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

Mark Bucklin

April 13, 2017

List of Tables

List of Figures

1 Cover Page

2 Acknowledgements

The support and patience I have received from my committee has gone far beyond what should be expected of anyone. I can't thank you enough. - Xue Han, Ph.D. - Jerome Mertz, Ph.D. - Ian Davis, Ph.D. - Tom Bifano, Ph.D. - David Boas, Ph.D.

---

3 Abbreviations

4 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.

4.1 Abstract:

The latest generation of genetically encoded sensors emerged 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 fluorescence 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 demonstrates 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.

zIntroduction

4.2 Optical Imaging of Neural Activity

Optical techniques for observing neural activity have advanced recently owing to both an evolution of digital imaging technology, and the development of engineered 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.

4.3 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.

4.4 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 multithreaded 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.

4.5 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 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, 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.

Streaming processing addresses the issues of processing and storing for sufficient learning from large networks possible. Additionally, I propose a strategy in the 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_presence_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.zProject Prologue {#sec:project-prologue} This chapter describes several projects that were

started early during my graduate studies. Each project is similar in that they are outside the realm of optical imaging of neural activity, which is the focus of the rest of this dissertation. Nevertheless, they are included here because the issues they bring up will later inform the approach I take in the work described in later chapters. The projects described in the following sections are also tied together by a common goal: to enable research in the neurosciences with translation potential for clinical applications.

4.6 Neuroscience Technology Development Background

- Electrophysiology, Histology, Functional brain imaging
  - pros/cons of each
  - very little compromise in tradeoffs
- Filling the gaps between these three approaches ## Behavior Box

I built an experiment apparatus for mice to enable a study being run by Jia-Min Zhuo. The goal of the study wasto elucidate the role of adult-born neurons on mouse behavior, specifically their performance in discrimination tasks. We called the apparatus the "Behavior Box" and modeled it after a commercially available but grossly over-priced box that itself came from other labs [(todo-cite-clelland2009), (todo-cite-creer2010)].

The chamber was constructed with black plastic walls, extruded aluminum framing, and a perforated metal mesh floor 1 cm above a plastic waste tray. A 10-inch infrared touchscreen (ITouch Systems) was mounted over a 10-inch LCD monitor forming one wall of the chamber. An opaque mask with seven windows was placed over the screeen to limit where the mouse could touch. A water pump with infrared detector was located at teh other end of the chamber to provide reward for the water-deprived mice in the study. A white LED strip encircled the chamber from the top, and multiple speakers positioned outside to deliver sound cues. A web camera was fixed above the chamber to record and monitor mouse activity. My contribution to this project was the program for interact with all the system components. This program controlled and recorded experiment progress. I developed the program in MATLAB, and the main components of its function are described below.

4.6.1 IR Touchscreen

The IR touchscreen provided a robust measure of the location of any contact with the animal's paws or nose. The screen was more reliable than either *resistive* or *capacitive* touchscreens, which are much more common in devices like POS systems and mobile phones respectively.

Files in this folder are used to run our "BehaviorBox" system, which features easily customizable control of experiments involving an infrared touchscreen and LCD display along with speakers, water-ports, lights, essentially anything that can be controlled electronically.

COGENT 2000 The graphics/visual-stimulation package used is missing from this folder due to size, but can be downloaded from the source

4.6.2 FrameSynx Toolbox

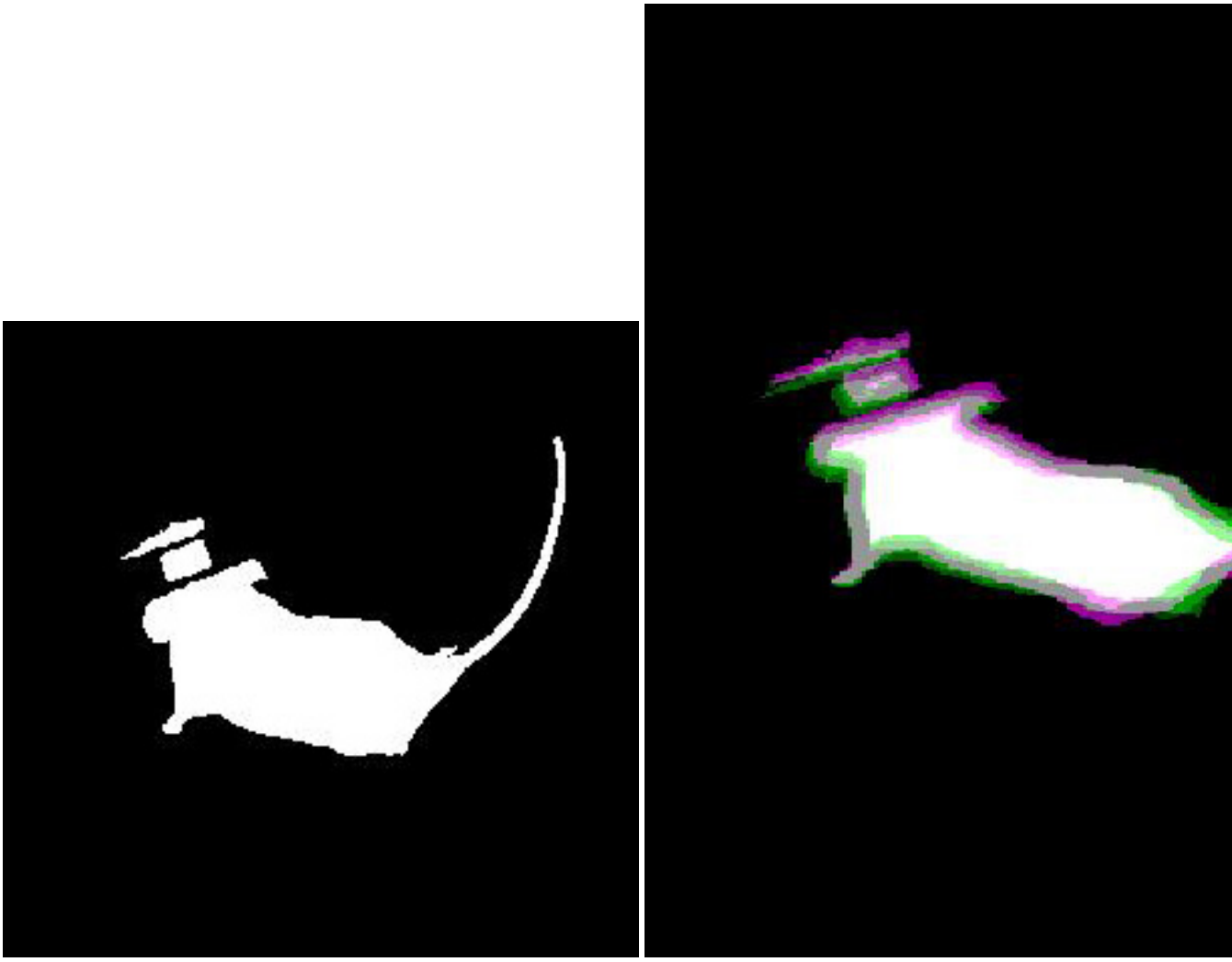
The FrameSynx toolbox for MATLAB was built to synchronize continuous image acquisition with experiments conducted in the neuroscience laboratory setting. While the experiments are conducted in separate software (and potentially on a different computer), FrameSynx listens for messages to start/stop the experiment, start a trial, etc. and responds accordingly by controlling one or multiple cameras and illumination devices, and synchronizing this information with the data acquired. The major contribution to the "Behavior Box" package, and also to later image processing packages is the procedure for definition and storage and of experimental data files, which will be touched on briefly in [(todo-ref-data-file-section)] ## Animal Tracking {#sec:animal-tracking}

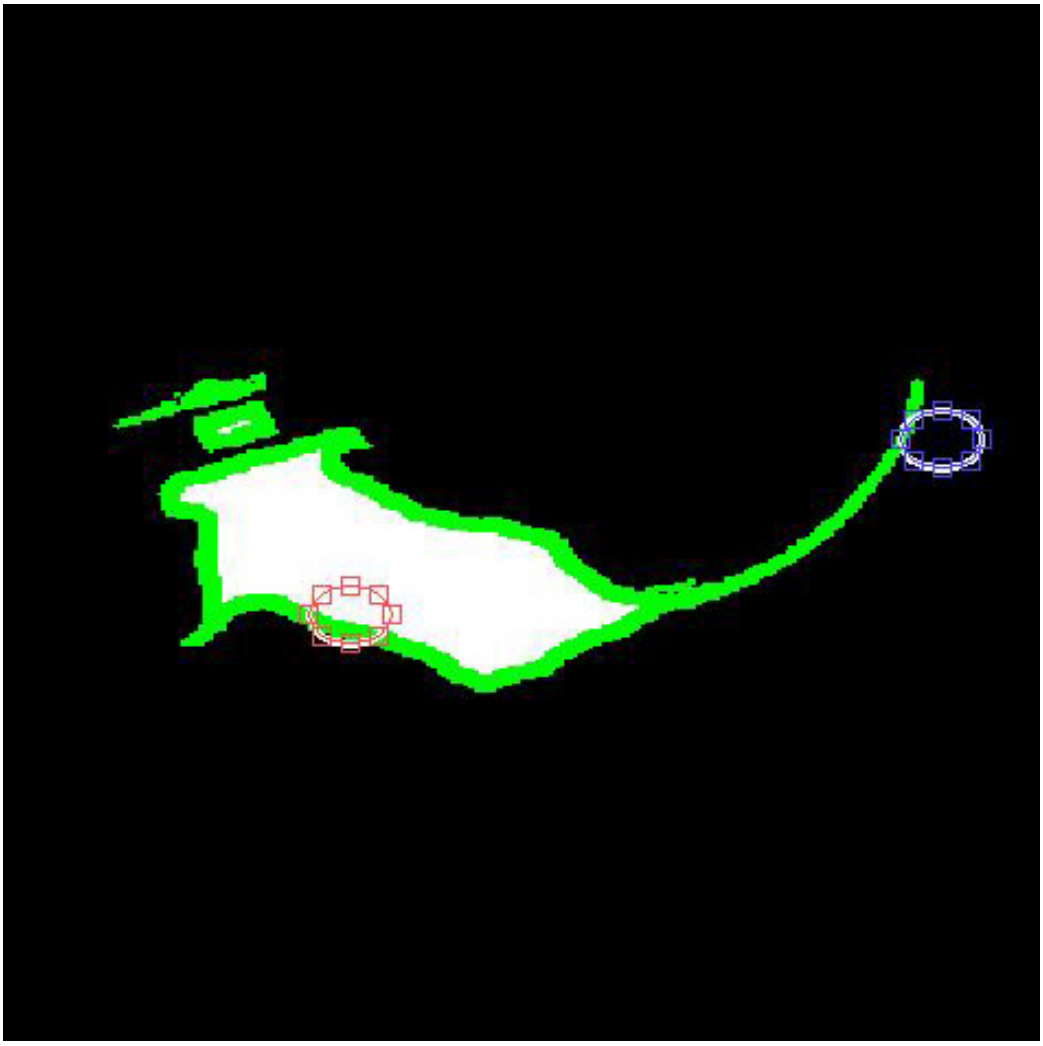
4.6.3 Using Computer Vision to track Position and Orientation

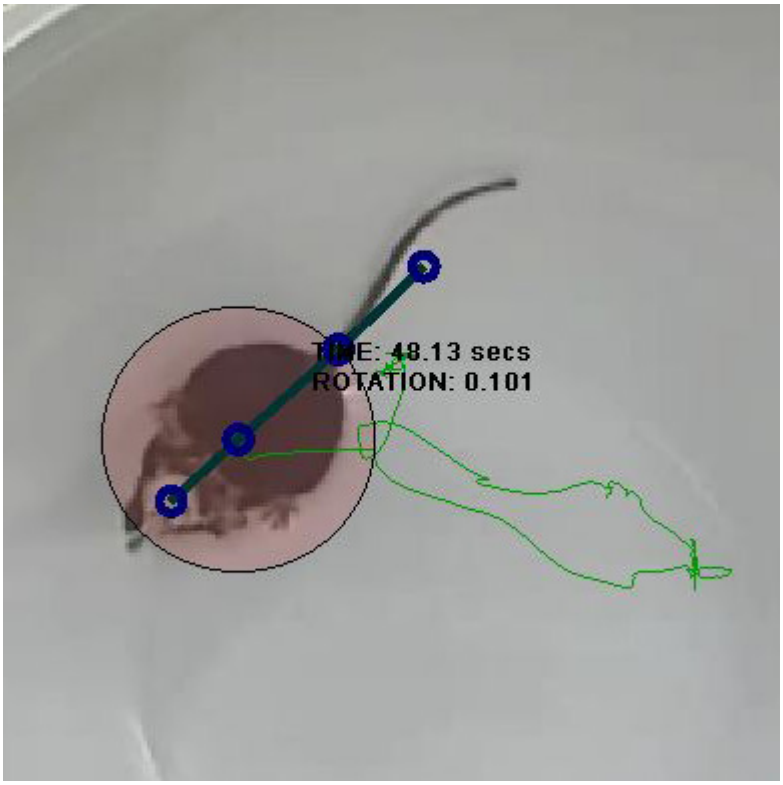
A webcam-based motion tracking box constructed to analyze the movement of our unilaterally lesioned PD mouse model. Video is recorded at 15 frames per second and processed on-line or off-line using a function written in MATLAB. Briefly, this function converts each frame to a black and white image (logical matrix), uses morphological filtering functions to isolate the mouse (remove

mouse excrement) and identify its body (remove the tail), then finds the center of mass in cartesian coordinates (maximum center of projection on x- and y-axes), and the rostral-caudal orientation measured in degrees off the x-axis. Orientation is determined by the index of maximum of a radon transform of the binary image. Processing is accomplished at a rate of 15-16 fps, using a single core, or 64 fps using parallel processing on a quad-core processor with multi-threading enabled. The advantage of this apparatus over the virtual-reality system is that it allows free movement of an untrained mouse, with real-time movement metrics at nearly the same rate as the spherical treadmill.

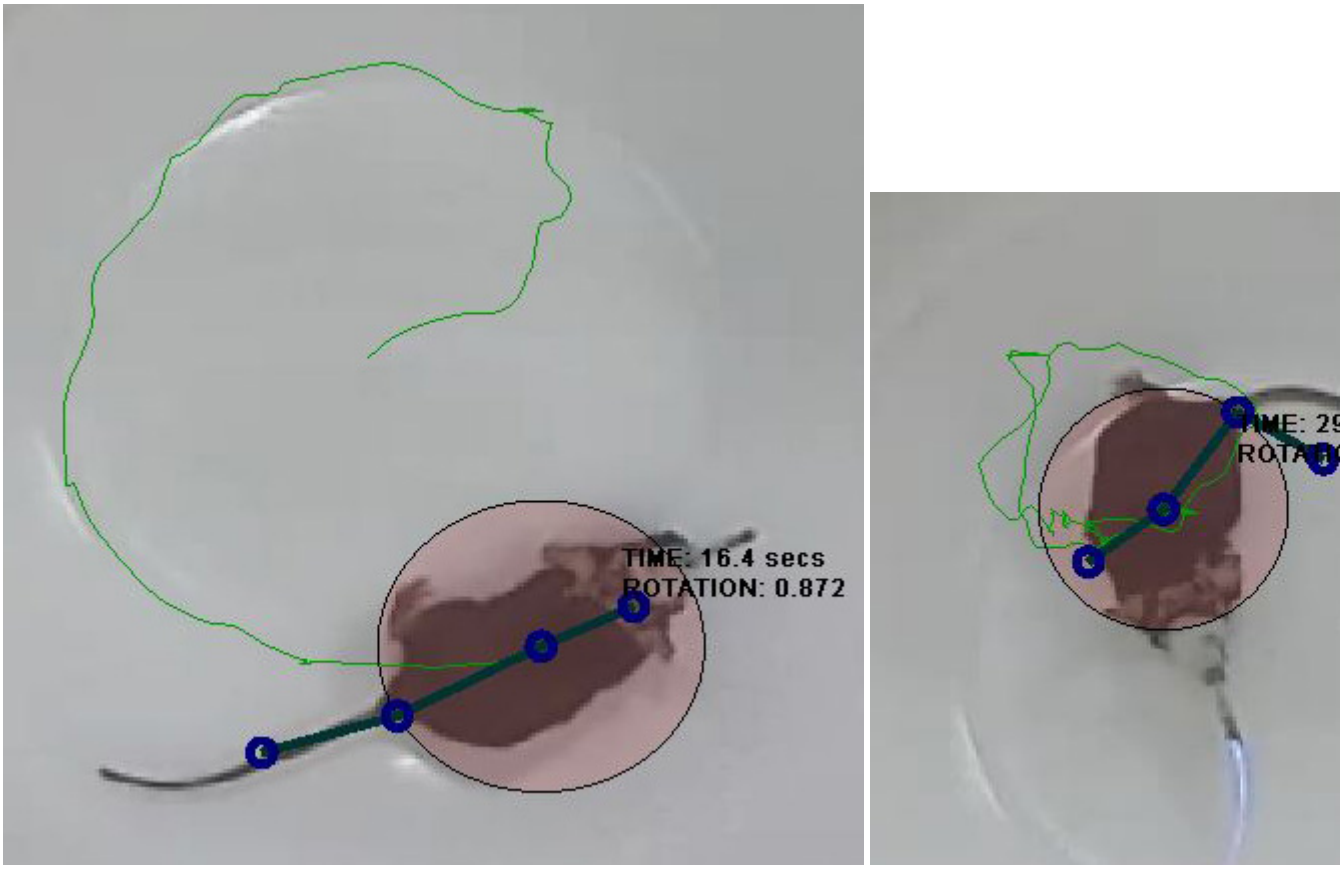
4.6.3.1 Figure:






TIME: 48.13 secs
ROTATION: 0.101


TIME: 128
ROTATION



Caption: Processing steps for automated rotation counting procedure used in hemiparkinsonian mouse study

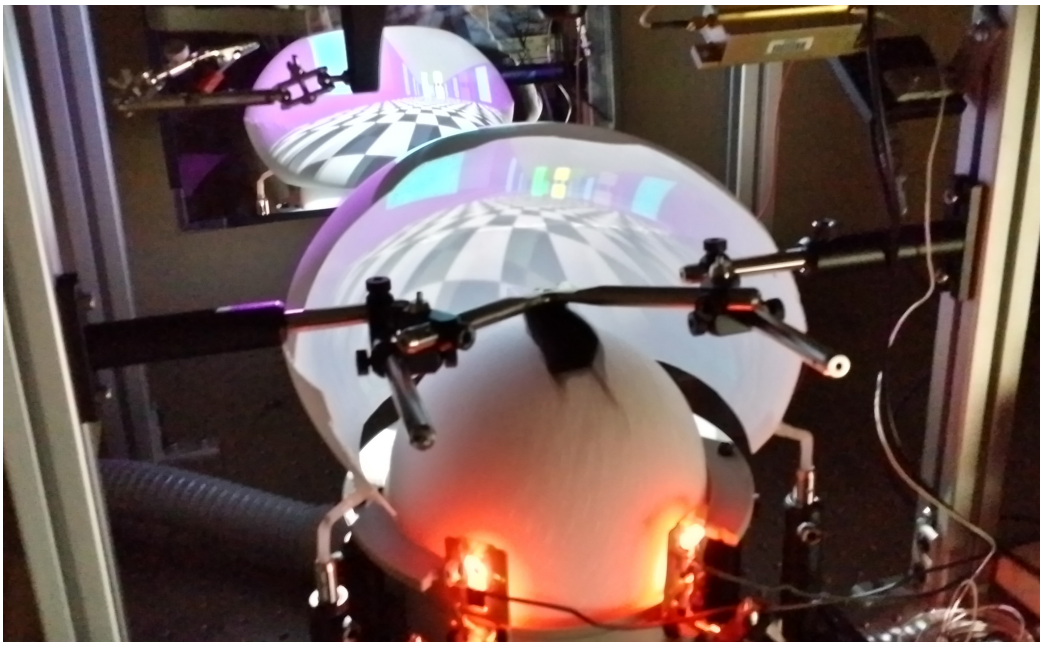
4.7 Spherical Treadmill & Virtual-Reality

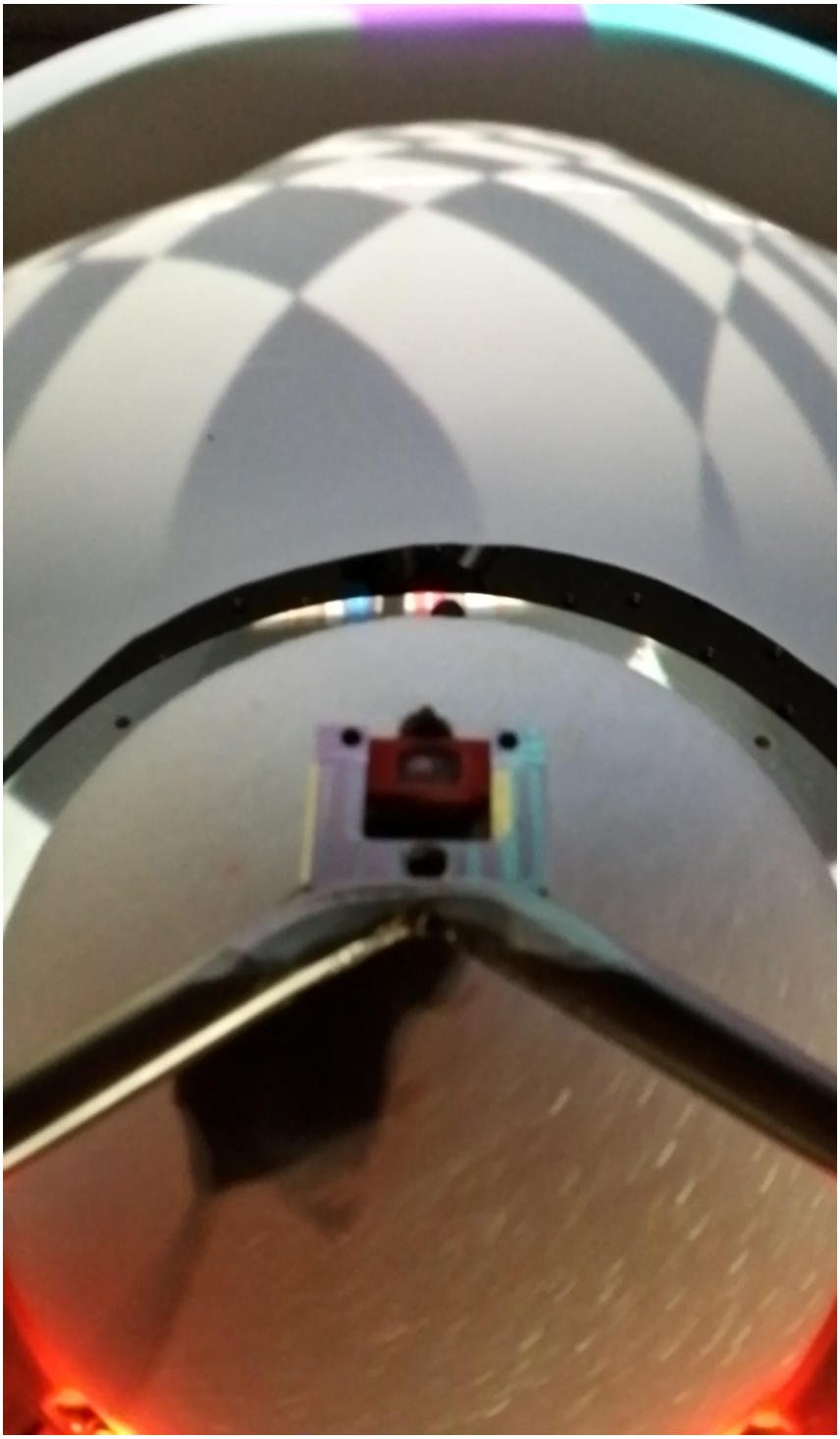
A virtual reality system was assembled, adopting methods from the Harvey lab lab [@harvey_intracellular_2009]. This system allows placement of a head-restrained mouse on an 8-inch diameter polystyrene foam ball supported by a cushion of compressed air, surrounded by a toroidal projection screen. Ball rotation is tracked with two optical computer mice placed orthogonal to each other. Movement vectors are fed into a virtual-reality engine that updates the image projected onto a toroidal screen surrounding the ball, simulating movement through any arbitrary virtual world. Movement vectors are recorded as an arbitrarily scaled translation in the mouse-relative X and Y

axes and rotation around the Z axis, at approximately 30 ms intervals. This behavioral apparatus has the advantage of allowing trivial measurement of the mouse's movement ability while the mouse is head-fixed. The disadvantage is the time and potential confounds involved with training individual mice to use the system.

4.7.1 Treadmill contstruction

4.7.1.1 Figure:





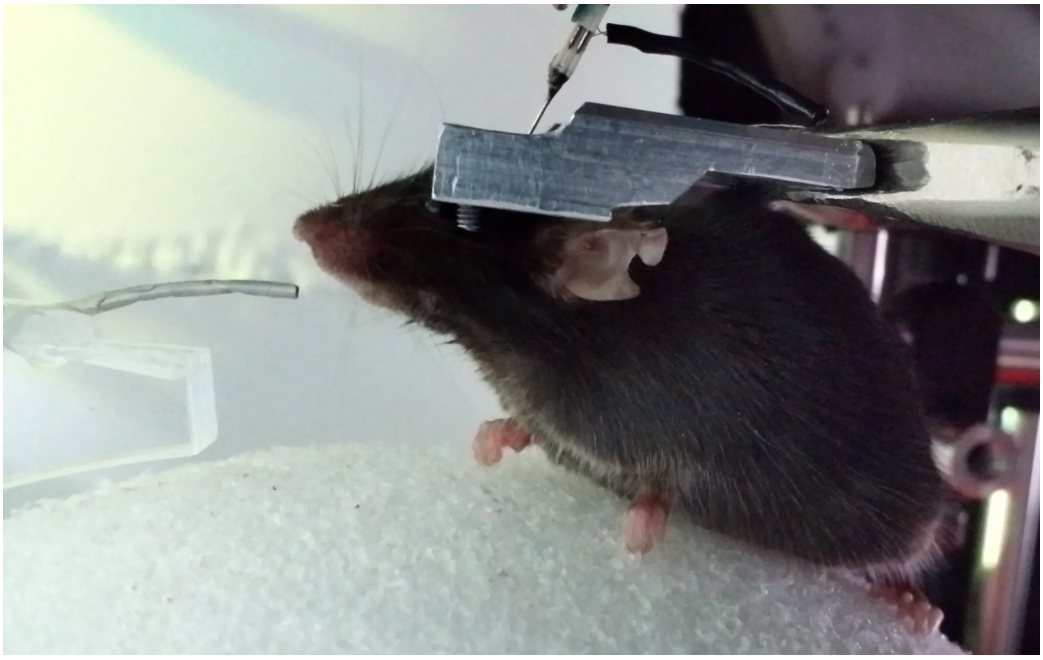
Caption: this

treadmill supports behavior.

4.7.2 Water Delivery

4.7.2.1 Figure:


9
6
3
Enter
2



Caption: waterport

4.7.3 Motion Sensors

Motion sensing was implemented using a linux computer and standard mice at first, and later using precision laser navigation sensors for "gaming" mice and custom firmware written to work with any arduino-compatible microcontroller.

4.7.3.1 Generic USB Computer Mouse with Minimal Linux

Run "mouse_relay.py" on any computer running linux to send xy-data from 2 USB optical computer mice to another computer over an RS-232 serial-port connection. The receiving computer (in this implementation) uses MATLAB to read the values and translate the xy-values from 2 mice on the surface of a sphere into 3 values corresponding to rotation of that sphere around 3 orthogonal axes (XYZ) with their origin at the sphere's center.

RECEIVING FUNCTIONS: The MATLAB class that receives the serial input (xy-values from both mice) is called "VrMovementInterface"

The MATLAB function that translates the double-stream of xy-values from the sphere's surface into rotation around its center is called "moveBucklin.m" and is located in the VIRMEN "movements" folder.

SERIAL FORMAT: XY-Values are transmitted in 'packets' using an ascii formatted string terminated by a newline. Each packet contains the Sensor Number (s) that the reading is coming from, followed by the X-Value (dx), then the Y-Value (dy). The python code looks like the following:

```
> datastring = s + 'x'+dx + 'y'+dy + '\n'
```

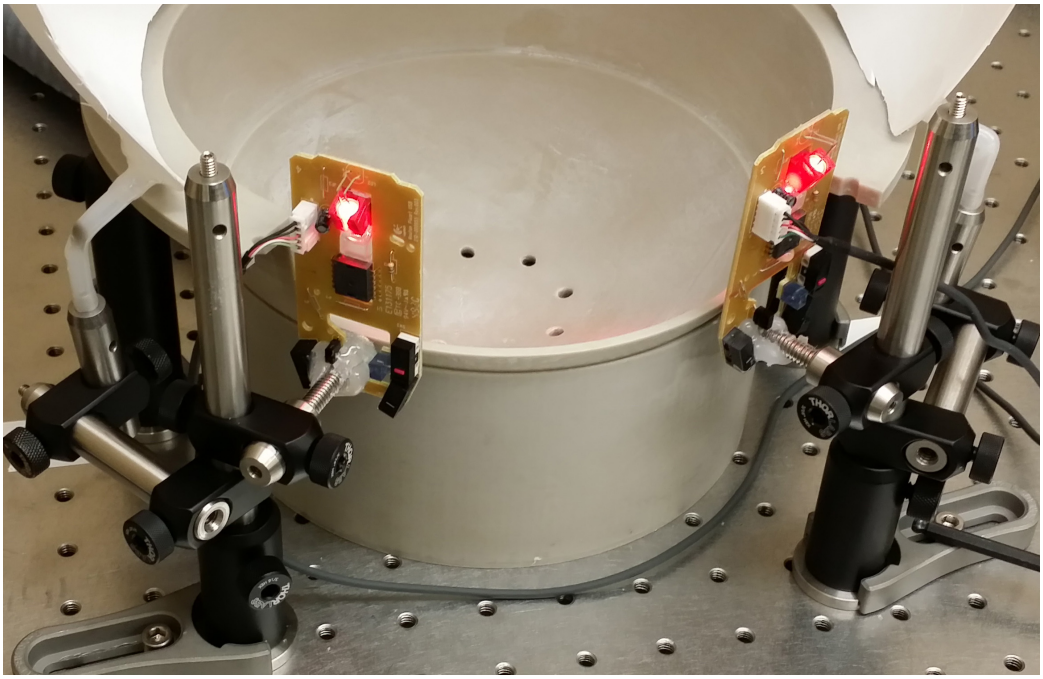
For example:

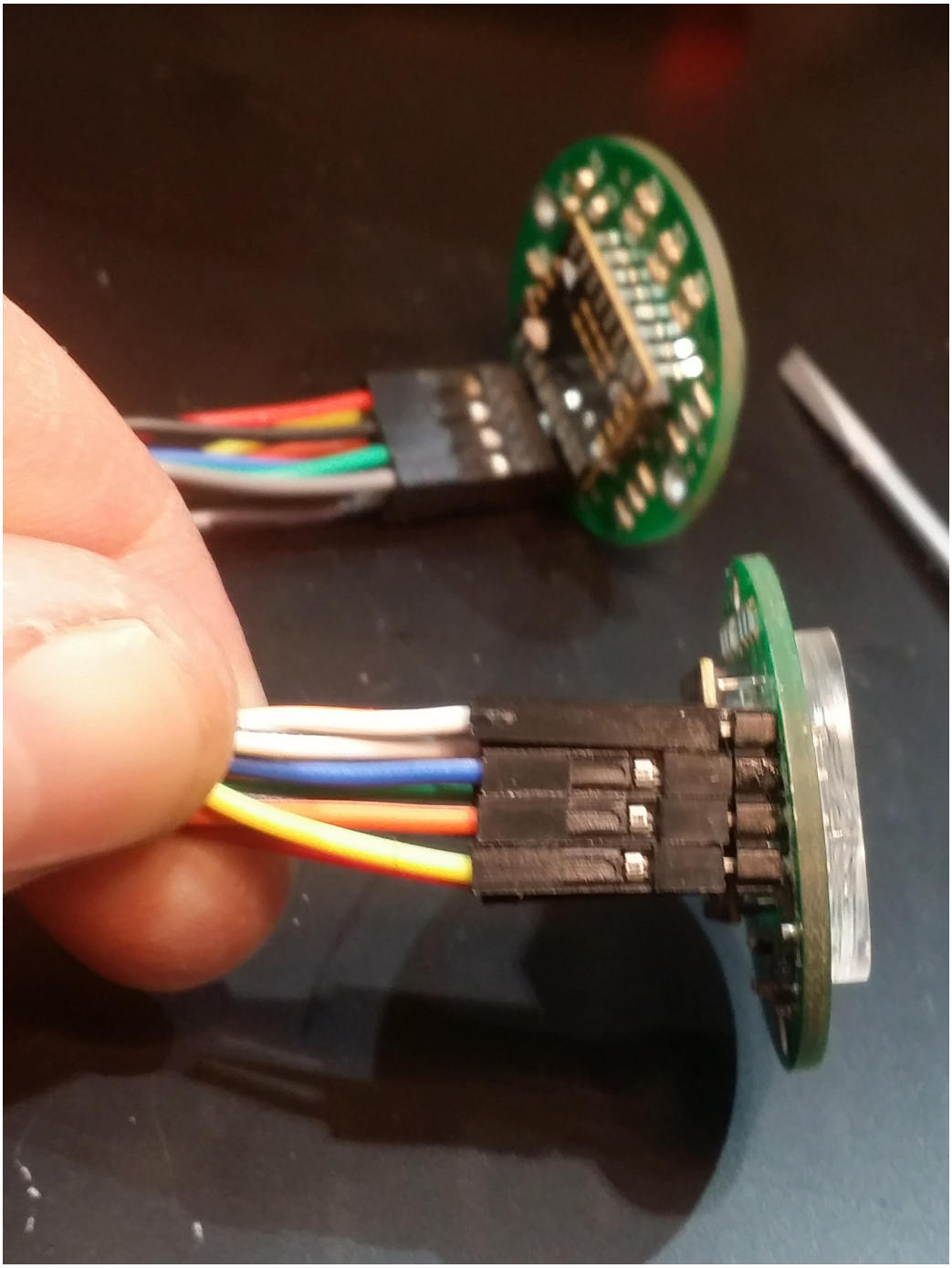
```
> s1x34y-3
```

4.7.3.2 Navigation Sensor Chip with Arduino

Works with ADNS library (Mark Bucklin) to pass [dx,dy] measurements from two ADNS-9800 laser mouse sensors (placed 45-degrees apart on surface of styrofoam ball).

4.7.3.3 Figure:





Caption: Motion Sensors for tracking ball movement. can be used to control

and record speed and direction of mouse movement within the VR world. ## Closed-Loop Diffuse Optogenetic Neuromodulation

Chronic implant for long-term multi-site recording and optogenetic neuromodulation. The implant is fixed to a mouse's skull with dental cement, and sits on top of bilateral 5 mm craniotomies. Electrodes are driven through 32 guiding-tunnels along pre-determined trajectories to bilateral targets in thalamus, striatum, prefrontal cortex, motor cortex and auditory cortex. 2 High-intensity LEDs are coupled to optical fibers driven to mediodorsal and centromedial thalamic nuclei. Electrode trajectories are computed in stereotaxic coordinates and imported into CAD model to construct guiding-tunnel features that facilitate correct placement of electrode tips in brain targets and connection to a circuit board. A-C) CAD model of implant. D) Circuit board for electrode termination and LED power.

Brain disease, often accompanied by enormous personal and economic costs, continues to emerge as among the most pressing contributors to the global disease burden. Unprecedented advances in biotechnology and in portable electronics support tremendous opportunity to conduct research with excellent potential for advanced understanding, improved treatments, and one day cures for these devastating diseases, disorders and conditions. The aim of this project was to would leverage technology from the fast-moving forefronts of electronics and biomedical research to build a next-generation neuroprosthetic.

4.7.4 Background

Deep Brain Stimulation (DBS) has been used clinically since the early 1990's, and is currently approved by the FDA for treatment of Parkinson's disease (PD) and essential tremor, with Humanitarian Device Exemptions for OCD and dystonia. In 1987 a French neurosurgeon observed during a thalamotomy procedure - where stimulation was applied for localization of ventralis intermedius - that high-frequency stimulation was able to suppress an extrapyramidal tremor (@benabid_combined_1987; @liker_deep_2008). Subsequent studies in a non-human primate Parkinson's model suggested high-frequency stimulation (~100-200 Hz) of the Subthalamic Nucleus (STN) was as effective as a lesion, in that it"jammed" neural activity at the tip of the electrode, yet it was tunable and reversible (@eusebio_does_2012).

Long term efficacy is around 50% using a standard clinical scale (UPDRS III)(@beuter_delayed_2009); DBS can provide symptomatic improvement in tremor, rigidity, and bradykinesia, and can also facilitate a substantial reduction in levodopa dosage and consequent reduction in its major side-effects, i.e. dyskinesia (@eusebio_does_2012). While DBS has substantially improved our ability to treat PD and other movement disorders, the therapeutic effect remains insufficient and unpredictable.

The mechanism by which DBS suppresses parkinsonian symptoms is still uncertain. However, there are growing evidence to suggest that increases in synchronized oscillatory activity in basal ganglia and thalamo-cortical circuits – so-called *Beta-frequency* oscillations – represents a pathologic and moreover *symptomatic* state in PD, the disruption of which is accomplished by high-frequency stimulation (@wilson_chaotic_2011). Meanwhile, synchronized activity in the beta range is also involved in many normal cognitive functions, and abnormal synchronization is likewise a mechanism for numerous other neurologic and psychiatric disorders (@velazquez_biophysical_2012),(@andrews_neuromodulation_2010). Consequently DBS has potential to become an effective treatment for epilepsy (@ii_dynamic_2013), OCD(@bourne_mechanisms_2012), depression (@mayberg_deep_2005, @mayberg_targeted_2009), Alien Hand Syndrome and Tourette's (@krack_deep_2010), obesity (@taghva_obesity_2012), alcoholism (@heldmann_deep_2012), addiction (@luigjes_deep_2012), autism (@sturm_dbs_2013), Alzheimer's disease (@laxton_deep_nodate), and schizophrenia (@white_dysregulated_2013). There are a large number of case-reports describing off-label DBS treatments, and a few small clinical trials (@kennedy_deep_2011-1; @sarem-aslani_industrial_2011). In 2009 Medtronic received a Humanitarian Device Exemption (HDE) to market a DBS device for OCD, called *Reclaim* ("Reclaim^{TM} Deep Brain Stimulation for Obsessive Compulsive Disorder (OCD) Therapy - H050003"2013). While closed-loop DBS might be even better suited for the complex dynamics of psychiatric disorders, most investigators have stuck with movement and seizure disorders (@sarem-aslani_industrial_2011), presumably feeling more comfortable with well-established animal models and FDA-approved applications; or perhaps feeling *less* comfortable with the ethical questions inherent to treating of behavior.

4.7.5 Project Plan

Deep brain stimulation (DBS) has revolutionized the treatment of neurological and psychiatric diseases over the past couple decades. A DBS device is essentially a cardiac pacemaker reconfigured to stimulate a small region of brain near the tip of an electrode with high-frequency pulses of electrical current; despite this simplicity, DBS has been used widely for treating movement disorders, such as Parkinson's disease (PD) and dystonia. The impressive performance of DBS on alleviating movement deficits associated with PD and dystonia has motivated a number of ongoing clinical trials on assessing its use in treating other brain disorders, such as obsessive compulsive disorder and depression. However, benefits of DBS treatment are unpredictable and show varying degrees and patterns of symptom suppression across patients. Unfortunately, this technology in its current iteration has been unable to overcome these limitations. This is likely due in part to a need for neural implant devices that are capable of more numerous and specific stimulation contact areas. Additionally, the DBS mechanism of action remains unknown and this further limits its ability to become wholly sufficient treatment. The central goal of this project is to develop a novel adaptive neuromodulation platform with increased neural contacts and their specificity as well as to examine the therapeutic mechanisms of DBS. My hypothesis was that aberrant neural network dynamics underlie the behavioral symptoms of brain disorders, and DBS therapeutic effect is through scrambling pathological neural network patterns. While I would primarily focus on PD here, the principles discovered through this study should be easily generalized to design novel therapies for other neurological and psychiatric disorders.

4.7.5.1 Design a novel adaptive neuromodulation platform for closed loop control of neural activities

Functional neural systems underlying brain disorders are likely complex, impacting interconnected brain areas. To reliably distinguish normal and pathologic neural network states, such as the "on/off" states in PD, it would be advantageous to simultaneously measure neural activities in multiple relevant brain areas. The platform proposed here consists of a bidirectional neural interface with penetrating silicon electrodes capable of simultaneously recording from multiple sites interspersed with LED-coupled optical-fibers

for multi-point optogenetic neuromodulation. I would use the commercially available multicontact silicon electrodes, known as Michigan probes, that are 5um thick and capable of recording up to hundreds of locations simultaneously. To perform neuromodulation with high spatiotemporal precision, I would use optogenetic methods to activate or silence specific neurons expressing rhodopsins in mice using certain colors of light. Even though this system is designed for mice in this study, the principles demonstrated here have a clear translational path to humans.

4.7.5.2 Implement a library of closed loop modulation protocols with machine-learning algorithms

To efficiently determine the neural network states responsible for a specific disease condition, I would use machine-learning algorithms to implement a library of optimized strategies. Specifically, this system consists of two parts: The first is a closed-loop control system responsible for gathering and analyzing streams of neural and behavioral data in real-time to estimate a neurobehavioral state, and then applying a state-specific optogenetic control with patterned light into the brain. The second component is a slower and more complex routine that records and analyzes the streams of neural and behavioral data, and assesses the performance of state-estimates and controls applied by the real-time component. State-response strategy updates would be periodically fed from the thoughtful controller (2^{nd} component) to the real-time decision maker (1^{st} component) using variations of traditional machine-learning algorithms to generate successively optimized strategies.

4.7.5.3 Systematically evaluate the neural circuit mechanisms underlying DBS therapeutic effect on PD

Even though the precise therapeutic mechanism of DBS is unclear, DBS has revealed tremendous information about the brain's functionings and dysfunctionings. For example, pathological beta oscillations, oscillations around 20Hz, have been widely observed in the cortical-basal gangion circuits in PD patients implanted with DBS electrodes. As the causal role of beta oscillations is yet to be established, it is likely that the neural network states identified in Aim 2 would conform and converge into distinct archetypes, such as beta oscillations within the cortical-basal gangion circuits relevant to

PD. I hypothesize that neuromodulation outside of the classical PD neural circuit would be therapeutic, in so far as such neuromodulation is capable of scrambling pathological network dynamics. Once proven correct, the proposed therapeutic mechanisms of DBS through altering neural network pathological representationswould provide a basis for a new generation of neural circuit based neuromodulation therapies.

4.7.6 Significance

This proposal focuses on developing a novel closed-loop neuromodulation system using optogenetics and an intelligent control algorithm. This approach enables automation of a search for individual-optimized stimulation patterns, and adaptation of these patterns over time in response to fluctuating symptoms.

4.7.6.1 Justification for This Approach

The novel closed-loop neuromodulation system as presented here has far reaching implications. These include improved patient symptoms, fewer surgical procedures (for battery replacement) and broader application in terms of patient population and disease profiles. The increase in number and range of stimulation sites and rapid real-time response rate of the proposed mechanism would yield such results (Lozano and Lipsman 2013). Furthermore, the information on DBS mechanism gleamed from this study would have immediate effects on the current technology

4.7.6.2 Shortcomings

Any device that relies on optogenetics to deliver stimulation to neurons inherently shares the same hurdles to clinical translation, the requirement for gene-therapy and its associated risks. Several early trials of viral transfection of cells had adverse effects including a greatly increased risk of carcinoma. In these early studies, the DNA insertion location was uncontrolled leaving important regions of DNA tumor suppressor genes exposed to damage. New methods that improve the safety of gene therapy have been developed. Several of the more recent methods utilize adeno-associated virus (AAV) and show

greater control regarding DNA insertion and decreased DNA damage. These more recent methods suggest a possibility that with continuing research, such a method may be developed without the potential for malignancy.

Working on a project that requires a technology that does not as of yet exist, represents one of the greatest educational benefits of this project. That leap of faith into a future that does not exist requires us to depend on each other as a team of collaborators, as each of our work depends on the others' success. In order to succeed, we must do so together; and without eachother, our therapeutics would never reach their ultimate goal, the patient. As such, we would share each success and setback in the same way whether such events occur within our own labs or labs across the country.

4.7.7 Research Design and Methods

The initial stages of this project were focused designing the physical components of the bi-directional neural interface – and the peripheral aspects of aim 2 – development of an automated behavioral assay for our prototypic animal model of Parkinson's disease.

4.7.7.1 Neural Interface

A *chronic neural implant* is required for this project, as the machine-learning component of this closed-loop system would require time and signal stability to function effectively. While our lab currently employs chronically implanted drivable tetrode arrays, this project would benefit from dropping this complexity in favor of *static* electrodes that are set to their prescribed position during the initial surgery, and never thereafter.

Sustained penetration of brain tissue with large (15 microns) diameter electrodes provokes a chonic inflammatory reaction to the foreign body eventually encapsulates the electrode in a "glial scar" and inhibits regeneration of neural process in the vicinity (@polikov_response_2005). The chronic implant designed for this project would use silicon electrodes to record from deep brain targets, and for shallow targets would employ *carbon fiber electrodes* (4-10 microns) using methods developed in a nearby collaborating lab (@guitchounts_carbon-fiber_2013). Carbon fiber is cheaply available with a

range of electrical properties, but the common properties of high modulus and small diameter – 5-10 micron vs. 30-50 microns for typical commercial microwire arrays (@ward_toward_2009) – allows carbon fiber to penetrate brain tissue without deforming, and to remain long-term without provoking an inflammatory reaction (@kozai_ultrasmall_2012).

4.7.7.2 LEDs

High-intensity LEDs would be incorporated into the implant in favor of a traditional fiber-coupled laser primarily for their ability to *diffusely* illuminate neurons over a *large area* and their *scalability*, which would be vital for *multi-site* and *chronic* modulation. LEDs do not couple efficiently to optical fibers of small diameter (~200 microns is typical in optogenetics labs), but they can provide a far greater *luminous intensity* in a greater variety of wavelengths than lasers, at a fraction of the cost. The proposed project would incorporate LEDs built into the implant, with their emitting surfaces *closely coupled to the brain surface*.

LED arrays, and Digital Micromirror Devices (DMDs) and Liquid Crystal on Silicon (LcoS) chips have been considered as a means of greater spatial control of light intensity; any of these would be incorporated if the technology improves during the lifetime of the project, but the current state of technology places heavy restrictions on space and/or power requirements to merit incorporation into the implant. Incorporating multiple LEDs (4 to 8) would enable semi-independent activation of neurons in widely-separated brain regions, the proposed project would need to *overlap* this diffuse control with more precise targeted delivery of opsins (@packer_targeting_2013).

4.7.7.3 Figure: PCB schematic for head-mounted carbon fiber electrode array

Caption: Circuit Board for connection to carbon-fiber electrodes and LEDs

Here I describe the design strategy for the physical implant outlined above that would provide pathways to and from the mouse brain for the the electrodes and LEDs described above, as well as the circuit board that sits within this implant and consolidates all *input/output* signals for connection with a computer. This project would strive to ensure the bidirectional chronic neural

implant is *resilient* to the abuse it would surely endure over the weeks to months of service *affixed to the cranium* of a laboratory mouse. The outcome we wish to avoid is *movement* of the penetrating electrodes within the brain tissue when external forces act on the implant, or when internal forces cause the animals brain to move relative to its skull. Moreover, if the electrode is unable to move with the brain there is substantial risk of electrode breakage.

We are designing the implant to address this potential issue from two directions: the first strategy is to make the implant as *tolerable* for the animal as possible by minimizing its weight, rounding all edges, and fabricating with a bioinert material such as polymethylmethacrylate (PMMA), polyetheretherketone (PEEK), and silicone elastomer. Elastomer or functionally similar material would be used to encase the printed circuit board (PCB) once electrodes are driven to their prescribed depth and electrical connections made. The encasement would cover the entirety of the board with the exception of two connectors for electrophysiology readout and LED power, and a heatsink connector for each LED . Heatsinks for the LEDs would either be copper tube or carbon fiber. The second strategy for minimizing electrode movement involves *floating* the connection between the electrodes and the PCB. The goal of this procedure is to allow the electrode – with one end in the brain and the other in a plated through-hole – to move vertically and laterally relative to the PCB while maintaining electrical contact with the pad. The method for accomplishing this is still being developed; fortunately its success is not critical to the project, but merely an improvement.

4.7.7.4 Electrophysiology

The neural interface proposed in this document would record 32 signals simultaneously. To minimize the risk that poor trajectory planning could have on the results described in [@sec:f31-aim3], we would specify targets redundantly. Thus, for each round of testing with our Parkinson's mouse model we would select 8 brain areas to record from redundantly and bilaterally. 7 of the 8 would be selected from a set of brain areas for which claims have been made in the literature suggesting functional relevance to Parkinson's disease; as a *control* for each round, the 8^{th} would be selected from a set of areas for which no connection with PD has ever been reported. Each round would begin with a modified CAD model of the implant (for modified electrode trajectories), which would be 3D printed or fabricated on a 4-axis

CNC-mill in the lab.

4.7.7.5 Front-end Digitization

Recording would be accomplished with the RHD2132 digital electrophysiology interface chip produced and made freely available by Intan ("RHD2000 Amplifier Chips | Intan Technologies" 2013). The chip accomplishes as much as a traditional electrophysiology system at a price that is orders of magnitude less, and a package that takes up much less space.

4.7.7.6 Signal Filtering, LFP and MUA

Electrophysiology data would be digitally filtered on the Intan chip using parameters selected by the machine-learning routine. The only hard limits on bandwidth would be dictated by the requirements of the analog to digital converter (ADC), i.e. a high-pass filter to remove drifting DC offsets and a low-pass filter to prevent aliasing. These limits would vary depending on electrode impedance and sampling frequency, but would have cutoff frequencies or roughly 0.1 Hz and 10 KHz respectively. While there is much published in regard to the spatial extent and information content in common named frequency bands, the algorithm that selects filtering parameters would not incorporate any of these assumptions.

That being said, we expect the learning algorithm would find more *reliably relevant* features in the classic Local Field Potential (LFP) range, which is typically low-pass filtered with a cut-off around 300 Hz. It's thought that LFP represents a sum of neural activity from cells within a 50-250 micron radius of the electrode tip, or more if activity is highly correlated (Lindén et al. 2011). This spatial scale allows for variability in electrode position across subjects, and for small movements of the electrode over time (Andersen, Musallam, and Pesaran 2004).

Closed-loop input/output would be handled at three levels by multiple computers.

4.7.7.7 Input/Output

The lowest level – implemented on the Intan RHD2132 amplifier chip and RHD2000 evaluation board – handles analog-to-digital conversion and filtering of electrophysiology data. Also at this level – but implemented on a different computer – are the movement tracking functions described above.

4.7.7.8 Closed-loop State Estimation and Adaptive Neuromodulation

Streams of electrophysiology and behavior/movement data are sent to the second level, a *real-time* computer system encapsulating all the *closed-loop* functions of the platform. This computer would use the *Real-time eXperiment Interface* (RTXI; www.rtxi.org), an open-source Linux-based operating system supported by the NIH (Lin et al. 2010). This system provides modules for data acquisition, storage, synchronization of output, and more. RTXI would be installed on an ARM-based embedded system such as the open-source *Puggle*, or a similar embedded processor from Texas Instruments. What is *Puggle*? _"Puggle is ARM-based, real-time data acquisition and processing tool. It is designed to sense, process, and react to both analog and digital input signals in hard real-time. Puggle is designed for closed-loop electrophysiology applications. . . _ (http://www.puggleboard.com)." The closed-loop routine running at this level would analyze neural and behavioral data using parameters from the next higher level. The analysis step would conclude with two estimates for *neural* and *behavioral state*, with this combination *mapped* to an output routine with parameters that are also passed down by the higher level. The specifics of the output routine would be handled by a connected microcontroller, would be interrupt-driven, and would ultimately control the power of optical output from each LED over time.

4.7.7.9 Generation and Evaluation of Stimulus Paradigm

The highest level of processing would be implemented in a *non-real-time* environment, MATLAB on Windows. Raw data and activity logs from the closed-loop routine would be written to disk continuously. A machine-learning routine would analyze these data and send updated parameters to the closed-loop computer as described below. All parameter updates would also be written to disk in synchrony with acquisition for retrospective analysis as described by [@sec:f31-aim3].

4.7.7.10 Machine-Learning & Optimization

The process for selecting exactly how to stimulate a brain follows a common routine both in the clinic and research laboratory. After having spatially localized an electrode in the intended position, physicians (and neuroscientists) would begin stimulation with a temporal pattern that is fully defined with two or three parameters (e.g. a binary pulse-sequence with constant amplitude, frequency, and duty-cycle). Effects of stimulation are often immediately apparent. The physician or neuroscientist – let's just call this person the "*decision-maker*" – would evaluate the subjects behavior and compare this evaluation to the desired *target* behavior. In the example case of PD the deviations from this "target behavior" include akinesia, gait-inbalances, and postural instability, but might also include iatrogenic responses such as dyskinesias or diplopia. The decision-maker then draws on their experience to estimate which parameters should be adjusted, and with what magnitude in order to shift behavior towards the intended target. In the absence of experience, the decision-maker relies on *education* (many years of it in either the physician or scientist's case). The first adjustment is applied, and the effect on behavior is evaluated once again. The inexperienced decision-maker learns something new, and can use this new information in combination (or perhaps even *in contrast!)* with their prior education to estimate the effect of adjustments in any or all available parameters, and make a decision that begins the cycle again.

Humans are excellent learners – undoubtedly better than any other species on Earth. Computers are not; they are painfully literal and would not flip a bit without being told to. Their persistence, however, is their saving grace, and is the aspect that makes machine learning possible, given creatively designed seeds of human thought and recursive and inversive instructions for expanding this seed. The process of cyclic evaluation and decision-making described above would be used as a model for a machine-learning algorithm with two goals: optimizing the identification of *disease-relevant neural states*, and optimizing a neuromodulation routine for *disease-related symptom reduction*.

4.7.7.11 Identification of Neural State

The algorithm used to optimize closed-loop identification of neural neural state would be developed from a set of common *pattern-recognition* algorithms and

semi-supervised learning models, including support vector machines (SVMs), logistic, polynomial, or linear regression, principal component regression (PCR), Kalman filtering, and hidden Markov models (HMM). These algorithms would be applied to recorded data and *selected for implementation* in the closed-loop routine based on measures of performance (discussed below) and ability to meet computation-time specifications (<~250 ms).

4.7.7.12 Stimulus Sequence Generation

Each neural state would be mapped to a neuromodulation routine that is applied continuously (with *sub-millisecond* response time) until the next state-estimate and routine update. Each stimulation sequence would initially be defined by random selection from a set of sequence *archetypes*, and further specification by random selection of *archetype-specific-parameters*. The real-time process would handle application of the state-mapped sequence, but the sequence formation routines would be run in the non-real-time process. Both open-loop (for control/comparison) and closed-loop stimulation archetypes would be generated from three sources, with generic examples from each source in the table below:

1. Common achetypes reported in literature, e.g. DBS-style constant pulse generation.

2. Novel archetypes generated in creative thinking and discussion sessions with colleagues.

3. Concocted archetypes from chaotic-generation algorithm

Constant DBS-type square wave with parameters for amplitude, frequency, and pulse-width/duty-cycle (open-loop) LED power (stimulation) in each area of control proportional to *arctangent of phase difference* between two areas. Chaotically generate functions by recombining elements from the novel and common archetypes, and also by applying every function available in MATLAB (use *what* and *lookfor* functions applied in *try... catch...* blocks to find statements that work with given data) *Triggered impulse response* with parameters defining a threshold or feature from any channel that triggers stimulation of some shape that initiates at some delay (Rosin et al. 2011) (closed-loop) Stimulation is continuously proportional a *ratio of band-pass filtered signal magnitude* from arbitrarily selected channels.

Phase Cancellation of filtered signal from any defined channel (Brittain et al. 2013) (closed-loop) Signal from one channel is used to construct *wavelet* then convolved with other channels with output determining pulse-shape

The number of possible archetypes used by sourcing in these ways would be extensive, so much so that trying them manually would be unmanageable. This is where the automated machine-learning algorithm which selects, applies, evaluates, and optimizes each routine for several hours every day becomes absolutely necessary.

4.7.7.13 Stimulus Optimization

As the real-time component runs its state-response cycle and logs data, the non-real-time machine-learning component would evaluate its performance and update parameters in an attempt to optimize *symptom suppression*. This evaluation process would identify "hot" parameters (those with large effect), and continue sending parameter updates until symptom suppression cannot be optimized further. This process would be slow to allow observation of subtle effects of stimulation in noisy data. If manipulation of all available parameters yields no changes in behavior, a new stimulation archetype would be tried.

This type of process is sometimes referred to as a *Genetic Algorithm*, where the parameters for a particular archetype would be viewed as the *genes* that define a species. The *fitness* of each individual combination genes (i.e. parameters) determines survival and consequent ability to pass genes to the next generation. In our implementation, we'd be running multiple species (i.e. archetypes) as well, with the expectation that many would be quickly declared extinct.

4.7.8 Systematic evaluation of potential DBS targets and mechanisms in PD mouse model:

To that end we are currently inducing a quantifiable PD-like state in mice with a unilateral injection of the neurotoxin 6-hydroxydopamine (6-OHDA) into the striatum, and subsequent administration of apomorphine to provoke side-biased motor deficits (Iancu et al. 2005) . Side-biased"turning" behavior

is quantified autonomously on two distinct platforms, a computer-vision system that allows free movement, and a virtual-reality spherical treadmill platform that simulates free movement.

4.7.8.1 Metrics of Behavior

Two testing platforms would be used to assess changes in behavior over time. Behavior is analyzed and quantified in real-time, and would be synchronized with electrophysiology and made available as another stream of input for the closed-loop routine. The quantification routine creates a signal that is representative of symptom severity. For our unilaterally lesioned mouse model of PD the most readily observable impairment is the inability to walk straight; mice would turn in circles contralateral to the lesion when given intraperitoneal apomorphine. The behavioral apparati are described below.zMicroscopy 1

This section describes the background in microscopy in the neurosciences, and also how it relates to imaging in healthcare and electrophysiology in neuroscience. It will also describe the basic elements necessary for the construction of a microscope in a laboratory where calcium imaging in an animal is available. It will also refer to later sections which cover the design and construction of mechanical elements for animal handling and optical access (i.e. the headplate and a chronic optical window).

Optical imaging has traditionally involved wide-field imaging or two photon imaging, each with their own distinctive advantages and disadvantages

In recent years, two photon microscopy has been a preeminent choice for imaging in tissue, because of its high spatial resolution, and tissue penetrating features

Two photon calcium imaging has been broadly applied to individual cells or subcellular components of neurons including spines and axons

Because two photon microscopy uses a scanning mechanism, the signal to noise ratio is influenced by the time spent imaging each point, and the spatial resolution is determined by the number of points scanned to obtain each image. As a result, the size of the imaging field is inversely correlated with the overall temporal resolution while maintaining a relatively high signal-to-noise

ratio, thus, two photon calcium imaging is often performed on a small area or on a sparse network of cells, when dynamic responses with high temporal fidelity is necessary.

Wide-field, or single photon imaging has been in use for several decades and was first used to characterize the functional architecture and hemodynamic responses in brain tissue

However, this technique has seen a renaissance recently due to its simple instrumentation, relatively inexpensive cost, and the improvements in neural signal indicators. Optical imaging and two photon microscopy have traditionally been performed in head-fixed preparations, but recent advances have also made it possible to perform wide-field calcium imaging in freely moving animals, through miniaturized and wearable microendoscope systems

While wide-field imaging lacks the spatial resolution to resolve fine subcellular structure or the penetrating properties available with two-photon, it is possible to obtain clear neurites and somatic features, including spike detection

Because a single photon microscope does not rely on scanning features, it can be used to sample a larger field of view without sacrificing sampling rates. Additionally, recording sessions may be less sensitive to fluorophore bleaching than other techniques, which makes it possible to perform sustained illumination and subsequent imaging for an extended period of time - a desired feature for analyzing neural networks during some behavior paradigms (*e.g.,* repeated trial learning paradigms). Thus, wide-field imaging offers an advantage if the objective is to simultaneously recording hundreds of neurons in the brain of a living and behaving animal with high temporal fidelity.

4.8 Background: Brain Imaging and Microscopy in Neuroscience

4.9 Fluorescent Proteins Background

- GCamP
  - vs dyes

4.9.1 Fluorescence Microscope Types Background

- 2p, confocal, wide-field
  - tissue scattering
    - depth comparison
  - cost & complexity
  - (address scalability later)

4.9.2 Widefield Microscope COnfiguration

- historical trend/shift from *finite* to *infinite* conjugate type
  - infinite type uses *infinity corrected lenses*

4.9.3 Filters

- excitation
- emission
- for epifluourescent microscope configuration: dichroic beamsplitter

4.9.4 Lenses

The simplest configuration of infinite wide-field microscope requires an excitation

4.9.5 Excitation

4.9.6 Emission

4.9.7 Emitted Light Collection and Image Formation (emission/collection)

- Lens selection
  - objectives

- SLR
    - spectral coating characteristics ### Mechanics

4.9.8 Microscopy and Functional Imaging Two core innovations in available

- technology 1. Synthetic bio (i.e. GCaMP) 2. Cameras
- scientific CMOS

4.10 Brain Imaging Awake Animals

4.11 Brain Regions

-hippocampus, cortex, striatum - cell types - cell sparsity

4.12 Analysisz# ## Cameras for Widefield Microscopy

Traditional widefield microscope or macroscope builds incorporate 'scientific grade' cameras. Compared to cameras built for other markets (e.g. consumer, industrial, studio, etc.), these cameras are often well tested and certified to offer low or well-characterised noise at moderate speeds, and a linear photo-response profile. Unlike consumer or studio cameras which are invaribaly configured for RGB color, they are preferably configured with 'monochrome' sensors - essentially identical to the analagous color sensor, without the bayer filter. Of much greater importance, one must consider the unique connectivity and control interface that scientific cameras come with. Standards exist, but are typically unique to this segment of the industry, with poorly defined specifications for translation to other electronic communication and connection interface standards, such as those used in studio and broadcast video, or those used with consumer cameras. The trait that is the most worthy of consideration, however, is the cost. See (todo-crossref-discussion) for details.

The in-vivo instrinsic-signal or fluorescent-dye imaging camera of 1 decade ago

had a 0.5“-1” monochrome CCD sensor with 0.1-1 MegaPixels, a large well-depth, and moderately low noise at speeds around 30 to 60 fps. Connection was often LVDS, with custom electrial connectors unique to each camera. A particularly popular and long-running model was the Dalsa 1M30, followed by the 1M60 in later years [@takahashi_vivo_2006].

4.12.1 Accelerated Sensor improvement

4.12.1.1 Sensor Size

4.12.1.2 Read noise and Speed

4.12.1.3 Interfaces: Connection

4.12.1.4 Interfaces: Protocol

4.13 Computer Workstation

Go to Puget Systems to find a computer configuration that is well tested and uses commercially available components to deliver high performance. If you don't feel like putting a computer together yourself, you could order directly from them.

4.13.1 Part Selection and Assembly

Motherboard:	Processing	Expansion Slots:	Memory:
ASUS WS C621E SAGE	Socket: 3647		

4.13.2 Configuration

Must configure bios to optimize for high-throughput writes to disk.

4.13.3 Package Managment

- Package Management & Binary Distribution Sites
- Bintools/JFrog
- Conan.io
- NPM
- Nuget
- VCPkg
- Chocolatey (win)
- Apt/yum (linux)
- Pacman (msys/mingw)
- Pact (babun/cygwin)
- scoop
- NPackd
- Pypi

4.13.4 Development Environment

- Version-Control
  - Github
  - Bitbucket
  - Gitlab
  - Sourceforge
  -
  - Comparison on Wikipedia
- Code Repositories
  - Matlab File Exchange
  - stack overflow
  - ~~(Google Code)[code.google.com]~~
- Research Entity Web Sites
  - From Laboratory Manifest
    - Janelia

        * INRIA
        * Schartz Center for Computational Neuroscience (SCCN)
            · (SCCN Homepage)[https://sccn.ucsd.edu/wiki/SCCN_Software]
            · (SCCN Github)[https://github.com/sccn]
            · EEGLAB, BCILAB,Lab-Streaming-Layer
        * Cohen Lab
            · LeverJS
    - NIH
    - NASA
- Binary Utilities
    - gtools
    - exetools
    - nirsoft
    - sysinternals
    - unxutils
    - x64tools
    - BU EngIT (butools)
- OSS Foundations
    - Apache
- Citations in Literature
    - from Biblio-Manifest
- SCC or Grid Utilities ## Head-Fixation Apparatus for Mice

Whether using a microscope or electrophysiology apparatus with the spherical treadmill, the requirements for a rigid connection to the animal's cranium are critical.zPipeline overview

4.14 phases

4.14.1 preprocessing

Image filtering operations Normalization Motion estimation and compensation Layer classification foreground/background Cell/neuropil/vessel

4.14.1.1 time series analysis

Prediction and factorization of temporal components/contaminants Model: Unidirectional trend (sources. . .) Oscillatory ("seasonal") Unpredictable

4.14.1.2 spatial signal deconvolution (unmixing)

4.14.2 signal localization and extraction

Segmentation of cells Overlap and "disconnected" regions Initial vs incremental cell identification

4.14.3 active signal analysis

Signal statistics Shape and stability of probabilistic distribution Pixel level vs cell level

4.14.3.1 independent cellular activity measures

Quality Complexity Spatial Temporal

4.14.3.2 network analysis

4.15 Computing cost:

boundary analysis for iOS and config

4.16 Storage cost:

compression and imposed boundarieszImage Processing

This section borrows from AIM-1 and AIM-2 of the prospectus.

4.17 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 indicate 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 np2 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.

4.17.1 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 multithreaded 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.

4.17.2 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.

4.18 Methods and Approach

Image processing is performed offline using MATLAB software. The goal of this procedure is to reduce the raw image sequence to a collection of one-dimensional traces, where each trace 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. We implement the process in 3 distinct stages as described below. The main novel contribution of this work is the efficient extension of segmented ROIs into the third dimension by clustering features of ROIs segmented separately in two dimensions. Online processing uses a similar approach, and the differences are explained in the next section.

4.18.1 Image Pre-processing: Contrast Enhancement and Motion Correction

Alignment of each frame in the image sequence with all other frames is essential to the methods we use in subsequent steps for identifying and tracking cells over time. Thus, the goal of the first stage is to correct for any misalignment caused by movement of the brain relative to the microscope and camera.

Many algorithms for estimating and correcting image displacement exist and are well described in the medical imaging literature. We elected to use phase-correlation to estimate the induced motion in each frame, as we found this method to be highly stable, moderately accurate, and (most importantly) fast, especially when implemented in the frequency domain and using a decent graphics card.

Phase-correlation estimates the mean translational displacement between two frames, one being the template or "fixed" frame, and the other being the uncorrected or "moving" frame. In the spatial domain this is accomplished

by computing the normalized cross-correlation, which implies a 2-dimensional convolution of large matrices. The equivalent operation in the frequency domain is a simple scalar dot-product of the discrete Fourier transforms of each image normalized by the square of the template, followed by the inverse Fourier transform. The intermediate result is the cross-correlation (or phase-correlation) matrix, which should have a peak in its center for correctly aligned images, or a peak near the center, the offset of which indicates the mean offset between the two images. This peak can be found with subpixel precision by interpolation to give a more accurate alignment, although at some moderate expense in computation time.

For the template image we used a moving average of previously aligned frames when processing frames sequentially, or alternatively a fixed mean of randomly sampled and sequentially aligned images from the entire set when processing files in parallel. The simplest way to perform this operation is to use the built-in MATLAB function normxcorr2, which makes optimization decisions based on image size and available hardware automatically. However, performance can be improved by tailoring the operation to your particular hardware and image size, i.e. using fft2 and ifft2 for large images and a good graphics card.

4.18.2 Region of Interest (ROI) identification & segmentation

The ROI detection process used an adaptive threshold on the z-score of pixel intensity to reduce each frame to binary 1's and 0's (logical true or false). These binary frames were then processed using morphological operations to find and label connected components within each frame. For example, beginning with a z-score threshold of 1.5, all pixels that were more than 1.5 standard deviations above their mean were reduced to 1 (true), and all others reduced to 0 (false). Pixels reduced to 1 were often pixels overlying a cell that was significantly brighter during that frame due to activation of GCaMP. This initial threshold was adjusted up or down based on the number of non-zero pixels detected with each threshold. This was done to prevent spurious motion-induced shifts of the image frame from producing ROIs along high contrast borders. All morphological operations were performed using built-in MATLAB functions from the Image Processing Toolbox, which have fast parallel versions if the operation is run on a graphics card (e.g. imclose,

imopen, etc.). Furthermore, the connected-component labeling and region formation operations were run using built-in MATLAB functions bwconncomp, and regionprops. Connected components were stored in a custom class and termed "single-frame ROIs," and these were then passed to the 3rd stage of processing, which merges them into a "multi-frame ROI" that represents the location and spatial distribution of each cell identified over the entire video.

4.18.3 Region of Interest (ROI) merging

The standard structure of region properties output by the MATLAB function regionprops (Area, BoundingBox, Centroid, etc.) are mimicked in a custom class called RegionOfInterest, where each field of the structure becomes a property of the custom class. We add additional properties for storing state information and data associated with each ROI, along with a number of methods for comparing, merging, manipulating, and visualizing the single-frame and multi-frame ROIs. The single-frame to multi-frame ROI merging procedure is essentially a clustering process that merges single-frame ROIs together using such criteria as the proximity of their centroids, as well as proximity of their bounding-box (upper-left and lower-right corners). Performing this operation quickly was highly dependent on pre-grouping ROIs based on centroid location in overlapping blocks of the image frame, as well as grouping by size. This enabled the clustering to be performed in parallel (across CPU cores) followed by a second iteration of clustering to deal with redundancy in overlapping regions.

4.18.4 Visualization

Once ROIs are established, all video data is reloaded and passed to a method in the *RegionOfInterest* class that extracts the 1-dimensional trace for each ROI representing the fluorescence intensity in that region over time. The ROIs and their traces can then be interactively visualized using another method in the *RegionOfInterest* class.

The *RegionOfInterest* class defines methods for rapid spatial comparison operations which can typically be viewed as an adjacency matrix using built-in image viewing commands. Visualization of the segmented cell overlay and

1D traces can be manipulated by assigning colors, removing ROIs, hiding ROIs, and more.

4.18.5 Predicting Activation State & Assessing Network Activity

The continuous signal intensity signals can be reduced to binary indicators of activity for each frame. This enables simplified and fast calculation of information theory measures, such as activation probability, joint and conditional probabilities, response entropy, mutual information, etc. The conversion from continuous to binary uses several abstractions of the signal applied to a Gaussian Mixture Model (GMM). The abstractions are calculated from the following:

1. Linear least-squares fits to moving windows with variable size to find slope of the line surrounding each point.
2. Skewness and Kurtosis of finite windows surrounding each data point.
3. Temporal difference of fluorescence intensity.

The gaussian mixture model employs all measures to infer periods of reliable distinct activation of neurons.

4.18.6 Parallel Processing

Many built-in MATLAB functions are implemented using efficient multi-threaded procedures, and these are used to the extent that they can be. However, for procedures that must operate on data in irregular formats (i.e. any format other than N-dimensional arrays of primitive data types), one also has the option of performing explicitly defined parallel operations by distributing data across multiple parallel processes, each with their own memory space. Below I give examples of how implementing in a multi-threaded fashion can substantially boost performance, and also an example of a situation where multi-threaded operations aren't possible without explicit calls for parallel distribution.

Standard elementwise operators like *plus* (+) and *times* (.*), as well as comparison operators like *equals* (==) and *less-than* (<) will be performed

efficiently using as many processing cores as available when applied to large n-dimensional arrays of the same size. However, when operand sizes differ a simple call to the built-in operation will not work. For example, if we wish to subtract the average from each pixel over time from all frames in the series we can accomplish this with a call to MATLAB's *bsxfun* function, which stands for Binary-Singleton-eXpansion-FUNction, as shown below:

```
Fmeansub = bsxfun( @minus, F, mean(F,3) );
```

This operation passes a function handle as the first argument (denoted by the '@' symbol) indicating the operation to perform. It then passes the entire [IxJxK] array of image data as the second argument, and it's temporal mean with size [IxJx1] is calculated once and passed as the third. The function efficiently expands the mean argument as needed for fast distribution across parallel threads.

4.18.7 Managing Continuity

Data such as baseline frames and frames for alignment must be passed between parallel processors to maintain continuity between data divided temporally between processors. However, the efficient application of this approach was limited by the system memory and number of cores available, and was meant to be applied in a cluster environment.

Building the set of functions for offline processing enabled application to data already gathered, and also served as a framework that informed the necessary procedures to be included in the online extension of this toolbox.

4.18.7.1 Computing Power and Connectivity

- Remote Clusters (AWS)
- Graphics Processing Units (NVIDIA GTX)
- Embedded Units (NVIDIA Tegra X2) 2. Well developed libraries
- ImageJ (so so)
- OpenCV (uses OpenCL)
- GStreamer (much better)

- OpenGL
- Shader

4.18.7.2 Image Processing

- Motion Correction
- Image Enhancement

4.18.7.3 Motion Correction Two approaches to find displacement

4.18.7.4 Spatially Homogeneous phase correlation

- aka normalized cross correlation - Feature Matching
- Detect features (i.e. corners) - Triangulate best

4.18.7.5 Image Enhancement

1. Contrast - Linear Scaling - Lookup Tables
2. Spatial and Temporal Filtering
3. Feature images - Gradients

4.18.7.6 Feature Extraction

1. Feature images (temporally independent)

- Gradients - Surface Curvature 2. Long Term Memory - Statistics
- changes (single pixel)
- Mutual information changes (inter-pixel)

4.18.7.7 Acceleration and Optimization Procedures for Online Video Processing

4.18.7.8 Incremental Update of Statistics

4.18.7.8.1 central moments

```
function [m1,m2,m3,m4,fmin,fmax] = updateStatistics(x,m1,m2,m3,m4))
      n = n + 1;

      % GET PIXEL SAMPLE
      f = F(rowIdx,colIdx,k);

      % PRECOMPUTE & CACHE SOME VALUES FOR SPEED
      d = single(f) - m1;
      dk = d/n;
      dk2 = dk^2;
      s = d*dk*(n-1);

      % UPDATE CENTRAL MOMENTS
      m1 = m1 + dk;
      m4 = m4 + s*dk2*(n.^2-3*n+3) + 6*dk2*m2 - 4*dk*m3;
      m3 = m3 + s*dk*(n-2) - 3*dk*m2;
      m2 = m2 + s;

      % UPDATE MIN & MAX
      fmin = min(fmin, f);
      fmax = max(fmax, f);
end
```

4.18.7.8.2 Extract Features

```
function [dm1,dm2,dm3,dm4] = getStatisticUpdate(x,m1,m2,m3,m4)
      % COMPUTE DIFFERENTIAL UPDATE TO CENTRAL MOMENTS
      dm1 = dk;
      m1 = m1 + dm1;
      dm4 = s*dk2*(n^2-3*n+3) + 6*dk2*m2 - 4*dk*m3;
      dm3 = s*dk*(n-2) - 3*dk*m2;
      dm2 = s;
      m2 = m2 + dm2;
      % NORMALIZE BY VARIANCE & SAMPLE NUMBER -> CONVERSION TO dVar, dSkew
      dm2 = dm2/max(1,n-1);
```

```
        dm3 = dm3*sqrt(max(1,n))/(m2^1.5);
        dm4 = dm4*n/(m2^2);
    end
```

4.18.7.9 Simple Processing on GPU

```
[dm1,dm2,dm3,dm4] = arrayfun(@getStatisticUpdate(x,m1,m2,m3,m4)
[dm1,dm2,dm3,dm4] = arrayfun(@getStatisticUpdate(rowidx,colidx)
```

4.18.7.9.1 Alternative Libraries

- NVIDIA Performance Primitives
- OpenCV
- VLFeat
- OpenGL
- OpenCL
- OpenVX
- CLosedDoesNotExist (. . . ?)
- Shader Languages
- GLSL
- HLSL
- WebGL
- Halide
- – FFmpeg
- GStreamer

4.18.8 Choice of Interface

4.18.8.1 Procedural Framework: Pipes, Streams, & Graphs

4.18.8.1.1 Concurrency: Parallel = Performance?

Not always, no. While concurrent processing of independent tasks or sequentially arriving data elements will almost always increase performance, this is not always the case. At a lower instruction-level than we typically program, synchronous operations can often be optimized in ways that asynchronous operations cannot, typically through strategic register allocation, or by taking cache-hit performance). "Globally Asynchronous Locally Synchronous"

4.18.8.1.2 Scheduling

4.18.8.1.3 Adaptive

4.18.8.2 Choice of Operations

- What is the goal?
- Is it effective?
- Is the computation cost worth the result?
- Are there side-effects or artifacts?
- Can they be reliably controlled or accounted for?

4.18.8.3 Motion Correction

In our application, the goal of a motion correction operation is to artificially suppress translation of the brain tissue parallel to the image plane. *Phase-Correlation* (also referred to as *normalized cross-correlation*) has consistent performance across a range of image sources with varying spatial noise characteristics. However, a large non-uniform change from reference frames - such as occurs when cells with low baseline fluourescence are first activated - can cause drastic errors that must be recognized and corrected by a supervisory

procedure. This can induce an undesirable, unpredicatable, and specifically inopportune latency Unfortunately in all the whole pipeline.

Unfortunately, “Globally Asynchronous Locally Synchronous”

as it’s In some experimental setups,

The phase correlation method of Motion Estimation - cost: 2-10 ms/frame - Frequently unstable (depending on video content)

4.18.8.3.1 Motion Compensation & Interpolation

- cost: 400-800 us/frame
- Requires infill with nearby or prior pixel values if frame size is to be maintained # Microscopy 2

4.19 Construction

4.19.1 Standards

4.20 Multispectral Imaging

4.20.1 Filter Selection

4.20.2 Lenses

4.20.3 Mechanics and Positioning

- Configuration
  - gantry
  - XY table with split Z
  - AB axial rotation mounts
    - center of rotation bi-coaxial with image plan
- Bearings
  - ball bearing slides vs crossed roller
  - flexures

  - air bearings
    * hybrid vacuum and compressed air
    * magnet and compressed air
  - simple teflon slip

4.21 Electronics

4.21.1 Photosensors

4.22 Cameras

This section details the evolution of cameras sensors and other sensors that provide bio-relevant data. Emphasis is on

4.22.1 Scientific CMOS (sCMOS)

- Correlated double sampling
- HDR
- On-sensor Fusion
- Commercial availability

4.22.2 Data-Transfer (Camera Interfaces)

- USB
- CameraLink
- COaXPress
- PCIe
  - gen2, gen3
  - x4, x8
  - copper, fiber

4.23 Image Acquisition

- Libraries
- Camera configuration
- Windows vs. Linux
- Setup and programming

4.24 Image Data Management

- Storage format
- storage location
- provenance ## Sensors

Development boards # Video Processing

4.25 Continuous Online Video Processing

4.26 Introduction

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 minimizing 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.

4.27 Method and Approach

The entire procedure for processing images and extracting cell signals can be performed in substantially less time than most commonly available tools using

the approach described in Aim 1, particularly the methods for restricting the spatial extent of pixel-association operations, and distributing operations across parallel processing cores using a Single Program Multiple Data (SPMD) archetype. However, the total time still exceeds that of the acquisition session. Inefficiency arises from the overhead involved with distributing data and passing information between separate parallel processes. Graphics cards, however execute in what's called Single Instruction Multiple Data (SIMD) fashion, to distribute computation across the thousands of processing cores.

The processing components are implemented using the MATLAB System-Object framework, which allows for slightly faster performance through internal optimizations having to do with memory allocation. Most system objects, each representing one step in the serial processing and signal-extraction procedure, also have companion functions that implement the computation-heavy components of each algorithm using a pre-compiled CUDA kernel.

4.27.1 Benchmarking & General Performance

Built-in MATLAB functions that execute on the GPU can be profiled with benchmarking functions like *gputimeit()*, or with the *tic/toc* functions. When execution isn't fast enough, they need to be replaced with custom functions. The custom functions typically achieve the speed up necessary by enabling the operation to carried out on several frames at once. This reduces the over-head costs inposed for each function call by spreading it over several frames. This solution is not ideal, as it increases the latency of solutions, however does not preclude implementation in real-time system if the procedures are adapted to run on a real-time hybrid system-on-module like NVIDIA's Tegra X1, which should involve minimal effort once a standard set of successful procedures is realized. The current implementation tests the processing time of each stage of the process to ensure that the sum is less than the acquisition time for each frame dictated by the inverse of the frame-rate (30-50 milliseconds).

4.27.2 Buffered Operations

Combining frames for each operation can result in near linear speedup. For example, for the phase-correlation step required for motion correction, the

FFT and IFFT are called on 16 image-frames at once, and the time take to accomplish is approximately the same as if the operation were called on 1 frame. This essentially leads to a 16x speedup, though the latency is also increased slightly. The best size to use is difficult to pre-determine, and typically must be measured for varying size 'chunks' using the benchmarking functions indicated above. The system objects manage the details necessary to allow buffered chunks of video to be passed to each stage without introducing artifacts at the temporal edges between chunks.

4.27.3 Image Pre-Processing & Motion Correction

Pre-processing is implemented as with the offline procedure, with a few changes. Images are aligned in chunks, and they are aligned sequentially to two templates. One template is the most recent stable frame from the preceding chunk. The other is a recursively temporal-low-pass filtered image that mitigates slow drifts. Aligning to the first template is usually more stable as the brightness of cells in the recent image will be more similar to those in the current chunk than will be the brightness of cells in the slow-moving average.

The displacement of each frame is found to sub-pixel precision, then used with a custom bicubic resampling kernel that replaces any pixels at the edges with images from the moving average.

4.27.4 Sequential Statistics

A number of statistics for each pixel are updated online and can be used for normalization and segmentation procedures later in the process. These include the minimum and maximum pixel intensity, and the first four central moments, which are easily converted to the mean, variance, skewness, and kurtosis. The formulas for making these calculations are given below, and are performed in a highly efficient manner as data are kept local to each processing core, and repeat computations are minimized.

```
n = n + 1;
f = F(rowIdx,colIdx,k);
d = single(f) - m1;
```

```
dk = d/n;
dk2 = dk\^2;
s = d\*dk\*(n-1);

% UPDATE CENTRAL MOMENTS
m1 = m1 + dk;
m4 = m4 + s\*dk2\*(n.\^2-3\*n+3) + 6\*dk2\*m2 - 4\*dk\*m3;
m3 = m3 + s\*dk\*(n-2) - 3\*dk\*m2;
m2 = m2 + s;
% UPDATE MIN & MAX
fmin = min(fmin, f);
fmax = max(fmax, f);
```

Furthermore, the value used to update each central moment at each point in time can be used as a measure of change in the distribution of each pixel caused by the current pixel intensity, as explained next.

4.27.5 Non-Stationarity & Differential Moments

Stationary refers to the property of a signal such that its statistics do not vary over time, i.e. its distribution is stable. Neural signals tend to specifically *not* have this property, in contrast to other measurable components such as those contributed by physiologic noise (heart-rate, respirations, etc.). Thus, by analyzing the evolution of statistical measures calculated for each pixel as frames are added in sequence gives a highly sensitive indicator of neural activity. This is done using a routine analogous to that for updating central moments given above, except the values returned are not only the updated moment, but also the updating component – essentially the partial derivative with respect to time. This is illustrated below, including the normalization functions which convert the partial-moment values to their variance, skewness, and kurtosis analogues:

> % COMPUTE DIFFERENTIAL UPDATE TO CENTRAL MOMENTS
>
> dm1 = dk;
>
> m1 = m1 + dm1;

dm4 = s*dk2*(n^2-3*n+3) + 6*dk2*m2 - 4*dk*m3;

dm3 = s*dk*(n-2) - 3*dk*m2;

dm2 = s;

m2 = m2 + dm2;

% NORMALIZE BY VARIANCE & SAMPLE NUMBER -> CONVERSION TO dVar, dSkew, dKurt

dm2 = dm2/max(1,n-1);

dm3 = dm3*sqrt(max(1,n))/(m2^1.5);

dm4 = dm4*n/(m2^2);

These functions run on images representing the image intensity, and also on images taken from sequential differences indicating the temporal derivative of image intensity. The combination of outputs from these operations indicate both when image intensities are significantly high relative to past distribution, and also when intensities are changing significantly faster than learned from their past distribution.

4.27.6 Surface Classification: Peaks, Edges, Curvature

Edge-finding methods are employed for establishing boundaries between cells, and first and second-order gradients are used to compute local measures of curvature from an eigenvalue decomposition of the local Hessian matrix. I won't go into detail, as the utility of these procedure in the most recent implementation has been lost, but nevertheless, the operation is optimized and ready to be plugged back in when further development calls for better accuracy informing cell-segmentation, or when a faster or more accurate motion-correction algorithm is called for.

4.27.7 Online Cell Segmentation & Tracking

Cells are segmented by first running sequential statistics on the properties of identifiable regions on a pixel-wise basis. That is, as regions are identified

in a method similar to that used offline in Aim 1, the region-properties are calculated (Centroid, Bounding-Box, etc.) and statistics for these properties are updated at each pixel covered by a proposed region. After sufficient evidence has gathered, Seeds are generated by finding the local peak of a seed-probability function that optimizes each pixel's proximity to a region centroid, and distance from any boundary. Regions are grown from these seed regions, and registered in a hierarchy that allows for co-labeling of cellular and sub-cellular components. Newly identified regions occur as new seeds, where as seeds overlapping with old regions are used to identify sub-regions, or to track regions over time.

4.27.8 Signal Extraction from Subcellular Compartments

I also have functions for the extraction of normalized Pointwise-Mutual-Information (nPMI), which can operate on a pixel-to-pixel basis or on a region-to-pixel basis. This operation accumulates mutually informative changes in all pixels in the maximal bounding-box (e.g. 64x64 pixels) surrounding each identified regions centroid. The weights given by this function can take on values between -1 and 1, and can be used to inform any reduction operations to follow. Additionally, spatial moments can indicate the subcellular distribution of activity across the identified region. In this context, the first spatial moment M_{00} indicates the mean signal intensity.

4.27.9 User Interface for Parameter Tuning

Some system-objects also incorporate a user interface to aid in parameter selection for tuning. ## Compression: as a Tool, a Goal, as an Explanation

This section describes general compression algorithms as well as compression algorithms specifically tailored to application in video, accomplished by searching for both temporal and spatial redunancy.

4.28 Data Management

Essential to the long-term success of any operation is an effective strategy for the management I've all data coming in and out of the system. Especially important is the insured facility for continued access and translation of data as systems evolve or as data is passed to other infrastructure. Strategies that meet the need of data generators and users often if it's somewhere on a spectrum of strictly defined data structures that minimize errors but are strict and difficult to evolve, versus those that require verbose documentation which hinders performance but are adaptable.

The new structures that scientists typically use are on the flexible and self-describing Spectrum and, these include two files for large image formats, fits files, cifs, Matlab structure definitions, which are an implementation of hdf5 file structures. And similarly standard JavaScript object notation in parentheses Json structures. At the other end exist Raw structures compression codex and other processing procedures, and cannot be accessed reliably without thorough knowledge of the source code for napi, or extensive packing and guessing. In the middle exist and number of serialization formats such as Google protobuf format, Thrift, Avril, Captain T, ND growing number of other formats that try to offer flexibility in evolution over time as well as variable levels of self-description and performance. These formats are often to find in simple files and are used by code generators 2 produce computational e efficient code for serialization and D serialization without losing cross platform compatibility or risking rapid deprecation as the needs for the format evolve with the needs of the users. Types can be added to provide new functionality without damaging backwards compatibility. Additionally this model is off in accompanied by a translator herbal component to JavaScript object notation which makes the format ultimately universal in the case of a desperate need for compatibility when performance no longer matters

Tags: data, directory, format, programming, serialization ## Data Scaling

This section describes the reality of how data scales as more and more sensors are added. Typically we humans think linearly, but the ramifications of increasing something like sensor size is often exponential. Furthermore, many operations we perform have costs that scale exponentially, and in my humble opinion are not even worth attempting if any such procedure must applied

continuously on a data=stream. ## Distributed Dataflow and Streaming

4.28.1 Language: Is MATLAB the best tool for this job?

- Standard language in many engineering programs
- Proprietary
- Performance
- Compatibility
- Need for a "SandBox"
- Lacks modularity
-

4.28.2 Alternatives Languages

- Python
- C/C++
- Java/Scala
- Javascript/Node
- GO, Haskell

4.28.3 Databases

4.28.4 Big Data

- not exactly
- disparate simple queries across

4.28.5 Computational Models

- Dataflow Processing

- Actors model
- Petri Nets
- Graph Processing
  - i.e. Tensorflow

---

5 Information and Informativity

This section presents a background in information theory in the context of behavior sensors and video streams. I further elaborate on what types of information neuroscientists and healthcare providers would hope to extract from these data, focusing on a larger picture of connecting the process of extraction directly the ultimate application.

Whereas previous attempts to process data focus largely on putting it in an intermediate state that is workable with current limitations on computation, visualization, and developer interaction with the data-streams, we now have an opportunity to skip intermediate states and develop with applications in mind.

Transition into compression by showing that previous attempts were essentially compression algorithms developed specifically for the data and preconceived notions about results, but that we can also apply very well developed general compression algorithms, with slight modifications to extract features such as localized bitrate that quantify (in an unbiased way) information content in an selectable region of a video-frame as it changes over time.

---

6 Discussion

6.1 Points to Address

- "Biomimicry" in Visual Processing

  - This section describes how image and video processing in the computer relate to visual processing in the mammalian brain. The overall goal is to emphasize the advantage and importance of biomimetic development.
  - Neuromorphic computing
    * On chip image processing
      · relation "edge computing"
    * event-based image sensors
      · "Artificial retina"
      · tittto
      · situations where event streams are advantageous
      · asynchronous
      · threshold-based
    * convolutional neural-nets and deep learning for specific tasks
      · similarities
      · differences (substantial)
    * genetic programming approaches to procedure optimization
      · minimize latency
      · maximize sensitivity and accuracy
      · minimize computational cost
      · minimize energy expenditure (metabolic efficiency)
  - Visual stream processing
    * feature extraction
    * motion estimation and compensation
  - Asymmetry of learning/training time and desired inference computation time
- Common themes across projects
  - finding common standards
    * sticking to non-proprietary *open source* conventions
      · optical parts (lens threads)
      · file formats
      · software libraries
      · programming languages
    * file transmission
      · web-based
  - Borrow from related sectors
    * better developed in many cases
    * surveillance

    * media streaming for web/entertainment
    * sports
    * astronomy/telescopes
    * medical imaging
    * automotive
–

7 Appendix

7.1 Software for Procedure Development

7.1.1 Development Environment

8 References