VScope

Multicamera image and electrophysiological data acquisition

By Daniel A. Wagenaar

Contact information:

Daniel A. Wagenaar
California Institute of Technology
1200 E. California Blvd. M/C 139-74
Pasadena CA 91125

Phone: +1-626-395-8567
Email: daw@caltech.edu
Web: www.danielwagenaar.net

Chapter 1

Overview

VScope is a software package for the acquisition and analysis of data from multiple cameras as well as electrophysiology. Its main intended purpose is to record fluorescent traces from neurons loaded with voltage-sensitive dyes along with associated electrophysiology. VScope can record simultaneously from any number of cameras and frame rates can be set independently for each camera. VScope can also record synchronized electrophysiology traces and digital channels. VScope is mainly intended to acquire fixed-duration trials, but it can also acquire electrophysiology continuously. A variety of electrical stimulation protocols can be created through an easy-to-use GUI.

Since VScope is intended for use in neuronal recordings, a central facility is the ability to define arbitrarily many regions of interest (ROIs) on the acquired images. VScope can calculate fluorescence traces for each ROI online and perform coherence analysis to visualize relationships between neurons. VScope ships with a library of Matlab/Octave functions to perform further analysis of acquired data.

1.1 Camera support

At present, cameras from Photometrics and QImaging are supported through the *PVCam* library. Support for other types of cameras may be added at a later date. VScope has been tested with the

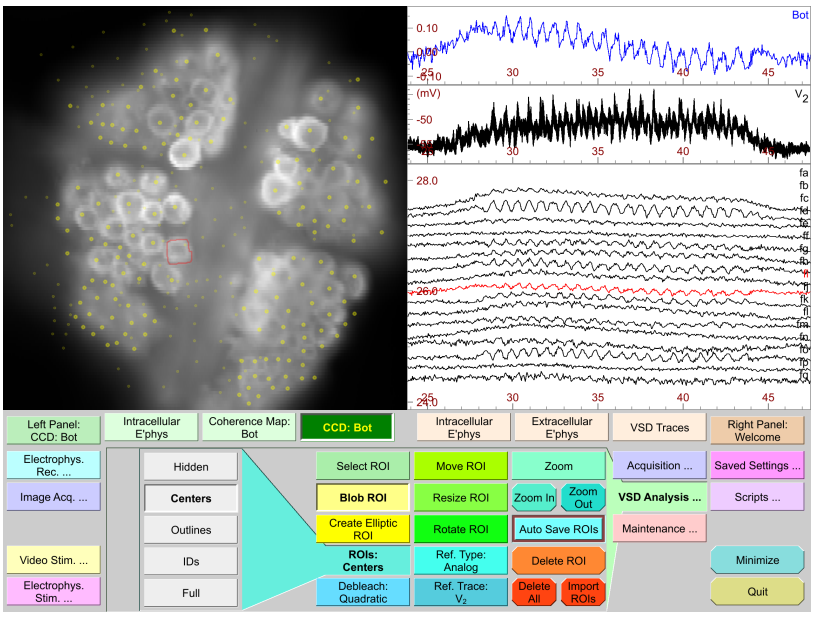

Figure 1.1: Example screenshot with voltage-sensitive dye data acquired from the nervous system of a medicinal leech during fictive swimming.

"QuantEM 512SC" and "Evolve 512delta" cameras.

1.2 Data acquisition system support

At present, National Instruments cards are supported through the *DAQmx* library. At a minimum, cards must support two hardware-timed digital lines and a single hardware-timed analog line. This includes the majority of NI cards. VScope has been tested with the USB-6229 and PCI-6221 cards.

1.3 Operating system support

A binary installation package is available for Windows 10 (64-bit). VScope can be compiled for other versions of Windows using Microsoft Visual C++ and the open-source *Qt* libraries.

In addition, a ".deb" package is available for Ubuntu Linux. At present, the Linux version can only be used for post-hoc analysis and not for data acquisition, because of the severely incomplete Linux support by National Instruments.

The Matlab/Octave libraries included with VScope are operating-system independent and can be installed on Windows, Mac, or Linux machines.

Chapter 2

Installation

2.1 Windows 10

2.1.1 Prerequisites

Before installing VScope, you must download and install camera and data acquisition support libraries:

- PVCam from Photometrics, version 3.1 or later:

  https://www.photometrics.com/support/software

  When using Firewire-connected cameras, it is crucial to use the drivers provided by Photometrics and not those provided by Microsoft. The correct drivers are "manufactured" by Thesycon and "digitally signed" by Roper Scientific. The Windows "Device Manager" can be used to verify which driver is in use.

- DAQmx from National Instruments, version 8.0 or later:

  https://www.ni.com/dataacquisition/nidaqmx.htm

2.1.2 Downloading and installing VScope

After installing the prerequisites, VScope itself can be downloaded from:

https://github.com/wagenadl/vscope/releases

Installation should be as simple as running the ".exe" file and following the prompts. The Matlab/Octave library is available from the same location as a ".zip" file.

2.2 Other versions of Windows

For other versions of Windows that are supported by PVCam and DAQmx, first install the prerequisites listed above and also:

- Microsoft Visual C++, version 2013 or later. Regrettably, this compiler is not free. VScope can probably be compiled with open-source compilers such as MinGW:

  http://www.mingw.org

  However, this has not been tested.

- Qt, version 5.6 or later, from:

    https://www.qt.io/download

Due to limited 64-bit support by National Instruments, it may be best to compile VScope using a 32-bit tool chain, even when using 64-bit Windows.

2.3 Ubuntu Linux

A binary package for Ubuntu 16.10 is available at:

https://github.com/wagenadl/vscope/releases

This package installs both VScope and the Matlab/Octave library.

2.4 Other versions of Linux

In other versions of Linux, compiling VScope should be straightforward. You will need Qt, version 5.6 or later, as well as the GNU C++ compiler (g++), version 4.8.1 or later. Download the source ("git clone" may be the easiest way), compile using "make", and install using "sudo make install" or "DESTDIR=$HOME/vscope make install".

2.5 Mac OS

At present, Mac OS is not supported. However, the Matlab/Octave components of VScope are operating-system independent. This means that data acquired by VScope on a PC can subsequently analyzed on an Apple laptop.

Chapter 3

First run and configuration

3.1 First run

After installation on Windows, VScope can be started by double-clicking the Desktop icon or from the Start menu. In Linux, VScope can be started by typing “vscope” in a terminal. When VScope starts, it shows its welcome screen (Figure 3.1). The welcome screen displays:

- The version of VScope you are using (top left);
- Which, if any, data acquisition system VScope has found to use (under “DAQ status”);
- Which, if any, cameras VSCope has found to use (under “Camera status”);
- Details of the available cameras (below that);
- A large range of buttons that will be explained in subsequent chapters.

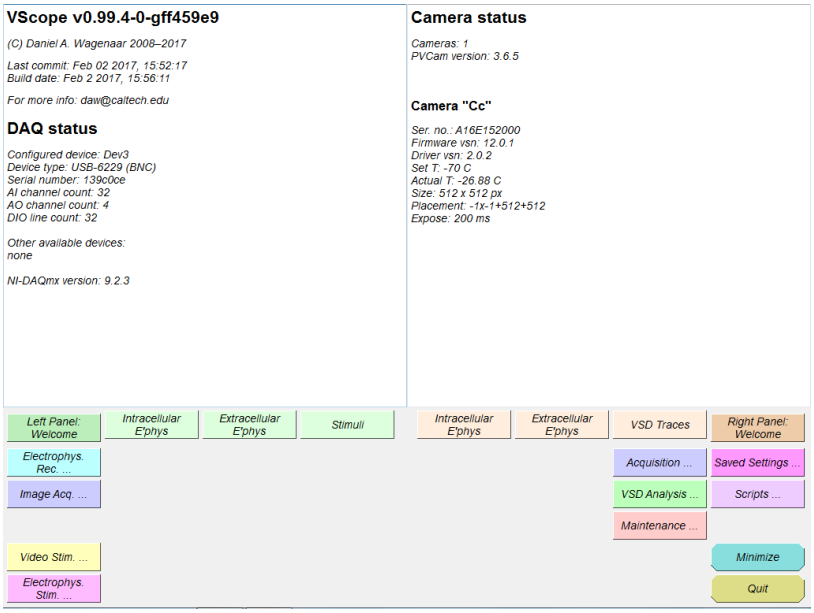

Figure 3.1: VScope’s welcome screen

3.2 Camera configuration

Probably, when you just started VScope, it complained that "Configured cameras could not be found". That's because VScope must be manually configured to find your camera (or cameras). On first run, VScope creates a folder called "*vsddata*" in your main "*Documents*" folder. In there, you will find a folder called "*_settings*", in which you will find a file called "*_connections.xml*". The contents of the default connections file is reproduced in Appendix A. Open this file using *WordPad* or your favorite text editor and configure your cameras in the <camera/> sections starting at line 52. Please read on carefully, because it is critical to get the details right.

The default configuration defines two cameras, named "Cc" and "Ox" (for "coumarin" and "oxonol", a pair of FRET dyes commonly used for voltage-sensitive dye imaging). Replace the default serial number of the "Cc" camera ("A08F101002") with the actual serial number of your first camera.

If your camera's pixel count is not 512x512, modify the xpix="512" and ypix="512" properties accordingly. If your pixels are not square, use the "transform" tag described below.

If you do have a second camera, replace the default serial number of the "Ox" camera ("A08B101016") with the actual serial number of your second camera. If your cameras do not operate as a donor/acceptor pair but as independent views, replace role="acceptor" with role="donor" and remove the partnerid="Cc".

The final line of each camera section specifies a geometric transformation of the camera image. If your camera has 512x512 pixels (as mine do), the topleft pixel will be occupy the square (0,0) to (1,1), and the bottom right pixel will occupy the square (511,511) to (512,512). The phrase <transform ax="1" ay="−1" bx="0" by="512"/> means that the image from the camera will be mirrored in the y-axis (ay="−1) and shifted by 512 pixels (by="512"), so that the resulting image once more occupy the area (0,0) to (512,512). This scheme can also be used to align cameras that may have different pixel sizes. For instance, if your first camera has 512x512 pixels of 8 μmx 8 μmand your second camera has 128x128 pixels of 16 μmx 16 μm, you might use <transform ax="1" ay="1" bx="0" by="0"/> for your first camera and <transform ax="2" ay="2" bx="128" by="128"/> for your second camera to co-align the two cameras. Naturally, you can use unequal scaling for x and y to accommodate non-square pixels.

Other properties in the 'camera' section are:

- lamp="*xxx*": along with a subsequent <dochannel id="Lamp:*xxx*" line="*nnn*"/> section, specifies that the lamp named *xxx* should be turned on by activating the digital output line numbered *nnn* when this camera is active. If no lamp needs to be turned on for this camera, you can say 'lamp="".

- shutter="*yyy*": along with a <dochannel id="Shutter:*xxx*" line="*nnn*"/> section, specifies that the shutter named *xxx* should be opened by activating the digital output line numbered *nnn* when this camera is active.

- focusexpose="*t* ms": specifies that this camera should expose each frame for *t* ms in "Focus" mode. This has no effect on normal image acquisition.

- role="donor": specifies that this camera functions to record from the donor fluorophore of a FRET pair.

- role="acceptor": specifies that this camera functions to record from the acceptor fluorophore of a FRET pair. Use partnerid="*id*" to indicate which camera records the corresponding donor fluorophore.

In addition to these edits in the "camera" sections, you will need to specify which digital lines are used to trigger each camera and which will be used to record the timing of the resulting frames.

(Although strictly speaking, this configures your DAQ board, this description is included here because it pertains to image acquisition.) The lines to modify are:

- <dochannel id="trigger:*id*" line="*nnn*"/> specifies that the digital line numbered *nnn* will be activated to trigger image acquisition on the camera known by *id* (e.g., "Cc"). (See section 4.2.2 for details of triggering relating to image acquisition duty cycle.)
- <dichannel id="frame:*id*" line="*nnn*"/> specifies that the digital line numbered *nnn* will be monitored to record when the camera known by *id* exposes a frame.

VScope will not work unless "trigger" and "frame" lines are specified for each camera.

3.3 DAQ board configuration

In addition to acquiring image sequences, VScope can record simultaneous analog and digital traces from a single data acquisition (DAQ) board as well as to produce analog and digital outputs through that same DAQ board. This is used to trigger cameras as mentioned above, and can also be used to generate electrophysiological stimuli and record, e.g., intracellular membrane voltage traces.

VScope should automatically detect your data acquisition (DAQ) board. On my computer, it found my USB-6229 automatically as "Dev3". If you have multiple DAQ cards, it may be that VScope autodetects the wrong one, in which case the right one should be listed under "Other available devices". If that is not the case, you probably have a problem with your DAQmx installation.

Specific devices can be selected by creating a "device" section in the "connections.xml" file. Either of the following forms can be used:

```
<device type="USB-6229"/>
<device id="Dev3"/>
<device serno="xxxxxx"/>
```

3.3.1 Configuring digital inputs

We already saw how the <dichannel/> tag is used to specify which digital lines are monitored to record frame times. Similarly, any number of additional digital lines can be configured as inputs to be recorded in each trial. The default connections file specifies several examples ("TrigOxMon", etc.) that you can modify or remove.

3.3.2 Configuring digital outputs

We already saw how the <dochannel/> tag is used to specify which digital lines are used to trigger cameras and activate lamps and shutters. You can define additional digital lines to serve as "stimulus" channels which will appear under the "Electrophys. stim" menu of VScope:

```
<dochannel id="xxx" line="nnn" isstim="true"/>
```

Here, *xxx* is an arbitrary name for the signal, *nnn* is the digital line on your DAQ board, and the phrase isstim="true" ensures that the line is listed in the "Electrophys. stim" menu.

3.3.3 Configuring analog inputs

Analog input channels can be used to record electrophysiological signals or anything else. They are configured using the <aichannel/> tag, which takes id and line properties just line <dichannel/>. Additional properties to be specified are:

- ground="RSE" specifies "referenced single-ended" grounding. You can also say "NRSE" for "nonreferenced single-ended" or "DIFF" for "differential." See the National Instruments documentation for details.
- range="5V" specified that the channel will record signals in the range of ± 5 V. Other ranges may be available, again see the NI docs.
- unit="mV" specifies that the display unit for the recorded signals is millivolts. You can also say "uV", "V", "pA", "nA", or "mA".
- scale="100" specifies that 1 V going into the DAQ board is equivalent to 100 mV before your amplifier. (If you had set unit="pA", then scale="100" would mean that 1 V is equivalent to 100 pA, etc.)
- offset="1 mV" specifies that an offset of 1 mV (or whatever) is to be added when signals from this channel are displayed.

Analog inputs with IDs starting with "E" and ending in a number are included in the "Extracellular E'phys" graph, all others in "Intracellular E'phys" graph.

3.3.4 Configuring analog outputs

Analog output channels can be used to generate stimuli, just like digital outputs. The <aochannel/> tag takes id and line, properties as you would expect. For the channel to be included in the "Electrophys. Stim" menu, you must specify isstim="true". You may also specify unit and scale properties, but these are currently ignored by the GUI. (Feel free to contribute a patch.)

3.3.5 Configuration problems

VScope is very sensitive to correct syntax in its configuration files. If you mistype anything in the "*_connections.xml*" file, you are likely to get an error message when VScope starts. Usually, the error message will tell you where in the file the problem was discovered. You may find it useful to keep a copy of the original file to compare your syntax to a known-good configuration.

Chapter 4

Using VScope

4.1 Overview of the program

VScope presents itself as a single window, normally operating in full-screen mode, and is intended to be controlled using a touch screen. Referring back to the figure on p. 7, the window is divided into several areas: two "display areas" on the top, a "quick access bar" immediately below each display area, "page selectors" along the left and right edges, and an area to display those "pages" in the middle. The buttons in the "quick access bar" can be used to quickly display a certain view on the recorded data, e.g., "Intracellular E'phys" to show a graph of intracellular traces, or, "CCD: Cc" to show images from the camera known as "Cc." If your desired data view is not listed, simply click on "Left Panel" or "Right Panel" to list all available options. Any data view can be displayed on either the left or right panel, though not in both at the same time.

The "page selectors" display various top-level pages with settings and controls. For instance, "Electrophys. Rec." displays a page that allows selection of recording channels and related options, while "Acquisition" displays a page with buttons to start data acquisition or to load previously recorded data.

VScope organizes data into "experiments" and "trials." An "experiment" is conceptually a collection of recordings from a single preparation taken in a single session. A "trial" is one such recording. There are two types of trials: "Single Frames," which are just image snapshots without any electrophysiology, and "Single Trials," which comprise a timed image sequence and associated electrophysiology traces. Ordinarily, recorded data is immediately saved to disk.[1] All data is saved in "*Documents**vsddata*" in subfolders arranged by experiments. Experiments can be given arbitrary names, but VScope defaults to using the current date (in the form "*YYMMDD*") for the first experiment on a given day, appending "a," "b," etc. for subsequent experiments. Trials are numbered 001, 002, etc. Trial metadata is stored in a file called "*nnn*.xml," with the actual data stored in "*nnn*-analog.dat," "*nnn*-digital.dat," and "*nnn*-ccd.dat" for the analog, digital, and image data. An extensive Matlab/Octave package allows for convenient post-hoc analysis of acquired data.

4.2 Preparing for data acquisition

Several pages of settings affect data acquisition, as outlined in the following sections.

[1] If you need to take some test snapshots that you don't want to save, you can enable the "Dummy" button in the "Acquisition" page.

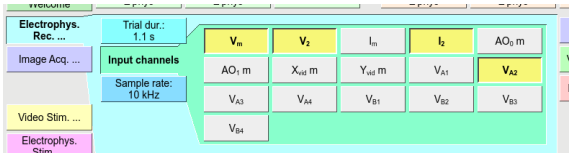

Figure 4.1: Screenshot showing the "Input channels" tab of the "Electrophys. Rec" page, with several analog inputs selected for recording.

4.2.1 Settings: *Electrophys. Rec.*

This page contains the following settings:

Trial dur. Duration of the trial. Click to open a submenu with various useful defaults. The italicized entry at the end of the list can be used to set a custom trial duration.

Input channels Click to make a selection of one or multiple analog input channels. (Digital input channels are always recorded.)

Sampling rate In our lab, we always sample electrophysiological signals at 10 kHz. Other settings are supported, but poorly tested.

Note that the current value of each setting is shown in the button that can be used to change that setting, except for "Input channels." Regrettably there is no space on the button for a complete list of channels.

4.2.2 Settings: *Image Acq.*

This page is organized into subpages, one per camera. There is a global "Enable" setting in the top left, with the names of the cameras below it. Click on a camera name to open a subpage with settings for that camera; double click on a camera name to enable or disable a camera. Per-camera settings are:

Master Normally set to "Self," this setting can be used to copy all other settings from another camera instead.

Binning When set to "1x1," each device pixel is read out as a pixel in the output image. When set to *N*x*M*, each group of *N* x *M* device pixels is read out as a single image pixel. This can improve maximum frame rates. For most cameras, increasing *N* has much less effect than increasing *M*. For some cameras, binning may have a negative impact on dynamic range as it involves pooling wells on the CCD.

Region Determines whether the whole pixel array of the camera is read out or some subset. "NW Quarter" means the top-left quarter of the image, etc.

VSD delay Amount of time between the start of the trial and the start of the first exposure.

VSD dur. During of the image sequence. The image sequence must fit entirely inside the "trial duration" set under "Electrophys. Rec."

Frame rate Rate of image acquisition, in frames per second. VScope presently does not enforce sensible limits; it is the user's responsibility to make choices that fit within the camera's capabilities. See note under "Duty cycle."

Duty cycle Set to "100%" to let the camera free-run as close to the prescribed frame rate as possible. When set to lower numbers, each individual frame is triggered at exactly the requested rate.

Note: The maximum frame rate for Photometrics cameras is affected by the "Duty cycle" setting. For duty cycles less than 100%, readout of each frame (i.e., transfer to the computer) must be completed before the next exposure starts. By contrast, at 100% duty cycle, readout of a frame can occur during exposure of the next frame. An oscilloscope attached to the "FRAME READOUT" and "EXPOSE OUT" signals of the camera is a useful tool to determine realistic limits on frame rate.

Port/speed Photometrics cameras have multiple internal "ports" for reading from their CCDs, each of which can be operated at different speeds. The "EM" port is faster than the "Std" port, but has greater readout noise. Choose the slowest setting that is compatible with your desired frame rate for best results.

Pre-illum The lamp associated with this camera will be turned on the given amount of time before each frame. (At 100% duty cycle, or when illumination periods overlap, the lamp is not turned off between frames.) Lamps cannot be turned on before start-of-trial, so "pre-illum" must not exceed "VSD delay."

Post-illum The lamp remains on for the given amount of time after each frame. The only reason why this cannot simply be fixed to a very small value is that at 100% duty cycle, the actual frame rate of the camera is typically slightly less than what is requested. In this case, "post-illum" can ensure that the final frame is fully illuminated.

There is not presently a visible setting to affect the timing of shuttering. A pre-frame shutter interval of 50 ms is hardcoded into the file "parameters.xml," as is a post-frame shutter interval of 100 ms. If you use a lamp that is permanently on, you could use the "lamp" connections from VScope to drive your shutter instead, and use the "pre-illum" and "post-illum" settings to control shutter timing. If you need control over both a lamp and a shutter for one of your cameras, please contact the author.

4.2.3 Settings: *Video Stim.*

These settings are intended to communicate with an external stimulation system and are currently not fully supported.

4.2.4 Settings: *Electrophys. Stim.*

These settings are used to define stimulus waveforms to be generated during each trial. Completely independent waveforms can be defined for each of the analog and digital output channels marked with isstim="true" in the "*settings.xml*" file. Per-channel settings are:

Delay Interval between start of trial and start of stimulation on this channel.

# of trains Number of trains of stimuli. A "train," here, is a short sequence of identical stimulus pulses.

Train period Start-to-start interval between consecutive trains. Immaterial when there is only one train.

Pulses/train Number of pulses in each train.

Pulse period Start-to-start interval between consecutive pulses in a train. Immaterial when there is only one pulse.

Pulse type This is the "shape" of each pulse. Choices include:

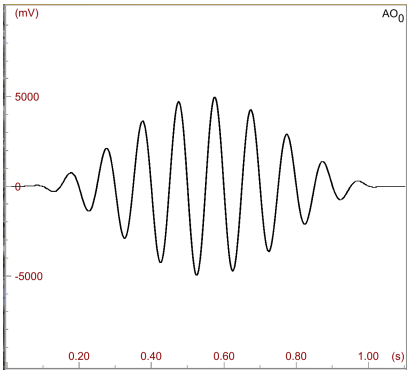

Figure 4.2: A 10-Hz sine-wave stimulus train constructed using 10 pulses at a 100-ms pulse period, with 500-ms slow start and end.

Square A simple pulse of fixed voltage and duration;

Biphasic A pulse consisting of two phases with separately controlled voltage and duration;

Ramp A shape that starts at one given voltage and gradually ramps to another voltage;

Sine A sinusoidal wave. The frequency of the wave is one over the specified "Pulse period."

On digital channels, only "Square" pulses are available, and the voltage is fixed to 5 V.

Amplitude The amplitude for a "Square" or "Sine" stimulus, or the starting amplitude for a "Biphasic" or "Ramp" stimulus.

Duration The duration for a "Square" or "Ramp" stimulus, or the duration of the first phase of a "Biphasic" stimulus. For "Sine" stimuli, this controls a gradual start of the stimulus.

2nd Ampl. The amplitude for the second phase of a "Biphasic" stimulus, or the final amplitude of a "Ramp" stimulus. Not applicable to "Square" or "Sine" stimuli.

2nd Dur. The duration of the second phase of a "Biphasic" stimulus. For "Sine" stimuli, this controls a gradual end of the stimulus. Not applicable to "Square" or "Ramp" stimuli.

The "Stimuli" display panel is very useful for instantly visualizing the effects of the various settings.

4.2.5 Saving (and reloading) settings

If you are planning to use the same settings in a subsequent experiment, you can save all your settings together using the "Saved Settings" control page. Click the "Save..." button to save your current settings under a new name. The names of previously saved settings appear in the same dialog. Clicking on a name loads those settings. (Saved settings are stored in the "*_settings*" folder in your main "*vsddata*" folder.)

4.3 Acquiring trials

All the action buttons for initiating actual data acquisition are collected in the "Acquisition" control page. The two most important buttons are:

Single Frame Simply takes a snapshot using each of the cameras enabled in "Image Acq."

Single Trial Initiate data acqusition using those cameras along with electrophysiology recording and stimulation as defined in "Electrophys. Rec." and "Electrophys. Stim."

Normally, data is automatically saved to disk, and the trial counter is updated. Sometimes it is useful to use the "Single Frame" button for focussing. In that case, automatic saving can be disabled using the "Dummy" button. Note, though, that data acquired in "Dummy" mode cannot subsequently be saved after all.

The "Acquisition" control page contains several other groups of controls, as follows.

- You can change the experiment name using the "Expt. Name" button and the trial number (using the "Trial No." button). The latter is useful if your experiment is broken down into sections: you can start a section with trial number 101 by setting the current trial to 100. (The number in the button is the number of the preceding trial; it will be auto-incremented *prior* to acquiring the next trial.)

- There are buttons for each of your cameras that track min/avg/max pixel values. Those can be used to optimize illumination intensity by repeatedly clicking "Single Frame" (probably in "Dummy" mode).

- Various buttons indicate clock time, time since start of experiment, and time since the most recent trial. Both "Since Start" and "Since Start" can be clicked to provide a simple timer that counts up from the time of your click.

- The "Log Note" button can be used to add arbitrary notes to the file "log.txt" that is generated in your experimental folder. In addition to your notes, this file contains the start times of all trials. It is meant to be read by humans, but the format is simple enough that parsing it mechanically should be straightforward too.

- The "Cont. E'phys" button enables (or disables) continuous electrophysiological recording. What that means is that analog and digital data is continuously recorded in the background. In that situation, the "Single Trial" button only records CCD data. The CCD data can be aligned post-hoc with the other data because each frame taken by the cameras results in a marking in the digital recording (specifically, on the "Frame:*camera-name*" channel).

- The "Auto Run" and "Trial period" buttons combine to automate data acquisition. When "Auto Run" is enabled, a trial is automatically started at the interval specified by "Trial period." This period is measured from start to start. If your trial period is short compared to the trial duration (in the "Electrophys Rec." page), it can be tricky to subsequently disable "Auto Run." Beware!

- Finally, the "Load Old Data" button opens a dialog from which you can load previously recorded data. When VScope displays old data in that way, certain operations are greyed out. In particular, you cannot retroactively change the settings pertaining to an old recording. To return to normal data acquisition mode, simply press the "Unlock" button that appears above the "Load Old Data" button.

4.4 Analyzing data

It is anticipated that most serious data analysis will be performed post-hoc using Octave or Matlab. However, the ability to quickly analyse voltage-sensitive dye recordings while the prep is still on the scope enables experiments that would otherwise be impossible: For instance, VSDs can be used to guide selection of neurons for electrophysiology. VScope supports this mode of operation under the "VSD Analysis" control page.[1] VScope allows the user to draw regions of interest (ROI) onto the

[1] Although the page is labeled "VSD Analysis," it can equally be used for analyzing calcium-dye recordings.

camera images and will perform basic analysis on the extracted signals. Two "styles" of ROI can be drawn: "elliptic" ROIs and "blob" ROIs. Elliptic ROIs can be drawn using the "Create Elliptic ROI" button and subsequently manipulated using the "Move ROI," "Resize ROI," and "Rotate ROI" buttons. Blob ROIs can be created and manipulated using the "Blob ROI" button. Drawing blob ROIs is especially convenient using pen tablets like those sold by Wacom. Once a blob ROI is drawn, it can be manipulated simply by dragging its edge around. The "Select ROI" button enables selecting various ROIs without changing them.

Ordinarily, ROIs are saved with the trial as you draw them. When you revisit a trial using "Load Old Data," this auto-saving is disabled to prevent accidental overwriting of previous work, but it can be re-enabled. In addition, it is possible to import ROIs from another trial using the "Import ROIs" button. Be careful: this replaces any ROIs you have already drawn for the present trial.

The "Zoom," "Zoom In," and "Zoom Out" buttons do what you would expect. The latter two use the currently selected ROI as their target.

If you draw a lot of ROIs, the display can become cluttered. Therefore, the "ROIs:" button allows you to select how to display ROIs. Choices are:

Hidden All ROIs except the presently selected one are completely hidden. (They can still be selected by clicking.)

Centers ROIs are indicated by a small dot in their centers.

Outlines ROIs are indicated by their outline shape.

IDs ROIs are indicated by short IDs in their centers.

Full ROIs are indicated by their IDs as well as their outlines.

Regardless, the currently selected ROI is always indicated by its outline.

The most basic online analysis, is simply extraction of traces from the ROIs. To see extracted traces, open the "VSD Traces" display panel. When an ROI is selected, a limited number of traces are drawn that include the selected ROI and some with neighboring IDs. When no ROI is selected, all traces are shown.

In many cases, a certain amount of photobleaching occurs over the duration of a trial. VScope can compensate for that by subtracting either a linear or a quadratic baseline from the signals. This is enabled using the "Debleach" button.

The second type of analysis that can be performed online in VScope is coherence analysis. This calculates the degree of coherence between optical signals and a chosen reference signal at the frequency of peak power of that reference signal. Alternatively, the coherence can be calculated between the optical signals and a pure sine wave of specified frequency, or between the optical signals and a stimulus. The reference signal is selected using the "Ref. Type" button and the button below it, which shows "Ref. Trace" for "Analog" references, "Digi. Ref." for "Digital" references, "Ref. Freq." for "Frequency" (sine wave) references, and "Ref. Stim." for "Train" or "Pulse" references. (The difference between those last two is that the coherence is calculated at either the frequency of trains or of individual pulses within the stimulus.)

The results of coherence analysis can be viewed in polar form in the "Coherence Graph" display panels, or as a color-coded overlay on top of the CCD images in the "Coherence Map" display panels.

The final buttons "Delete ROI" and "Delete All" practically speak for themselves. The former deletes the currently selected ROI, the latter deletes all ROIs. It requires double-clicking to activate.

4.5 The “Maintenance” control page

The “Maintenance” control page contains a subpage that allow to check focus and alignment of your cameras, and one that allows VScope to function as a virtual oscilloscope. The options should largely speak for themselves.

4.6 Scripting VScope operation

For advanced use, it is possible to control the timing and parameters of data acquisition using scripts. VScope’s scripting language is very basic. A script is a sequence of lines that consist of a command followed by zero or more arguments. Commands are:

set Changes a parameter setting. Must be given two arguments: a parameter name, and a parameter value. Parameter names can be found in the “parameters.xml” file in the VScope source code, or, more conveniently in any “.xml” file saved as a trial. Parameter names in scripts are written as slash-separated paths. For instance, the trial duration is written as “acqEphys/acqTime”. Array parameters use colons to indicate elements. For instance, “stimEphys/channel:AO0/enable”. Values are specified along with their correct SI units. VScope understands “V” for volts, “A” for amperes, “s” for seconds. It also understands metric prefixes “m” (milli), “u” (micro), “n” (nano), and “p” (pico).

trial Runs a trial. No arguments.

snap Take a single frame. No arguments.

ival Sets the time to wait after the subsequent trial before starting the trial following that. Argument specifies time, measured start-to-start, and must be specified with units.

loadsettings Load all settings from a saved-settings file. Single argument specifies the filename.

loop Causes the script to run indefinitely, either starting from the start (when no argument is given) or from the specified starting line.

As an example, the following is a valid script (assuming the settings file “simplestim” exists):

```
loadsettings simplestim # Note: no path, no “.xml”
ival 60 s
set stimEphys/channel:AO0/pulseAmp 1 V
trial # happens right away
set stimEphys/channel:AO0/pulseAmp 2 V
trial # happens 60 s later
# etcetera
```

VScope ignores any text following a “#” in scripts.

If a scripted trial takes longer than the interval specified with “ival,” the next trial happens very quickly after its end: a watchdog timer checks for the trial’s end every second.

Scripts can be saved and reloaded later. The “Check” button is very useful to confirm that your script contains legal code. If your script is flawless, it will display “Script OK.” Otherwise, it will report the first error it contounters. Its error messages are rather terse, but hopefully better than nothing.

The “Run” button starts the script running and can also used to abort a running script at any time.

Chapter 5

Post-hoc data analysis in Octave or Matlab

VScope comes with a library of Octave/Matlab functions to load data acquired using VScope and to analyze it in an environment more suited for in-depth analysis than VScope can provide inside its GUI. The most important functions are:

vscope_load	Load data from a single VScope trial
vscope_extractrois	Extract traces from regions of interest
vscope_cohanalysis	Perform coherence analysis
microcorrect	Perform automatic micromotion correction
vscope_debleach	Remove a bleaching artifact from traces

The full list of functions is given in Appendix B. Each of these functions is documented in the standard Octave/Matlab fashion. For instance, to learn more about vscope_load, simply type "help vscope_load" within either Octave or Matlab.

Conclusion

I hope that VScope will be useful to you. I love to hear happy users' stories. I also welcome bug reports of all kinds. Although I cannot make any guarantees (see the GPL license text!), it almost certainly can be fixed. And I would be happy to try and help.

Appendix A. Default “connections” file

The following is the contents of the defaults “_connections.xml” file created by VScope in Documents\vsddata_settings.

```
<?xml version="1.0" encoding="UTF-8" standalone='yes'?>
<!DOCTYPE vscope>
<vscope version="20120810">
<connections>

  <!-- These are default values. Modify them for your rig. -->

  <device id="" serno="" type=""/>

  <aichannel id="ME1_10Vm" line="0"
     ground="RSE" range="5V" unit="mV" scale="100"/>
     <!-- That is, 1 V at ADC corresponds to 100 mV in real world -->
  <aichannel id="ME2_Vm" line="1"
     ground="RSE" range="5V" unit="mV" scale="100"/>
  <aichannel id="ME1_I" line="2"
     ground="RSE" range="5V" unit="nA" scale="10"/>
  <aichannel id="ME2_I" line="3"
     ground="RSE" range="5V" unit="nA" scale="10"/>
  <aichannel id="Stim0Mon" line="4"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="Stim1Mon" line="5"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="Stim2Mon" line="6"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="Stim3Mon" line="7"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="60HzRef"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVA1" line="16"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVA2" line="17"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVA3" line="18"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVA4" line="19"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVB1" line="20"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVB2" line="21"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVB3" line="22"
     ground="RSE" range="5V" unit="V" scale="1"/>
  <aichannel id="ExtVB4" line="23"
     ground="RSE" range="5V" unit="V" scale="1"/>

  <aochannel id="AO0" line="0" isstim="true" unit="nA" scale="1"/>
  <aochannel id="AO1" line="1" isstim="true"/>
  <aochannel id="VidX" line="2"/>
  <aochannel id="VidY" line="3"/>
  <!-- Output units are ignored in the current version. -->

  <camera id="Cc" serno="A08F101002" role="donor"
     xpix="512" ypix="512" lamp="LED" shutter=""
```

```
      focusexpose="200 ms">
    <transform ax="1" ay="-1" bx="0" by="512"/>
  </camera>
  <camera id="Ox" serno="A08B101016" role="acceptor"
      xpix="512" ypix="512" lamp="LED" shutter=""
      focusexpose="3 ms" partnerid="Cc">
    <transform ax="1" ay="1" bx="0" by="0"/>
  </camera>

  <!-- The following digital channel names are used in the code.
       "*" respresents a camera id, lamp id, or shutter id.

       - Lamp:* - output to switch on lamp "*"
       - Shutter:* - output to open shutter "*"
       - Trigger:* - output to trigger camera "*"
       - Frame:* - input for frame time marker from camera "*"

       In particular Lamp:Video is for video illumination.
       -->

  <dichannel id="Frame:Ox" line="28"/>
  <dichannel id="Frame:Cc" line="29"/>
  <dichannel id="TrigOxMon" line="26"/>
  <dichannel id="TrigCcMon" line="27"/>
  <dichannel id="DO0Mon" line="8"/>
  <dichannel id="DO1Mon" line="9"/>
  <dichannel id="DO2Mon" line="10"/>
  <dichannel id="DO3Mon" line="11"/>
  <dichannel id="ExcShtrMon" line="24"/>
  <dichannel id="VideoLightMon" line="17"/>

  <dochannel id="Lamp:LED" line="7"/>
  <dochannel id="Trigger:Ox" line="30"/>
  <dochannel id="Trigger:Cc" line="31"/>
  <dochannel id="DO0" line="0" isstim="true"/>
  <dochannel id="DO1" line="1" isstim="true"/>
  <dochannel id="DO2" line="2" isstim="true"/>
  <dochannel id="DO3" line="3" isstim="true"/>
  <dochannel id="Lamp:Video" line="16"/>
</connections>
</vscope>
```

Appendix B. All Octave/Matlab functions

The following is a complete list of all Octave/Matlab functions included in the VScope package. Not all are useful to the end user, but they *are* all fairly well documented through the online "help" system.

bestshift — Calculate best shift for image based on two reference images
bestshiftrois — Calculate best shift for image based on two reference images
marginlessdiff — RMS difference between images sans margins
microcorrect — One step microcorrection
microcorrectrois — One step microcorrection
microcorrectroisslow — One step microcorrection - slow version with optimization
microcorrectslow — One step microcorrection - optimization version
microcorrectslow2 — One step microcorrection - optimization version
microperspective — Microperspectived image
microperspectivex — Microperspectived image
microrotate — Microrotated image
microscale — Microscaleed image
microshift — Microshifted image
microshift1 — Microshifted image
microshiftrois — Microshifted image
pf_microcorrect — One step microcorrection using parfarm
pf_microcorrect1 — One step microcorrection using parfarm
vscope_allroimask — Return mask corresponding to all ROIs
vscope_ccdtime — Construct a vector of time points for CCD data
vscope_cohanalysis — Comprehensive coherence analysis for VScope
vscope_cohcolor — Color values for coherence analysis
vscope_cohcontrol — Multitaper coherence estimate control data for VScope
vscope_cohere — Calculate coherence between signals and reference
vscope_coherence — Multitaper coherence estimate for VScope
vscope_cohplotimage — Color ROIs by coherence on top of CCD image
vscope_cohplotpolar — Polar plot of coherence results
vscope_cohplotsignals — Proof sheet of coherent signals
vscope_cohprettyimage — Pretty version of vscope_cohplotimage
vscope_contephys — Load continuous e'phys data from VScope
vscope_debleach — Debleach vsd data
vscope_downsample — Reduce resolution of CCD images in VScope data
vscope_ephysatccd — Interpolate ephys data at camera frame times
vscope_ephystime — Construct a vector of time points for ephys data
vscope_extractrois — Extract averages from ROIs in CCD images

vscope_globaltrend — Extract global trend from VSD data
vscope_imagestack — Image stacking for VScope
vscope_load — Load data from VScope
vscope_loadrois — Add ROI data to a loaded trial
vscope_plotpolargrid — Helper function for vscope_cohplotpolar
vscope_preptaper — Prepare tapers for PSDEst and CohEst
vscope_prettyimage — Overlay some color info over VSD image
vscope_psd — Multitaper power spectrum estimate for VScope
vscope_ratio — Extract ratioed data from CCD images
vscope_realtime — Time of start and end of trial of VScope data
vscope_roicoords — Returns a list of all pixels inside an ROI
vscope_roicoords_cam — Returns a list of pixels inside ROIs for a camera
vscope_roiid — Convert ROI number to string
vscope_roinum — Convert ROI string to number
vscope_roioutline — Returns a polygon specifying the edge of a ROI
vscope_roioutline_cam — Returns a polygon specifying the edge of a ROI
vscope_showrois — Show ROIs overlaid on camera images
vscope_unpackbits — Unpack a bit array stored in uint32 to booleans