Development & Application of a Closed-Loop Continuous Optical Neural Interface

Mark Bucklin

April 13, 2017

Abstract

The latest generation of genetically encoded sensors to emerge from molecular engineering labs are highly sensitive. These - combined with equally critical advances in the performance of affordable image sensor – have been put to use in labs conducting research neuroscience research to enable high-throughput detection of neural activity in behaving animals using both multi-photon and traditional wide-field fluorescence microscopy. Unfortunately, expanded sensing capability can generate a flow of data in proportions that challenge the standard procedures used to process, analyze, and store captured video. The torrent can easily overwhelm and debilitate, even when applying the latest and greatest from our ever-expanding arsenal of cluster computing resources. Sensing capabilities available to scientists, physicians and engineers will continue to grow exponentially, while traditional raw data storage and batch-processing routines will impose the same limits on throughput utilization.

The work presented here demonstrates the ease with which a dependable and affordable wide-field fluourescence imaging system can be assembled, and integrated with behavior control and monitoring system such as found in a typical neuroscience laboratory. Application of standard image processing and computer vision routines demonstrate the remarkable value of such a system, but also highlights the woeful inability of standard batch processing routines to manage the volume of data available. After describing a slew of marginally successful naive attempts to pre-shrink long streams of raw video data to more manageable proportions, a more likely plan is presented. Here you will find the strategic ingredients to consider if your intent is to transform an abundant flow of raw data into proportionally informative knowledge. Certainly, aggressive deployment of streamed computation on graphics processing hardware will be vital component, but not solely sufficient. A likely solution will also recognize opportunities afforded by implementing performance-tuned data structures, modular and dynamically reconfigurable data processing elements, and graph oriented stream semantics coordinating data-flow.

Forward

I have structured this document to roughly coincide with a chronological account of 6 years spent in a neuro-oriented biomedical engineering lab. My role in the lab was centered around exploratory device design and development, mostly targeting application in neuroscience research, with intended users being neuroscientist colleagues. One of the lab's most remarkable assets is the breadth and diversity of its constituents in terms of their skills and experience, both within and between the engineering/development and the science/medical sides of the lab. All efforts stood to benefit from the close proximity to skilled colleagues, most notably for the complementary guide and provide roles that assisted the development process of new devices and the experiments they were intended for.

My initial experience in optoelectronic device development was as an undergrad at Columbia University where I was advised by Elizabeth Hillman, and developed a device that combined thermography and near-infrared spectroscopy in a portable and inexpensive device intended to provide early detection of adverse neoplastic changes through at-home daily monitoring, particularly targeting use by patients with high-risk for breast cancer.

I then went to the Das Lab where I developed macroscopic imaging systems used for intrinsic imaging in the visual cortex of awake primates.

As a MD/PhD student, I attempt to maintain a potential to adapt the end-products of each development for clinical applicability.

The story presented here is rather unusual in that success precedes failure. The volume of tangible presentable results is greatest toward the beginning stages of the work described here. This unusual inversion is what make this story worth hearing, however. Thank you for taking the time to read this. I hope that at least the technical information provided herein, if not the procedural insight, is valuable in your current or future endeavors.

Introduction

Optical techniques for observing neural activity have advanced recently owing to both an evolution of digital imaging technology, and the development of synthetic proteins that act as fluorescent indicators of neural activity. Image sensors, like those found in scientific-CMOS (sCMOS) cameras are larger, faster, and more sensitive than what was previously available in science-grade cameras. Meanwhile, the latest generation of Genetically Encoded Calcium Indicators (GECIs), collectively called GCaMP6, reports fluctuations in neural activation with extremely high fidelity. This combination of developments enables neuroscientists to open a wider channel to the brain than previously possible – using

conventional epifluorescence microscopy techniques – enabling simultaneous recording from hundreds to thousands of neurons. Expanding the fraction of the observable neurons in an interconnected network may provide insight into mechanistic properties of neural disease, or may lead to a better understanding of neural coding. Additionally, feeding a large set of neural response information to a machine learning algorithm in a neuroprosthetic application may provide improved predictive performance, even if the exact mechanism of prediction remains difficult to discern. However, a few major challenges currently prevent realization of the potential benefits that these new technologies offer:

1. The increased size of raw data from a single imaging session can easily overwhelm the computational resources typically used to process similar but smaller sets of data.

2. The accumulation of raw data on disk over multiple imaging sessions quickly exceeds the data-storage capacity of most lab-scale servers, forcing researchers to halt data collection to process and delete, a nightmare scenario for some.

3. The experimental design and data analysis procedures that neuroscientists are familiar with applying for network activity data when there are 5 to 10 cells will produce highly biased spurious results, unless provided with many more stimulus-response repetitions, i.e. trials. The number of repeated trials sufficient for producing an accurate description of the neural response to any stimulus is on the order of 2^N, where N is the number of neurons being measured.

The objective of this project is to establish procedures that can address these challenges, then use these procedures to evaluate the effect that expanding available neural response input has on performance of a closed-loop encoder. This closed-loop encoder will attempt to predict changes in motor state of a mouse running on a ball, using sensors on the ball to train the encoder. It will then use the predicted motor state to modulate motor state in another mouse using opsins. This can be thought of as a model neuroprosthetic whos function is to overcome dysfunction caused by pathologically disconnected brain areas, such as exists in Parkinson's disease (PD). The goal will be to increase synchronization of mice beyond chance, such that they tend to run together and rest together.

Below I provide some background on the general procedure for offline video processing. I also discuss some of the issues with carrying out these procedures on a large dataset, and the variety of approaches that I and others have attempted for dealing with the issue. I then introduce the streaming approach (i.e. Aim 2), which is capable of processing video during acquisition and extracting signals directly, saving relevant signals only and discarding or compressing the raw video. This approach relies on GPU programming, so I also provide some background on the application of graphics cards for computationally demanding tasks. Using a graphics card for programming in the MATLAB environment is also discussed.

Capturing wide-field fluorescence images at high spatial and temporal resolution enables us to measure functional dynamic changes in many cells within a large interconnected network. Extracting a measure for each cell in a way that preserves spatial and temporal continuity with uniform/unbiased sampling of the observed signal is achievable, but implementing a procedure to accomplish the task can be made difficult by a number of factors. One class of computer-vision procedure commonly applied to this task is image-segmentation (cell-segmentation in histology applications), a procedure that seeks to represent distinct objects in an image by association of each image pixel with one of any number of abstract objects, or with the background. A variety of algorithms exist for performing this operation efficiently on single images. Most methods can be extended to operate in a 3rd dimension, applied to stacks of image frames to enable tracking cells at multiple depths, or equivalently over time.

However, motion induced by physiologic changes and animal movement necessitates alignment of all frames in the sequence. Moreover, the massive fluctuations in signal intensity from individual and spatially overlapping cells can breed unstable solutions for alignment and radically complicate cell identification routines by disrupting temporal continuity. Implementing a reliable procedure for identifying and tracking the same cells in each frame throughout the sequence thus becomes non-trivial.

Procedures for Calcium Imaging

The general goal of processing image data from functional fluorescence imaging experiments is to restructure raw image data in a way that maps pixels in each image frame to distinct individual cells or subcellular components, called 'Regions-Of-Interest' (ROI). Pixel-intensity values from mapped pixels are typically then reduced by combination to single dimensional 'trace' time-series. These traces indicates the fluorescence intensity of an individual neuron over time, and the collection approximates the distinct activity of each and every neuron in the microscope's field of view. However, this task is made difficult by motion of the brain throughout the experiment, and also by the apparent overlap of cells in the image plane captured from the camera's 2-dimensional perspective. These issues can be partially mitigated with a few image pre-processing steps – alignment of images to correct for motion being the most critical. These options are described in the Methods & Approaches section below. Most software packages geared specifically toward functional imaging implement either of two basic classes of pixel->cell mapping algorithms. One approach is to use image-segmentation routines for computer vision, which seeks to combine adjacent pixels into distinct spatially segregated regions representing objects in the image.

The other common approach is to perform an eigenvalue decomposition on the covariance matrix from a stack of image frames (also called spectral decomposition, or Principal Component Analysis, PCA), resulting in an assembly of basis vectors defining the weighting coefficients for each pixel. Multiplying the

basis-vectors (i.e. "components") with all frames produces a one-dimensional trace for each component. The linear combination is similar to the weighted image-segmentation method in that it assigns fractional coefficients to pixels. However the procedure for computing the covariance matrix employed by PCA operates on as many pixels as are in the image, multiplying each with every other pixel – a problem with np^2 complexity, where p is the number of pixels in the image. I mention these issues inherent to PCA not because this project will attempt to address them, but because this project was initiated following tremendous difficulty attempting to use PCA-based cell sorting methods with large datasets.

Computer Software Environments for Image Processing

The widespread usage of MATLAB in neuroscience communities lends potential for greater usability and easier adaptation to software developed in this environment. While software development environments with a focus on "ease-of-use" have traditionally presumed crippling sacrifices to computational performance, this assumption is getting to be less accurate.

Standard programs include ImageJ, the built-in routines in MATLAB's Image Processing Toolbox, Mosaic from Inscopix, which is merely a compiled version of MATLAB routines which uses the MATLAB engine, Sci-Kits Image for Python, and a remarkable diversity of other applications. MATLAB is a commercial software development platform which is geared toward fast production and prototyping of data processing routines in a high-level programming language. It implements several core libraries (LINPACK, BLAS, etc.) that make multi-threaded operations on matrix type data highly efficient. While MATLAB has traditionally been a considered the standard across neuroscience research labs, it was also well recognized that its performance was lacking for routines that aren't "vectorized", when compared to applications developed using lower-level languages like FORTRAN, C, and C++. Nevertheless, it remained in common use, and recent releases have added features that can drastically mitigate its performance issues, particularly through the development of a "Just-In-Time" compiler that automatically optimizes the deployment of computation accelerator resources for standard MATLAB functions. This feature enables code that performs repeated operations using for-loops or while-loops nearly as fast as equivalent code written in C. Additionally, code can be compiled into executable format using the Matlab Compiler toolbox, or used to generate equivalent C or C++ code using Matlab Coder.

Computational Resources for Processing Large Data Sets

Routines for extracting the activity in each cell from a collection of raw imaging data rely on an ability to simultaneous access many pixels separated over space and time (and consequently separated on disk). For long recording sessions,

however, the size of the collection of stored image data grows dramatically. This substantial increase in the size of data easily exceeds the capacity of system memory in the typical workstation computer available to researchers. Thus, performing the necessary processing routines using standard programs is often unfeasible.

Another popular approach to this challenge is the migration of processing routines to a cluster-based system. In this way image data can be distributed across many interconnected computer nodes capable of performing all locally restricted image processing procedures in parallel, then passing data to other nodes in the cluster for tasks that rely on comparisons made across time. Access to clusters capable of performing in this way has historically been restricted to those working in large universities or other large organization, and the diversity of cluster types is sizeable, with clusters often having very particular configuration requirements for implementing data processing jobs efficiently. These issues would pose some difficulty to the use and shared development of software libraries for image processing routines, although the growth of "cloud computing" services such as Amazon's EC2 and the Google Compute Engine, and also collaborative computing facilities like the Massachusetts Green High-Performance Computing Center (http://www.mghpcc.org) mitigate many of these issues. Additionally, efforts to produce a standardized interface for accessing and distributing data, and for managing computing resources across diverse computing environments have seen appreciable success. Apache's release of the open-source cluster computing framework, Hadoop, and a companion data-processing engine called Spark (http://spark.apache.org/), has encouraged a massive growth in collaborative development projects, a consequently increased the availability of robust shared libraries for data processing in a variety of applications. The Spark API can be accessed using the open-source programming Python, and also using other languages like Java, Scala, or R. One project specifically geared for image processing of neural imaging data is the Thunder library, a Spark package released by the Freeman lab and developed in collaboration with a number of other groups at Janelia farm and elsewhere.

Many applications will find the recent improvements in accessibility and standardization make cluster computing an attractive and worthwhile option for processing a very large set of reusable data. However, this strategy would impose harsh limitations for a neuroscientist with a project that is continuously generating new data, as the time required to transfer entire imaging data sets across the internet may be prohibitive. Unfortunately, storage on the cloud is not so unlimited that it can manage an accumulated collection of imaging data generated at anything near the rate that sCMOS cameras are capable of producing. This rate imbalance is a central motivating issue for Aim 2 this project, and is discussed in more detail below.

The current generation of sCMOS cameras can capture full-frame resolution video at either 30 fps or 100 fps, depending on the data interface between camera and computer (USB3.0 or CameraLink). At 16-bits per pixel and 2048x2048

pixels, the maximum data rate for the USB3.0 camera is 240 MB/s. Imaging sessions typically last 30-minutes or less. However, pixels are typically binned down 2x2, and frame rate often reduced; processing speed and storage constraints are the primary motivation for doing so. The effect of doubling resolution on processing time when using the graphics card is nearly negligible, however. By identifying ROIs online and extracting the traces of neural activity allows us to discard acquired images and instead store the traces only, or feed them into an encoder for online analysis.

Graphics Processing Units were traditionally developed for the consumer gaming market. They are optimized for the process which involves translating a continuous stream of information into a two-dimensional image format for transfer to a computer monitor. In the context of gaming, the stream of information received by a GPU describes the state of objects in a dynamic virtual environment, and is typically produced by a video game engine. These processors are highly optimized for this task. However, they are equally efficient at performing the same type of procedure in reverse – reducing a stream of images to structured streams of information about dynamic objects in the image – and thus are popular for video processing and computer vision applications.

Any GPU architecture will consist of a hierarchy of parallel processing elements. NVIDIA's CUDA architecture refers to the lowest level processing element as "CUDA Cores" and the highest level as "Symmetric Multiprocessors." Typically data is distributed across cores and multiprocessors by specifying a layout in C-code using different terminology, "threads" and "blocks." Blocks are then said to be organized in a "grid." Adapting traditional image processing or computer vision algorithms to run quickly on a GPU involves finding a way to distribute threads efficiently, ideally minimizimg communication between blocks.

MATLAB makes processing data using the GPU seemingly trivial by overloading a large number of built in functions. Performance varies, however, and often the fastest way to implement a routine is by writing a kernel-type subfunction – written as if it operates on single (scalar) elements only – that can be called on all pixels at once, or all pixel-subscripts, which the function can then use to retrieve the pixel value at the given subscript. The kernel-type function is compiled into a CUDA kernel the first time it's called, then repeated calls call the kernel directly, having minimal overhead. Calls go through the *arrayfun()* function.

Data transfers between system memory and graphics memory is often the major bottle-neck. Therefore, this operation is best performed only once. However, once data is on the GPU, many complex operations can be performed to extract information from the image, all while staying under the processing-time limit imposed by the frame-rate of the camera sending the images.

The function of the brain is to translate/encode sensory input into neural output that actuates an effect that promotes survival of the organism or propagates to promote the survival of offspring (generation of a response). It does this

by communicating input through interconnected neurons via converging and diverging connections which comprise the neural network. One way we study the brain is by testing and observing the properties of individual neurons and the response to changing conditions at the direct connections they form with others. Another way is by observing a collection of neurons and to measure their response to variable conditions in their external environment, either by recording or stimulating variations in sensory input, or measuring an organisms physical/behavioral response.

One might presume that the expansion of information provided by being able to measure activity from a larger proportion of cells in a network would make it easier to analyze stimulus-response type experiments and gain insight about underlying functional mechanisms. Unfortunately, the correlation and information theoretic procedures traditionally used to make these associations suffer from a systematic bias that grows exponentially with the number responses considered for each stimulus (i.e. the number of cells included). The number of trials necessary for overcoming this bias gets exponentially large, though methods do exist for bias correction, such as through shuffling/resampling tests.

A systems neuroscience experiment will benefit from online feedback in one or both of two ways:

1. For an experiment that seeks to learn the neural response/pattern associated with a *specific stimulus*, it can inform the user whether the current number of trials – i.e. repeated presentations of the stimulus – will be sufficient for overcoming *limited sampling bias*. This could be done by testing pattern hypotheses online to subsets of the collected data and assessing their stability.

2. If the intention of the experiment is to study neural coding in general, for which it's sufficient to have an *arbitrary stimulus*, then online pattern recognition feedback can aid in maximizing the information in the response about that stimulus, either by directing modification of the stimulus, or directing modification of the field-of-view.

Online streamed processing, as specified by Aim 2, addresses the issues of processing and storingfvy for sufficient learning from large networks possible. Additionally, I propose a strategy in the Aim 3 methods section by which incorporating this online processing stream into stimulus-response-type experiments could help correct *limited sampling bias*, enabling neural coding analysis in large populations of neurons (Ince 2009).

Overall, however, the third goal of this project will focus on the ability to use the expanded information made available by the first two project components to train an encoder that predicts intended motor states from one healthy mouse, and uses the predictions to direct neuromodulatory control of another mouse. This setup will simulate pathologic disconnection in a brain, and will test the ability to distinguish intention to start or stop running, and apply that in a way that performance is easily measureable.