect%20matrix>.

FPGAs use a family of languages called Hardware Description Language (HDL), of which there are two main dialects, VHSIC Hardware Description Language (VHDL) and Verilog, with System Verilog a newer version of Verilog. HDL allows capture of a design down to a level called Register Transfer Logic (RTL) which breaks down the computation into fundamental digital logic elements such as flip-flops(registers), state-machines, shifters, muxes, adders, AND and OR gates, etc. This flexibility of course permits almost any kind of computation and data transfer with the only limitations being speed and resources of the device. The memory of the FPGA exists throughout the device and can therefore be allocated as shared or global memory depending on the exact application. Additionally, modern FPGAs also consist of interfaces, DDR memory as one example, that allow for the expanse of external memory, thus greatly enlarging the memory available for logic to process. Finally, since an FPGA is essentially a sea of gates that operate with many clocks, concurrency is implicit as a result.

<Show picture of GPU, CPU and FPGA>

1. Methods

While there are many ways to solv

Diagram

Description automatically generated

1. In theory a lens less system has all the information available at the sensor. However, the problem is ill-posed. Introducing a diffuser gives structure to our point spread function, or random matrix, allowing us to solve an otherwise intractable problem. If we enforce sparsity as a prior, and non-negativity (no negative pixels), minimize for the least squares we are trying to solve the equation below
2. Text

   Description automatically generated
3. Here in (1) we do not show the regularization term that helps enforce sparsity, instead we want to show the main thrust of how we approach minimization overall
4. Text

   Description automatically generated
5. Here, the significance of (4) is that we have reduced the problem of computing A which could be a very large matrix to the adjunct of M and C, a much more manageable problem. (C is introduced as a cropping matrix)
6. A screenshot of a computer

   Description automatically generated with low confidence
7. Here there are a few points to note. The first is that the M matrix in (5) has already been decoupled from A, and this is because if we included the cropping effect into A, the matrix would be ill-conditioned. Continuing with (5), we have just expressed the convolution of “Mv” as a product of Fourier and the inverse of that dot product but have expressed the equation as products of matrices.
8. Text

   Description automatically generated
9. Here we substituted (7) back into (4) to give us (8) and taken the adjunct of (8) to give us (9). At this point (8) and (9) we can implement into code to solve our iteration (10).
10. Text

    Description automatically generated

A picture containing text, gallery, different, clock

Description automatically generated

ADMM

Gradient descent

Algorithm ( from Paper in Fall)

1. System Implementatio

What is multi-row 1D FFT

For both CPU and GPU we perform a column order 1-D FFT followed byua 1D-FFT row-order computation

Sh Diagram

Description automatically generatedow figure

Describe steps of our block diagram and show

In our cropped system where we have reduced the size of an image thru don

We will use a fixed point distance of 16 bits, note this can be adjusted empirically as we begin to implement our system in Simulink.

With CPU

With GPU

( Reference paper is in file : IEEE under FPGa folder/ “An Efficient, Model-Based CPU-GPU Heterogeneous FFT Library”, by Ogata, et al. CUDA can help in some areas here.( See reference [11] from Ogata paper above)

For the Cores (CPU and GPU)we see ( from Ogata [5] and [8]))

Memory limitations

Programming limitations that relate to DirectX or OpenGL

With combined CPU and GPU

With FPGA show our diagram

( From Proposal). How we do what we set out to do. Some of this comes from proposal

1. Results

( Running in Simulink)

Figure that shows images between diffused image and FFT of it.( Maybe get this picture with Python)

Estimate with FPGA

In our system we will only compare the 2- dimensional FFTs since they account for the major processing modules in our FISTA deconvolution design.

Estimates with CPU

Estimates with GPU

Estimate with CPU and GPU in tandem.

Do the times meet our system requirements for processing 30 frames a second for a size image that has been downsampled from the original full CMOS array.

1. Discussion and Conclusion

Talk about calcium imaging to be done Other types of processors

The reason we want to use a diffuser is for the tradefoss that we have discussed but also the reason that we are interested in a fast way to do compitations is that the processing of the steps for calcium imaging also requires a certain amount of time

??? Why even try to comnibe in real time the functional processing of diffusion with calcium imaging

Well I would say that threre are applications and papers that discuss real time calcium imaging BCAS(UCLA folk) and that adding the diffusion we are really just chaging the front end of how we collect light to better enhance the overall system. Said otherwise we have just added another pre-processor step tlo the systemn that will allow us to improve the accuracy of detecting and extracting the neurons of intereste…

There are models that can be used to calibrate

Talk about models and variant and invariant

While in this paper we focus on the front-end processing of reconstructing the image,

A little about optogenetics

1. References

FPGAs

<https://www.sciencedirect.com/topics/engineering/field-programmable-gate-arrays#:~:text=FPGA%20is%20an%20integrated%20circuit,switches%20in%20the%20interconnect%20matrix>.