5-24-2012

The Automation of Electrophysiological Experiments and Data Analysis

David Morton
Washington University in St. Louis

WASHINGTON UNIVERSITY IN SAINT LOUIS

Department of Physics

The Automation of Electrophysiological Experiments and Data Analysis

by

David L. Morton

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in Saint Louis
in partial fulfillment of the the
requirements for the degree of
Doctor of Philosophy

May 2012

Saint Louis, Missouri

Abstract

The role of computation in science is continually growing and neuroscience is no exception. Despite this, a severe lack of scientific software infrastructure persists, slowing progress in many domains. In this thesis, we will see how the combination of neuroscience and software engineering can build infrastructure that enables discovery.

The first chapter discusses the Turtle Electrophysiology Project, or TEP, an experiment-automation and data-management system. This system has allowed us to automate away some of the most tedious tasks involved in conducting experiments. As a result, we can collect more data in less time, and with fewer errors related to the loss of metadata (information about how the data were collected). Also, since all of the metadata is automatically digitized during the experiment we can now completely automate our analyses.

Chapters two and three are examples of research conducted using the ever-evolving TEP system. In the first instance, we used TEP to deliver visual stimuli and handle data-management. In chapter three, the experiments involved delivering electrical stimuli instead of visual stimuli, and much more rigorous analysis. And even though TEP was not specifically designed to handle collecting data this way, the flexible tags system enabled us to do so.

Finally, chapter four details the construction of a robust analysis tool called Spikepy. Whereas TEP is specially designed for the turtle preparation we have, Spikepy is a general-purpose spike-sorting application and framework. Spikepy takes flexibility to the extreme by being a plugin-based framework, yet maintaining a very easy to use interface.

Contents

List of Figures

Chapter 1

Turtle Electrophysiology Project

1.1 Introduction

The turtle electrophysiology project, from here on referred to as TEP, is designed to provide structure to the electrophysiological workflow that is associated with the turtle whole-brain preparation. Refer to chapters 2 and 3 for more about this preparation. This project provides tools for data-acquisition, storage/organization, and libraries to assist with analysis. It is written almost entirely in Python, the exception being a portion of the data-acquisition that is written in LabView. This chapter will consist of a user's guide followed by a section discussing some of the architecture underlying the project.

In the users guide we will explain how to install TEP and set everything up, how to run a typical experiment and how to access the data from an experiment. The guide will only cover the architecture that is relevant to the typical user. In the architecture section we will cover how the stimulus delivery system works and how to create new visual stimuli as well as how the data and metadata are managed.

1.2 User's Guide

This guide is written for the typical TEP user and is broken into three parts. Part one covers setup and installation, as well as an overview of how the project is organized. Part two walks through how to run a typical experiment in some detail. And part three describes how to gain access to the data acquired during an experiment, how one should organize analysis code, and an overview of the most relevant analysis tools included with the project.

Names that are formatted like `SomeName` refer to database objects (see subsection on data access 1.2.3). Some other text uses the `fixed-width font` to indicate that it is either a command or file path. Items that are **bold font** represent buttons in a GUI program.

1.2.1 Installation

The TEP project consists of two computers that each have specific roles to play (Figure 1.1). The so-called acquisition computer is dedicated to running the LabView data-acquisition programs and nothing else. This computer is likely to be a Windows-based computer, but its possible that it could be any computer with a working installation of LabView. The other computer is responsible for all the other aspects of TEP, such as data storage, running the experiment, and delivering visual stimuli. Strictly speaking, the stimuli could be delivered via a third (dedicated) computer, but this setup is non-standard and will not be covered here. We therefore refer to this second computer as the stimulus computer. These two computers communicate with one another in two ways that will be described in detail later, but consist of using the computer's parallel (i.e. printer) port as well as utilizing a typical network connection.

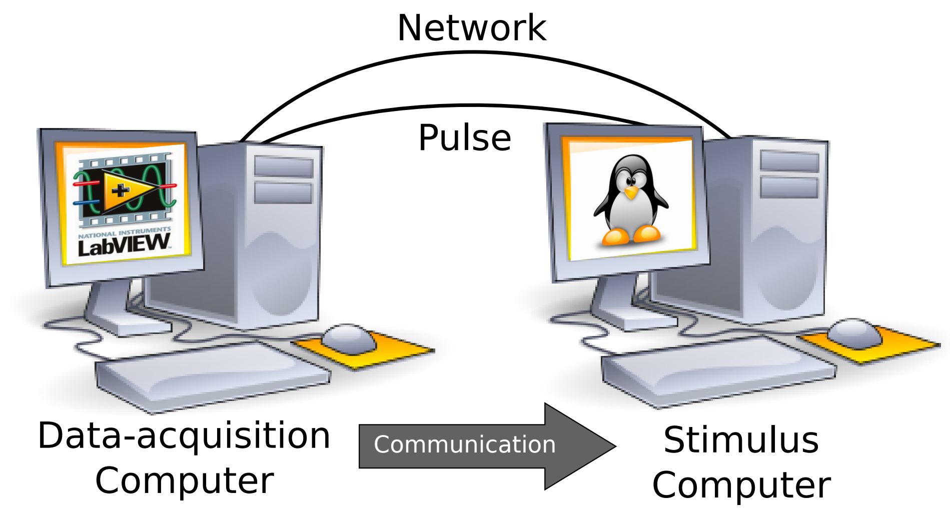


Figure 1.1: Computer structure. The data-acquisition computer runs LabView and sends information to the stimulus computer. There are two modes of communication, via the network, and via a pulse delivered to the parallel port of the stimulus computer.

Subversion

Before we get to the detailed installation procedure, it is important to understand how software version control works. The days are over when people just worked on code in a directory and then made copies of the directory each time they made "important" changes. Software version control allows us to avoid this by setting up a repository from which we can check out a working-copy of the code. With version control software, when we change the working-copy we can just update the repository with those changes. This system allows us to undo any changes to any file we have, and lets more than one person work on a project at the same time without the need to manually combine the changes from each person. And as a perk, it makes installing the software on a new system really easy. Moreover, since the repository is stored on an internet server somewhere, it also ensures that we have a robust backup system (at least for the software).

Subversion is a software version control application that is easy to use and very popular. The

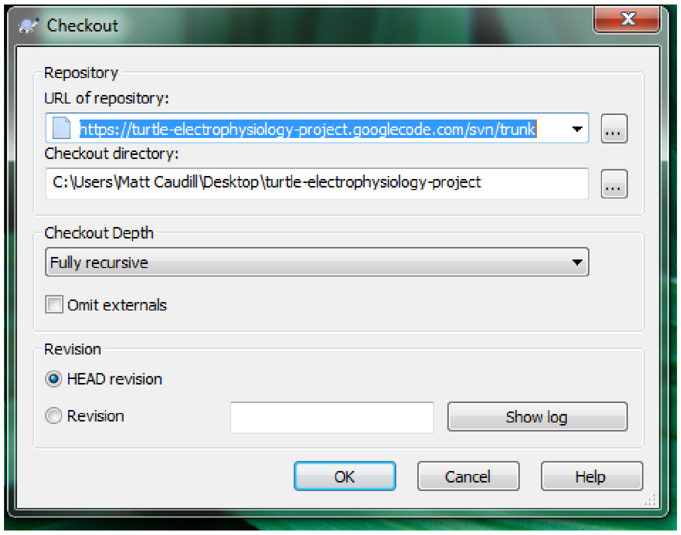
TEP software is managed by subversion and the repository is hosted on a googlecode server. Although there are many great tutorials on the internet covering how to use subversion, the most important activities are:

- `svn checkout` *repository* - This pulls down a working-copy from the repository onto the computer. You only have to perform this activity once per installation.
- `svn update` - This will update the project to the latest version from the repository. This should be done before every experiment and before editing any code.
- `svn status` - Prints the status of files in the project. The status is displayed as a single character preceding the filename. The most common status characters are, M=modified, ?=not managed by subversion, and C=conflict which must be handled manually (google for how to handle these).
- `svn add` *filename* - Allows subversion to track the status of the file. This only has to be done once per file/directory.
- `svn commit` - Updates the repository with the changes that you've made to the project. If you don't commit the changes you've made, then other computers using TEP cannot access those changes.

The most common daily workflow is `update`, do some work, `add` new files, and finally `commit` your changes to the repository.

Data-acquisition computer

The computer with a working installation of LabView is responsible for acquiring electrophysiology data during the experiment and is therefore called the data-acquisition computer. The LabView programs needed for data-acquisition are in the TEP subversion repository, so we need to `svn checkout` a working copy of the project. Since this computer is likely running Windows, the easiest way to do this is by installing a program called TortoiseSVN from http://www.tortoisesvn.net. Once that is installed, simply right click on the desktop and there should be an option for **svn checkout**. Fill out the dialog as seen in Figure 1.2. After clicking **OK**, subversion will check out a copy of the entire turtle-electrophysiology-project to the designated directory. Create shortcuts on the desktop to `mxACQUISITION.vi`, `mxREADFILE.vi`, and `mxSPEEDBOAT.vi`. These programs are for data-acquisition, viewing data, and a less-featured but more memory-efficient data-acquisition respectively.



Repository
URL of repository: https://turtle-electrophysiology-project.googlecode.com/svn/trunk
Checkout directory: C:\Users\Matt Caudill\Desktop\turtle-electrophysiology-project

Checkout Depth: Fully recursive
☐ Omit externals

Revision
[✓ HEAD revision]
Revision

Figure 1.2: Tortoise SVN checkout dialog. The URL of repository tells TortoiseSVN where to locate the repository. The checkout directory will be created locally and filled with the contents of the repository.

The data-acquisition computer communicates with the stimulus computer in two ways. The stimuli are triggered via the parallel port (aka DB-25, or printer port). There should be cables in the lab that route the 5V ttl pulse from the BNC 2080 to the stimulus computer. These cables connect the 5V ttl output (DAC0out or DAC1out) to pin number 13 (aka paper-out pin) on the parallel port of the stimulus computer. The data, collected via the LabView program, should be set to save to a shared folder on the stimulus computer (see Figure 1.3).

Stimulus computer

The computer responsible for delivering visual stimuli and recording and tracking metadata is referred to as the stimulus computer. To get this computer ready we need to `svn checkout` the

TEP codes:

http://code.google.com/p/turtle-electrophysiology-project/source/checkout

Two other things need to be done in order to get everything setup. First, set the PYTHONPATH environment variable to include the directory you checked out the project to. And then run the scripts included in the setup directory. The script `get_tep_packages.sh` will download and install all the external packages that TEP needs to function, and `setup_shared_folder.sh` will create and share a folder (`$HOME/tep_databases/uploads`).

1.2.2 Running an experiment

The objective of an experiment is to collect recordings of two types, visually driven recordings are `VisualRecordings` and all others are `SearchingRecordings`. The two computers will work in tandem to achieve this, where the data-acquisition computer is used to simply record from the brain while all other functions are carried out on the stimulus computer. It is important that experiments are as well planned out as possible, to ensure that you collect all the data you can in the limited time you have. This means not only having procedures written down, but all the needed stimuli set up as `ProtocolElements` and `Protocols` as you'll read about later.

In this section, we will first touch on the data-acquisition process, and then focus in on how the stimulus computer guides you through collecting data and metadata related to the experiment. The program `run_experiment.py` on the stimulus computer organizes an experiment into four distinct stages. The first stage collects the most basic information about the experiment. Next, the user sets up the eye-cup coordinate system and describes the electrodes that will be used. The third stage allows the experimentalist to search for visual responses and to collect data that is not synchronized with a visual stimuli. The last stage is devoted to collecting `VisualRecordings` which are synchronized to the visual stimulus.

A typical experiment will run through these stages in order, but may involve back-tracking to the preparation stage (stage two) when electrodes are repositioned. The experiment then proceeds to stage three and four. While the experiment is being performed, all the data that is entered is stored in a relational database. This makes it possible (see 1.2.2) to search, group, or sort based on any of this data when performing analyses.

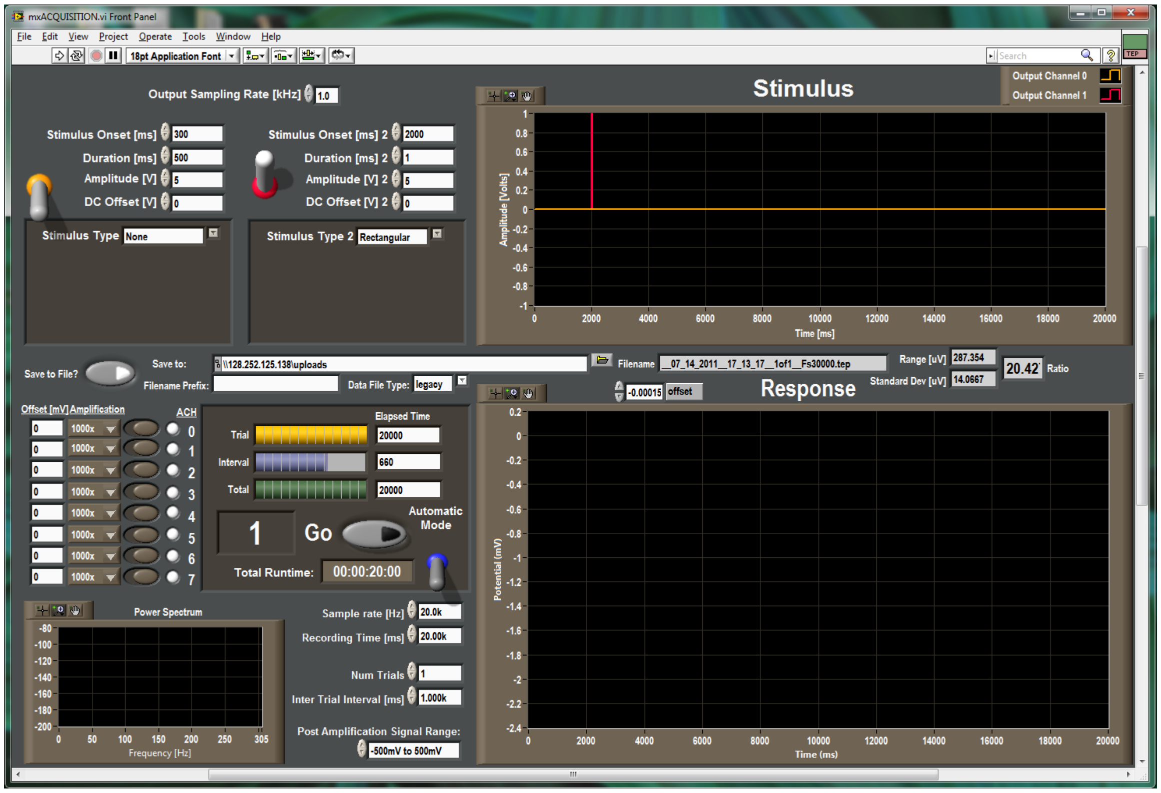
Data-acquisition

We begin at the point where the specimen has been dissected and prepared for recordings, but before any electrodes have been positioned. On the data-acquisition computer, open up the data-acquisition LabView program `mxACQUISITION.vi` which should have a shortcut on the Desktop. With this program you can control the two digital output channels as well as all the analog input (recording) channels. Familiarize yourself with the controls of this program (Figure 1.3).

Stages

The experiment will be broken up into four stages, where you will enter in information regarding how the experiment is being carried out. The stages help to break up the experiment into bite-sized pieces that are much less complicated than the whole experiment taken all together. Once you've completed a stage the next stage will become available. You'll also be able to go back to a previous stage if you need to. The information gathered at each stage will build off the information from previous stages. This means that you won't be asked for information more than once, unless you go back and change something.

On the stimulus computer open up a terminal and run the program `run_experiment.py` and that will start the program that runs an experiment. The program may prompt you for a password at this point, just to gain access to the parallel port (needed to trigger visual stimuli). The first



Output Sampling Rate [kHz] 1.0

Stimulus Onset [ms] 300
Duration [ms] 500
Amplitude [V] 5
DC Offset [V] 0
Stimulus Type None

Stimulus Onset [ms] 2 2000
Duration [ms] 2 1
Amplitude [V] 2 5
DC Offset [V] 2 0
Stimulus Type 2 Rectangular

Save to File?
Save to: \\128.252.125.138\uploads
Filename Prefix:
Data File Type: legacy
Filename _07_14_2011__17_13_17__1of1__Fs30000.tep
Range [uV] 287.354
Standard Dev [uV] 14.0667
20.42 Ratio
-0.00015 offset

Offset [mV]	Amplification	ACH
0	1000x	0
0	1000x	1
0	1000x	2
0	1000x	3
0	1000x	4
0	1000x	5
0	1000x	6
0	1000x	7

Elapsed Time
Trial 20000
Interval 660
Total 20000
1
Total Runtime: 00:00:20:00

Sample rate [Hz] 20.0k
Recording Time [ms] 20.00k
Num Trials 1
Inter Trial Interval [ms] 1.000k
Post Amplification Signal Range: -500mV to 500mV

Figure 1.3: LabView data-acquisition program. The top area contains controls to adjust the digital output channels (DAC0out and DAC1out). The bottom area controls the data-acquisition through any or all of the analogue input channels (ACH0 - ACH15 on the BNC 2080). Note that some controls are off the screen. In the lower-left there is a plot that will display the power-spectrum of the data after acquisition. In the lower-right there is a large area to plot the data.

screen presented will display some information about the database, such as how many experiments have been run, and who ran the last experiment and when it was performed. Click on the **Start Experiment** button to proceed to the first stage of the experiment.

Experiment

The first stage of an experiment prompts you for information about:

1. Experiment Date
2. The Specimen – Who performed the dissection and how did it go?
3. The Researcher – Who will perform the rest of the experiment?
4. The Experiment – What is the purpose of this experiment?

As you enter information, the label for that data will turn red to indicate that this has not been committed to the database yet. Once you have filled out everything on the page click on the **Submit** button and the red labels should return to black. Also, after you have clicked **Submit** the **Goto Preparation Stage** button will become enabled, allowing you to proceed to the next stage.

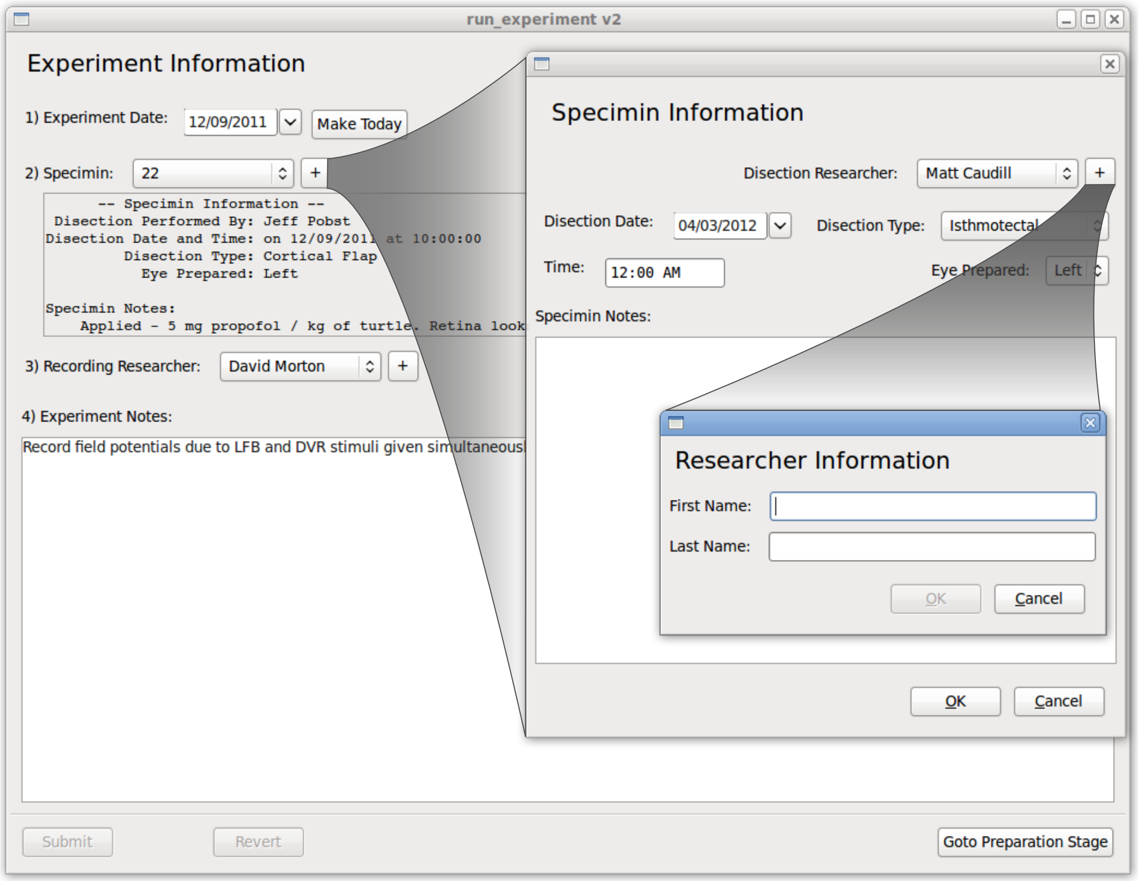
Preparation

The preparation stage is concerned with collecting data related to:

1. Electrodes – What electrode(s) will be used and where will they be placed?
2. Eye Cup – How is the screen projected down into the eye-cup?
3. The Preparation – additional (optional) notes about the preparation

Figure 1.5 shows how to add electrodes to a `Preparation`. After adding an electrode to the list, you will be able to press the **Set Position** button (see Figure 1.6). This will allow you to choose a stereotactic map and then position the electrode on that map with the mouse. After doing this for all the electrodes (both recording and stimulating), press the **Set Eye Cup** button.

When you press the **Set Eye Cup** button, you can choose the monitor and will then be put into eye-cup mode where you can adjust how stimuli will draw on the screen (see Figure 1.7). Dim


run_experiment v2

Experiment Information

1) Experiment Date: 12/09/2011 Make Today

2) Specimin: 22 +

```
       -- Specimin Information --
 Disection Performed By: Jeff Pobst
Disection Date and Time: on 12/09/2011 at 10:00:00
          Disection Type: Cortical Flap
            Eye Prepared: Left

Specimin Notes:
    Applied - 5 mg propofol / kg of turtle. Retina look
```

3) Recording Researcher: David Morton +

4) Experiment Notes:

Record field potentials due to LFB and DVR stimuli given simultaneous

Specimin Information

Disection Researcher: Matt Caudill +

Disection Date: 04/03/2012

Disection Type: Isthmotectal

Time: 12:00 AM

Eye Prepared: Left

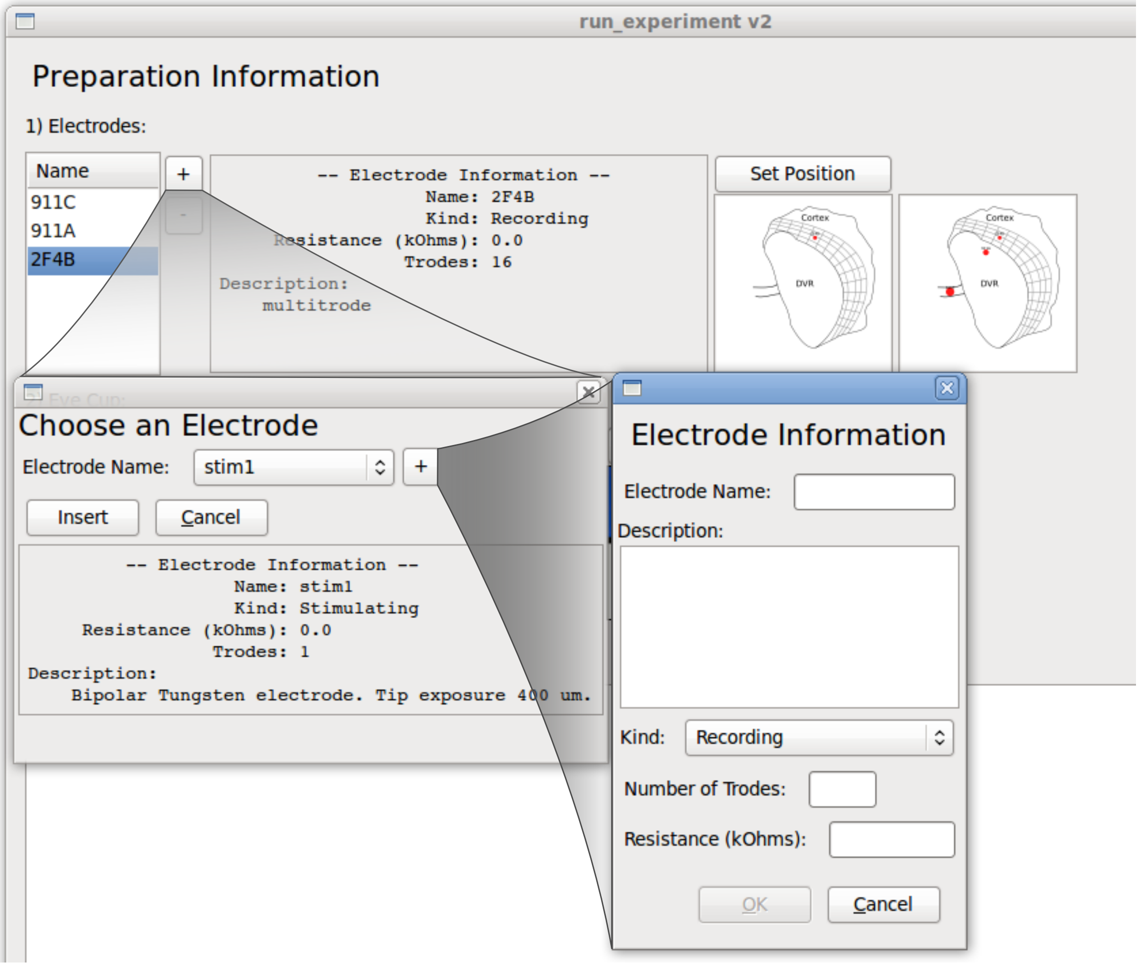
Specimin Notes:

Researcher Information

First Name:

Last Name:

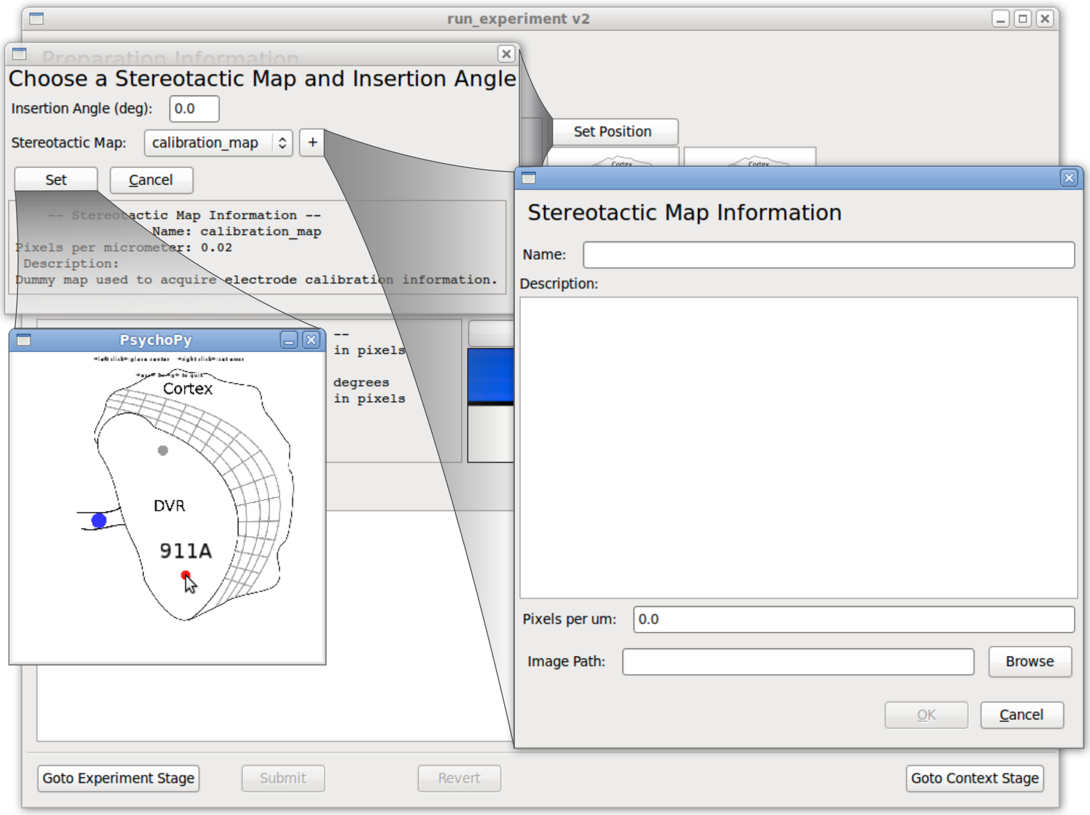
Figure 1.4: Starting an `Experiment`. You may select an existing specimen, if you're performing another experiment on the same specimen, or define a new specimen. Defining new researchers is also very easy.



```
           -- Electrode Information --
                     Name: 2F4B
                     Kind: Recording
      Resistance (kOhms): 0.0
                   Trodes: 16
Description:
    multitrode
```

```
           -- Electrode Information --
                     Name: stim1
                     Kind: Stimulating
      Resistance (kOhms): 0.0
                   Trodes: 1
Description:
    Bipolar Tungsten electrode. Tip exposure 400 um.
```

Figure 1.5: Adding electrodes to a preparation. Clicking on the + button brings up a dialog allowing you to select an electrode to add to this `Preparation`. Within that dialog, clicking on the + button next to the electrode selector, opens a dialog allowing you to define new electrodes. This means you only have to define electrodes once, and can use them in multiple experiments.



Choose a Stereotactic Map and Insertion Angle

Insertion Angle (deg): 0.0

Stereotactic Map: calibration_map

```
   -- Stereotactic Map Information --
                Name: calibration_map
Pixels per micrometer: 0.02
 Description:
Dummy map used to acquire electrode calibration information.
```

```
--
in pixels

degrees
in pixels
```

Stereotactic Map Information

Name:

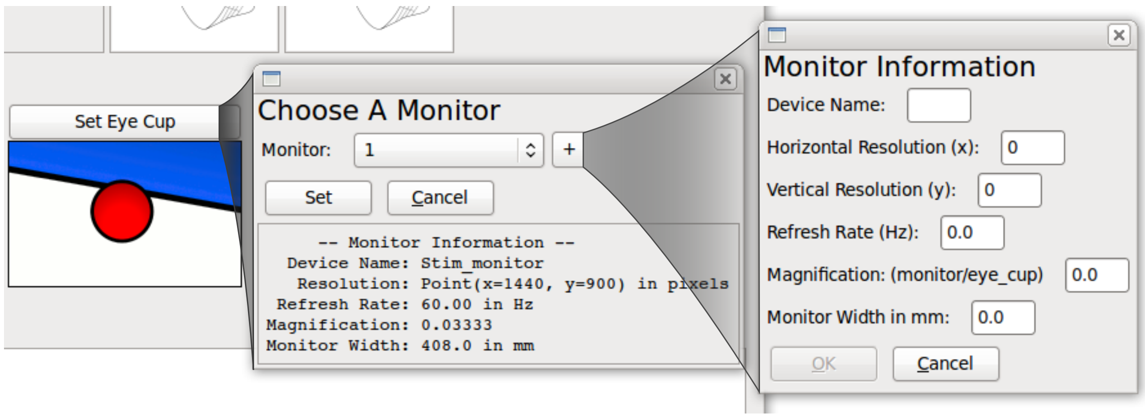
Description:

Pixels per um: 0.0

Image Path:

Figure 1.6: Positioning electrodes. You can indicate where electrodes are located using simple drawings of the brain region of interest that we call `StereotacticMaps`. New maps can be added in easily as well.

the lights and while looking inside the experimental chamber at the eye-cup, use the video game controller to move the eye-cup on the screen into place and adjust its size to match the real eye-cup. Also, tilt the visual streak on the screen to match the visual streak of the real eye-cup. This will ensure that when visual stimuli are delivered that they are scaled, translated and rotated correctly. Since the monitor and the magnification are fully described, the software can now convert from visual angle (in degrees) to pixels and vice-versa.



```
    -- Monitor Information --
  Device Name: Stim_monitor
   Resolution: Point(x=1440, y=900) in pixels
 Refresh Rate: 60.00 in Hz
Magnification: 0.03333
Monitor Width: 408.0 in mm
```

Figure 1.7: Defining the monitor to eye-cup transformation. You can choose a monitor that has already been described, or add a new monitor. The magnification is all that is needed to allow the software to convert pixels into visual angle (degrees).

Finally, you can record any notes related to the preparation and click the **Submit** button. This will again, make all the red labels turn black (letting you know the database has been updated) and enable the **Goto Context Stage** button. We are now ready to proceed to the context stage.

Context

The context stage is the first opportunity to collect recordings. The context stage is where we input data about:

1. Searching Technique – How did you decide that this is where we should record?
2. Electrode Depths – How far down did you lower each of the electrodes?

3. Chemicals/Pharmacology – What (if any) drugs or chemicals are relevant to this set of recordings?
4. Responsive Area – If you want, you can "chalk out" a responsive area using the mouse to move stimuli around and clicking where a response occurred.
5. Stimulus Origin – If you want to draw stimuli centered not on the eye-cup but on some other area, this allows you to set where using the mouse.
6. The Context – additional (optional) notes about the context.

In addition to all the above information, this is where you will be able to save recordings you have made that are not synchronized with a visual stimulus (`SearchingRecordings`). When you click **Submit** the data will be moved to a permanent storage location and the final stage will become available. If you are not intending to collect `VisualRecordings` then you can simply repeat the process of saving and submitting on this panel for the remainder of the experiment.

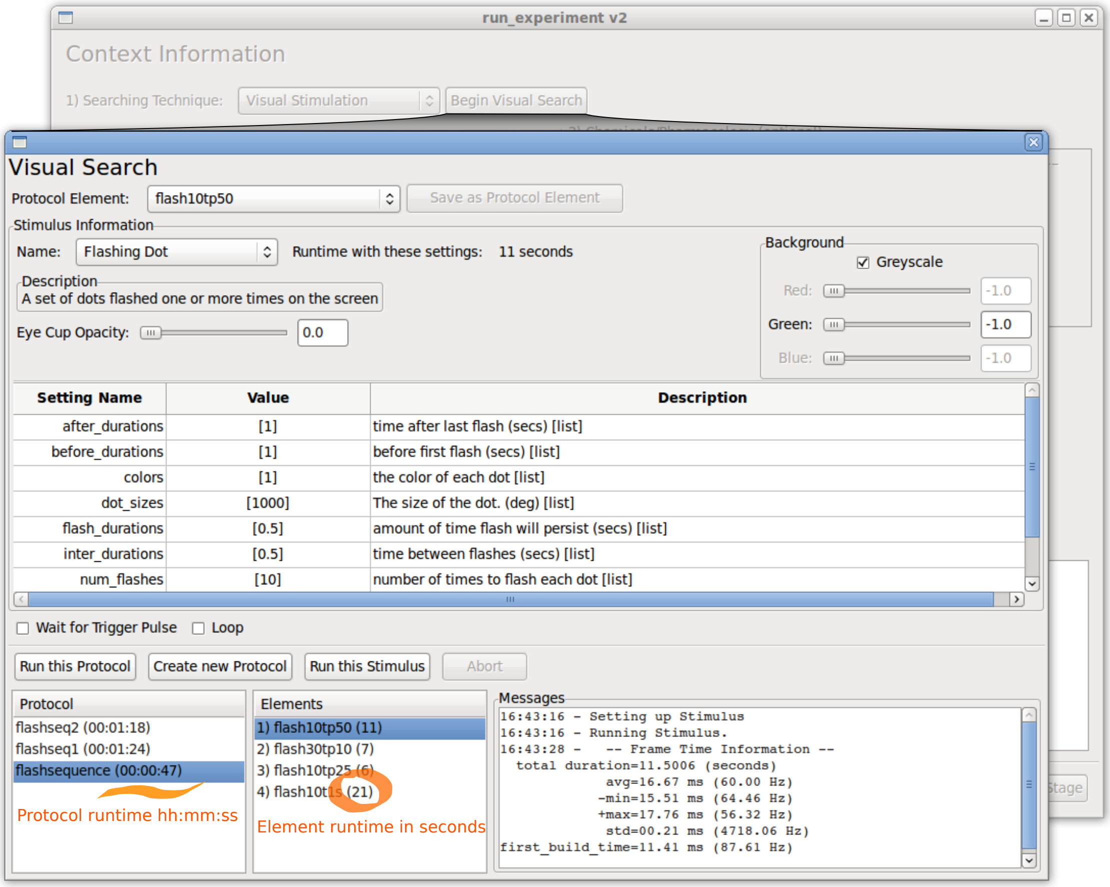
Collecting recordings

To collect recordings, set the data-acquisition computer to save data to the shared folder (see Figure 1.3). Go ahead and record as many recordings as you wish to save. Once the data has been recorded, click **save...** on the stimulus computer and a dialog will open, allowing you to confirm the files you wish to save. The dialog also allows you to enter notes or attach `Tags` to the data that can be used later to group or sort data. This process can be repeated as often as you wish.

Visual recordings

The last stage in the `run_experiment.py` program will allow you to save `VisualRecordings`. This panel allows you to adjust any of the settings related to how the visual stimulus is constructed (except those related to the eye-cup as it was set up earlier). Many different visual stimuli are available, and each of them has a number of settings such as colors, speeds, and directions.

After setting up the stimulus how you want it to be displayed, it can be previewed. With the



1) Searching Technique: Visual Stimulation

Protocol Element: flash10tp50

Name: Flashing Dot   Runtime with these settings: 11 seconds

Description: A set of dots flashed one or more times on the screen

Eye Cup Opacity: 0.0

Background: ☑ Greyscale; Red: -1.0; Green: -1.0; Blue: -1.0

Setting Name	Value	Description
after_durations	[1]	time after last flash (secs) [list]
before_durations	[1]	before first flash (secs) [list]
colors	[1]	the color of each dot [list]
dot_sizes	[1000]	The size of the dot. (deg) [list]
flash_durations	[0.5]	amount of time flash will persist (secs) [list]
inter_durations	[0.5]	time between flashes (secs) [list]
num_flashes	[10]	number of times to flash each dot [list]

☐ Wait for Trigger Pulse ☐ Loop

Protocol
- flashseq2 (00:01:18)
- flashseq1 (00:01:24)
- flashsequence (00:00:47)

Elements
1) flash10tp50 (11)
2) flash30tp10 (7)
3) flash10tp25 (6)
4) flash10t1s (21)

Messages
```
16:43:16 - Setting up Stimulus
16:43:16 - Running Stimulus.
16:43:28 -   -- Frame Time Information --
  total duration=11.5006 (seconds)
            avg=16.67 ms (60.00 Hz)
           -min=15.51 ms (64.46 Hz)
           +max=17.76 ms (56.32 Hz)
            std=00.21 ms (4718.06 Hz)
first_build_time=11.41 ms (87.61 Hz)
```

Figure 1.8: Displaying a visual stimulus. This panel contains settings used to control the visual stimulus. The background (what is shown before and after the stimulus) can be controlled via a set of sliders. The units of many settings are in degrees of visual angle instead of pixels, since the software can do the conversions for you. The duration of the `ProtocolElements` and the `protocols` (marked in orange) are displayed in the lists, next to their names.

settings correct, click on the **Run this stimulus** button and the stimulus will be prepared and the stimulus computer will begin waiting for a triggering pulse. On the data-acquisition computer, set the digital output channel to send a 5V pulse ($>$1ms duration) and begin recording. When the pulse is sent, the stimulus will display and that is how the stimulus and recording are synchronized. The data are saved to the shared folder and will be stored in permanent storage in exactly the same way as `SearchingRecordings`. You can add notes and attach `Tags` in the same way as well. If you plan to deliver more than one stimulus during an experiment it is useful to define a `Protocol`.

Protocols

Having a lot of settings for each stimulus gives us a lot of flexibility, but it can also be a hassle. If you plan to deliver the same stimulus during multiple experiments, or even interleave stimuli with different settings, you should define a `Protocol`. A `Protocol` is a sequence of stimuli with specific settings that will be run one after the other.

The first step in creating a new `Protocol` is to create the individual parts or `ProtocolElements`. After setting up a stimulus, simply click on the **Save as Protocol Element** button and give it a name. Once you have the `ProtocolElements` you want to show, click on the **Create new Protocol** button and a new dialog will open. This dialog will let you create a sequence of `ProtocolElements` and save that sequence with a name.

To run a `Protocol` simply select it from the list on the panel and click on the **Run Protocol** button. This will run each of the stimuli in the sequence in order, saving data after each one. On the data-acquisition computer, you should set it up to run as many trials as there are stimuli. Just make sure there is enough time in the inter-trial interval to allow the data to be transferred over the network.

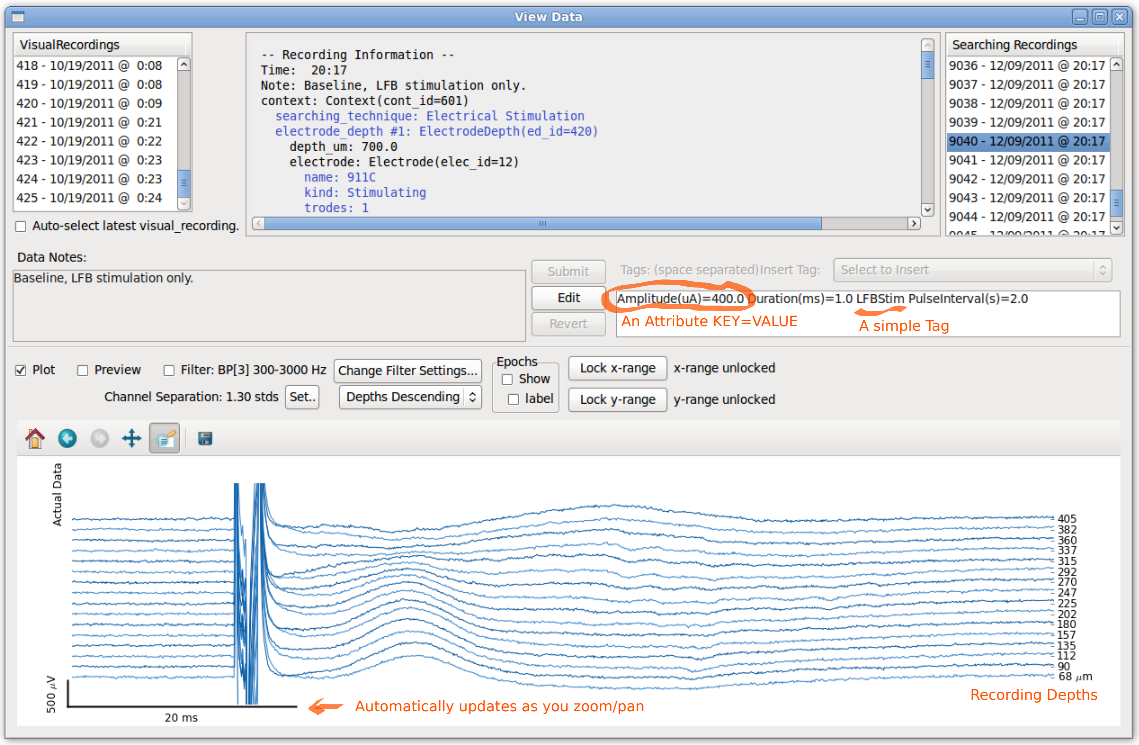
`Protocols`, once defined cannot be renamed, modified, or deleted. This is important because the `Protocol` describes exactly what stimulus was shown during an experiment. If you could modify it in any way then you wouldn't know how the stimuli were shown in the past. In fact, the `Protocol` even stores the subversion revision number of the stimulus. This way it should be impossible to not know exactly what the screen was drawing during any experiment.

Data viewer

During an experiment, as data are collected you may view them with the data viewer (see Figure 1.9). This window also displays all the information gathered about the recordings and allows you to change the notes and `Tags` associated with the recordings. There are many options available for how to plot the recordings, including filtering and displaying the stimulus epochs, which describe when certain events occurred in a stimulus. In addition, double-clicking on the plot will bring up the navigation toolbar that has tools for zooming and panning. Notice that the scale-bars drawn will automatically adjust as you zoom and pan (see Figure 1.9).

1.2.3 Analysing data

During the experiment, the `run_experiment.py` program collected a lot of information about the recordings you took. This information, or metadata, is stored in an SQLite database. In this section we will walk through how to search through this database and how to write programs to analyze the data (recordings). In the first section we will cover the basic usage of the search GUI, a powerful tool to curate your data. Next we will cover how to access the database using just python, which will enable you to start creating analysis scripts. There already exists a nice library of code to help with this and we will describe how it is organized and how the database itself is organized.



View Data

VisualRecordings
418 - 10/19/2011 @ 0:08
419 - 10/19/2011 @ 0:08
420 - 10/19/2011 @ 0:09
421 - 10/19/2011 @ 0:21
422 - 10/19/2011 @ 0:22
423 - 10/19/2011 @ 0:23
424 - 10/19/2011 @ 0:23
425 - 10/19/2011 @ 0:24

☐ Auto-select latest visual_recording.

```
-- Recording Information --
Time:  20:17
Note: Baseline, LFB stimulation only.
context: Context(cont_id=601)
  searching_technique: Electrical Stimulation
  electrode_depth #1: ElectrodeDepth(ed_id=420)
    depth_um: 700.0
    electrode: Electrode(elec_id=12)
      name: 911C
      kind: Stimulating
      trodes: 1
```

Searching Recordings
9036 - 12/09/2011 @ 20:17
9037 - 12/09/2011 @ 20:17
9038 - 12/09/2011 @ 20:17
9039 - 12/09/2011 @ 20:17
9040 - 12/09/2011 @ 20:17
9041 - 12/09/2011 @ 20:17
9042 - 12/09/2011 @ 20:17
9043 - 12/09/2011 @ 20:17
9044 - 12/09/2011 @ 20:17

Data Notes:
Baseline, LFB stimulation only.

Tags: (space separated) Insert Tag: Select to Insert

Amplitude(uA)=400.0 Duration(ms)=1.0 LFBStim PulseInterval(s)=2.0

An Attribute KEY=VALUE

A simple Tag

☑ Plot ☐ Preview ☐ Filter: BP[3] 300-3000 Hz

Channel Separation: 1.30 stds

Depths Descending

Epochs ☐ Show ☐ label

x-range unlocked

y-range unlocked

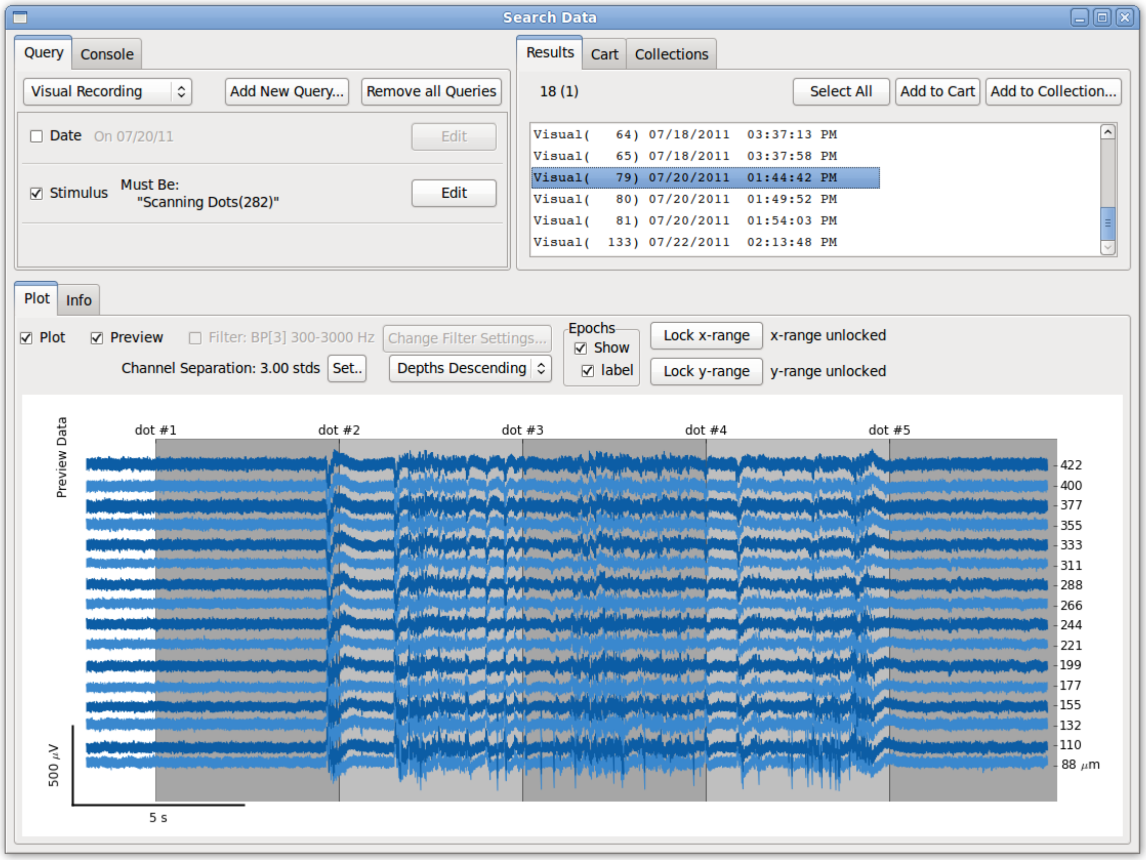
Automatically updates as you zoom/pan

Recording Depths

Figure 1.9: Data viewer. This window can be opened up at anytime during an experiment. Data that has been collected can be selected and plotted with various options. The `Tags` and `Attributes` (noted in orange) associated with the data can be modified here as well. The electrode's recording depths are calculated based on the information entered during the `Preparation` stage and the `Context` stage.

Using the search GUI

The search GUI (located at `$HOME\turtle-electrophysiology-project\tep\search\search.py`) will allow you to browse through your data as well as create `Collections` of recordings. You can construct queries that allow you to filter the data by date, stimulus-type, or by `Tags`. The database can be queried on any of the metadata that was collected during the experiment as well (using Python), but GUI components haven't yet been written to perform any other types of queries. For this reason, the console is also available within the GUI which will allow you to construct arbitrarily complex queries.



☐ Date On 07/20/11

☑ Stimulus Must Be: "Scanning Dots(282)"

18 (1)

```
Visual(   64) 07/18/2011  03:37:13 PM
Visual(   65) 07/18/2011  03:37:58 PM
Visual(   79) 07/20/2011  01:44:42 PM
Visual(   80) 07/20/2011  01:49:52 PM
Visual(   81) 07/20/2011  01:54:03 PM
Visual(  133) 07/22/2011  02:13:48 PM
```

☑ Plot ☑ Preview ☐ Filter: BP[3] 300-3000 Hz

Channel Separation: 3.00 stds

Depths Descending

Epochs: ☑ Show ☑ label

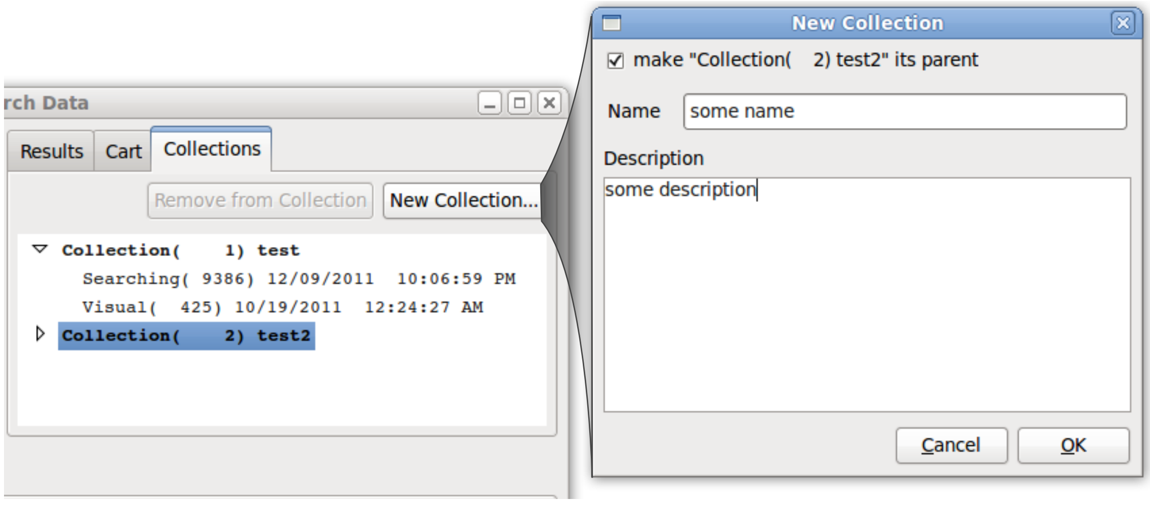
x-range unlocked

y-range unlocked

Figure 1.10: The search GUI. The panel in the upper-left allows you to construct simple queries, or (on the other tab), use the Python console to dig through the database. When you select a recording in the upper-right panel, it is plotted in the lower panel. The upper-right panel also has tabs related to creating and maintaining `Collections` of recordings.

In addition to querying the database, recordings can be plotted in the same way they were in the data viewer (see previous section). When the `run_experiment.py` program is shut down, a dialog will ask you if you want to archive your data now. If you say yes, the data will be compressed and previews will be constructed. These previews will allow you to very quickly see what your recordings look like without having to load the entire recording from the disk.

As you sort through your data, you will likely want to create `Collections` of data that are somehow related. This can be done within the GUI as well. `Collections` can contain any number of `SearchingRecordings` and or `VisualRecordings` and even other `Collections`. `Collections` can have `Tags` and `Attributes` associated with them as well. Curating your data into collections should be the first step in writing your analysis scripts.



```
Collection(     1) test
    Searching( 9386) 12/09/2011   10:06:59 PM
    Visual(   425) 10/19/2011   12:24:27 AM
Collection(     2) test2
```

☑ make "Collection(2) test2" its parent

Name: some name

Description: some description

Figure 1.11: Creating a new `Collection`. On the collections tab of the upper-right panel you can create and maintain `Collections` of recordings. `Collections` can contain `SearchingRecordings`, `VisualRecordings`, and other `Collections`.

Accessing the database in Python

In addition to being able to access the data through the search GUI, the TEP project has tools to allow you to access the database in Python. Normally, with a relational-database like the one used in TEP and elsewhere the database would be accessed by writing SQL queries. SQL queries are difficult to construct and can become rather complicated, so software engineers invented Object Relational Mappers or ORMs. In TEP, we use an ORM called SQLAlchemy that makes SQL queries unnecessary. Instead of SQL queries, SQLAlchemy makes database access ’invisible’ by providing regular Python classes that handle all the database access for you. We will cover here how to open up a database and how to access it using Python.

If you just want to quickly open the database and browse through it you can simply execute the command `read_only_database_shortcut.py` and it will open up an iPython session with the database already opened up in read-only mode. Similarly, there is a shortcut for altering the database called `alter_database_shortcut.py` that will open up the database in read-write mode after giving you stern warnings. For writing analysis scripts there are library functions that will take care of opening up the database as well.

- `choose_database_from_list` – Takes no arguments and displays a list on the screen letting you choose a database. Returns the `database_dir` that is needed for the other two functions.
- `backup_database` – Requires `database_dir` and `backup_suffix` that will be applied to the backed up database file. Backed up databases are saved in the `database_dir` under a folder named `database_backups`.
- `open_database` – Opens a database, given its `database_dir`. Optional arguments allow you to open in read-only mode, or read-write mode. If you choose read-write mode, this function will call `backup_database` before opening opening up the database.

After opening the database, you'll want to access it using Python. This requires that we import some objects that help with that (see Figure 1.12). If you use the command `read_only_database_shortcut.py` this will all be done for you. To see how these objects allow us to access the database it is helpful to understand how the database is organized. Figure 1.13 shows the overall database organization. An illustrative example of how to navigate through the database is shown in Figure 1.14

```
 1  from tep.acquire.database.open_utils import \
 2          choose_database_from_list, open_database
 3
 4  # Choose what database is opened.
 5  database_dir = choose_database_from_list()
 6  open_database(database_dir, read_only=True)
 7
 8  #   Now that the database is opened, we can import things that
 9  # help us to access it.
10  from tep.acquire.database.alchemy_database_manager import adbm
11  from tep.acquire.database.alchemy_classes import *
12  from tep.acquire.database import alchemy_globals
```

Figure 1.12: Accessing the database with Python. This is a typical way to start off an analysis file. Lines 1 - 6 are responsible for opening up the database. To access the database we need the `alchemy_database_manager` or `adbm`. It is also very handy to have access to the classes that represent objects within the database (line 11) and the global variables associated with the database (line 12).

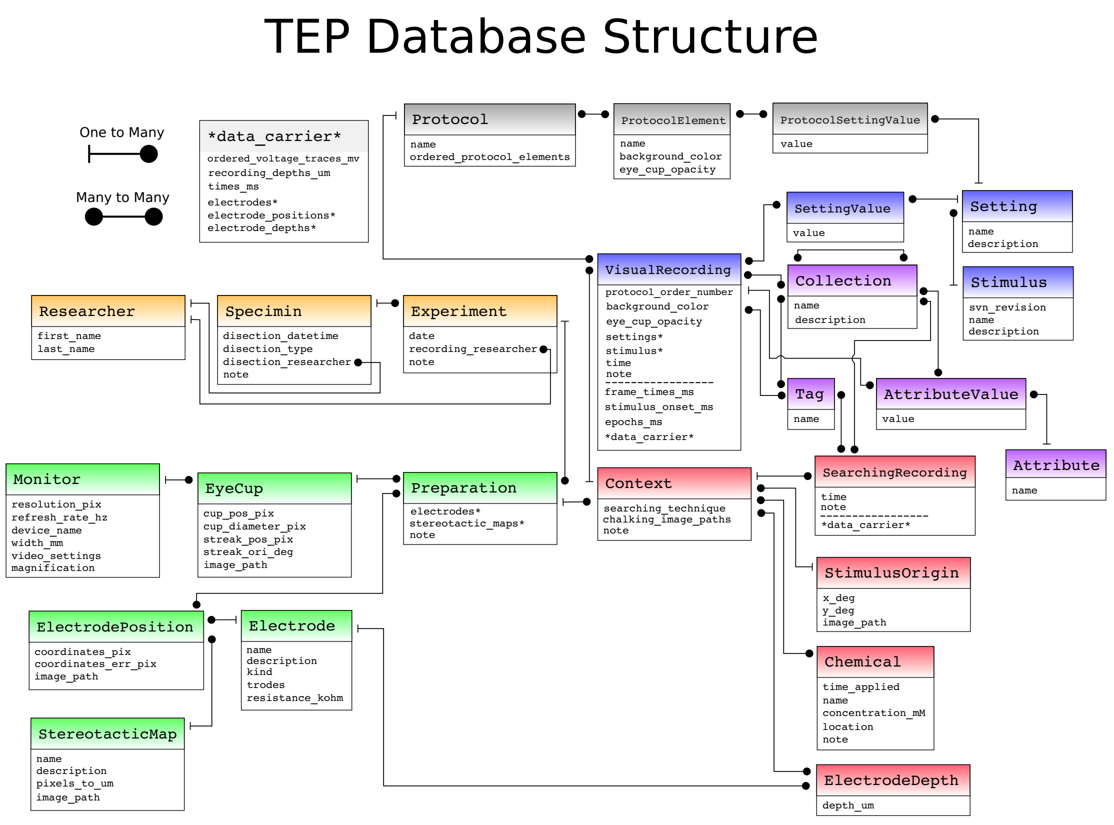
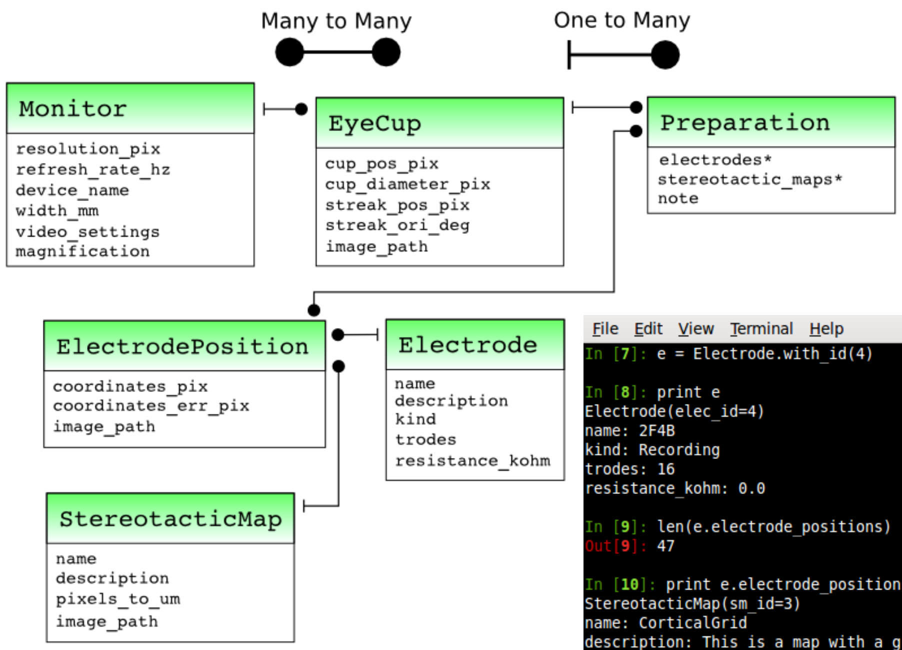
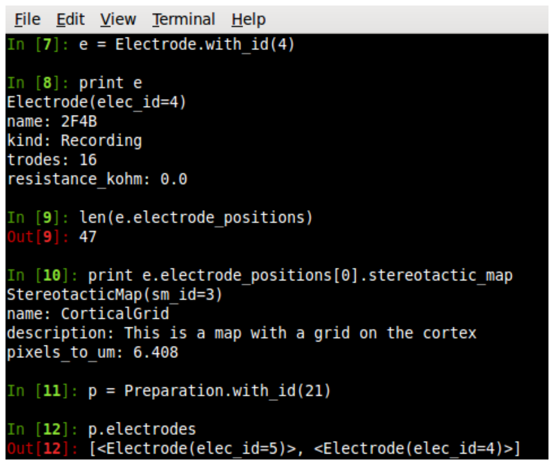


Figure 1.13: How the TEP database is organized. Each box represents a table in the database. Relationships between tables are shown as lines. The symbol on the ends of the lines tells you what kind of relationship it is. The heading colors distinguish the stages discussed in section 1.2.1.





```
In [7]: e = Electrode.with_id(4)

In [8]: print e
Electrode(elec_id=4)
name: 2F4B
kind: Recording
trodes: 16
resistance_kohm: 0.0

In [9]: len(e.electrode_positions)
Out[9]: 47

In [10]: print e.electrode_positions[0].stereotactic_map
StereotacticMap(sm_id=3)
name: CorticalGrid
description: This is a map with a grid on the cortex
pixels_to_um: 6.408

In [11]: p = Preparation.with_id(21)

In [12]: p.electrodes
Out[12]: [<Electrode(elec_id=5)>, <Electrode(elec_id=4)>]
```

Figure 1.14: An illustrative example of how the `Preparation` section of the database is organized and accessed. The names of the tables correspond exactly to the names of the Python objects imported on line 11 of Figure 1.12. The terminal inset (lower-right) shows a sample iPython session started with the command `read_only_database_shortcut.py`. Notice on line 10 how the `ElectrodePosition` object is accessed via the variable `electrode_positions` (pleural) reflecting that the relationship between `Electrodes` and `ElectrodePositions` is one-to-many. That is, each `Electrode` can have multiple positions, but for each `ElectrodePosition` there is only one `Electrode`.

Analysis code library

The TEP repository also houses a number of libraries for writing analysis code in Python. The directory `plotting` contains libraries that are rather TEP specific and help with things such as plotting current-source-density and maps showing electrode positions. The directory `utils` is for more general-purpose analysis libraries. There are libraries for doing current-source-density analysis, filtering, plotting, and much more. There are far too many libraries to describe them all here, you should just browse around and become familiar with what is available. There's nothing worse than spending a few days writing code only to find out it has already been written for you.

Where to put your code

Keeping a large project like TEP organized is a constant challenge. The subversion repository is a great tool to help us but we also have to be disciplined with regards to how we organize our code. Here are a few guidelines:

- The most general-use libraries should go in `tep/utils`. These libraries should be further subdivided by category.
- TEP specific but database-agnostic plotting libraries go in `tep/plotting`.
- Database-specific scripts go in a directory related to the database. There are currently two such directories `tep_382` and `tep_654`, but as additional databases are added (as more recording rigs are brought up) you should add new directories. Inside these directories, take care to keep things organized so that your analyses can be replicated in the future. These directories are also excellent sources of analysis script examples that should help those who are unfamiliar with how to write analysis scripts.

1.3 Architecture

There are many parts to the TEP system, so here we describe how it was constructed and give some instruction as to how it can be extended. Much of the complexity of the TEP system is hidden from the typical user, but will be important to those who will maintain and extend it. This complexity is mostly a result of the desire for flexibility from the perspective of the end-user. As you make a system both easy-to-use and flexible, the back-end tends to become more complex.

Figure 1.15 shows an overview of how the TEP system is designed. In this section we will describe the "database" and how access is mediated through SQLAlchemy. In addition, we will show how the stimulus system works and how to write new stimuli.

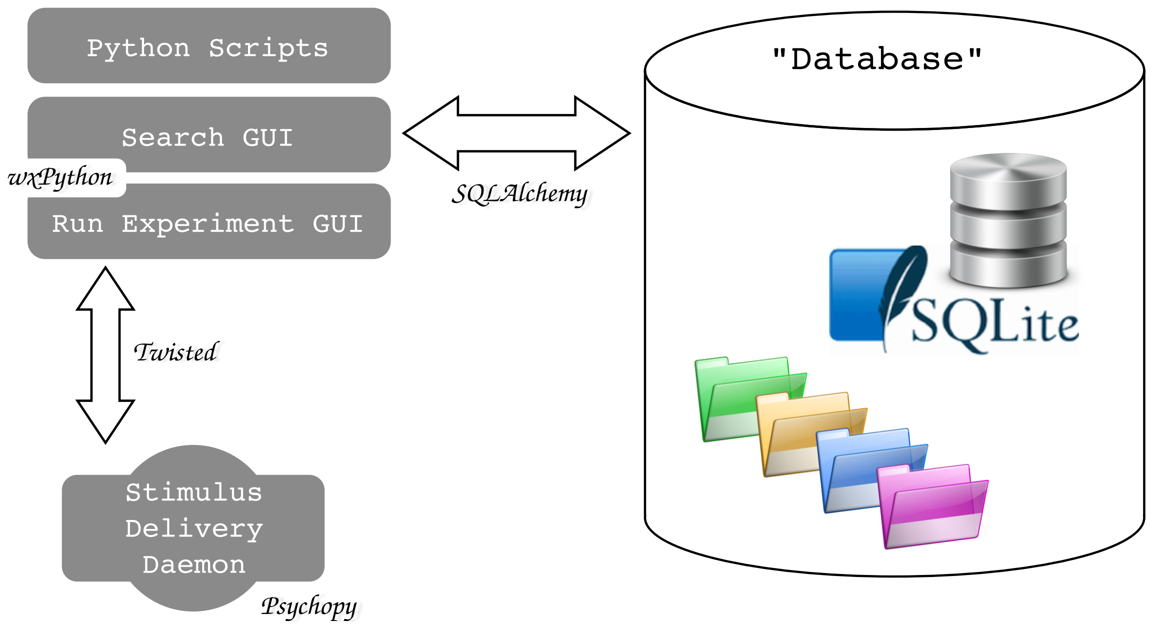


Figure 1.15: An overview of the TEP architecture. The "database" consists of an SQLite relational database that holds metadata and the raw-data files that are held in system-managed directories. The end-user accesses this "database" in one of three ways: by writing Python scripts, by using the Search GUI or by running the Run Experiment GUI. All interactions with the database are mediated by an Object Relational Manager called SQLAlchemy. The Stimulus Delivery Daemon is responsible for drawing stimuli to the screen (using psychopy) and it communicates with the Run Experiment GUI via the twisted networking library. The GUI programs are written using wxPython, a cross-platform GUI framework.

1.3.1 Stimulus delivery system

Stimuli in TEP are written in Python using the open-source psychopy library. This library is an abstraction layer above two other open-source Python libraries named pyglet and pygame. These libraries are themselves built on top of OpenGL which provides hardware level graphics optimizations. This means that we should be able to display graphics at very high throughput without dropping frames.

In order to ensure that we get the best performance, the stimuli are delivered via a dedicated daemon process that is started when the researcher starts an experiment. This means that even if the `run_experiment.py` GUI is demanding processing, the stimulus will draw without interruption. The same result could have been achieved by using threads or multiprocessing, however, the daemon approach is more flexible. If needed, for performance, one could have a dedicated stimulus computer that runs this daemon process.

The main program, `run_experiment.py`, communicates with the stimulus delivery daemon using the open-source Python networking library called twisted. Admittedly, twisted is perhaps overkill for this purpose, since twisted is a very full-featured networking library. The use of twisted, however, makes it possible to have a separate dedicated stimulus computer if you really need high performance.

Delivering a stimulus

Once the `run_experiment.py` program has gathered all the settings from the user and the user has chosen to run the stimulus, the program packages up all these settings as well as the `EyeCup` (see below) and sends the information to the stimulus delivery daemon. The daemon, receives the information, looks up the stimulus, and begins to prepare the stimulus for delivery. After the OpenGL window is created, the stimulus settings are set, and then the `setup()` function of the stimulus is run. The job of this function is to prepare all the textures and pre-calculate (if possible) the positions, sizes, and rotations of the textures that will be shown during the stimulus. This ensures that as little processing goes on during stimulus delivery as possible.

After preparation, the stimulus delivery daemon will start a process that waits for the trigger pulse. As soon as the pulse is delivered the stimulus will begin its drawing loop. During the drawing loop the two functions `evolve()` and `draw()` of the stimulus will be called until the `evolve()` function returns `False`.

As soon as the stimulus has completed, the recorded `frame_times` and the `estimated_epochs` are written to files and saved in `tep/acquire/tmp`. If the stimulus was delivered using the `run_experiment.py` program, then these files as well as the data files are moved to a system-controlled directory structure. The `frame_times` are saved as the stimulus is being delivered so that the stimulus can be recreated later. It is simply too computationally expensive to save the frames as they are drawn, so the `frame_times` are saved instead. With the `frame_times` and the stimulus settings, we can recreate the stimulus at a later time, when we are not worried about spending cpu cycles. The `estimated_epochs` are a description of the times when major events occurred in a stimulus such as when a dot appeared or started moving. These `estimated_epochs` along with the `frame_times` allow the system to show you when important events occurred on a plot of the data. (see Figure 1.11)

Eye coordinate system

Since stimuli are drawn to a computer monitor and then projected through a lens and mirror system down onto the hemisected eye-cup of the turtle, we need to ensure that the stimuli will look the same no matter how the eye-cup is arranged in the recording chamber. To do this the researcher running an experiment uses a program to setup the eye coordinate system. With this system stimuli are then translated, rotated, and scaled properly.

In the software, the object responsible for maintaining the information about this coordinate system is the `EyeCup` or the `DrawableEyeCup`. The former is an SQLAlchemy object whereas the latter is used by the stimulus delivery system. Although they have a very similar interface, care must be taken to use the appropriate one depending on where it is being used. Luckily, you shouldn't have to create these manually, since the TEP system handles all of that for you.

Creating a new stimulus

Stimuli in the TEP system are each contained in a single python file in the `tep/acquire/stimuli` directory. Within the `.py` file you must define a class that inherits from `VisualStimulus` (see Figure 1.16). The files `stimulus_template.py` and `another_stimulus_template.py` that are located in `tep/acquire/stimuli/utils` are useful as a starting point to create a new stimulus. In addition, the existing stimuli serve as examples of how to construct new stimuli.

1.3.2 Data management

To keep TEP as flexible as possible, the raw-data are stored as regular files in a software managed directory structure, instead of being stored in the SQLite database itself. This makes it so that the actual format of the

```
1   from psychopy import visual
2   import numpy
3
4   from tep.acquire.stimuli.utils.visual_stimulus import (VisualStimulus,
5           StimulusSetting)
6
7   class SomeNewStimulus(VisualStimulus):
8       description = 'Is shown in the run_experiment GUI'
9       some_scalar_setting = StimulusSetting(default=0.0,
10              description='Some kind of description, usually ends with (units).')
11      some_list_setting = StimulusSetting(default=[0, 0],
12              description='Description must end with [list]. [list]')
13
14      def runtime(self, eye_cup):
15          # Returns the amount of time (in seconds) the stimulus will run.
16          # You can access self.settings to calculate this value.
17
18      def estimated_epochs(self, eye_cup):
19          # Returns a list of (start_time, end_time, string_description) tuples that
20          # describe the important events or epochs of a stimulus.
21          # You can access self.settings to calculate this.
22
23      def setup(self, window, refresh_rate, eye_cup):
24          # Prepares textures and everything else needed for the evolve() and draw() loop.
25          # Here window is a psychopy.visual Window object, refresh_rate is a float, and eye_cup is a
26          # tep.acquire.stimulus.utils DrawableEyeCup object.
27
28      def evolve(self, loop=False):
29          # Prepares everything for the next call of the draw() function.
30          # Usually just moving or resizing textures.
31          # Usually returns True, when it returns False, the stimulus will end.
32
33      def draw(self):
34          # Simply draws the stimulus to the screen.
35          # Usually just calling the .draw() function of all the textures.
36
```

Figure 1.16: Creating a new stimulus. This template shows all the major parts needed to create an new stimulus. **(line 7)** A class must be defined (the name of the class will be the name of the stimulus) and inherit from `VisualStimulus`. **(lines 9 and 11)** `StimulusSettings` are created as class-variables an will then become accessible through `self.settings` to be used throughout the stimulus. **(lines 14-33)** The member functions that need to be defined in order for the stimulus to function.

data does not matter to the TEP system, so if the data format changes it will not be a problem. The SQLite database holds only metadata, that is data about the data, which includes where in the directory structure to find the raw-data files.

The SQLAlchemy objects that have raw-data associated with them are `SearchingRecordings` and `VisualRecordings`. Both of these classes inherit from `DataCarrier` which provides an interface to the data through variables. The variable `data_path` is agnostic to the format of the data. Many of the other variables assume that the data can be opened with `tep.util.file_io.read_data_file` such as `voltage_traces_mv` and `sampling_freq_hz`. Other variables query the database and are provided for convenience like `electrode_depths` and `recording_electrode_names`.

The database manager

The alchemy database manager, adbm, is responsible for setting up the connection to the database and provides access to the database session so you can perform queries and manipulate the database. To perform a query you can use the `adbm.query()` function, as explained in the SQLAlchemy documentation (http://docs.sqlalchemy.org/en/latest/orm/query.html). The so called `SomeMappedClass` in the documentation refers to any of the classes in `tep.database.alchemy_classes`. The SQLAlchemy session is `adbm.session` in case you need it to `add()` or `delete()` entities. When you are done manipulating the database, you will have to call `adbm.flush()` to make the changes permanent.

Chapter 2

Response Properties of Visual Neurons in the Turtle Nucleus Isthmi

Debajit Saha, David Morton, Michael Ariel, and Ralf Wessel

Published in Journal of Comparative Physiology A

October 22, 2010

The optic tectum holds a central position in the tectofugal pathway of the non-mammalian species and is reciprocally connected with the nucleus isthmi. Here, we recorded from individual nucleus isthmi pars parvocellularis (Ipc) neurons in the turtle eye-attached whole-brain preparation in response to a range of computer-generated visual stimuli. Ipc neurons responded to a variety of moving or flashing stimuli as long as those stimuli were small. When mapped with a moving spot, the excitatory receptive field was of circular Gaussian shape with an average half-width of less than 3 deg. We found no evidence for directional sensitivity. For moving spots of varying sizes, the measured Ipc response-size profile was reproduced by the linear Difference-of-Gaussian model, which is consistent with the superposition of a narrow excitatory center and an inhibitory surround. Intracellular Ipc recordings revealed a strong inhibitory connection from the nucleus isthmi pars magnocellularis (Imc), which has the anatomical feature to provide a broad inhibitory projection. The recorded Ipc response properties, together with the modulatory role of the Ipc in tectal visual processing, suggest that the columns of Ipc axon terminals in turtle optic tectum bias tectal visual responses to small dark changing features

in visual scenes.

2.1 Introduction

The nucleus isthmi (parabigeminal nucleus) is a visually responsive midbrain structure in vertebrates [6, 7, 12, 23, 26, 33, 34, 47, 48, 57, 59, 61] that influences visual processing by direct modulation of tectal circuits [10, 28, 52, 53, 56, 60]. This modulation is mediated by reciprocal connections between the nucleus isthmi and optic tectum [16, 17, 36, 45, 55, 58].

The turtle magnocellular isthmic complex contains two cytoarchitectonically distinct isthmic nuclei (Figure 2.1), which receive sensory information from the ipsilateral optic tectum [36, 45]. In order to make comparisons with avian systems we have adopted the nomenclature consistent with the avian isthmotectal system [55, 58] following the example of Powers and Reiner [36]. When we refer to the two isthmic nuclei as the pars parvocellularis (Ipc) and the pars magnocellularis (Imc), these correspond to the caudal Imc and rostral Imc, respectively from earlier turtle literature [45]. Retinal ganglion cell (RGC) axons terminate in the superficial layers of the tectum (Figure 2.1) where they innervate the radial, narrow dendrites of the tectal stratum griseum periventriculare (SGP) neurons [44]. The axons of SGP neurons project to the Ipc and Imc nuclei [21]. In the Imc, neurons have large, sparsely branched dendritic fields overlapped by local axon collaterals. The axons of Imc neurons nontopographically project to both the deeper layers of the tectum and to the Ipc [45]. In the Ipc, neurons have medium-sized, elongated somata, flattened bipolar dendritic fields, and axons that project topographically back to the tectum. Each Ipc axon terminates as a compact swarm of boutons within a cylinder about 150 m in diameter and 400 m tall, placed mainly in the upper tectal layers, where it spans less than one percent of the tectal surface [45]. The topographically-organized, columnar Ipc axon terminals spatially overlap with the RGC axons and the SGP neuron dendrites (Figure 2.1).

Cholinergic isthmic neurons (turtle: [8, 36] frog: [11, 17, 60] bird: [30, 51] mammal: [16, 19, 49]) provide a major source of acetylcholine (ACh) to the optic tectum (superior colliculus). Although the cholinergic neurons are hypothesized to be modulators of tectal activity [14], the information represented in the spatiotemporal pattern of ACh release is largely unknown. In general, the extensive studies of cholinergic

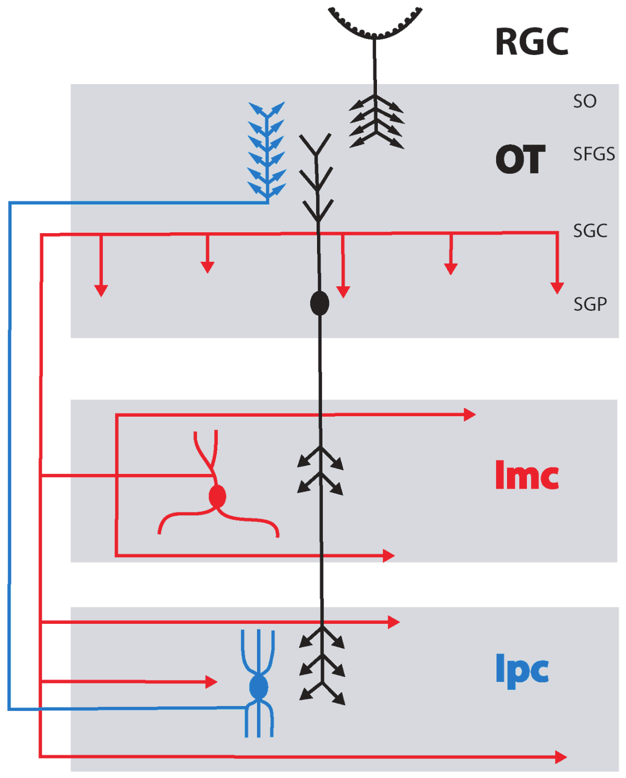


Figure 2.1: Schematic drawing of the turtle isthmotectal feedback circuitry. Neurons in the optic tectum (OT) with somata in the SGP layer receive RGC inputs at their narrow apical dendrites in the SFGS layer. These glutamatergic SGP neurons (black) project to the two isthmic nuclei: pars parvocellularis (Ipc) and pars magnocellularis (Imc). The GABAergic Imc neurons (red) have large, sparsely branched dendritic fields, and axons that project nontopographically to the Ipc and to the deeper tectal layers (SGC, SGP) in addition to local axon collaterals. The cholinergic Ipc neurons (blue) have flattened bipolar dendrites and project topographically to the upper layers of ipsilateral tectum. We use the revised nomenclature, Ipc and Imc, for isthmic nuclei, which is consistent with the nomenclature in the avian isthmotectal system

modulation in the brain have focused largely on the postsynaptic effect of ACh release [24, 25, 29, 31]. Cholinergic Ipc neurons, being visually responsive and reciprocally connected exclusively to the tectum, are an ideal system to study both spatiotemporal pattern of ACh release and its postsynaptic effect on tectal neurons. While this study does not address the postsynaptic effects of ACh in the optic tectum, an understanding of the visual responses of Ipc neurons is a necessary first step in the direction of correlating visual stimulation with spatiotemporal ACh release.

To investigate the visual response properties of the Ipc, we conducted extracellular recordings from Ipc neurons during visual stimulation to the contralateral retina [**?**] in a turtle eye-attached whole-brain preparation [20, 40]. Apart from preserving the long range visual circuitry, this whole-brain preparation mitigates the primary in vivo complications such as heart beat vibrations and the lack of control of the extracellular media. We presented a wide variety of visual stimuli (see METHODS) and found that the Ipc responds strongly to small stimuli preferably dark compared to the background. Ipc responses to static stimuli show significant temporal adaptation. For dynamic stimuli, the Ipc responses are tuned to the stimulus size with no directional preference.

2.2 Methods

2.2.1 Surgery

18 adult red-ear turtles (Trachemys scripta elegans, 12-15 cm carapace length) were used in this study. Procedures used in this study were approved by the Washington University Institutional Animal Care and Use Committee and conform to the guidelines of the National Institutes of Health on the Care and Use of Laboratory Animals. Following cryanesthesia ($> 15\ \mathrm{min}$ of hypothermia in ice water), rapid decapitation and then cannulation was performed on neck vasculature to infuse cold saline to rinse out cranial blood. Within 20 minutes of decapitation, the telencephalon was removed. The eye was kept attached to the brainstem but freed from its orbits by cutting the conjunctiva and extraocular muscles. The eye was then hemisected and drained of its vitreous. The dura surrounding the brain was removed, as was the pia covering lateral mesencephalon. The preparation was then transferred to the superfusion chamber positioned on an air table with

the eye-cup beneath a focusing lens. The preparation was bathed in physiological media (in μM; 85 NaCl, 2 KCl, 2 $MgCl_2$, 45 $NaHCO_3$, 20 D-glucose, 3 $CaCl_2$ bubbled with 95% O_2 and 5% CO_2), adjusted to pH 7.4 at room temperature.

2.2.2 Extracellular recordings

Extracellular recordings from Ipc neurons were achieved with tungsten microelectrodes (A-M Systems, parylene-C insulated, 250 μm core diameter, 12 $^\circ$ tapered tip) of 1 $M\Omega$ impedance (measured at 1 kHz, 0.1 nA p-p current). Since the Ipc is a surface structure, its location was identified by a characteristic protrusion just caudal of the optic tectum (Figure 2.2). To find a responsive recording site, the microelectrode was advanced smoothly into the Ipc (100 - 400 μm), using a hydraulic drive (FHC 50-12-9 Manual Drum Drive Unit) while the search stimulus (see below) was focused on the retina of the contralateral eye-cup. The voltage traces were obtained using a differential AC amplifier (A-M Systems, Model 1800). The signal was passed through a 300 - 5000 Hz analog band pass filter and then sampled at 10 kHz (PCI-MIO-16E-4) using a LabView-controlled data acquisition system (National Instruments). Recordings were continuously monitored using an audio monitor (AM Systems 3300) and an oscilloscope (Tektronix TDS 210).

2.2.3 Visual stimulation

Visual stimuli were created by a computer and delivered with an LCD monitor (Samsung 19”, 1440x900 pixels, contrast ratio = 20000:1, response time = 2 ms). The image on the monitor was projected onto the retinal surface of the hemisected eye-cup with a converging lens system (Figure 2.2). A monitor pixel corresponded to 7 μm on the retina or $\sim 0.08^\circ$ of visual angle [2, 34]. Stimuli were created using psychopy, an open-source psychophysics module written for the Python programming language [35].

Eye-cup layout

Software tools were written to allow the experimenter to characterize the projection of the computer screen onto the retina. A computer game pad was used to control the measurements of size and position of the eye-cup interactively, relative to the monitor. The eye-cup parameters, including the position and orientation

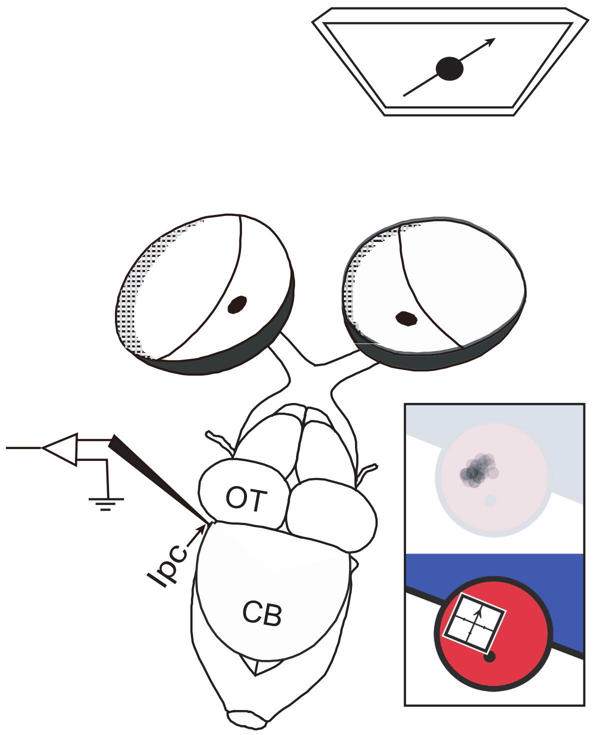


Figure 2.2: Schematic drawing of a dorsal view of the turtle whole-brain preparation with the eye-cups attached and the telencephalon removed. Computer-generated visual stimuli are presented on a monitor (moving black spot) and are projected with appropriate optics (not shown) onto the retinal eye-cup. Surface landmarks guide the placement of microelectrodes for extracellular and blind-patch whole-cell recordings from Ipc. Inset The monitor-to-eye-cup projection and the presentation of visual stimuli. Top, evaluating a responsive area in the visual field with small moving spot stimuli. Bottom, defining a stimulation coordinate system (white square) covering the entire responsive area.

of the visual streak, and the size and position of the optic disk were documented. An image based on these parameters (Figure 2.2, bottom inset) was projected onto the eye cup preparation (in the recording chamber) allowing for interactive adjustments. The visual streak represents a high density of photoreceptors and RGCs in the turtle retina [5]. It is identified by visual inspection as a faint white linear structure along the nasal-temporal direction of the eye.

Search stimulus

The search stimulus consisted of 5 - 8 black spots of different sizes (diameter 2 - 8°) moving pseudo-randomly on a white background with different directions and speeds (2 - 5°/s). The spots moved in a straight line until they reached the edge of the eye cup region whereupon they randomly changed direction but not speed or size. Unless specified elsewhere, the default computer screen background was white in a darkened room.

Stimulation coordinate system

To define the stimulation coordinate system, we monitored neural activity while a black spot moved on the screen, starting outside the eye cup region. The movement and size of the spot was controlled by a computer mouse. As the spot moved towards the eye cup center, single/multi unit responses were audibly detected at some locations with the use of an audio monitor. The spot locations and sizes were saved at the onset of a response by clicking the mouse. After multiple such identifications of responses, an image of all the response-triggering spots overlaid on each other was generated (Figure 2.2, top inset). This image was used as a guide to define a stimulation coordinate system that was about twice the size of the active area and oriented parallel to the visual streak (white square in Figure 2.2, bottom inset). All experimental stimuli were subsequently scaled and rotated using this normalized stimulation coordinate system, giving us the ability to generate a whole set of stimuli which can be presented independently of the particulars of each monitor-to-eye-cup projection setup. The nasal retina or temporal visual field parallel to the visual streak was defined 0° (180°) in the right (left) eye. The superior and inferior visual fields were defined as 90° and 270°, respectively for both eyes.

Visual stimuli

Various stimuli, both static and moving were shown from a pool of computer-generated stimuli. During an experiment, stimuli were shown time-locked with the data acquisition. The data acquisition system sent a 5-V trigger pulse to the stimulus computer's parallel port, triggering the display of the stimulus. With the video card (ASUS EN8600GT, 256 MB), the psychopy codes maintained sub-millisecond frame-rendering precision. Interstimulus intervals of at least 30 s were maintained between different sets of stimuli.

To study the Ipc visual response properties to different stimuli, 7 different types of stimuli were used. S1) Black moving spot on white background A small spot of diameter 6 - 8° moved at about 6 - 8°/s radially through the center of the stimulation coordinates in 8 directions (0, 45, , 315 degrees relative to the visual streak) pseudo randomly. There was $\geq$ 500 ms wait between different directions being presented within the stimulus set. S2) White moving spot on black background This set of stimuli was identical to stimulus S1 except that the contrast was inverted. S3) Black leading edge on white background A bar moved along its long axis so that only the leading edge crossed the screen. The short axis was the same size as the diameter of spots in stimuli S1 and S2, likewise for the speed. S4) Randomly changing check pattern The stimulation coordinate system was divided into a 4x4 grid, and each grid space pseudo-randomly displayed white or black with equal probability. Each grid space (check) was about 8° square. All 16 checks were updated every 10 frames (168 ms). This stimulus was displayed for 25 seconds. S5) Drifting sinusoidal grating A grating was displayed with motion perpendicular to the visual streak. The spatial frequency was 0.02 cycles/deg, and the temporal frequency was 1 Hz. The stimulus was displayed for 25 seconds. S6) Flashing black spot on white background A small spot of diameter 6 - 8° was displayed for 5 seconds in the center of the stimulation coordinate system. S7) Whole screen flash from white to black The entire screen was set to display black for 5 seconds. Note that except for S2, the computer screen background was white before and after the trials of stimuli.

2.2.4 Extracellular data analysis

To determine the number of active units at the recording site, sample recordings were analyzed with waveClus, a spike-sorting algorithm based on spike detection and sorting using wavelets and superparamagnetic clustering [37]. Single units were isolated using criteria that included a well-defined spike shape with no kinks in the standard deviation of all classified spikes and a refractory period of 2 ms in the interspike interval distribution. Only single-unit responses were used for the analysis. For each experiment we obtained recordings for at least 3 trials, with exceptions noted in the figure captions.

Difference-of-Gaussians analysis

The stimulus size-response profile of Ipc neurons were analyzed using the Difference-of-Gaussians (DOG) model [13, 39, 43], which assumes the linear and independent sum of two concentric Gaussians, excitatory and inhibitory, yielding the net response to the stimulus. We used moving spots (stimulus S1) of different sizes to study the stimulus size tuning of Ipc neurons. The assumption was made that only the spots leading edge is generating the Ipc response. With this assumption, for a linear sweep of a moving spot of radius s, the net response R(s) follows the relation,

$$R(s) = R_0 + K_e \int_{-a/2}^{a/2} e^{-(2x/a)^2} dx \int_{-s}^{s} e^{-(2y/a)^2} dy - K_i \int_{-b/2}^{b/2} e^{-(2x/b)^2} dx \int_{-s}^{s} e^{-(2y/b)^2} dy$$

where the parameters K_e and K_i represent the strength of excitatory and inhibitory receptive fields, respectively, while a and b represent the spatial extent of excitatory and inhibitory Gaussian receptive fields. The baseline activity R_0 is assumed to be zero due to low ($>$ 1 Hz) spontaneous activity of Ipc units. Notice that the equation is equivalent to one commonly used for flashing spot stimuli of different radius with an additional constant multiplicative factor due to the integration of the whole Gaussian profile along one direction.

2.2.5 Blind-patch whole-cell recording

For blind-patch whole-cell recordings from Ipc neurons, patch pipettes were fabricated on a Flaming-Brown horizontal puller (P-97, Sutter Instruments) from Corning #7052 glass capillary tubing (OD 1.5 mm, ID 0.86 mm) to yield 5 to 8 $\mathrm{M\Omega}$ pipette resistance. The pipette solution for ruptured patch recordings contained (in mM) 117 $KMeSO_4$, 5 KCl, 5 NaCl, 1.2 $MgCl_2$, 10 HEPES, 5 EGTA; pH 7.3; with osmolarity 260 mOsMol. Pipette solutions were filtered before use. Electrodes were advanced through the Ipc with a motorized micromanipulator (MP-285, Sutter Instruments) while maintaining a constant positive pressure. Electrode resistance was monitored continuously by applying short current pulses (Axoclamp 900 amplifier; -1.0 nA, 10 ms duration, 10 Hz). An increase of the resulting voltage pulse height (an increase in series resistance) was observed as the pipette tip approached a cell membrane. The pipette was advanced until the series resistance increased by about 10 MΩ. Current was then reduced to -0.1 nA and the positive pressure was reversed to a small negative pressure by pulling gently on a syringe plunger. This gentle suction was applied until the pipette formed a GΩ seal. Additional brief suction was applied to rupture the patch of membrane within the pipette. The pipette capacitance was compensated by the amplifier's circuitry and the access resistance was monitored. Analog data were low-pass filtered (4-pole Butterworth) at 1 kHz, digitized at 10 kHz, stored and analyzed on a PC equipped with a PCI-MIO-16E-4 and LabView software.

For stimulation of presynaptic Imc axons, a concentric, bipolar tungsten electrode (tip diameter = 4 μm; core diameter = 76 μm) was lowered into deep tectal layers (Figure 2.1, 2.10). A 200 μA current pulse of 0.5 ms duration was used to stimulate Imc axons in deep tectal layers while recording from the Ipc neuron intracellularly. No visual stimuli were applied during intracellular recordings.

2.3 Results

2.3.1 Stimulus selectivity

To evaluate to what extent Ipc visual response properties are stimulus specific, we conducted extracellular single-unit recordings. We recorded while showing different stimuli in a pseudo random order. Stimuli were shown to the contralateral eye of the recorded Ipc neuron. A continuously illuminated white screen elicited almost no response (mean = 0.1 Hz, SD = 0.2 Hz, n = 8 units). However, a dark spot (diameter 6 8°) on a white screen moving through the center of the excitatory receptive field of an Ipc neuron at a speed of 6 8°/s elicited strong spiking responses (Figure 2.3a and 2.4; spike count 23 - 72, n = 5 units).

For all stimuli and Ipc units tested, responses are quantified as the average spike count during the first 5 s of stimulus presentation (Figure 2.4). The Ipc response was somewhat reduced when the dark spot was replaced by a moving leading edge of similar width and speed (spike count 17 - 52, n = 4 units) or when

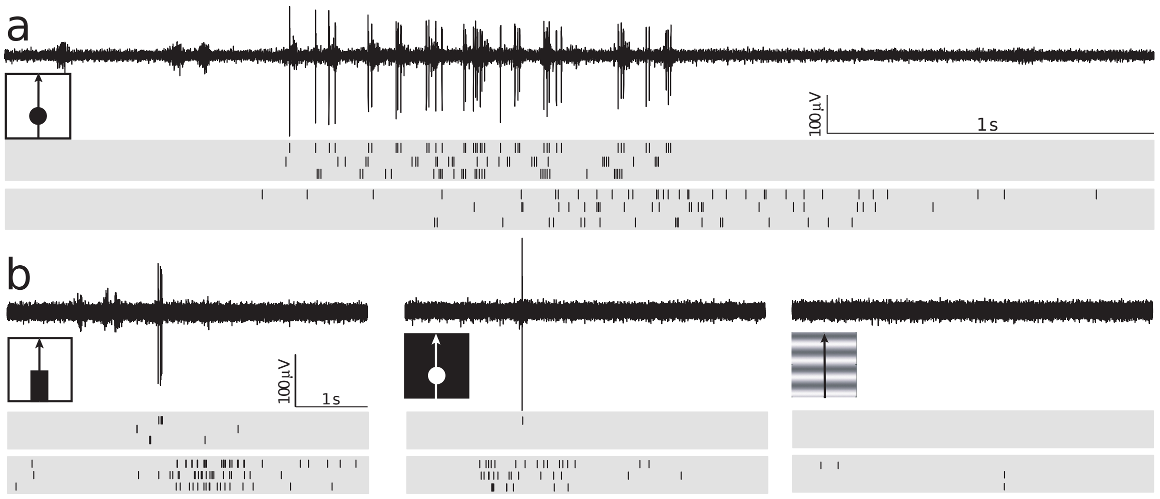


Figure 2.3: Stimulus-selective visual responses of Ipc neurons. **a** Responses of two single-unit Ipc neurons to a black spot moving perpendicular to the visual streak. Schematic of the stimulus is shown at the left of the respective voltage trace. The square box represents the stimulation coordinate system, which covers the entire responsive area of the unit and is aligned parallel to the visual streak. The direction of stimulus movement is indicated by the arrow. Top to bottom, raw voltage trace of one unit, raster of three trials of that unit and raster for another unit. The gray boxes indicate the duration of stimulus movement. Spot diameter: 7°, 8°; speed: 7°/s, 8°/s respectively, for two units. **b** Responses to other moving stimuli for the same units shown in (a). From left to right, moving leading edge, moving white spot (same dimension and speed as of black spot in 3a) and drifting sine grating (spatial frequency 0.02 cycles/deg, temporal frequency 1 Hz).

a white spot moved on a dark background (spike count 2 - 31, n = 5 units) (Figure 2.3b and 2.4). Among the moving stimuli we tested, the whole-field sinusoidal drifting grating (stimulus S5) elicited the weakest response (spike count 0 - 5, n = 5 units; Figure 2.3b and 2.4).

For stationary stimuli, Ipc neurons responded to a whole-field random checkerboard stimulus (spike count 1 - 29, n = 5; Figure 2.4) and to a small dark spot flashed at the center of the receptive field (spike count 2 - 50, n = 5 units; Figure 2.5). Ipc neurons responded little to whole-field diffuse illumination changes from white to black (spike count 0 - 5, n = 5; data not shown). In conclusion, Ipc neurons respond to a variety of moving or flashing stimuli as long as they are small.

2.3.2 Excitatory receptive field (ERF) structure

We used the results obtained with the small moving black spot (stimulus S1) to elucidate the excitatory receptive field structure of Ipc neurons. An analysis was developed to measure the gross structure of the ERF even when the location of the ERF center was only estimated during stimulus presentation. Single-unit Ipc responses were collected in response to a small black moving spot (stimulus S1). For each direction,

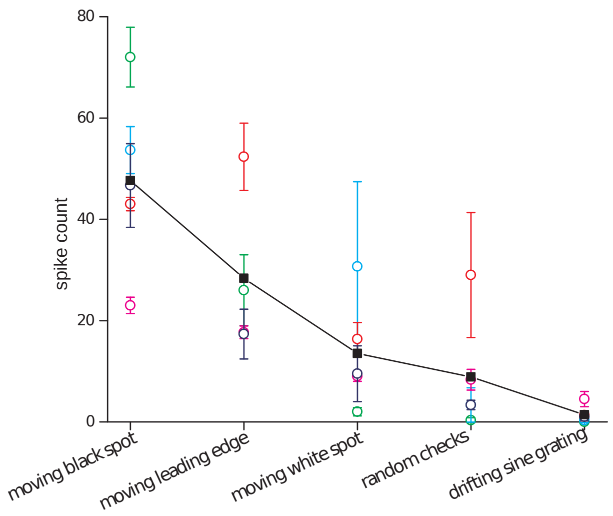


Figure 2.4: Population study of single-unit Ipc responses to spatiotemporally changing stimuli. Presented are 5 single-units, collected from 5 sites of 4 different brains. Responses for Ipc neurons to different stimuli are quantified by the average spike count, which is the number of spikes for the first 5 s after stimulus onset and averaged over multiple trials. Small moving stimuli; black spot, black leading edge and white spot are presented in the stimulus coordinate system of each unit (S1, S3, S2 respectively, see METHODS). Spot diameter, edge width and speed are kept constant for one unit but varied slightly from unit to unit (spot diameter/edge width 6 - 8°, speed 6 - 8°/s). For each unit the maximum value of average spike count along any one direction (out of 8) is plotted for each small moving stimulus. The cyan unit was not tested for leading edge stimuli. Error bars represent SD for multiple trials (m = 2 for white moving spot stimulus of blue and magenta units, m = 3 for all others). For broad spatiotemporally changing stimuli; random flashing checkerboard and drifting sine grating (S4, S5), the average spike counts are plotted with the SD values. The black squares indicate the average response of all single-units for any given stimulus. The green and red circles indicate the Ipc units shown in Figure 2.3.

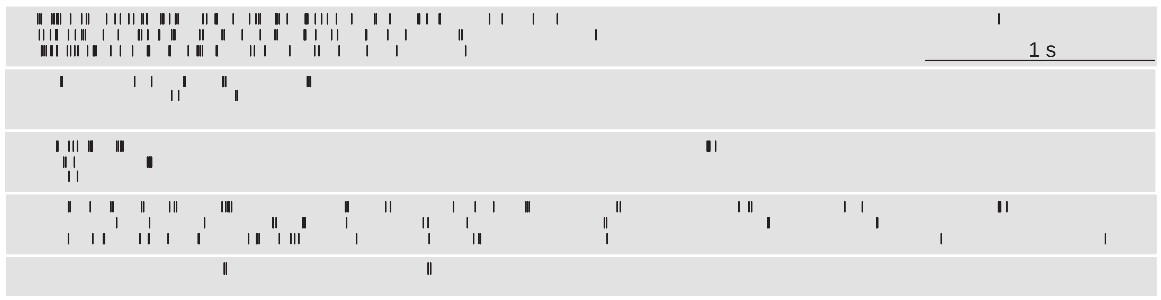


Figure 2.5: Adaptation of single-unit Ipc responses to a stationary flash (stimulus S6, see METHODS). Spike raster (m = 3, except bottom unit where m = 2) of 5 single-unit Ipc responses (same as in Figure 2.4) are shown. Flashing spot size kept constant for one unit but varied slightly (6 - 8°) from unit to unit. The gray boxes indicate the onset and duration of the stimuli.

peristimulus time histograms, PSTHs (bin 200 ms, smoothed by 400 ms window) were calculated. The delay due to response latency for each unit was adjusted by overlaying the PSTHs of opposite directions. This delay varied between 100 - 300 ms (n = 5 units), which was consistent with the response delay obtained from flashing small dark spots at the center of the receptive field of recorded units. The PSTHs were averaged over multiple trials and translated into 2D spatial coordinates in accordance with the stimulus. These data suggested a 2D Gaussian shape. By taking slices of a 2D Gaussian surface and comparing to the spike histograms, we were able to fit a 2D Gaussian to each unit. A genetic algorithm [9] was utilized to determine the best position, height, widths and orientation angle of the 2D Gaussian functions, by minimizing the squared difference between the 2D Gaussian function and the data (Figure 2.6, top panels, red: average rate; pink: SD; black: Gaussian fit). This method enabled us to reproduce the ERF structure (Figure 2.6, bottom panels) even if the spot missed the exact center (e.g. Figure 2.6a). For the 9 units analyzed, the half-widths (half-width 1: Avg. 2.4°, SD 0.6; half-width 2: Avg. 2.9°, SD 0.8) of the 2D Gaussian fits (Figure 2.6, bottom panels) did not differ significantly. This indicates that Ipc neurons have spatially restricted excitatory receptive fields with circular structure.

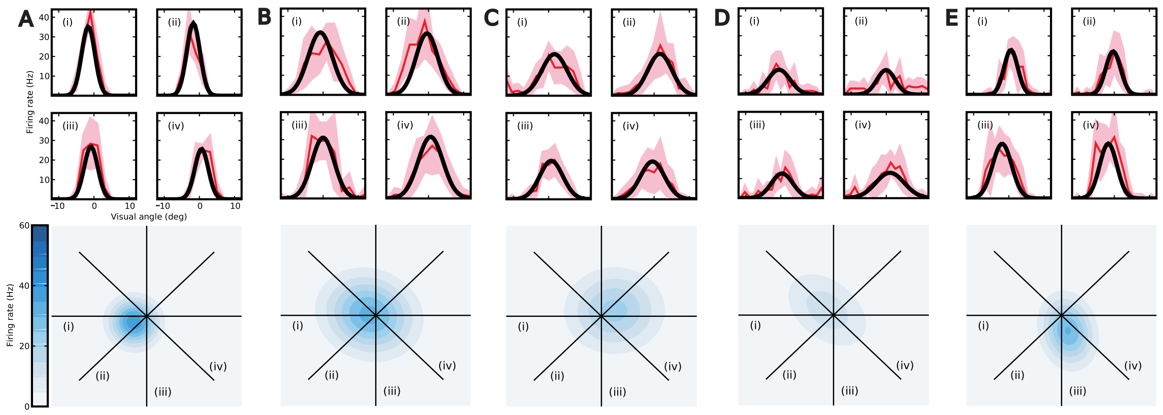


Figure 2.6: Two-dimensional shape of the excitatory receptive field, presented for 5 Ipc units (out of the nine analyzed). For each unit, top 4 panels show the average firing rate (red line) with SD (m = 3, pink) along 4 directions, indicated by Roman numbers. The black line in each panel represents the 2D Gaussian fitting along that direction (see METHODS). The bottom plot (blue) shows the location and size of the Gaussian fit (the contour plot windows are all 22° square). The scale bar represents the firing rate. All 5 units are shown in the same scale.

2.3.3 Stimulus-response profile

We studied responses of the Ipc neurons to different sizes of black moving spots (Figure 2.7a). For each spot size, the stimulus moved in 8 directions (stimulus S1) through the center of the stimulus coordinate system tracing the Ipc excitatory receptive field. The firing rates along all 8 directions were calculated by dividing the total spike count along one direction by half of the stimulus duration. To avoid any artifact due to missing the center of the Ipc ERF along any one direction, the maximum value of average firing rates among 8 directions was chosen to represent the corresponding spot size.

Spot size zero corresponded to the average response to the white screen. Single-unit Ipc responses (n = 4) followed a classic response-size profile that increased up to a certain spot size then asymptotically decreased to a certain value with increasing of spot size. For the 4 units tested, the Ipc neurons showed a strong spiking response for a spot of radius of about 3 - 5° of visual angle (Figure 2.7b). The response decreased considerably with increasing spot radius ($> 5^\circ$) for three out of four units (except the blue unit, Figure 2.7b). For the 4 units studied, the firing rate decrease with increasing spot size was independent of direction of stimulus movement. To quantify this response-size profile, we fitted the phenomenological Difference-of-Gaussian model with four independent variables (see METHODS) to these data (lines in Figure 2.7b). For 3 units, the half-widths of the fitted excitatory and inhibitory receptive fields varied between 1 - 2° and 3 - 5° respectively. This estimate of the excitatory receptive field half-width is consistent with the results obtained using the previous analysis (Figure 2.6). The unit with the largest ERF (5o; blue unit of Figure 2.7b) also had the weakest surround inhibition. Overall the excitatory receptive field turned out to be narrower and stronger than the overlapping wider and weaker inhibitory receptive field for all 4 units (Figure 2.7c).

2.3.4 Direction tuning

To investigate the potential directional sensitivity of Ipc neurons, a small black moving spot (stimulus S1) was used as it elicited the strongest firing in Ipc neurons. Directional preferences to movement along each radial axis were calculated using directional sensitivity index (DSI).

$$\text{DSI} = \frac{R_{\text{max}} - R_{\text{min}}}{R_{\text{max}} + R_{\text{min}}}$$

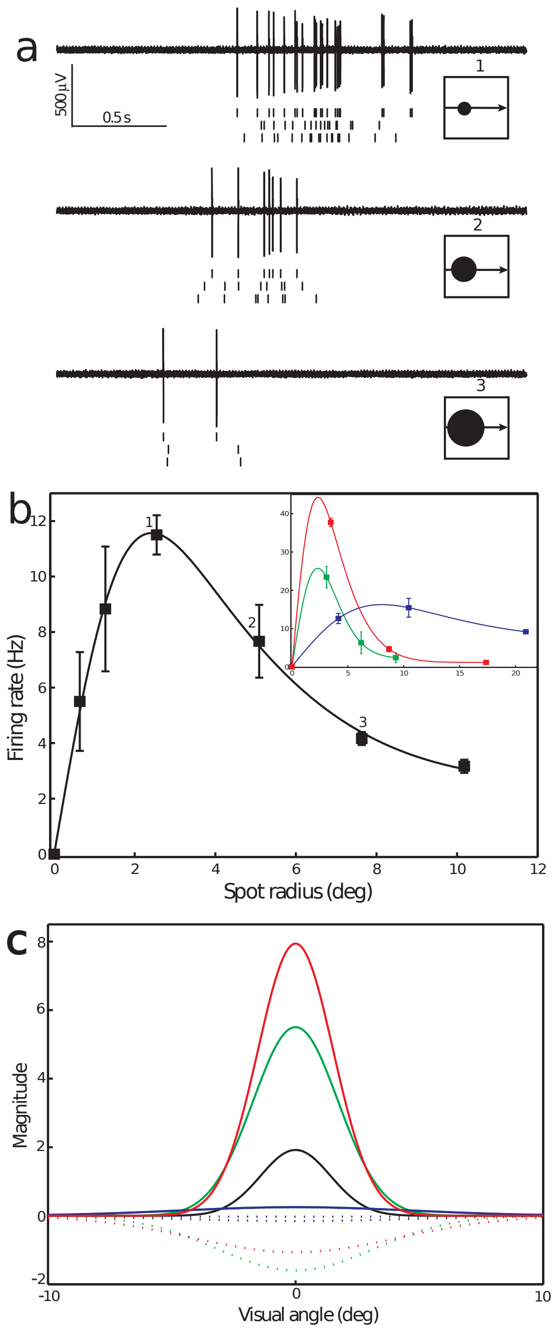


Figure 2.7: Ipc responses vary with the size of a small moving black stimuli. **a** The Ipc responses are shown for different spot radii of 2.5, 5 and 7.5° as the spot moves 9.5°/s along the visual streak through the estimated center of the Ipc excitatory receptive field. For each spot size, one raw voltage trace and 3 raster plots are shown. The spot size is scaled to the stimulus coordinate system (square box) in the drawn stimuli. **b** The response-size profile is shown for the same unit. A difference of Gaussian model is fitted to the result. Numbers indicate the average firing rate corresponding to spot radius shown in (a). **Inset** Population study of single-unit Ipc responses (3 units) are shown for varying spot radius and fitted with Difference of Gaussian model. Error bars indicate the standard deviation between multiple trials (m = 3). Speed kept constant for each unit but ranged from 5.5 to 9.5°/s. **c** The excitatory (solid) and inhibitory (dotted) Gaussians fitted to the response-size profiles of the units in (b) from the Difference of Gaussian model (same color-coded).

Where R_{max} and R_{min} are the higher and lower average firing rate, respectively recorded during the presentation of the stimulus (stimulus S1) along two opposite directions. By convention, units are classified as directionally sensitive, when $R_{\text{max}} \geq 2R_{\text{min}}$ (i.e. DSI $\geq$ 0.33) [41, 42]. Opposite directions (such as 0°, 180° etc) were evaluated together as directional pairs. Since the spot moved along 8 directions with 45° separations, each unit produced 4 directional pairs. The DSI was calculated for total of 36 directional pairs (n = 9 units, Figure 2.8). We found that the DSI of all but one of the pairs fell below the criterion of 0.33, and thus were not considered directionally sensitive. The absolute values of R_{max} and R_{min} were plotted for each unit. The distribution, though above the $R_{\text{max}} = R_{\text{min}}$ line by definition, did not exceed the DSI criterion line of $R_{\text{max}} = 2R_{\text{min}}$, indicating no directional tuning (Figure 2.8, inset).

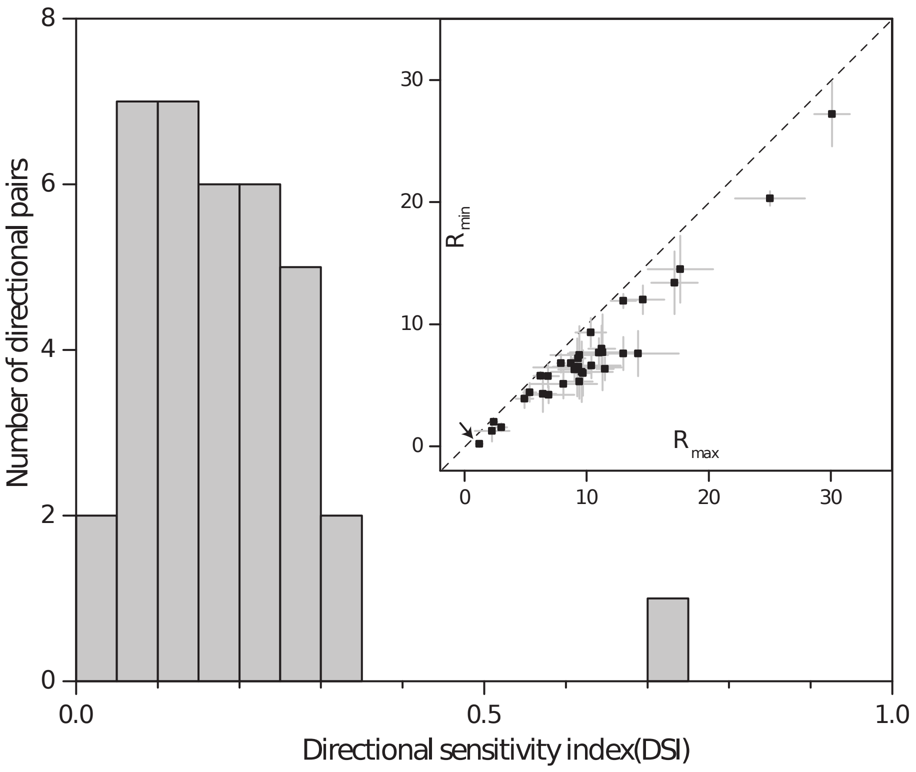


Figure 2.8: Histogram distribution of the directional sensitivity index (DSI, see METHODS) is plotted for 36 directional pairs from 9 single-unit Ipc neurons (9 recording sites, 8 brains). The spot diameter and speed are kept constant for multiple trials in each unit but varied slightly between different units (diameter 5 - 9°, speed 6 - 10°/s using stimulus S1). DSI index of all but one pair falls below 0.33. Inset R_{max} and R_{min} for all the directional pairs are plotted with their absolute value of firing rate (Hz) with SD (gray bars). R_{max} and R_{min} represent the average value of the higher and lower firing rates over multiple trials (n = 3) for each directional pair. The dotted line represents $R_{\text{max}} = R_{\text{min}}$. The arrow indicates the directional pair with lowest firing rate that has a DSI just above the criterion.

2.3.5 Speed tuning

The speed tuning of Ipc units was also investigated with a small moving spot (stimulus S1). A medium sized spot (8 - 9° of visual angle) that elicited strong spiking activity was moved with different speeds across the RF. For the 2 units studied, speeds up to a certain value elicited similar responses (8 - 15°/s in Figure 2.9a, 3-7°/s in Figure 2.9b), whereas the response decreased for higher speed (27°/s in Figure 2.9a, 17°/s in Figure 2.9b). The speed tuning curves of both the units show a peaked response profile (Figure 2.9c, d). For higher speeds, both units responded with a larger delay compared to the response latency for slower speeds. Speed analyses were carried out for all 8 different directions of motion for each unit. The results were independent of the direction of spot movement.

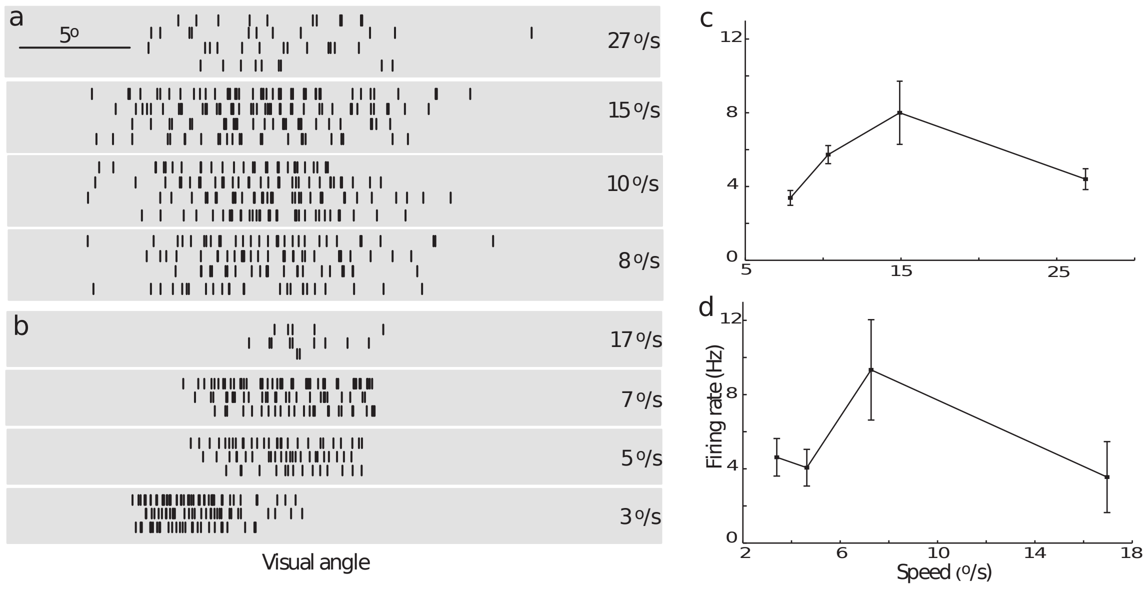


Figure 2.9: Variation of Ipc response with different speeds of small black moving stimuli (stimulus S1, see METHODS). Spikes are plotted at the position they occurred in the receptive field for different speeds with a response latency of 300 ms. Each gray box represents 30° of visual angle. Shown are each cell's strongest response among all 8 directions presented. **a** Raster of 4 trials are shown for a moving spot of diameter 9° with different speed values (mentioned at the far right). For different speeds the average spike count and SD values are (format: Speed, Mean SD); 27°/s, 11 1.6; 15°/s, 36 8.9; 10°/s, 37.2 3.7; 8°/s, 28.8 3.9. **b** Raster of 3 trials are shown for a different unit for a spot (diameter 8°) moving with different speeds (similar to those shown in (a)). The mean and SD values of spike count for different speeds are 17°/s, 5.3 3.5; 7°/s, 32.7 11.6; 5°/s, 22.3 6.7; 3°/s, 34.7 9.3. **c** Speed tuning plot of the unit presented in (a). The firing rate is calculated by dividing the total number of spikes by the duration of the stimulus movement. Error bars represent SD. **d** Speed tuning plot of the unit shown in (b).

2.3.6 Synaptic projection from Imc to Ipc is inhibitory

The Ipc response-size profile suggested the involvement of inhibition in shaping Ipc visual responses. One source for inhibition is the GABAergic Imc neuron. Imc neurons receive broad tectal input and innervate the Ipc, so are well-suited to provide large-field inhibition to Ipc neurons. To test this possibility, we conducted whole-cell blind-patch recordings from Ipc neurons while stimulating the Imc axons in the deep tectal layers. A stimulus electrode (red vertical line, Inset Figure 2.10) was lowered into deep tectal SGC layers (500 $\mu\mathrm{m}$ below the surface) to stimulate Imc axons antidromically. The resulting Imc spikes traveled from the optic tectum to the Imc and continued to the Ipc. Because of the broad and dense Imc projection pattern, a recorded Ipc neuron (blue dot) had a high probability of receiving synaptic inputs from the stimulated Imc axons (see Figure 2.10). To avoid direct stimulation of the Ipc axon of the recorded neuron or SGP neurons, the topography of Ipc-tectum projection was taken into account [46] and recordings were made from the dorso-lateral part of the Ipc while stimulating the rostro-medial tectum. Six Ipc neurons had response latencies less than 10 ms which is consistent with antidromic Imc axon stimulation and displayed substantial long-lasting (200 - 400 ms) inhibition. Additionally, there were 3 cells investigated by this stimulation protocol that were excluded. Among them, two cells showed long inhibition but were excluded from the analysis because of their longer response latencies, which indicate an indirect activation of the Imc neurons. In one cell, the deep tectal stimulation showed long inhibition along with short excitation, which is consistent with stimulation of both GABAergic Imc and glutamatergic SGP neurons in deep tectal layers. To determine the source of this inhibitory current, current clamp experiments were conducted (Figure 2.10). The inhibitory post synaptic potential due to Imc stimulation reversed at around -79 mV (Figure 2.10, bottom).

2.4 Discussion

We show that turtle Ipc neurons have a localized excitatory receptive field with surround inhibition, a preference for small stimuli, as well as significant adaptation to static stimuli. Here, we compare our findings with those from other animals, discuss possible mechanisms based on our results and the underlying circuitry of the system, and provide an outlook for the role of Ipc cholinergic feedback for tectal visual processing in

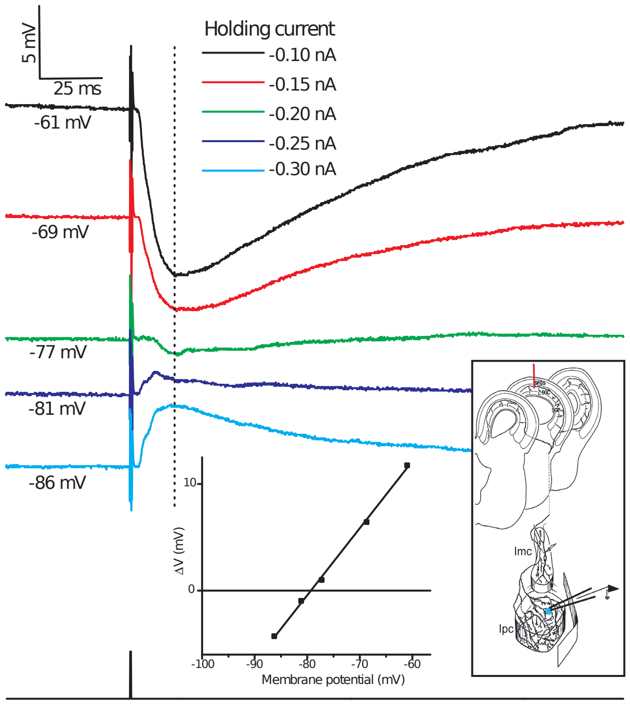


Figure 2.10: Synaptic response of an Ipc neuron to deep tectal stimulation, presumably activating Imc axons directly. Single pulse stimulation (0.5 ms, 200 μA) resulted in a long-lasting postsynaptic potential (PSP). Recordings were obtained for five different holding currents corresponding to membrane potentials between -61 and -86 mV. Each trace represents an average of three trials. The peak amplitude of the PSP, ΔV, was measured as the difference between the membrane potential at the holding current and the membrane potential at 18 ms after the stimulus pulse (vertical dotted line). The PSP peak amplitude increases linearly with increasing membrane potential and reverses sign at -79 mV. **Inset** Schematic of stimulating electrode in tectum (red line) and recording from an Ipc neuron (blue dot). The GABAergic Imc neuron's axon splits into two major branches. One branch projects nontopographically to the Ipc and the other branch projects nontopographically to tectal layers SGC and SGP. The two isthmic nuclei have been enlarged relative to the tectum for clarity. The axon split and the broad projection pattern allows for Imc stimulation via its axons in the tectum. Inset reproduced from Sereno and Ulinski 1987 with additions.

turtle.

The small moving stimuli generate vigorous spiking responses in the turtle Ipc neurons as well as in isthmic neurons of many species (pigeon: [23, 26–28, 57, 61] fish: [12] cat: [48]. The spontaneous firing rates of turtle Ipc neurons and pigeons nucleus isthmi cells are found to be low [57] compared to the high spontaneous activity of PBN neurons [7, 14, 48]. We have found that the turtle Ipc neurons are not directionally sensitive (Figure 2.8), which is consistent with the weakly directionally sensitive pigeon Ipc neurons [22, 57] and cat PBN neurons [48]. However, evidence of directional selectivity of isthmic neurons has been found in other species (reptiles: [54] birds: [61]. We have found oscillatory bursts in the turtle Ipc neurons in response to visual stimulation (Figure 2.3a), which has also been observed in avian Ipc neurons [27, 28] and fish isthmic neurons [33]. Visual response habituation has been observed in nucleus isthmi and optic tectum of teleosts [33]. Habituation is also present in neurons in bird optic tectum [27, 32]. The temporal adaptation with considerable variability in time-course from unit to unit is prominent in bird Ipc neurons while responding to a small bright spot stimulus [26]. Another similarity we observed with isthmic neurons of other species is their broad speed tuning (turtle: Figure 2.9a, b; bird: [57] cat: [48]).

The turtle Ipc excitatory receptive field is shaped in the pathway from retina, to tectum, to Ipc. The radial narrow dendrites of the tectal SGP neurons overlap with retinal axon terminals in superficial tectal layers [21]. In turn, axons from the SGP neurons project to the ipsilateral Ipc in a topographic fashion (Figure 2.1). The SGP apical radial dendrites form dendrodendritic synapses within the SFGS layer [44], thus providing the hardware for the lateral spread of local retinal inputs and suggesting broad Ipc receptive fields. Yet, our recordings reveal circular excitatory Ipc receptive fields of Gaussian shape with an average half-width just below 3 deg. Cat parabigeminal neurons (PBN) have a similar excitatory receptive field with a little less than 3 deg in diameter [48]. In contrast, pigeon Ipc neurons have circular excitatory receptive fields with diameters in the range from 10 to 20 deg [22, 27, 28] and the excitatory receptive fields of frog nucleus isthmi neurons are even larger [6, 60].

To determine the inhibitory receptive field structure of Ipc neurons, we investigated the Ipc response-size profile using different spot sizes (Figure 2.7). Based on a Difference-of-Gaussian analysis the inhibitory receptive field turned out to be weaker and about twice as big as the excitatory receptive field of Ipc neurons.

This stimulus size tuning, which is the result of a surround inhibition, is found in cat PBN [48], bird Ipc [57] and tectal [32] neurons. A similar response-size profile can also be observed in the LGN [1, 50] and cortical VI neurons [43].

In the turtle isthmotectal system, the GABAergic Imc neuron is a likely candidate for the generation of this inhibitory surround through its broad projection to the Ipc. Consistent with this hypothesis, we have found that the Imc-Ipc synapses are inhibitory (Figure 2.10). In addition, retinal and tectal processing [4, 15] may contribute to the Ipcs visual response properties. Our study cannot separate these influences from the role of Imc neurons in providing inhibition to the Ipc neurons.

The feedback pathway from nucleus isthmi to tectum in turtle consists of (Figure 2.1) a narrow, topographic, cholinergic projection to multiple tectal layers from Ipc neuron and a broad, inhibitory projection from Imc neuron to deep tectal layers [46]. This anatomical connection underscores the fact that the cholinergic feedback mediated by Ipc neurons at one tectal locus faces competition with feedback produced at other tectal loci via long distance suppression by Imc neurons. These interactions might provide spatially specific modulation of tectal circuitry and mediate mechanisms of visual processing in the tectofugal pathway in different species [18].

The cholinergic Ipc projections also play an important role in upstream visual processing in the tectofugal pathway. Large-field neurons in the SGC layer of the turtle tectum project to the nucleus rotundus in a non-topographic manner, forming a key element of tectofugal pathway [3, 38]. The tectal SGC neurons' dendritic branches extend into the superficial retinorecipient tectal layers (SFGS). The topographically organized columnar Ipc axon terminals spatially overlap with the RGC axons and the SGC dendrites. Thus the release of ACh from Ipc axon terminals across several tectal layers provides ample potential for a spatially specific cholinergic modulation of retino-tectal synaptic transmission and in turn, the ascending visual pathway. This conjecture is supported by the observations that isthmic activity enhances calcium influx into the optic nerve fiber terminals in frog [10] and by findings that the local inactivation of the Ipc prevents visual responses in the spatially corresponding ascending RGC-SGC visual pathway to the nucleus rotundus in birds [28]. Clearly, a deeper understanding of the Ipc visual responses will be a prerequisite for gaining insight into the role of spatiotemporal ACh release for visual processing in the optic tectum or superior colliculus.

References

[1] Henry J Alitto and W Martin Usrey Ã. Corticothalamic feedback and sensory processing. *Current Opinion in Neurobiology*, (Figure 2):440–445, 2003.

[2] Michael Ariel, Naoki Kogo, Monique Maurice, Henri Gioanni, and Anick Abourachid. Direction Tuning of Inhibitory Inputs to the Turtle Accessory Optic System Direction Tuning of Inhibitory Inputs to the Turtle Accessory Optic System. *Journal of Neurophysiology*, pages 2919–2930, 2001.

[3] Margarita Belekhova, Natalia Kenigfest, Jean-Paul Rio, Jacques Repérant, Roger Ward, Nikolai Vesselkin, and Olga Karamian. Tectothalamic visual projections in turtles: their cells of origin revealed by tracing methods. *The Journal of comparative neurology*, 457(1):37–56, February 2003.

[4] B Y D B Bowling. Light Responses of ganglion cells in the retina of the turtle. *J Physiol*, 299(1):173–196, 1980.

[5] Kenneth T Brown. A linear area centralis extending across the turtle retina and stabilized to the horizon by non-visual cues. *Vision Research*, 9(9):1053 – IN5, 1969.

[6] Matthew S Caudill, Adam T Eggebrecht, Edward R Gruberg, and Ralf Wessel. Electrophysiological properties of isthmic neurons in frogs revealed by in vitro and in vivo studies. *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, 196(4):249–62, April 2010.

[7] He Cui and Joseph G Malpeli. Activity in the parabigeminal nucleus during eye movements directed at moving and stationary targets. *Journal of neurophysiology*, 89(6):3128–42, June 2003.

[8] P H Desan, E R Gruberg, and F Eckenstein. A cholinergic projection from the nucleus isthmi to the optic tectum in turtle and frog. In *Proc. Society for Neuroscience Annual Meeting (SFN'84)*, volume 10, page 575, November 1984.

[9] Shaul Druckmann, Thomas K Berger, Sean Hill, Felix Schürmann, Henry Markram, and Idan Segev. Evaluating automated parameter constraining procedures of neuron models by experimental and surrogate data. *Biological Cybernetics*, pages 371–379, 2008.

[10] Elizabeth A Dudkin and Edward R Gruberg. N ucleus isthmi enhances calcium influx into optic nerve fiber terminals in Rana pipiens. *Brain Research*, 969:44–52, 2003.

[11] Elizabeth A Dudkin, Joel B Sheffield, and Edward R Gruberg. Combining Visual Information from the Two Eyes : The Relationship between Isthmotectal Cells That Project to Ipsilateral and to Contralateral Optic Tectum Using Fluorescent Retrograde Labels in the Frog , Rana pipiens. *Comparative and General Pharmacology*, 54(August 2006):38–54, 2007.

[12] Shawn P Gallagher and David P M Northmore. Responses of the teleostean nucleus isthmi to looming objects and other moving stimuli. *Visual Neuroscience*, pages 209–219, 2006.

[13] GC DeAngelis, RD Freeman, and I Ohzawa. Length and Width Tuning of Neurons in the Cat s Primary Visual Cortex. *Neurophysiology*, 71(1), 1994.

[14] C Alex Goddard, Eric I Knudsen, and John R Huguenard. Intrinsic excitability of cholinergic neurons in the rat parabigeminal nucleus. *Journal of neurophysiology*, 98(6):3486–93, December 2007.

[15] a M Granda and J E Fulbrook. Classification of turtle retinal ganglion cells. *Journal of neurophysiology*, 62(3):723–37, September 1989.

[16] A M Graybiel. A satellite system of the superior colliculus: the parabigeminal nucleus and its projection to the superficial collicular layers. *Brain Res*, 145:365–374, 1978.

[17] E R Gruberg and S B Udin. Topographic projections between the nucleus isthmi and the tectum of the frog, Rana pipiens. *J Comp Neurol*, 179:487–500, 1978.

[18] Edward Gruberg, Elizabeth Dudkin, Yuan Wang, Gonzalo Marín, Carlos Salas, Elisa Sentis, Juan Letelier, Jorge Mpodozis, Joseph Malpeli, He Cui, Rui Ma, David Northmore, and Susan Udin. Influencing and interpreting visual input: the role of a visual feedback system. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 26(41):10368–71, October 2006.

[19] W C Hall, D Fitzpatrcik, L L Klatt, and D Raczkowski. Cholinergic innervation of the superior colliculus in the cat. *J Comp Neurol*, 287:495–514, 1989.

[20] R Kriegstein. Synaptic Responses of Cortical Pyramidal Neurons Stimulation in the Isolated Turtle Visual System. *Journal of Neuroscience*, 7(August):2488–2492, 1987.

[21] H Kunzle and H Schnyder. The isthmus-tegmentum complex in the turtle and rat: a comparative analysis of its interconnections with the optic tectum. *Exp Brain Res*, 56:509–522, 1984.

[22] D P Li, Q Xiao, and S R Wang. Feedforward construction of the receptive field and orientation selectivity of visual neurons in the pigeon. *Cereb Cortex*, 17:885–893, 2007.

[23] Da-Peng Li, Qian Xiao, and Shu-Rong Wang. Feedforward construction of the receptive field and orientation selectivity of visual neurons in the pigeon. *Cerebral cortex (New York, N.Y. : 1991)*, 17(4):885–93, April 2007.

[24] E Lucas-Meunier, P Fossier, G Baux, and M Amar. Cholinergic modulation of the cortical neuronal network. *European journal of physiology*, 446(1):17–29, April 2003.

[25] Estelle Lucas-Meunier, Cyril Monier, Muriel Amar, Gérard Baux, Yves Frégnac, and Philippe Fossier. Involvement of nicotinic and muscarinic receptors in the endogenous cholinergic modulation of the balance between excitation and inhibition in the young rat visual cortex. *Cerebral cortex (New York, N.Y. : 1991)*, 19(10):2411–27, October 2009.

[26] Kristin a Maczko, Phyllis F Knudsen, and Eric I Knudsen. Auditory and visual space maps in the cholinergic nucleus isthmi pars parvocellularis of the barn owl. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 26(49):12799–806, December 2006.

[27] Gonzalo Marín, Jorge Mpodozis, Jorge Mpdozis, Elisa Sentis, Tomás Ossandón, and Juan Carlos Letelier. Oscillatory bursts in the optic tectum of birds represent re-entrant signals from the nucleus isthmi pars parvocellularis. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 25(30):7081–9, July 2005.

[28] Gonzalo Marín, Carlos Salas, Elisa Sentis, Ximena Rojas, Juan Carlos Letelier, and Jorge Mpodozis. A cholinergic gating mechanism controlled by competitive interactions in the optic tectum of the pigeon.

The Journal of neuroscience : the official journal of the Society for Neuroscience, 27(30):8112–21, July 2007.

[29] D A McCormick. Actions of acetylcholine in the cerebral cortex and thalamus and implications for functions. *Prog Brain Res*, 98:303–308, 1993.

[30] L Medina and A Reiner. Distribution of choline acetyltransferase immunoreactivity in the pigeon brain. *J Comp Neurol*, 342:497–537, 1994.

[31] R Metherate. Nicotinic acetylcholine receptors in sensory cortex. *Learn Mem*, 11:50–59, 2004.

[32] Shreesh P Mysore, Ali Asadollahi, and Eric I Knudsen. Global inhibition and stimulus competition in the owl optic tectum. *J Neurosci*, 30(5):1727–1738, February 2010.

[33] D P Northmore and S P Gallagher. Functional relationship between nucleus isthmi and tectum in teleosts: synchrony but no topography. *Vis Neurosci*, 20:335–348, 2003.

[34] D P Northmore and A M Granda. Ocular dimensions and schematic eyes of freshwater and sea turtles. *Vis Neurosci*, 7:627–635, 1991.

[35] J W Peirce. Generating Stimuli for Neuroscience Using PsychoPy. *Front Neuroinformatics*, 2:10, 2008.

[36] A S Powers and A Reiner. The distribution of cholinergic neurons in the central nervous system of turtles. *Brain Behav Evol*, 41:326–334, 1993.

[37] R Quian Quiroga, Z Nadasdy, and Y Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput*, 16(8):1661–1687, August 2004.

[38] A Reiner. Laminar distribution of the cells of origin of ascending and descending tectofugal pathways in turtles: implications for the evolution of tectal lamination. *Brain Beh Evol*, 43:254–292, 1994.

[39] R W Rodieck. Quantitative analysis of cat retinal ganglion cell response to visual stimuli. *Vision Res*, 5:583–601, 1965.

[40] A F Rosenberg and M Ariel. Visual-response properties of neurons in turtle basal optic nucleus in vitro. *J Neurophysiol*, 63:1033–1045, 1990.

[41] A F Rosenberg and M Ariel. Electrophysiological evidence for a direct projection of direction-sensitive retinal ganglion cells to the turtle's accessory optic system. *J Neurophysiol*, 65:1022–1033, 1991.

[42] H Sato, N Katsuyama, H Tamura, Y Hata, and T Tsumoto. Mechanisms underlying direction selectivity of neurons in the primary visual cortex of the macaque. *J Neurophysiol*, 74:1382–1394, 1995.

[43] M P Sceniak, D L Ringach, M J Hawken, and R Shapley. Contrast's effect on spatial summation by macaque V1 neurons. *Nat Neurosci*, 2:733–739, 1999.

[44] Pb Schechter and P S Ulinski. Interactions between tectal radial cells in the red-eared turtle, Pseudemys scripta Elegans: An analysis of tectal modules. *J Morph*, 162:17–36, 1979.

[45] M I Sereno and P S Ulinski. Caudal topographic nucleus isthmi and the rostral nontopographic nucleus isthmi in the turtle, Pseudemys scripta. *The Journal of comparative neurology*, 261(3):319–46, July 1987.

[46] M I Sereno and P S Ulinski. Caudal topographic nucleus isthmi and the rostral nontopographic nucleus isthmi in the turtle, Pseudemys scripta. *J Comp Neurol*, 261:319–346, 1987.

[47] H Sherk. Visual response properties and visual field topography in the cat's parabigeminal nucleus. *Brain Res*, 145:375–379, 1978.

[48] H Sherk. A comparison of visual-response properties in cat's parabigeminal nucleus and superior colliculus. *J Neurophysiol*, 42:1640–1655, 1979.

[49] H Sherk. Connections and visual field mapping in cats tectoparabigeminal circuit. *J Neurophysiol*, 42:1656–1668, 1979.

[50] A M Sillito and H E Jones. Corticothalamic interactions in the transfer of visual information. *Philos Trans R Soc Lond B Biol Sci*, 357:1739–1752, 2002.

[51] E M Sorenson, D Parkinson, J L Dahl, and V A Chiappinelli. Immunohistochemical localization of choline acetyltransferase in the chicken mesencephalon. *J Comp Neurol*, 281:641–657, 1989.

[52] S R Wang. The nucleus isthmi and dual modulation of the receptive field of tectal neurons in non-mammals. *Brain Res Rev*, 41:13–25, 2003.

[53] S R Wang, Y C Wang, and B J Frost. Magnocellular and parvocellular divisions of pigeon nucleus isthmi differentially modulate visual responses in the tectum. *Exp Brain Res*, 104:376–384, 1995.

[54] S R Wang, K Yan, Y T Wang, S Y Jiang, and X S Wang. Neuroanatomy and electrophysiology of the lacertilian nucleus isthmi. *Brain Res*, 275:355–360, 1983.

[55] Y Wang, D E Major, and H J Karten. Morphology and connections of nucleus isthmi pars magnocellularis in chicks (Gallus gallus). *J Comp Neurol*, 469:275–297, 2004.

[56] Y Wang, J Xiao, and S R Wang. Excitatory and inhibitory receptive fields of tectal cells are differentially modified by magnocellular and parvocellular divisions of the pigeon nucleus isthmi. *J Comp Physiol A*, 186:505–511, 2000.

[57] Y C Wang and B J Frost. Visual response characteristics of neurons in the nucleus isthmi magnocellularis and nucleus isthmi parvocellularis of pigeons. *Exp Brain Res*, 87:624–633, 1991.

[58] Yuan Wang, Harald Luksch, Nicholas C Brecha, and Harvey J Karten. Columnar projections from the cholinergic nucleus isthmi to the optic tectum in chicks (Gallus gallus): a possible substrate for synchronizing tectal channels. *The Journal of comparative neurology*, 494(1):7–35, January 2006.

[59] W Wiggers and G Roth. Anatomy, neurophysiology and functional aspects of the nucleus isthmi in salamanders of the family Plethodontidae. *J Comp Physiol A*, 169:165–176, 1991.

[60] D E Winkowski and E R Gruberg. The representation of the ipsilateral eye in nucleus isthmi of the leopard frog, Rana pipiens. *Vis Neurosci*, 19:669–679, 2002.

[61] K Yan and S R Wang. Visual responses of neurons in the avian nucleus isthmi. *Neurosci Lett*, 64:340–344, 1986.

Chapter 3

Laminar Specific Inputs in a Three-Layered Visual Cortex

Matthew Caudill, David Morton, and Ralf Wessel

Manuscript in Preparation

One of the striking features of the synaptic organization of cortical circuits is the degree to which distinct populations of afferent axons synaptically target specific laminae. Despite the wealth of anatomical knowledge on the synaptic organization of the six-layered mammalian cortex, the functional significance of layer specific synaptic inputs remains a challenge. Here we perform a current source-density analysis with a model system, the three layered visual cortex of turtles. In this model system, principal cells are confined to a single layer. This feature provides a clear advantage in the interpretation of spatio-temporal synaptic events following afferent stimulation. Our results reveal three synaptic current sinks that are spatially segregated along the soma-dendrtic axis. This analysis confirms previous experimental findings and provides predictions about additional synaptic connectivity in the turtle cortical network.

3.1 Introduction

The local field potential (LFP) is low frequency electrical activity recorded with an extracellular electrode that primarily reflects summed dendritic synaptic activity in a volume of tissue [27]. It has recently received much interest because it is a remarkably precise measure of the neural processes that underlie perception, attention and memory [27]. In brain regions such as the cerebellum, hippocampus and neocortex, large field potential arise from the laminar organization and synchronous activation of populations of neurons. These potentials generate microscopic current flows in the extracellular space [20, 24] that reveal information about the laminar activation of neural ensembles. These currents can be obtained from the LFP using current source-density analysis (CSD) [24]. However, in the mammalian neocortex, the interpretation of the CSD can be complicated by the presence of multiple cell types within each laminae. Thus model systems featuring laminated structures with identifiable cell types confined to specific layers have been developed to probe outstanding questions concerning laminar specific inputs.

The turtle dorsal or visual cortex is a highly ordered structure with two principal cell types arranged into three layers: spiny pyramidal neurons and aspiny stellate neurons[5, 6]. (Figure 3.1) Pyramidal cells are confined to the middle layer (layer 2) and receive a distributed projection from the thalamus onto their distal dendrites located in the outer third of layer 1. Stellate neurons appear predominantly in the superficial and deep layers and have flattened vesicles with symmetrical profiles that have been correlated with inhibitory function [8]. The laminar organization and the confinement of excitatory principal neurons to a single layer make the turtle cortex an interesting model for investigating the laminar organization of synapses in cortical networks.

Here we show that afferent fiber, intra-cortical, and extrastriate stimulation evoke large field potentials in the turtle visual cortex and analyze these responses using an inverse current source-density method. Our purpose is to further clarify the intrinsic circuitry and identify laminar specific excitatory inputs within the turtle cortical microcircuit. The results demonstrate that feedforward, lateral and recurrent excitations, common in cortical connectivity patterns, target specific laminae of the turtle dorsal cortex.

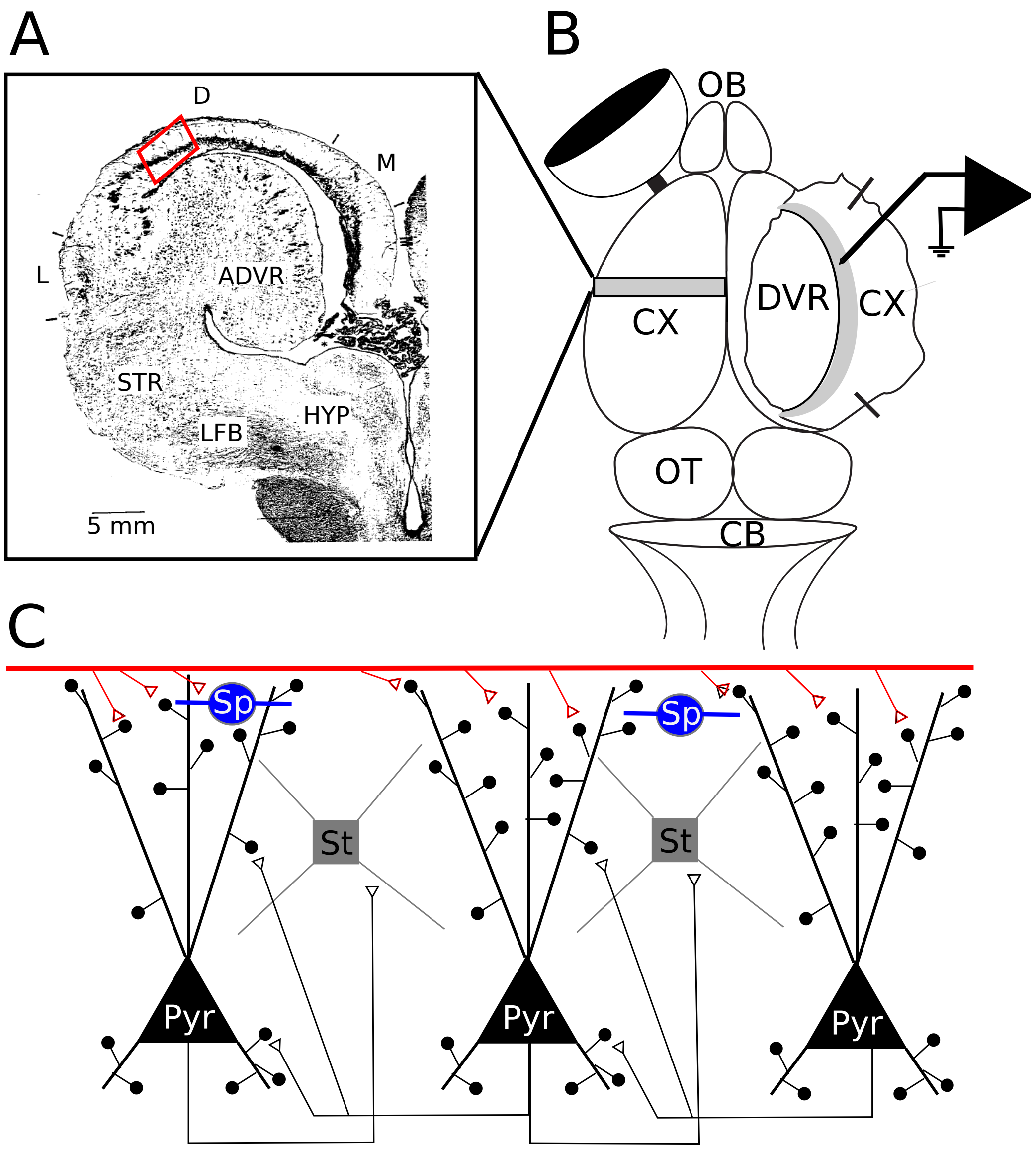


Figure 3.1: Turtle cortical anatomy and circuitry. A. Transverse section through the turtle telencephalon. Cortex is the 'scroll-like' flap of tissue. It is separated into lateral (L), dorsal (D), and medial (M) sections. B. Whole-brain preparation and diagram of recording electrode used in this study. Cortex is cut and pinned open so that the medial cortex is furthest from the DVR. The dorsal cortex is shown in gray. C. Zoom of the boxed region in A. Lateral forebrain inputs (LFB) carrying geniculate axons bear varicosities *en-passant* and contact a line of pyramidal (Pyr) and subpial (Sp) neurons. Pyramidal neurons are reciprocally connected to other pyramidals and stellate (St) neurons. Abbreviations are anterior dorsal ventricular ridge (ADVR), Striatum (STR), hypothalamus (HYP), cortex (CX), olfactory bulb (OB), optic tectum (OT), and cerebellum (CB).

3.2 Methods

3.2.1 Animal preparation

Adult red-eared slider turtles (*pseudemys scripta elegans*) of either sex, with carapace lengths of 8-15 cm, purchased from Niles Biological (Sacramento, CA) were used in this study. Turtles were cooled to 0°C, decapitated, and the brains were surgically removed with the eyes and optic nerves attached. The retina was prepared for visual stimulation by removing the anterior part of the eye (see Figure 3.1). Access to the visual cortex was obtained by making a rostral-caudal incision along the medial cerebral hemisphere and unfolding the cortex to expose the ventricular surface (see Figure 3.1 a,b). This procedure maintains the laterally coursing geniculate afferents (see Figure 3.1).The preparation was maintained at room temperature in a recording chamber continuously perfused with aerated (95% O_2 5% CO_2) turtle ACSF containing (in mM): 96.5 NaCl; 2.6 KCl; 4.0 $CaCl_2$; 2.0 $MgCL_2$-$6H_2O$; 31.5 $NaHCO_3$; and 10 dextrose, at a pH of 7.4.

3.2.2 Stimulation and Recording

Electrical Stimulation

To evoke cortical field potentials, concentric bipolar stimulating electrodes (TM33CCINS, World precision instruments, Sarasota, FL) were placed into either the lateral forebrain bundle (LFB), below the ventricular surface of the cortex, or in the superficial anterior dorsal ventricular ridge (ADVR) with a manual manipulator (Narishige model M-3333, East Meadow, NY). Electrical stimuli consisted of 100 μs biphasic current pulses with amplitudes between 50-800 μA generated with an isolated pulse stimulator (AM-systems model 2100, Carlsborg, WA).

Recordings

Cortical field potentials were recorded using 16-channel silicon-based recording electrodes (A1x16) purchased from Neuronexus Technologies (Ann Arbor, MI). The Iridium contacts were arranged linearly and had site areas of 177 μm^2 with a center-to-center distance of 25 μm. The stated impedance values of the contacts ranged from 0.61 to 1.24 MΩ. This variation in impedance led to slightly different voltage gains on each channel. To account for these gain differences, we applied a sine wave of continuously varying frequencies between 10 and 300 Hz to the bath before each experiment to allow us to calibrate the gain of each channel. This frequency range covered the local field potential responses observed during electrical stimulation. The electrode arrays were positioned with a motorized drive purchased from Sutter Instruments (model MP-285, Novato, CA) and were slowly lowered into the ventricular surface in 25 μm steps over a period of 30 minutes to reduce cortical dimpling. Signals recorded by the probe were amplified with a 16-channel microelectrode amplifier (AM-systems model 3600, Carlsborg WA) and band-passed filtered between 10 and 20,000 Hz (roll off rate 40 dB/decade).

3.2.3 CSD Analysis

After collecting the field potentials, a current source-density analysis was performed using a Python software package that we have released as an open-source project called Pycsd [16]. Prior to the analysis, we conditioned the data by removing the stimulus artifact, removing the DC component from each channel (estimated from averages of the pre-stimulus signal), and low pass filtering with a second order Bessel filter at 300 Hz. We further conditioned the data by separating the even and odd channels before applying the CSD analysis. This was necessary because significant correlations in the noise across paired channels was detected and indicated cross-talk in the wires leading from the probe to head-stage (Figure 3.2). The CSD analysis was performed on the even and odd channels separately and the resulting CSDs were then recombined and then multiple trials were averaged. Lastly, we smoothed the average CSDs in depth using a 5 point Hamming window. All figures were generated using Matplotlib, an open-source python plotting library.

CSD estimation

Commonly, CSD estimation consist of estimating the second spatial derivative of the extracellular potential [24]. This corresponds to the approximation that current sources are infinitely thin planes with no in-plane variation in neural activity. However, this estimation suffers from at least two problems. Firstly, the CSD can not be estimated at the top and bottom electrode positions because the second derivative requires the potential at electrode positions above and below each contact. Secondly, estimating the CSD near the tissue-

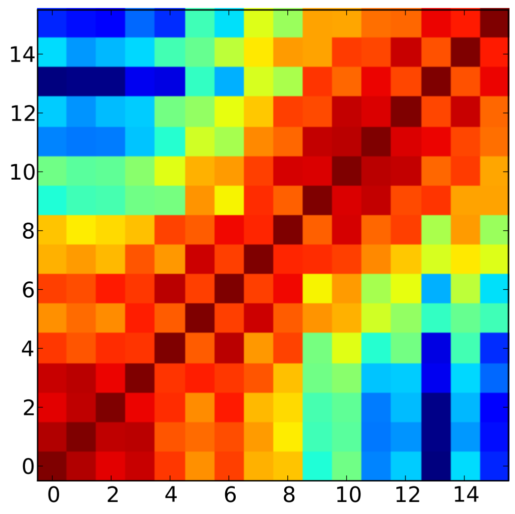


Figure 3.2: Correlated noise in contacts on laminar probe. Significant "cross-talk" between the channels were measured for noise measured in recording chamber. X-Y coordinate pairs are color coded for the amount of correlation. Stripping indicates correlations between even/odd channels.

saline boundary with this method is problematic due to the abrupt conductance change. Pettersen et al. [28] described a method that overcomes these limitations and we use a variant of their general method called the step-iCSD.

The step-iCSD method utilizes the forward solution $\mathbf{F}$ of $\mathbf{\Phi} = \mathbf{FC}$ where $\mathbf{\Phi}$ is the extracellularly measured electrical potentials, $\mathbf{C}$ is the CSD and $\mathbf{F}$ is the transfer function which depends on the geometry and boundary conditions. With the approximation that $\mathbf{C}$ is a step-wise constant in depth, $\mathbf{F}$ can be determined. With the forward solution, $\mathbf{F}$, in hand, we invert the problem to estimate the CSD from a given set of potential measurements, $\mathbf{F^{-1}\Phi} = \hat{\mathbf{C}}$ where $\hat{\mathbf{C}}$ is the CSD estimate. The forward solution matrix elements for the step-iCSD method are

$$F_{ji} = \int_{z_i - \frac{h}{2}}^{z_i + \frac{h}{2}} \frac{1}{2\sigma} \left(\sqrt{(z_j - z\prime)^2 + R^2} - |z_j - z\prime| \right) dz\prime \tag{3.1}$$

where z_i and z_j are the ith and jth contact depth along the shank electrode, R is the radius of the model current-source disks, h is the width of the disks, and σ is the extracellular conductivity. This forward solution corresponds to the boundary condition where the conductivity above the brain surface is equal to the conductivity in the extracellular medium. This solution can be extended to incorporate effects of different conductivities above the cortical surface using the method of images. The case we are concerned with is when the conductivity above the surface of the brain is much greater than the extracellular medium. In this case, elements of the forward solution matrix can be found exactly as,

$$\begin{aligned} F_{ji} = &\int_{z_i-\frac{h}{2}}^{z_i+\frac{h}{2}} \frac{1}{2\sigma}\left(\sqrt{(z_j - z\prime)^2 + R^2} - |z_j - z\prime|\right) dz\prime \\ &- \int_{z_i-\frac{h}{2}}^{z_i+\frac{h}{2}} \frac{1}{2\sigma}\left(\sqrt{(z_j + z\prime)^2 + R^2} - |z_j + z\prime|\right) dz\prime. \end{aligned} \tag{3.2}$$

3.3 Results

3.3.1 Spatiotemporal profile of evoked field potentials

Evoked field potential responses to lateral forebrain bundle stimulation

The lateral forebrain bundle (LFB) contains geniculate axons that pass below the anterior dorsal ventricular ridge (ADVR) and course in the lateral-to-medial direction. These axons bear *en passant* synapses that contact the distal dendrites of pyramidal cells along a lateral-to-medial line within the cortex (Figure 3.1) [25]. We analyzed field potential events from 33 recording sites within the cortex in response to different levels of current applied to the LFB. During these experiments, we noted that the shapes of the field potentials could vary drastically (Figure 3.4). We first analyzed the stability of the field potentials at a each recording location for different amounts of stimulating current (Figure 3.3). We found that the amplitudes of the field potentials increased with increasing current and that the latency to the peak of the field potential relative to the stimulus decreased slightly with increasing current (Figure 3.3). On the other hand, the overall shape of the field potential did not vary for different stimulus current amplitudes (Figure 3.3). This stability allowed us to begin classifying field potential shapes. We found that we could reliably classify field potentials based on whether peaks inverted in depth and whether they contained multiple local extrema (Figure 3.4).

Field potentials that did not invert in depth were encountered in 16 of the 33 recording sites (Figure 3.4). We wondered whether this inversion failure could be due to a bias in our recordings. This bias could potentially occur because our array of electrode contacts was placed 400-450 μm down into a cortex which has a thickness of 500-600 μm in small turtles. Thus, distal synaptic inputs occurring at depths greater than 450 μm are not represented in our dataset. However, a careful examination of the field potential amplitudes in depth revealed that the amplitudes generally increased with increasing depth. This observation makes it unlikely that an inversion would be observed at any depth within the cortex at these sites.

Field potentials that inverted in depth were encountered in 17 of the 33 recording sites (Figure 3.4). We estimated inversion depths with an error on the order of the contact spacing of our electrode array (< 25 μm) and mapped these depths onto the ventricular surface of the cortex (Figure 3.5). We observed a slight tendency for the inversion depth to be deeper at locations that were more medial (i.e. closer to the ADVR). Comparisons between sites with and without field potential inversions indicated that inversions were more likely encountered in the rostral than the caudal cortex.

In addition to examining the field potential inversion depths, we also mapped the field potential response latency. This was measured as the time between the stimulus onset and the first peak in the field potential. A map of this latency on the ventricular cortical surface revealed latencies in the rostral cortex that were shorter than the latencies measured in the caudal cortex (Figure 3.5). These results are consistent with previous studies that demonstrated a rostral to caudal spread of activity to diffuse flashes of light measured using voltage-sensitive dyes [31, 34].

ADVR stimulation

The turtle visual cortex receives afferents from several subcortical structures throughout the telencephalon [39]. Interestingly, like mammals, turtles possess an extrastriate visual area called the anterior dorsal ventricular ridge (ADVR, homolog to mammalian V2). This structure receives visual input from the tectorecipient nucleus rotundus (RT) (homolog of the mammalian pulvinar) and shares many hodological, functional and developmental features that characterize the isocortex of mammals [32]. We examined the field potentials for this secondary visual route to the cortex would look like. Stimulation of the lateral ADVR evoked both biphasic and triphasic field potentials that did not invert in depth (n=3 recording locations). As with the LFB stimulation we found that at a given recording site the field potential shape was stable across all current values applied through the stimulating electrode (Figure 3.6). The amplitudes of the field potentials increased with

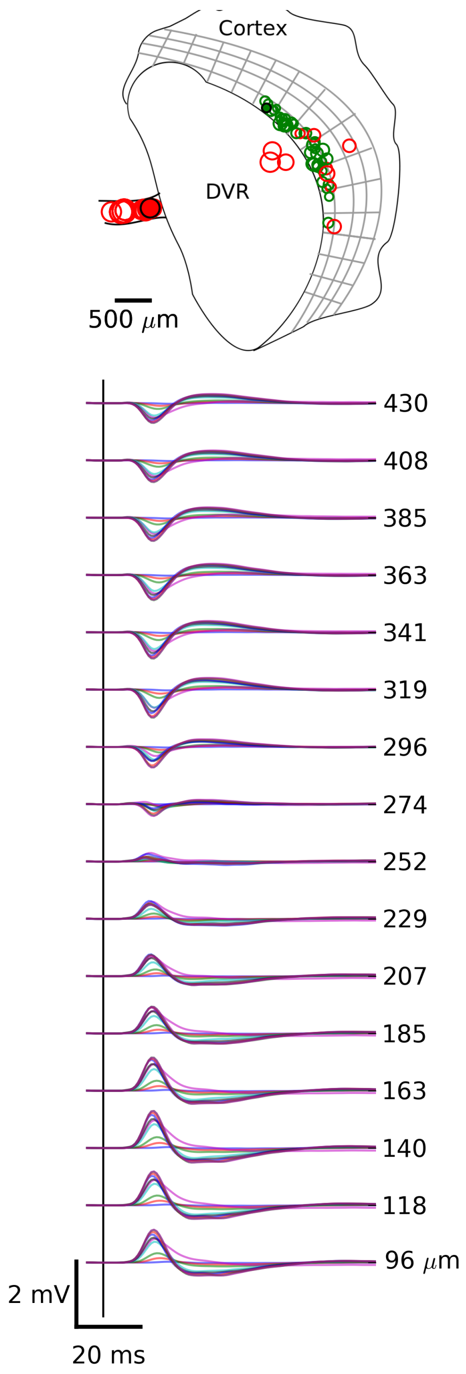


Figure 3.3: ./images/Stability of evoked cortical field-potentials in response to stimulation of the LFB. Field potentials were recorded at the location denoted by a green circle with a black rim to a stimulating current applied at the position denoted by a red circle with a black rim. All other circles are the other locations of the stimulating and recording electrode positions reported in this paper. Traces show a double extrema field potential that inverted in depth recorded at this site. Each field potential trace is an average of twenty trials for a given stimulus current. Various stimulus currents are represented as unique colors in the field potential plots. The stimulating currents varied from 50 to 500 μA in 50 μA steps. Note that the field potential amplitudes increase while the latency to the peak response decreases with increasing stimulus current.

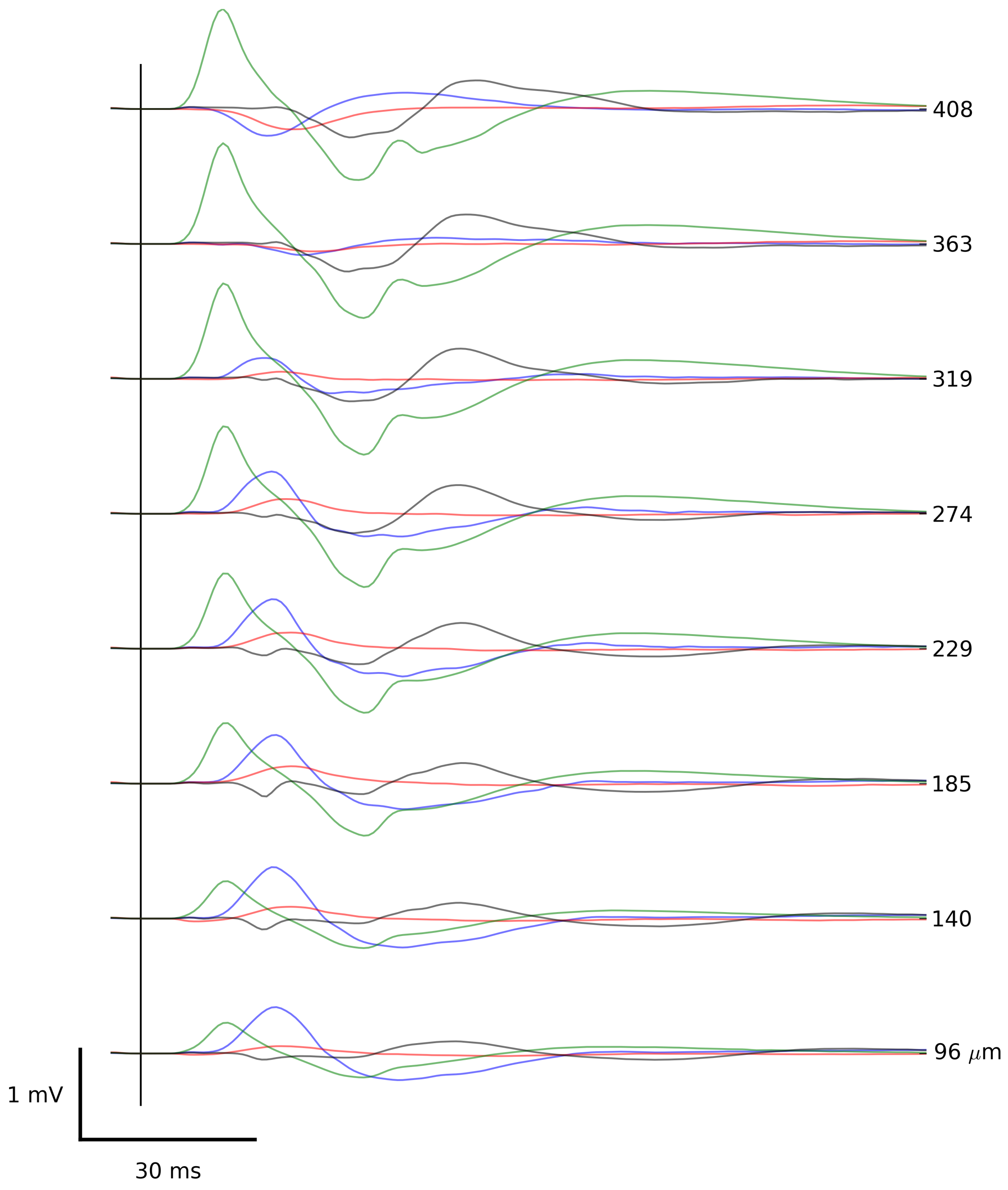


Figure 3.4: Evoked field potential patterns. Summary of four distinct field potential patterns recorded in response to LFB stimulation at a current of 300 μA. Black and green traces are stereotypical biphasic and triphasic responses that do not invert in depth. Red and blue are stereotypical monophasic and biphasic responses that invert at a depth of approximately 363 μm. Note depths are relative to the ventricular surface of the cortex.

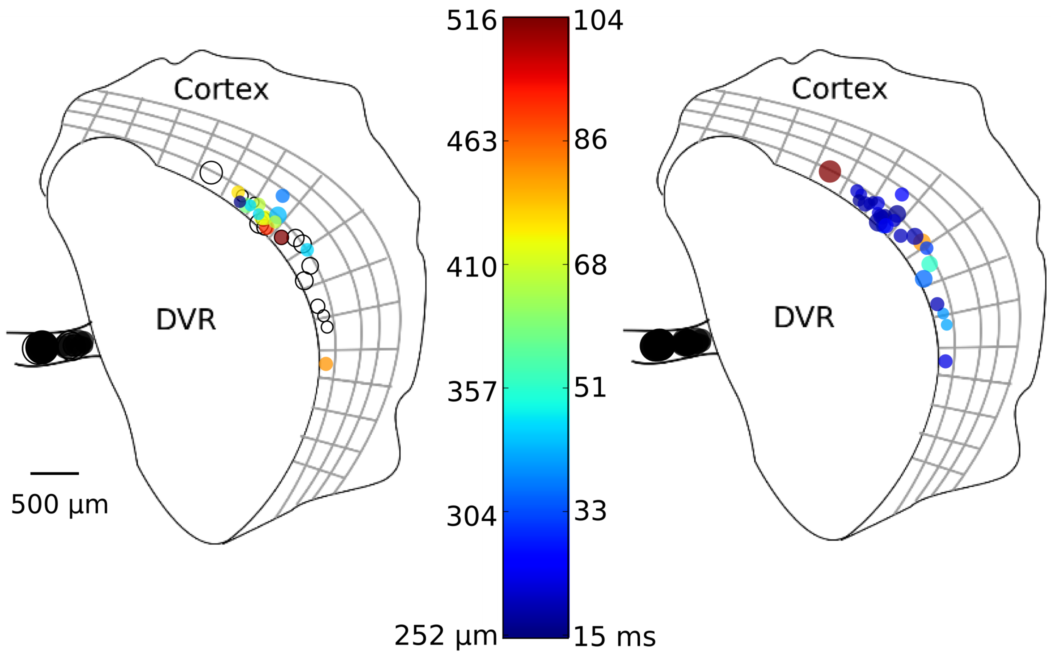


Figure 3.5: Field potential inversion depths and latencies mapped to cortical locations. Left. Heat map of inversion depths across 33 recording locations to stimulation of the LFB. Open circles are sites where no inversion was observed. Notice a slight tendency in sites recorded closer to the ADVR to have greater inversion depths than sites more medial. Right. Heat map of the latency to first peak in the field potential. Recording sites in the rostral cortex have noticeably shorter latencies than sites in the caudal cortex.

increasing stimulus current. We also observed that the latency of the responses decreased with increasing stimulus current by a few milliseconds.

Intra-cortical stimulation

Long-range horizontal connections spanning several millimeters have long been recognized as an important feature in the synaptic organization of mammalian cortical networks [12]. Local injections of horse radish peroxidase into the turtle cortex has found similar long-range connectivity patterns [7, 25]. We tested whether we could electrophysiologically observe these long range connections and whether they target specific cortical layers. We placed our stimulation electrode into the ventricular surface of the turtle cortex several millimeters away from the recording site in order to stimulate the associational fibers (Figure 3.1). We found that stimulation of the cortex reliably evoked large cortical field potentials at short-latencies for stimulation sites several millimeters from the recording location that were stable over all stimulating current intensities (Figure 3.7). The shape of the field potential was either biphasic or triphasic and did not invert over the depth of the cortex. Again we found increasing field potential amplitudes and decreasing latencies when the stimulus current was increased.

3.4 Current Source Density

3.4.1 Current source density analysis of LFB stimulation

Current source density analysis extracts the laminar distribution and time course of summed excitatory membrane currents (i.e. current sinks) from a set of extracellular potentials recorded at a series of depths [24, 26]. We applied this technique to our field potential data recorded in response to LFB stimulation. In particular, we chose the subset of data (17 of 33 cortical sites) that had field potentials that inverted in depth (Figure 3.4) because this feature was found to correlate with the presence of localized sources and sinks. We then further divided this subset into two groups, those that had a single local extremum (Figure 3.3) and those with multiple extrema.

Field potentials with a single extremum that inverted in depth (Figure 3.4) yielded a CSD profile (Fig-

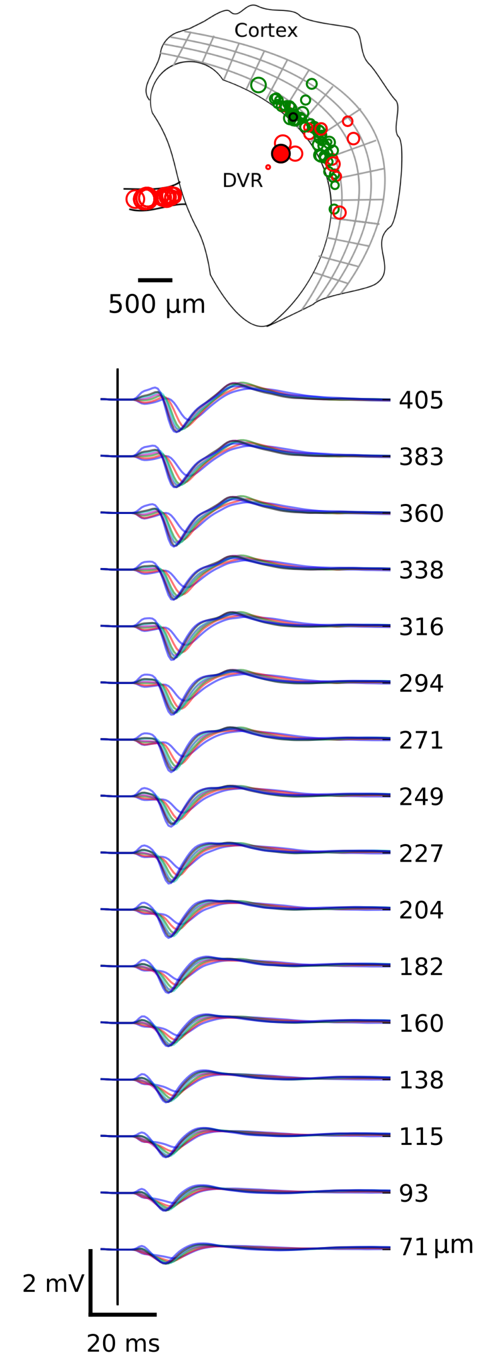


Figure 3.6: ./images/Evoked cortical field-potentials in response to stimulation of the ADVR. Diagram above shows the stimulation electrode position (red circle with black rim) and recording electrode position (green circle with black rim). Field potential traces below show cortical responses to current shocks with amplitudes of 100 to 500 μA in 100 μA steps. Each color represents a unique current. Notice the saturation of the amplitudes and decrease in the response latency for stronger stimulation currents.

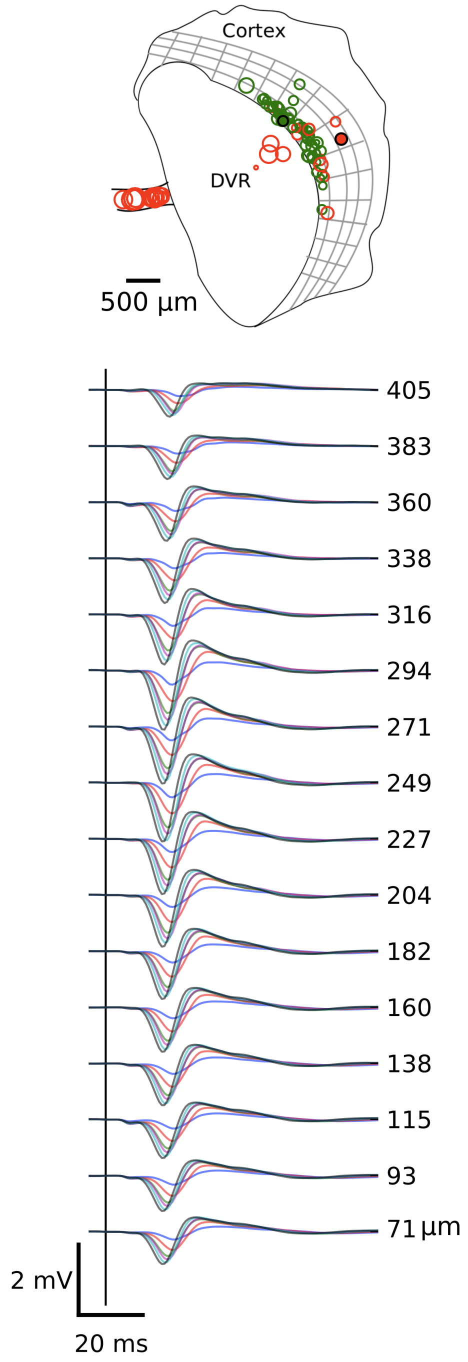


Figure 3.7: ./images/Evoked cortical field-potentials in response to stimulation of a distal cortical area. Diagram above shows the stimulation electrode position (red circle with black rim) and recording electrode position (green circle with black rim). Field potential traces below show cortical responses to current shocks with amplitudes of 100 to 500 μA in 100 μA steps. Note the graded response in the amplitudes and decrease in response latencies with increasing stimulus current intensity.

ure 3.8 A-C) with an isolated sink in deep layer 1 and a set of concurrent sources located at varying depths in layers 1 and 3. The average latency to the peak of this sink was 21 ms (range 19.5-23 ms) and occurred at depths between 350 and 430 μm (Figure 3.8 A). We were able to infer the relative position of this sink with respect to the cell layer by examining the laminar locations of somatic spikes that could be occasionally elicited by stronger stimulating currents. In addition, we noticed an absence of sources and sinks at depths between 100 and 250 μm (Figure 3.8 B). This is consistent with histological and electrophysiological evidence that has shown that the depth of the cellular layer decreases with increasing medial distance (i.e. greater distance from the ADVR) and is densest in the 100 to 300 μm depth range within the dorsal cortex [21]. Given these reference positions, the location of this sink is near the distal dendrites of pyramidal cells and is consistent with electron micrograph studies that found excitatory thalamic synapses in the outer third of layer 1 [38].

Field potentials with multiple extremum (Figure 3.4) were the most commonly observed response to LFB stimulation (13 of 17 recording sites). The CSD of these potentials revealed two to three synaptic sinks that occurred in separate laminae within the visual cortex (Figure 3.9). The initial sink occurred at depths between 350 and 455 μm, a depth that is near the distal dendrites of pyramidal neurons. This sink event occurred with an average latency of 17.7 ms (range 14.9 to 21.8 ms) and was accompanied by concurrent sources located in layers 1 and 3 (Figure 3.9 A). The time course and laminar position of this sink was similar to that obtained from field potentials possessing a single inverting extremum (see Figure 3.8). Again, we observed a noticeable absence of sources or sinks at the depths corresponding to the positions of pyramidal somata (i.e. layer 2) (Figure 3.9 B). Following this initial distal sink, a shallower secondary sink developed at depths between 200 and 350 μm (proximal layer 1) with a latency from the first sink of 17.4 ms (range 14.9 to 22.9 ms). This sink was noticeably weaker than the initial sink and was accompanied by weaker sources located in layers 1 (Figure 3.9 B). The laminar position of this sink is consistent with electron micrograph studies reporting excitatory asymmetrical contacts onto small spines along the inner two thirds of layer 1 [8]. In addition to sink currents in layer 1, a tertiary sink at depths less than 150 μm was observed (Figure 3.9 B). This depth corresponds with the location of pyramidal cell basil dendrites. The latency of this sink relative to the proximal sink was very short ($<$ 4 ms) and in many cases overlapped in time (Figure 3.9 C). We noted that this

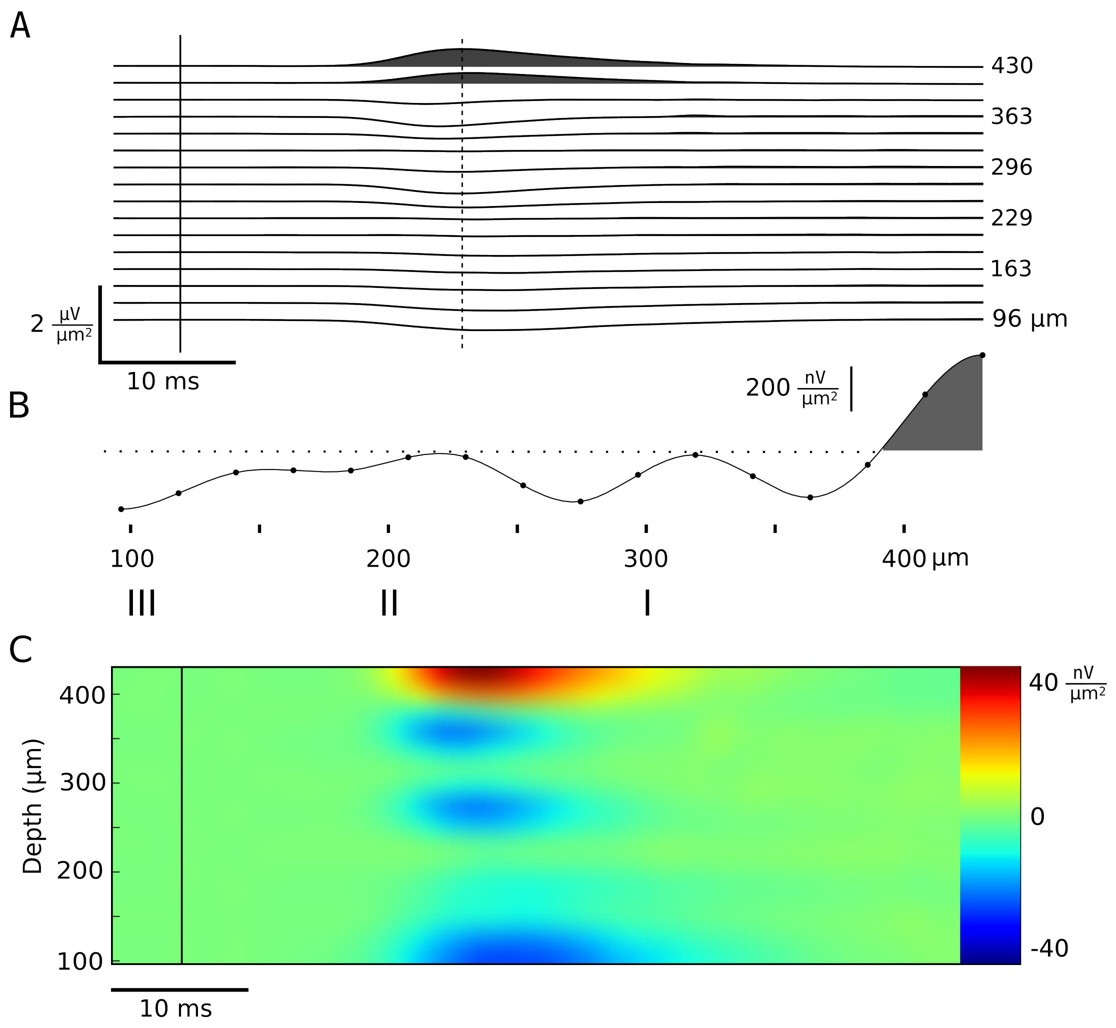


Figure 3.8: Current source density analysis of single extremum field potential data evoked by LFB stimulation. A. Time course of current sources and sinks in the cortex. Layer 1, where the proximal and distal dendrites of pyramidal cells are located, is at depths greater than 250 μm. Layer 2, where the somata of pyramidal cells are located, is at depths between 150 and 250 μm. Layer 3, where the axons and basilar dendrites of pyramidal cells are located, is below 150 μm. B. Depth profile of CSD at the time shown by the dashed line in A. A large sink is present at distal dendritic depths and is accompanied by sources located in layers 1 and 3. Note the absence of sinks and sources at the putative somatic depth. C. Heat map of the spatio-temporal profile of the CSD. The strongest source is in layer 3 and occurs concurrently with the sink at distal dendritic depths in layer 1. Data points in B and C are interpolated with a cubic spline.

sink was stronger than the secondary sink of layer of 1 and is consistent with studies that have simultaneously measured large negative field potentials and excitatory post-synaptic potentials in pyramidal cells while stimulating fibers in layer 3 [19]. Thus in summary, we found that afferent stimulation evoked three current sink events that were confined to the distal, proximal and basilar dendritic regions of pyramidal neurons.

3.4.2 Current source density analysis of intra-cortical stimulation

Afferent stimulation evoked two to three sinks in discrete laminae within the cortex. We tested whether we could selectively evoke one or more of these sinks by antidromic stimulation of associational fibers within the cortex. This could potentially reveal the laminar specific targets of the associational connections in the turtle cortical microcircuit. We applied a CSD analysis to the field potentials recorded in response to electrical stimulation of layer 3, where the axons and basilar dendrites of pyramidal cells are located. Stimulation of this layer evoked three sink events (Figure 3.10). The first two events were confined to the inner two-thirds of layer 1 (proximal dendritic depth) and layer 3 (basilar dendritic depth). These sinks occurred concurrently and were accompanied by sources located in the outer one-third of layer 1 (distal dendritic depth). Consistently, we again found no sources or sinks in the putative cell layer (layer 2). Following these initial sinks a tertiary sink developed in the outer one-third of layer and was accompanied by sources in the inner two-thirds of layer one and layer 3.

3.4.3 Current source density analysis of dorsal ventricular ridge stimulation

The ADVR is a secondary route by which visual information is relayed to the cortex [39]. This route serves to link the tectothalamic nuclei to the cortex via the nucleus rotundus (homolog of the mammalian pulvinar). We wondered whether this pathway targeted the same lamina in the cortex as observed in the LFB stimulation. We performed the CSD analysis on field potential responses recorded in the cortex to stimulation of the lateral ADVR and found three sink events (Figure 3.11). The earliest sink event occurred in layer three where the basilar dendrites of pyramidal cells are located. The secondary sink occurred in the inner third of layer 1 where the proximal dendrites are located. The last sink event occurred in the distal dendrites and was stronger than the previous two sink events. Thus the sequence of current sinks evoked by stimulation of the ADVR

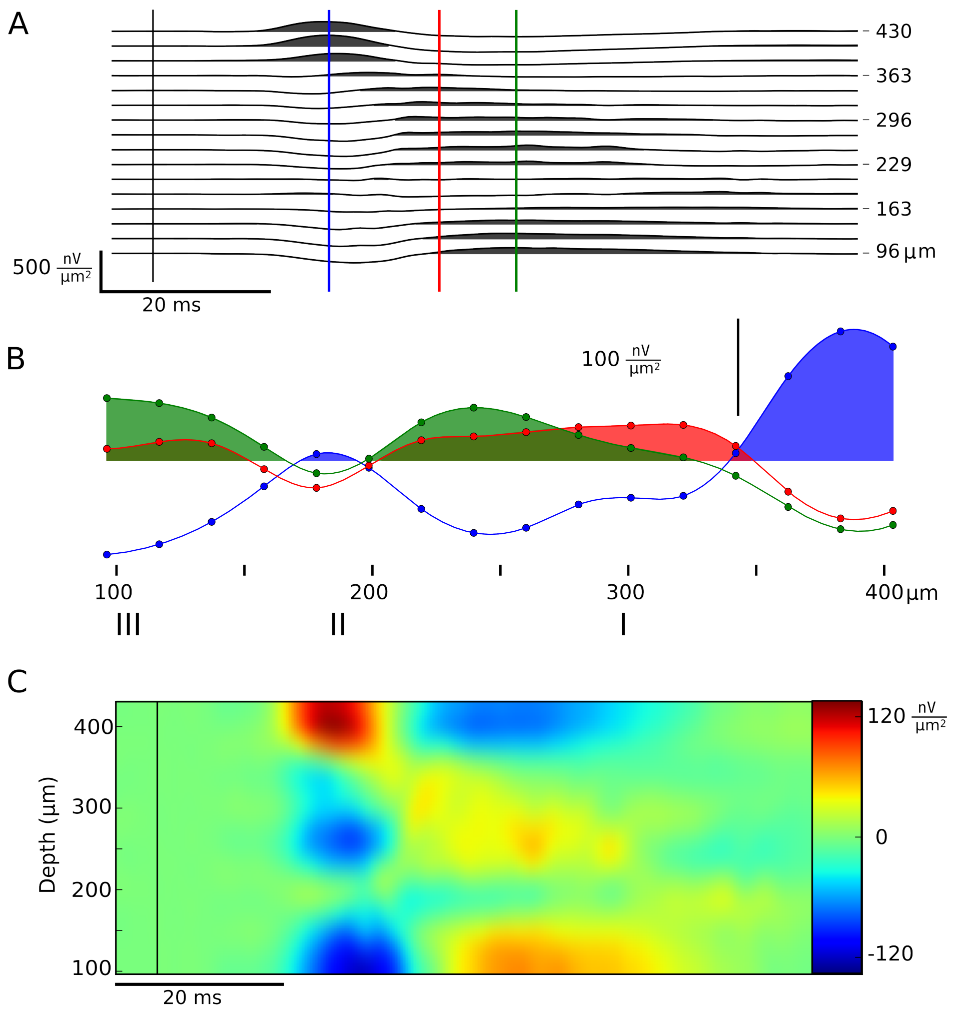


Figure 3.9: Current source density analysis of multiple extrema field potential data in response to LFB stimulation. A. Time course of current sources and sinks in the cortex. Layer 1, where the proximal and distal dendrites of pyramidal cells are located, is at depths greater than 250 μm. Layer 2, where the somata of pyramidal cells are located, is at depths between 150 and 250 μm. Layer 3, where the axons and basilar dendrites of pyramidal cells are located, is below 150 μm. B. Snapshots in time of the depth profile of CSD at the times shown by the colored lines in A. Blue trace depicts the initial sink event in the distal dendrites and the concurrent sources in layers 1 and 3. Red trace shows the development of a secondary sink in the inner two thirds of layer 1 and the development of the tertiary sink in layer 3. Green trace depicts an increase in the strength of the tertiary current sink in layer 3 along with concurrent sources located in layer 3. Data points are interpolated with a cubic spline. C. Heat map of the spatio-temporal profile of the CSD described in A and B, using a cubic spline interpolation.

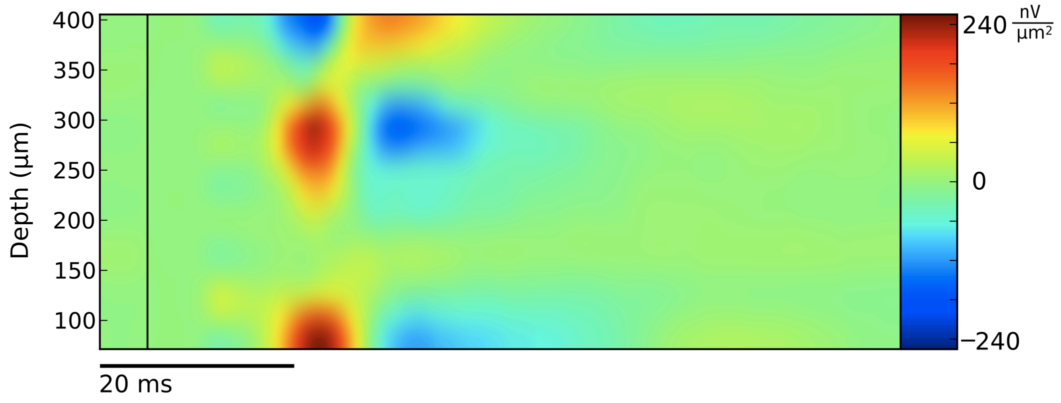


Figure 3.10: Current source density analysis of intra-cortical field potential data. Heat map of the spatio-temporal profile of the CSD obtained from intra-cortical field potentials. Stimulation evoked three sink events. The first two occurred in the inner two-thirds of layer 1 and layer 3. These events occurred simultaneously and were accompanied by sources in the outer third of layer 1. Following these initial sinks a third weaker tertiary sink developed in the outer third of layer 1 and was accompanied by sources in the inner two-thirds of layer 1 and layer 3.

are the reverse of the synaptic currents evoked by LFB stimulation.

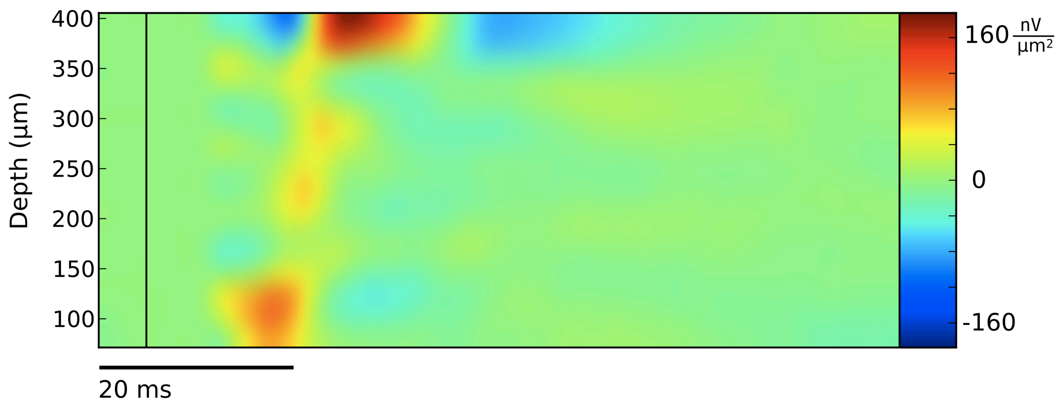


Figure 3.11: Current source density analysis of ADVR stimulation. Heat map of the spatio-temporal profile of the CSD obtained from intra-cortical field potentials reveals three sink events. The first sink begins in layer 3 and partially overlapped in time with a sink in the inner 2/3 of layer 1. These sinks were followed by a stronger sink in the outer 1/3 of layer 1.

3.5 Discussion

3.5.1 Turtle dorsal cortex as a model for laminar specific synaptic inputs

Comparisons of the synaptic organization of cortical circuits within and across species has begun to identify common circuit elements in the six-layered neocortex [37]. These elements include strong excitatory afferents onto the distal dendrites of pyramidal cells, recurrent excitation through pyramidal cell axon collaterals, and recurrent inhibition by inhibitory neurons. In mammals, these distinct types of inputs confine their synapses to distinct laminae within the cerebral cortex [10, 33]. Here we have investigated the laminar distribution of excitatory inputs in a reduced cortical model system, the turtle dorsal cortex.

The turtle dorsal cortex is a three-layered structure with three electrophysiologically distinguishable neuron types, namely excitatory pyramidal [6, 19] and inhibitory subpial and stellate neurons [3, 5, 22] (Figure 3.1 C). The physiological properties of these neuron types are very similar to their mammalian cortical counterparts [6]. These neurons receive a projection from the thalamus (Figure 3.1) via geniculocortical axons that bear varicosities *en passant* and contact a line of pyramidal and subpial neurons [8, 15, 25] (Figure 3.1 C). This distributed input onto the distal dendrites of pyramidal cells is a feature that is shared with the lateral olfactory tract input to the olfactory cortex [2, 13, 14] and the perforant pathway onto CA3 pyramidal neurons in the hippocampus [1, 4, 17]. Here, we investigated the laminar distribution of thalamic inputs to the turtle dorsal cortex.

In mammalian cortices it is well-establised that pyramidal neurons are linked by a dense set of cortico-cortical connections that can span several millimeters in the horizontal direction [12, 13]. In the hippocampus, recurrent excitation by Schaffer collaterals is confined to the strata radiatum in CA1 [36]. Similarly in the olfactory cortex, recurrent excitation from superficial pyramidal cells is confined to the apical dendrites of deep pyramidal neurons [35]. Anatomical evidence [7] suggests that long-range horizontal connections are also present in the turtle dorsal cortex but the laminar distribution has not been explored. We have determined the laminar distribution of long-range excitatory synaptic connections in the turtle cortical system. Beyond pyramidal cell connectivity, striking similarities in the synaptic organization of the inhibitory populations are shared between turtle and mammalian cortices. Specifically, subpial neurons are strongly activated by thalamic input and intracellular studies have revealed that this input constitutes a strong feedforward inhibition onto pyramidal cells [5, 18, 19, 21, 22]. In primary sensory cortices of mammals, this early inhibition has been shown to play an important role in shaping the response of cortical pyramidal neurons to excitatory sensory input [9, 11, 29, 40]. We have examined the excitatory inputs in the turtle cortical system to determine whether a CSD analysis supports prior intracellular evidence of feedforward inhibition in the turtle cortical network.

Lastly, the turtle dorsal cortex supports a population of interneurons, the stellate cells, that are thought to provide feedback inhibition to the pyramidal cell population [5, 18, 19, 21, 22]. In mammals, recurrent inhibitory synaptic patterns are common across all cortical regions [37]. In turtles, stellate interneurons are found in all layers so it was previously not clear what the laminar distribution of excitation onto these neurons would look like. Our CSD analysis has revealed sinks in specific laminae that may indicate the recruitment of laminar specific interneurons following afferent stimulation.

3.5.2 Field potentials

LFB evoked cortical potentials

The laminar organization of pyramidal cells coupled with a distributed synchronous synaptic input from the thalamus leads to large field potentials in response to afferent stimulation of the LFB. We found that we could classify these field potentials into two broad groups, those that invert in depth and those that do not. We mapped the inversion depths and found that sites located in the rostral cortex were more likely to invert than field potentials recorded in the caudal cortex. In addition, we found that the inversion depths were deeper at recording sites closer to the ADVR. This is consistent with anatomical evidence that has demonstrated that the LFB axons synapse onto the proximal dendrites of neurons located in the lateral cortex and shifts closer to the pial surface to contact the distal dendrites of neurons as the lateral-to-medial cortical distance increases [39]. In addition, we determined the latency of the field potential with respect to the stimulus current and found that the latency tended to increase with increasing rostral-to-caudal distance. This finding agrees with earlier voltage-sensitive dye studies that found that retinal excitation evokes waves of electrical activity in the cortex that typically begin in the rostral cortex and spread caudally in time [30, 31, 34].

Intra-cortical evoked cortical field potentials

Intra-cortical stimulation evoked biphasic and triphasic field potentials that did not invert in depth. Interestingly, we found that stimulation of the cortex several millimeters away from the recording site could reliably evoke field potentials with weak stimulating currents. This finding suggest that excitatory collaterals from pyramidal neurons affect not only neighboring pyramidal cells but affect the responsiveness of distant cortical locations as well. This supports earlier findings which have described large cortical receptive fields that respond to small moving stimuli anywhere in the visual field [23].

ADVR evoked cortical field potentials

The ADVR is homologous to the extrastriate cortex of mammals. We found that stimulation of the ADVR evoked biphasic and triphasic field potentials that did not invert over the depth of the cortex. The latency of the peak of these field potentials was nearly identical to the latency measured for LFB stimulation and raises the interesting question of how inputs from the tecto-thalamic and retino-geniculate pathways are integrated at the level of the cortex.

3.5.3 CSD analysis

We applied a current source-density analysis to field potentials evoked by LFB fiber stimulation. We found that the resulting CSDs could be grouped into two response patterns that were correlated with the shape of the field potentials. Field potentials with a single extremum yielded a CSD profile with a single sink located at depths consistent with the distal apical dendrites of pyramidal cells. The latency of this sink was approximately 20 ms from the stimulus and lasted for approximately 10 ms. Early work examining the responses of pyramidal cells to LFB stimulation found that pyramidal cells typically respond with a long duration ipsp following weak stimulation. However, at higher current intensities pyramidal cells responded with a single spike that rose abruptly from the ipsp [19]. Given that the responses were primarily inhibitory, the authors concluded that the observed spikes resulted from an excitation that was electronically remote from the somatic recording site [19]. Our results here confirm this finding by demonstrating that thalamic excitation occurs on the distal apical dendrites of pyramidal cells. Furthermore, the laminar position and strength of this excitation supports anatomical [15, 38] and intracellular studies [5, 18, 19, 21, 22] that have suggested that thalamic excitation strongly activates interneurons in layer 1 that provide feedforward inhibition onto pyramidal neurons.

Field potentials with multiple extrema yielded CSD profiles featuring two to three current sink events. The initial current sink occurred at depths consistent with the distal dendrites of pyramidal cells. As discussed in the single extremum case, this current sink is likely thalamic excitation of pyramidal cells and subpial interneurons.

Following this initial sink a second sink occurred about 15 ms later and was isolated at a depth consistent with the proximal dendrites of pyramidal cells. This finding supports electron micrograph (em) studies that found asymmetrical synapses on spines within the inner third of layer 1 [**?**]. In addition, prior intracellular studies found that excitation of associational fibers in layer 3 evoked epsps in pyramidal neurons [19]. Our data further supports these findings by demonstrating that the timing of this current sink is consistent with a recurrent disynaptic excitation. Lastly, stimulation of afferent fibers evokes late onset inhibition at the soma of pyramidal cells [22]. Our results provide evidence for excitation of stellate neurons in the inner 2/3 of layer 1 that are likely responsible for this late inhibition.

A third sink that often occurred concurrently with the proximal current sink was observed in layer 3. The depth of this sink is consistent with the location of pyramidal cell basil dendrites. This synaptic connection, to our knowledge, has not been described in prior anatomical or physiological studies. The density of neurons in this layer is quite low so we tested whether this sink could reflect intra-cortical connections between pyramidal neurons. We stimulated the associational fiber layer (layer 3) and found that sinks could be evoked at recording locations several millimeters away. The spatio-temporal profile of this CSD showed three sink events. The first and second sinks occurred concurrently in the inner 2/3 of layer 1 and layer 3. This is consistent with excitatory cortico-cortical connections observed in stimulation of the LFB. These sinks were then followed by a distal apical dendritic sink. It is unlikely that this sink results from antidromic activation of the thalamus followed by orthodromic excitation of pyramidal cells because the latency of this sink relative to the basilar sink was less than 10 ms. This latency is far shorter than the 20 ms latency recorded in response to LFB stimulation. A second and more likely possibility is that recurrent pyramidal cell collaterals also synapse near the distal apical dendrites where they possibly contact subpial and stellate interneurons. We did not detect this sink current in our LFB stimulation recordings because the distal dendritic sink was always followed by a source in time. This source likely masked this distal current sink. Future experiments that severe the geniculo-cortical fibers would be valuable in confirming this hypothesis.

In addition to geniculate inputs, the dorsal cortex is reciprocally connected to the ADVR (homolog of mammalian extrastriate cortex). We examined the laminar distribution of excitatory sinks in response to stimulation of the ADVR and compared these results with LFB fiber stimulation. The CSD profile again revealed three sink currents confined to specific laminae. The first sink occurred at a depth consistent with the laminar position of the basilar dendrites of pyramidal neurons. The latency of this sink was on the order of 10 ms, shorter than LFB stimulation latencies. Following this initial sink, a second sink occurred at a depth consistent with the laminar position of the proximal dendrites of pyramidal cells. Lastly, a third sink occurred at a depth consistent with the distal dendrites of pyramidal cells. Again, the latency of this sink was not consistent with antidromic stimulation of the thalamus. Thus stimulation of the ADVR results in sinks that begin in the basilar dendritic region and shift to more proximal and then distal locations in time. This raises an interesting question about the integration of excitatory inputs from the tectal and geniculate pathways that can be addressed through concurrent stimulation.

To summarize our results, we have provided further evidence that the turtle cortical system shares many of the same synaptic features that characterize the olfactory, hippocampal and neocortical circuitry. These include synaptic inputs to the distal dendrites of pyramidal neurons, recurrent excitation to the proximal and basilar dendrites, and likely activation of inhibitory neurons within specific laminae. In addition, we also determined the laminar distribution of extrastriate (ADVR) inputs into the cortex and found differences in the sequence of laminar inputs when compared with LFB stimulation.

References

[1] P Andersen. Interhippocampal impulses i. origin, course and distribution in cat, rabbit and rat. *Acta Physiologica Scandinavica*, 47(1):63–90, 1960.

[2] MA Biedenbach and CF Stevens. Synaptic organization of cat olfactory cortex as revealed by intracellular recording. *Journal of neurophysiology*, 32(2):204–14, March 1969.

[3] MG Blanton, JM Shen, and AR Kriegstein. Evidence for the inhibitory neurotransmitter gamma-aminobutyric acid in aspiny and sparsely spiny nonpyramidal neurons of the turtle dorsal cortex. *The Journal of comparative neurology*, 259(2):277–97, May 1987.

[4] R Cajal. Le lobe optiques des vertebrates inferiuers, toit optique des oiseaux. *Histologie de systeme nerveux d l'homme et des vertebrates*, pages 196–212, 1911.

[5] JB Colombe, J Sylvester, J Block, and PS Ulinski. Subpial and stellate cells: two populations of interneurons in turtle visual cortex. *The Journal of comparative neurology*, 471(3):333–51, April 2004.

[6] BW Connors and AR Kriegstein. Cellular physiology of the turtle visual cortex: distinctive properties of pyramidal and stellate neurons. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 6(1):164–77, January 1986.

[7] CE Cosans and PS Ulinski. Spatial organization of axons in turtle visual cortex: intralamellar and interlamellar projections. *The Journal of comparative neurology*, 296(4):548–58, June 1990.

[8] FF. Ebner and M Colonnier. A quantitative study of synaptic patterns in turtle visual cortex. *The Journal of Comparative Neurology*, 179(2):263–276, 1978.

[9] D Ferster and B Jagadeesh. EPSP-IPSP interactions in cat visual cortex studied with in vivo whole-cell patch recording. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 12(4):1262–74, April 1992.

[10] E Förster, S Zhao, and M Frotscher. Laminating the hippocampus. *Nature reviews. Neuroscience*, 7(4):259–67, April 2006.

[11] L Gabernet, SP Jadhav, DE Feldman, M Carandini, and M Scanziani. Somatosensory integration controlled by dynamic thalamocortical feed-forward inhibition. *Neuron*, 48(2):315–27, October 2005.

[12] CD Gilbert and TN Wiesel. Clustered intrinsic connections in cat visual cortex. *The Journal of Neuroscience*, 3(5):1116, 1983.

[13] LB Haberly. Summed potentials evoked in opposum prepyriform cortex. *Journal of Neurophysiology*, 36:775–788, 1973a.

[14] LB Haberly. Unitary analysis of opposum prepyriform cortex. *Journal of Neurophysiology*, 36(4):762, 1973b.

[15] WC Hall and FF Ebner. Thalamotelencephalic projections in the turtle (Pseudemys scripta). *The Journal of comparative neurology*, 140(1):101–22, September 1970.

[16] https://github.com/davidlmorton/pycsd. : Link to current source density analysis code hosted on github code repository.

[17] ER Kandel, WA Spencer, and FJ Brinley. Electrophysiology of hippocampal neurons. I. Sequential invasion and synaptic organization. *Journal of Neurophysiology*, 24(3):225, 1961.

[18] AR Kriegstein. Synaptic responses of cortical pyramidal neurons to light stimulation in the isolated turtle visual system. *The Journal of Neuroscience : the official journal of the Society for Neuroscience*, 7(8):2488–92, August 1987.

[19] AR Kriegstein and BW Connors. Cellular Physiology Synaptic Properties of the Turtle Visual Cortex : and Intrinsic Circuitry. *The Journal of Neuroscience*, 6(January):178–191, 1986.

[20] J Malmivou and R Plonsey. *Bioelectromagnetsim*. Oxford Univ Press, New York, 1995.

[21] JG Mancilla, M Fowler, and PS Ulinski. Responses of regular spiking and fast spiking cells in turtle visual cortex to light flashes. *Visual neuroscience*, 15(5):979–93, 1998.

[22] JG Mancilla and PS Ulinski. Role of GABA(A)-mediated inhibition in controlling the responses of regular spiking cells in turtle visual cortex. *Visual neuroscience*, 18(1):9–24, 2001.

[23] PZ Mazurskaya. Organization of receptive fields in the forebrain ofEmys orbicularis. *Neuroscience and Behavioral Physiology*, 6(4):311–318, October 1973.

[24] U Mitzdorf. Current source-density method and application in cat cerebral cortex: investigation of evoked potentials and EEG phenomena. *Physiological reviews*, 65(1):37–100, January 1985.

[25] KA Mulligan and PS Ulinski. Organization of geniculocortical projections in turtles: isoazimuth lamellae in the visual cortex. *The Journal of comparative neurology*, 296(4):531–47, June 1990.

[26] C. Nicholson and R. Llinas. Real time current source-density analysis using multi-electrode array in cat cerebellum. *Brain Research*, 100(2):418 – 424, 1975.

[27] B Pesaran. Uncovering the mysterious origins of local field potentials. *Neuron*, 61(1):1–2, January 2009.

[28] KH Pettersen, A Devor, I Ulbert, AM Dale, and GT Einevoll. Current-source density estimation based on inversion of electrostatic forward solution: effects of finite extent of neuronal activity and conductivity discontinuities. *Journal of neuroscience methods*, 154(1-2):116–33, July 2006.

[29] C Poo and JS Isaacson. Odor representations in olfactory cortex: ”sparse” coding, global inhibition, and oscillations. *Neuron*, 62(6):850–61, June 2009.

[30] JC Prechtl, TH Bullock, and D Kleinfeld. Direct evidence for local oscillatory current sources and intracortical phase gradients in turtle visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 97(2):877–82, January 2000.

[31] JC Prechtl, LB Cohen, B Pesaran, PP Mitra, and D Kleinfeld. Visual stimuli induce waves of electrical activity in turtle cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 94(14):7621–6, July 1997.

[32] A Reiner. Neurotransmitter organization and connections of turtle cortex: implications for the evolution of mammalian isocortex. *Comparative Biochemistry and Physiology Part A: Physiology*, 104(4):735–748, April 1993.

[33] JR Sanes and M Yamagata. Formation of lamina-specific synaptic connections. *Current Opinion in Neurobiology*, 9(1):79–87, 1999.

[34] DM Senseman. Spatiotemporal structure of depolarization spread in cortical pyramidal cell populations evoked by diffuse retinal light flashes. *Visual neuroscience*, 16(1):65–79, 1999.

[35] GM Shepherd. Current-Density Evoked Potentials Analysis of Summed Prepyriform Cortex in Opossum. *Interpretation A Journal Of Bible And Theology*, 1973.

[36] GM Shepherd, editor. *The synaptic organization of the brain*. Oxford, New York, 5th edition, 2004.

[37] GM Shepherd. The microcircuit concept applied to cortical evolution: from three-layer to six-layer cortex. *Frontiers in Neuroanatomy*, 5(0), 2011.

[38] LM. Smith, FF. Ebner, and M Colonnier. The thalamocortical projection in pseudemys turtles: A quantitative electron microscopic study. *The Journal of Comparative Neurology*, 190(3):445–461, 1980.

[39] PS Ulinski. *The cerebral cortex of reptiles*. Chicago, 1990.

[40] WB Wilent and D Contreras. Dynamics of excitation and inhibition underlying stimulus selectivity in rat somatosensory cortex. *Nature neuroscience*, 8(10):1364–70, October 2005.

Chapter 4

Spikepy: A Flexible Spike-sorting Application and Framework

David Morton and Ralf Wessel

Manuscript in preparation to be submitted

to Frontiers in Neuroinformatics

The analysis of extracellular neural recordings usually begins with spike-sorting. There is an active discussion within the research community about how to best perform spike-sorting. Spikepy is a flexible spike-sorting application that serves to facilitate this discussion in two important ways. Firstly, it provides an extensible plugin-based framework that can be used as a test bed for new spike-sorting algorithms. And importantly, it is an easy-to-use application so the average electrophysiologist can have access to the latest spike-sorting advances.

4.1 Introduction

Recording the electrical potential from extracellular electrodes remains a popular technique in neuroscience. The extracellular signals recorded may include spikes (the remnants of neuronal action potentials) originating from multiple neurons. The category of analysis that involves both detecting these spikes and associating them with the correct source is called spike-sorting.

Spike-sorting research has been around for more than two decades and is still an active area of research [1, 8]. Much of the early tools developed required a great deal of manual control, but within the last ten years or so a lot of effort has been spent to automate the spike-sorting process [1, 2, 7, 9, 10]. This is due in no small part to the growing popularity and shrinking cost of multielectrodes, which make manual sorting techniques unappealing.

The automation of spike-sorting processes should result in a lively discussion of the merits and costs of one algorithm over another, yet this is largely absent. The process of spike-sorting is complex, and often involves many steps to complete. Algorithm developers often focus on only one step in the process. When they then go to implement their new algorithm, they are forced to also implement the other steps, before they can evaluate the performance of the algorithm. As a result, the comparisons are not direct, since these other steps often influence performance in a significant way.

This paper introduces a framework that aims to address this by providing a standardized platform to perform spike-sorting. Spikepy is a plugin-based framework, where each step is a type of plugin, allowing algorithm developers to focus on just the step they are interested in. The framework takes care of creating both a graphical user interface as well as a full featured API. Additionally, the framework handles reading data in and exporting to various file formats via plugins. Many of the most popular algorithms are already written as plugins, making Spikepy a spike-sorting solution that is ready to be used without writing any code at all.

Spikepy is free and open-source software, released under the GPL version 3 software license. Spikepy will run in Windows, Mac osX and Linux. You can download Spikepy from http://code.google.com/p/spikepy where there are installation instructions, sample data, as well as other documentation. There is also a mailing list at http://groups.google.com/group/spikepy-users.

4.2 Methods

4.2.1 Plugin Framework

Spikepy is built around the plugin concept. Plugins are responsible for almost everything in Spikepy from loading data, to processing data, even visualizing results. Spikepy is responsible for coordinating these plugins and providing the user interface either through the GUI or as an API. This coordination is made possible through the use of `Trial` objects and their associated `Resources`.

`Trial` objects are a way of grouping data from various stages of processing together. The primary role of `Trial` objects is to provide access to their `Resources` which are where the data are stored. `Resources` store information about how the data were last altered and provide a mechanism to lock/unlock access to the data to facilitate multi-processing. Spikepy plugins require that some `Resources` are available in order to run and provide `Resources` as their outputs. (Figure 4.1)

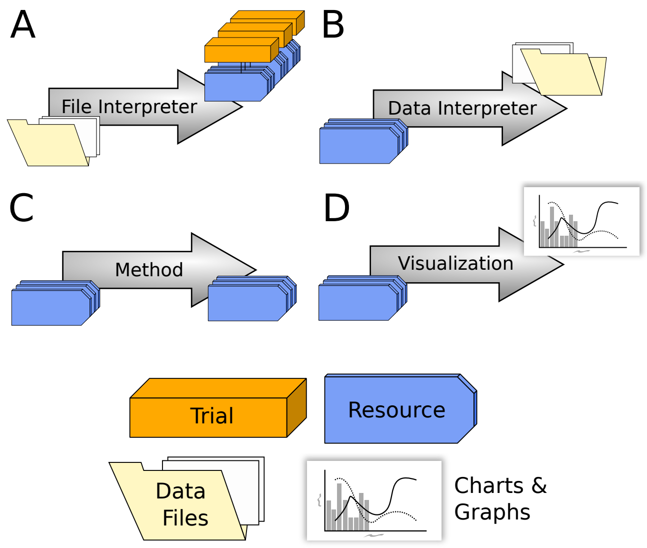


Figure 4.1: Spikepy plugin types. **(A)** `FileInterpreters` read in data files and create `Trial` objects. **(B)** `DataInterpreters` take `Resources` and produce data files. **(C)** `Methods` carry out the various spike-sorting processing stages. They require `Resources` as well as produce them. **(D)** `Visualizations` crate charts and graphs from `Resources`.

4.2.2 Processing Stages

Spike-sorting is often carried out in four steps starting with filtering, spike-detection, feature-extraction, and then clustering. Spikepy generally follows this paradigm but allows for the possibility of different filtering to happen before spike-detection and feature-extraction. (Figure 4.2) This makes it possible to filter once to detect spikes, disregarding the spike-shape distortion associated with the filter. Another filtering stage before the extraction stage is performed when spike-shape is potentially more important. This process is further generalized to include any number of auxiliary steps that can occur before or after any of these required processing stages.

Each of the stages in Figure 4.2 corresponds to a method-plugin family, including the auxiliary stages. Spikepy determines the order in which processing will occur by attempting to solve the dependencies of a plugin (what `Resources` the plugin requires). This is possible since all plugins tell Spikepy what `Resources` they produce as well as what `Resources` they require. When it is possible multi-processing will occur, utilizing all available processors on the computer (settings allow you to limit the number of processes).

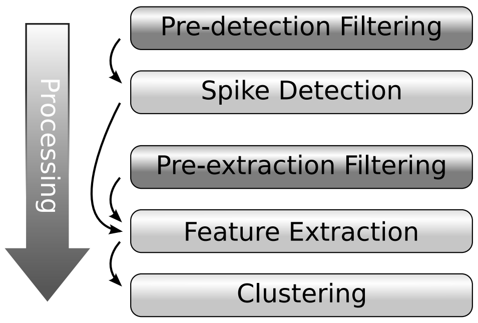


Figure 4.2: Spike-sorting processing stages. The five main processing stages are shown generally starting with filtering and ending with clustering. The two filtering stages are colored more darkly to indicate that they are entry-points to the spike-sorting process. `AuxiliaryMethods` can be run before or after any of the five required stages. Curved arrows indicate standard requirements (e.g. spike detection requires pre-detection filtering to have finished).

4.2.3 Extending Spikepy

Substantial effort has been made to ensure that writing plugins for Spikepy is simple and straightforward. Authors of new plugins do not have to know how the internals of Spikepy work. Figure 4.3 show the actual code that creates an auxiliary plugin that resamples a signal to a new sampling frequency. The actual function that resamples the signal is not shown, only the source that is required to turn it into a plugin.

Plugins are Python classes that inherit from one of the base-classes corresponding to the four plugin

```
 1 from spikepy.developer.methods import AuxiliaryMethod
 2 from spikepy.common.valid_types import ValidInteger
 3 from spikepy.utils.resample import resample
 4
 5 class ResampleAEF(AuxiliaryMethod):
 6     name = 'Resample after Extraction Filter'
 7     description = 'Resample the signal after running the Extraction Filter stage.'
 8     runs_with_stage = 'extraction_filter'
 9     requires = ['ef_traces', 'ef_sampling_freq']
10     provides = ['ef_traces', 'ef_sampling_freq']
11
12     # -- method parameters --  (become keyword arguments to run)
13     new_sampling_frequency = ValidInteger(10, 100000, default=30000)
14
15     def run(self, signal, sampling_freq, new_sampling_freq=30000):
16         return [resample(signal, sampling_freq, new_sampling_frequency),
17                 new_sampling_frequency]
```

Figure 4.3: Writing a Spikepy plugin. This code segment is all that is required to create a plugin that will resample the signal following the extraction-filtering stage. On lines 6 - 10 the class-variables tell Spikepy how to handle this plugin. Line 13 defines a plugin setting that allows the user to select the sampling frequency for the resampled signal. Spikepy will automatically build the GUI element for these settings. The run method on line 15 has positional arguments corresponding to the `Resources` that the plugin "requires", and keyword arguments that correspond to the plugin settings. Lines 16 - 17 return the data for the two `Resources` that this plugin "provides".

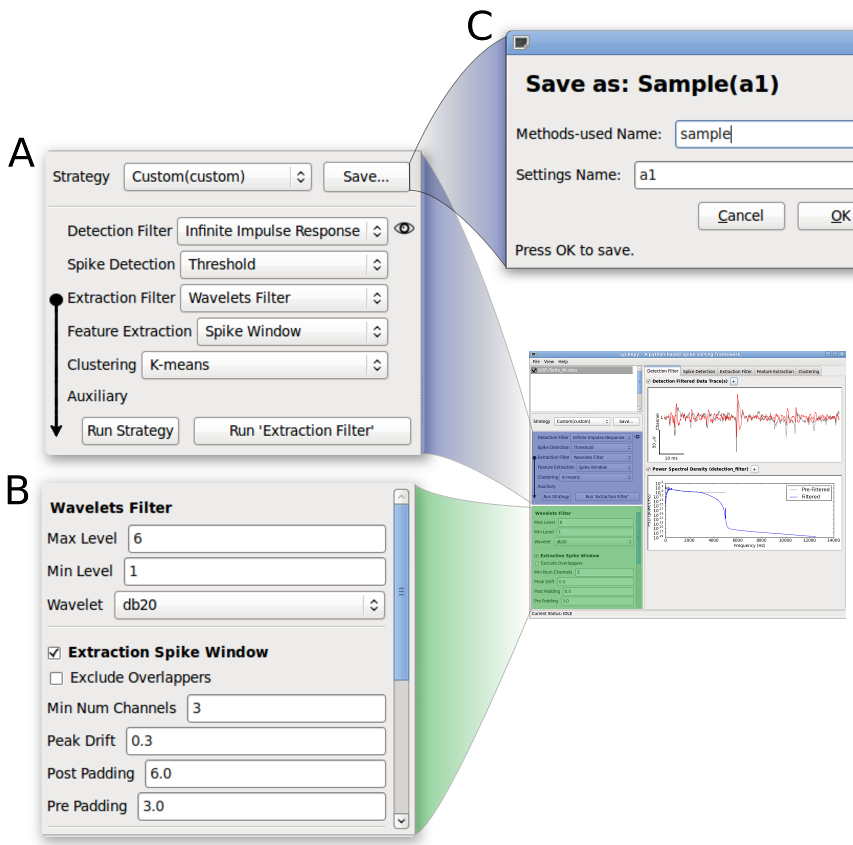
types. (Figure 4.1) Plugins have class-variables that tell Spikepy how to handle the plugin. Obligatory class-variables for `SpikepyMethods` include `requires` and `provides` which are lists of `Resource` names. There are also optional class-variables (described in the documentation) such as `is_stochastic` that help Spikepy figure out when it is or is not necessary to run a `SpikepyMethod` plugin again with the same settings.

4.2.4 Strategies

One consequence of the flexibility that the plugin system provides is that the user makes many choices about how the data are processed. These decisions should be recorded in some way so that the research is reproducible and so that data can be batch processed. In Spikepy a strategy is an index of all the plugins used as well as all of the settings associated with those plugins. These strategies can be saved and shared easily, making it possible to communicate clearly how you analyzed your data.

4.2.5 Graphical Interface

Spikepy has an easy to use, cross-platform GUI that allows for a more interactive spike-sorting session (Figure 4.4). The interface lets the user choose plugins to use and enter processing settings and either run the



Save as: Sample(a1)

Methods-used Name: sample

Settings Name: a1

Press OK to save.

Strategy: Custom(custom)

Detection Filter: Infinite Impulse Response

Spike Detection: Threshold

Extraction Filter: Wavelets Filter

Feature Extraction: Spike Window

Clustering: K-means

Auxiliary

Wavelets Filter

Max Level: 6

Min Level: 1

Wavelet: db20

☑ **Extraction Spike Window**

☐ Exclude Overlappers

Min Num Channels: 3

Peak Drift: 0.3

Post Padding: 6.0

Pre Padding: 3.0

Figure 4.4: The Spikepy graphical user interface. In the background is a screen-capture of the main window in the Spikepy GUI. **(A)** A zoomed in view of the strategy panel where users can choose the method for each processing stage, choose previously saved strategies, and run stages or whole strategies. **(B)** A zoomed in view of the control panel for the Extraction Filter stage. The first section is for the currently selected method (Wavelets Filter) while the controls below this are for auxiliary methods that are associated with the Extraction Filter stage (not all are shown – notice the scrollbar). Note: all GUI control panels are automatically built by Spikepy, so plugin authors do not need to know how to program a GUI.**(C)** A view of the save strategy dialog. Users are prompted for a methods name and a settings name. All strategies using the exact same set of methods share a methods name making it easier to keep strategies organized.

entire strategy or just one stage of processing at a time. The GUI then generates all the relevant visualizations of the results of processing. Under the hood though, the GUI is accessing the exact same API that is available to anyone who wants to write Python scripts that use Spikepy.

4.2.6 Application Programming Interface

The Spikepy API turns Spikepy into a Python library that can be utilized in any Python program. The API is organized around the `Session` object. The `Session` object provides methods for opening files, creating and updating strategies, and all the other actions available through the GUI. In addition to those methods, you can gain access to the underlying data in order to customize how the data are processed.

The arguments to many of the functions in the API are designed to make interactive programming (i.e. using iPython) easy. All arguments which are names, such as trial names, strategy names and plugin names will accept any unique substring instead of the entire name. For example, if the name of a plugin was `Some Long Plugin Name`, then the string `Long` would suffice as an argument, provided that there are no occurrences of `Long` in any other plugin names. That is, any unambiguous substring of a name can be used instead of the entire name.

4.3 Results

Spikepy is a plugin-based framework which means that it is easily extensible, in addition though, it comes prepackaged with a number of very useful plugins. In this section we will describe these builtin plugins, and compare and contrast them when appropriate. We will follow the standard processing order of stages as shown in Figure 4.2, and discuss auxiliary plugins throughout the section.

4.3.1 Filtering

There are two filtering stages in Spikepy, one that occurs just before the spike-detection stage, and one that occurs just before the feature-extraction stage. This means that a filter that is specially well suited to help with spike-detection can be used before the spike-detection stage and another filter that is more beneficial for feature-extraction can be used before that stage. From the standpoint of the plugin system, they are all simply filtering plugins.

There are three builtin filtering plugins as of Spikepy version 0.82, they are:

1. Infinite Impulse Response – IIR filters commonly referred to as Butterworth and Bessel type filters are so named because using them to filter an impulse (Dirac delta), the response will continue on infinitely in time.
2. Finite Impulse Response – FIR filters are also referred to as sinc type filters because they can be implemented by convolving a windowed sinc function with the signal. If you filter an impulse, the response will not continue on infinitely in time.

3. Wavelets – The signal is filtered by first decomposing the signal using wavelet decomposition, and then recomposing it using only some of the resulting wavelet coefficients, setting the others to zero.

These filters vary from one another in many ways. In terms of complexity the IIR filters are the least complex, followed by the FIR filters and finally the Wavelets filters. In terms of computation time required, the IIR filters are the fastest, followed by the Wavelets filters and finally the FIR filters. What is probably one of the most important qualities though is the ability of the filter to preserve spike shape while removing unwanted frequencies from the signal. In that regard, the FIR and Wavelets filters are about equally good, with the IIR filters distorting the waveform substantially more (Figure 4.5).

One drawback to using the Wavelets filter over the FIR filter, is that with the former you cannot specify the cutoff frequencies directly, instead you must specify the min-level and max-level. A simple formula relates these parameters to the cutoff frequencies:

$$\omega_{\text{low}} = \frac{\omega_{\text{s}}}{2^{L_{\text{min}}}}$$
$$\omega_{\text{high}} = \frac{\omega_{\text{s}}}{2^{(L_{\text{max}}+1)}}$$

where ω_{low} is the lower cutoff frequency, ω_{high} is the upper cutoff frequency, and ω_{s} is the sampling frequency of the signal. L_{min} and L_{max} are the min and max levels respectively. This means that the sampling frequency of the signal essentially limits the choices of cutoff frequencies that the Wavelets filters can achieve. This limitation is absent in the FIR filters.

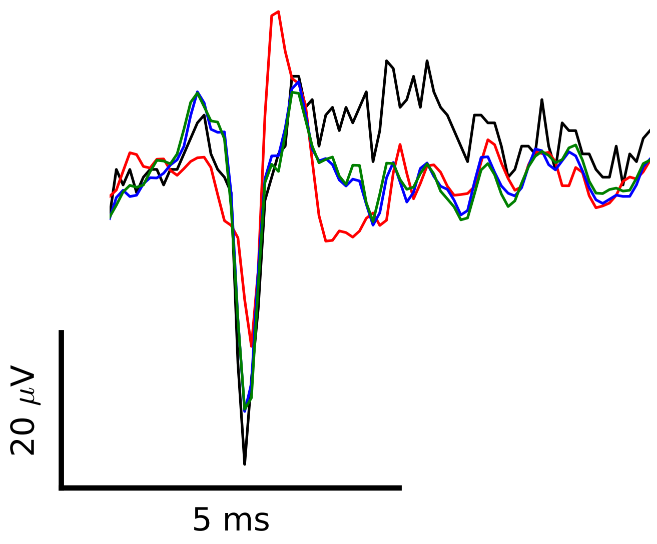


Figure 4.5: Comparing filter methods. **(black)** unfiltered **(red)** IIR filter: 3rd order Butterworth, band-pass (300-2,500 Hz) **(blue)** FIR filter: window=Hanning, taps=1001, band-pass (300-2,500 Hz) **(green)** Wavelets filter: wavelet=db20, min_level=2, max_level=4 (equivalent to band-pass (312-2,500 Hz) for this sampling frequency (10,000 Hz))

4.3.2 Spike detection

There is only one `DetectionMethod` plugin that comes with Spikepy and it offers threshold spike-detection. Before we describe this plugin however, it should be noted that there are a couple of auxiliary plugins that are useful in conjunction with the spike-detection plugins. First, there is an auxiliary plugin that will resample the signal after filtering. This is usually used to upsample the signal so that spikes are better aligned after detection. Also, there is an auxiliary plugin that implements the non-linear energy operator [4]. This plugin accentuates the spikes relative to the noise (Figure 4.6), and may aide in spike-detection.

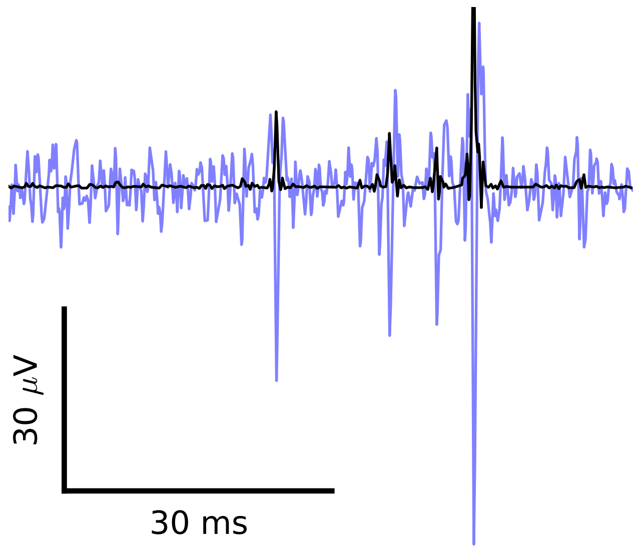


Figure 4.6: The non-linear energy operator auxiliary plugin. **(light-blue)** FIR filter: window=Hanning, taps=1001, band-pass (300-2,500 Hz) **(black)** Signal filtered in the same way, but applying the non-linear energy operator afterward. (result scaled by 30x)

The threshold spike-detection plugin that Spikepy comes with has a number of useful options. The plugin allows you to specify either one or two thresholds (see Figure 4.7). These thresholds can be specified in mV, or as a multiple of the standard deviation of the signal. Another possibility is to specify the threshold(s) as a multiple of the median value of the absolute value of the signal (calculated after removing the mean-value of the signal). In addition to the threshold(s), you can specify a max-spike-duration which will cause the algorithm to ignore spikes which are unphysiologically long (i.e. from recording artifacts or low frequency oscillations). And finally, you can enforce a refractory period, causing the algorithm to remove spikes which occur too soon after previously detected spikes.

Looking forward, we would like to see additional spike-detection algorithms implemented. One promising method uses the Cepstrum of Bispectrum [12, 13]. Another method involves using wavelets as spike-templates [6]. Yet another possibility is to threshold the temporal derivative(s) of the signal. This could be achieved by adding another auxiliary plugin similar to the nonlinear energy operator plugin.

4.3.3 Feature extraction

Spikepy currently has two plugins to extract features from the signal given the spike times. The first is a simple spike-windowing plugin where a portion of the extraction-filtered signal is cut out surrounding each

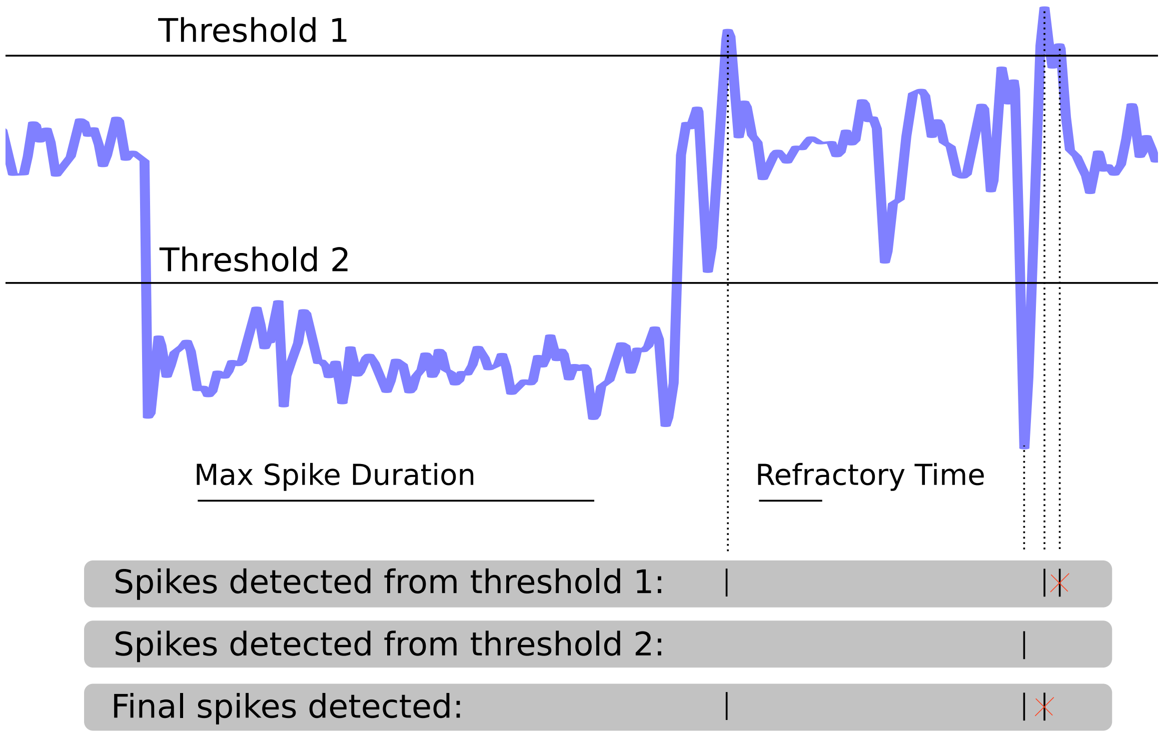


Figure 4.7: Threshold spike-detection plugin. You can specify up to two thresholds, a maximum spike duration, and a refractory time. The thresholds can be in various units such as mV, or standard deviations of the signal. The maximum spike duration and refractory time are in ms. Spikes that violate the maximum spike duration or refractory period are thrown out.

detected spike (Figure 4.8). Since the output of the detection stage is simply a list of spike times, one list per recording channel, this plugin must first determine if the spikes recorded on multiple channels are in fact the same event. The settings `peak_drift` and `min_num_channels` specify how many channels must register a spike, and within what period of time for them to be grouped together as a single event.

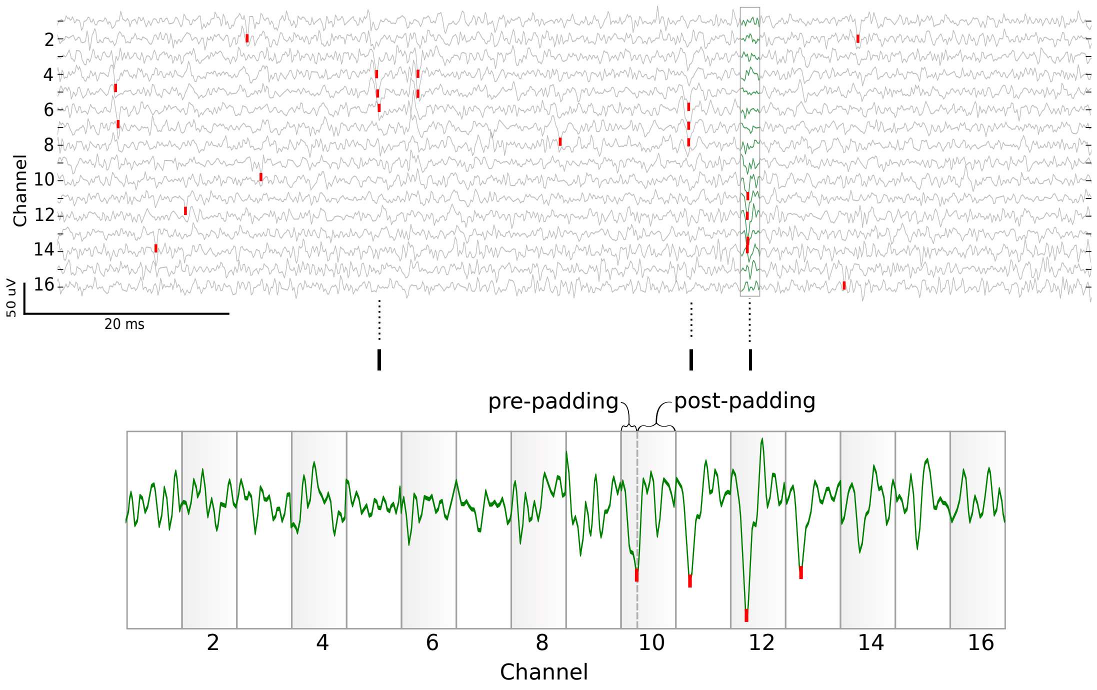


Figure 4.8: Spike window feature-extraction. **(top)** A short segment of a 16-channel extracellular recording showing spikes detected (red tick marks). **(middle)** A raster of spikes (in black) that meet the criteria of being on at least 3 channels within 0.3 ms of one another (`peak_drift=0.3` and `min_num_channels=3`). **(bottom)** The highlighted region in the top traces has been serialized to form a feature-vector. The parameters `pre_padding` and `post_padding` determine how much time before and after the spike to snip out of the traces.

The other feature-extraction plugin that comes with Spikepy is the method used in a paper by Rodrigo Quiroga et al. [10]. This method uses the previous method as a starting point, then it utilizes wavelet decomposition to obtain wavelet coefficients. The distribution of these coefficients are then tested for normality and those coefficients with the lowest scores (largest difference from the normal distribution) are chosen. This results in a lowering of the dimensionality of the feature-vector, while maintaining or perhaps even enhancing the separation of events in feature-space.

Many other feature-extraction methods exist, and are often called dimensionality-reduction techniques since they take the high-dimensional spike-window and return something with a much smaller feature-vector.

Principal Component Analysis, or PCA could be made into a feature-extraction plugin rather easily. Also easy to implement would be something as simple as spike-height and spike-width, or other scalar features of spikes such as spike-power.

4.3.4 Clustering

Spikepy has only one builtin clustering plugin as of version 0.82, the plugin implements the k-means clustering algorithm. There are a number of algorithms that we desire to have included in future releases of Spikepy. The super paramagnetic clustering algorithm that is used by WaveClus [10], a mixture-model algorithm (gaussians, student-t distributions, ect), and hierarchical clustering all seem interesting.

Spikepy has a number of clustering quality metrics implemented as an auxiliary plugin (Figure 4.9). These are the metrics described by Schmitzer-Torbert et al. [11], isolation distance, and L-ratio. The isolation distance is better if it is large, and the L-ratio is better if it is small. The actual magnitude of these metrics is dependent on the number of features that are being clustered, as well as the size of the feature vectors. They are therefore not useful for comparing the clustering quality achieved with other datasets unless they are of similar size. Additional clustering metrics should be made into auxiliary plugins such as those described in a recent paper by Hill et al. [5].

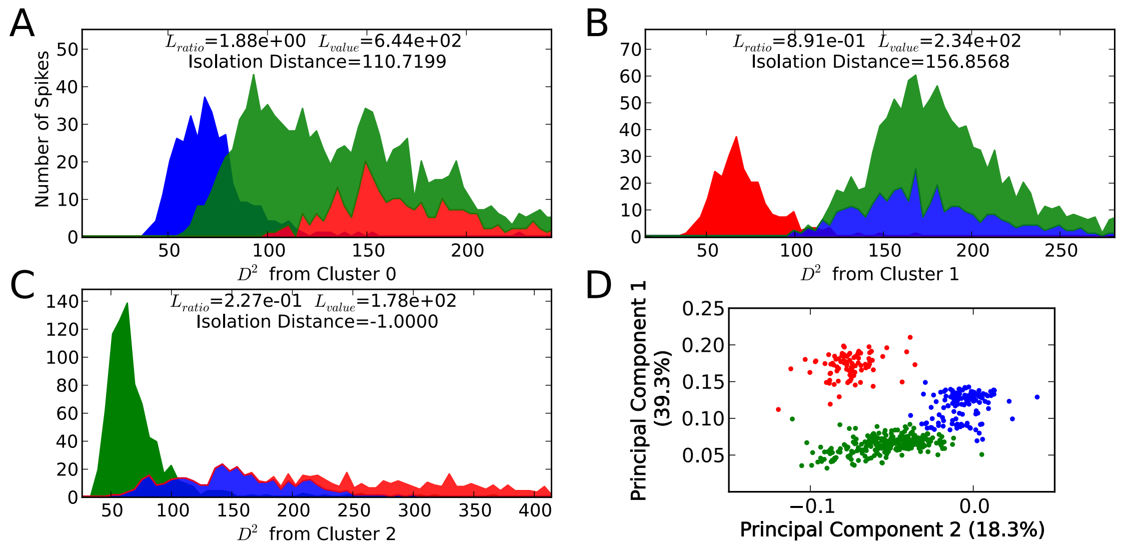


Figure 4.9: Clustering quality metrics. **(A-C)** A histogram of the squared Mahalanobis distance from the cluster center is shown for each of the three clusters. The histograms of the two 'other' clusters are drawn stacked on top of each other. The L-ratio, L-value, and isolation quality are also shown for each cluster. Notice in (C) that the isolation distance is undefined, this is because this cluster contains more features than the other two clusters combined. **(D)** The data projected onto the first two principal components gives some idea of the clustering quality. The percentages shown indicate the amount of variance explained by the first two principal component vectors.

4.4 Discussion

Even as new high-tech imaging techniques that are capable of recording the spiking activity of neurons are gaining favor, recording with extracellular electrodes remains very popular. To get the most out of these recordings, reliable spike-sorting must be performed. Because of the trend towards more and more electrodes, as well as the numerous brain/computer interface applications, this process must become automated. A significant impediment to reliable and automated spike-sorting is the lack of a common framework that would allow easy comparisons between competing spike-sorting algorithms.

Existing spike-sorting solutions such as Spike2 or WaveClus are either proprietary software, or require proprietary software in order to run. In addition, they are not written with extensibility in mind, so new algorithms cannot be added easily. Notable exceptions to this include an open-source project called open-electrophy [3] and the cloud computing system called Carmen [14], although neither is designed specifically for spike-sorting and require you to commit your entire dataflow to use their system.

We have described the construction of both a flexible plugin-based framework to perform spike-sorting as well as a core set of plugins. It is cross-platform and entirely free open-source software meaning that it can be adopted in virtually any laboratory and will be maintained for as long as the community finds it useful. Those who need to perform spike-sorting in their workflow will find the combination of a user-friendly graphical user interface and a powerful application programming interface makes Spikepy very useful. Additionally, researchers who are developing new spike-sorting algorithms will want to take advantage of the plugin-based framework that allows for very quick prototyping (they won't have to write an entire spike-sorting application) and will give them the ability to easily compare the effectiveness of their algorithms to others. Moreover, it connects these two populations by getting the latest advancements into the hands of those who need to perform spike-sorting but who may not be experts in the specific sub-domains underlying such advancements.

References

[1] Gaute T Einevoll, Felix Franke, Espen Hagen, Christophe Pouzat, and Kenneth D Harris. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current opinion in neurobiology*, 22(1):11–7, February 2012.

[2] Felix Franke, Michal Natora, Clemens Boucsein, Matthias H J Munk, and Klaus Obermayer. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *Journal of Computational Neuroscience*, pages 127–148, 2010.

[3] Samuel Garcia and Nicolas Fourcaud-Trocmé. OpenElectrophy: An Electrophysiological Data- and Analysis-Sharing Framework. *Frontiers in neuroinformatics*, 3(May):14, January 2009.

[4] Sarah Gibson, Jack W Judy, and Dejan Marković. Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 18(5):469–78, October 2010.

[5] Daniel N Hill, Samar B Mehta, and David Kleinfeld. Quality metrics to accompany spike sorting of extracellular signals. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 31(24):8699–705, June 2011.

[6] Kyung Hwan Kim and Sung June Kim. A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio. *IEEE transactions on bio-medical engineering*, 50(8):999–1011, August 2003.

[7] Sunghan Kim and James McNames. Automatic spike detection based on adaptive template matching for extracellular neural recordings. *Journal of neuroscience methods*, 165(2):165–74, September 2007.

[8] M S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network (Bristol, England)*, 9(4):R53–78, November 1998.

[9] Christophe Pouzat, Ofer Mazor, and Gilles Laurent. Using noise signature to optimize spike-sorting and

to assess neuronal classification quality. *Journal of neuroscience methods*, 122(1):43–57, December 2002.

[10] R Quian Quiroga, Z Nadasdy, and Y Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput*, 16(8):1661–1687, August 2004.

[11] N Schmitzer-Torbert, J Jackson, D Henze, K Harris, and a D Redish. Quantitative measures of cluster quality for use in extracellular recordings. *Neuroscience*, 131(1):1–11, January 2005.

[12] Shahjahan Shahid and Leslie S Smith. Cepstrum of bispectrum spike detection on extracellular signals with concurrent intracellular signals. *BMC Neuroscience*, 10(Suppl 1):P59, 2009.

[13] Shahjahan Shahid, Jacqueline Walker, and Leslie S Smith. A new spike detection algorithm for extracellular neural recordings. *IEEE transactions on bio-medical engineering*, 57(4):853–66, April 2010.

[14] L S Smith, J Austin, S Baker, R Borisyuk, S Eglen, J Feng, K Gurney, T Jackson, M Kaiser, P Overton, S Panzeri, R Quian Quiroga, S R Schultz, and E Sernagor. The CARMEN e-Science pilot project : Neuroinformatics work packages . *PLoS Computational Biology*, (September):591–598, 2007.