**Is the eye’s pupil sensitive to blue light?**

**STANDARD OPERATING PROCEDURE (SOP):  
 PUPILLOMETRY TESTING**

Ethics Approval Reference: R54409/RE005

Prepared by: Dr. Joel Martin <joel.martin@eng.ox.ac.uk>

A person standing in front of a computer

Description automatically generated

Figure 1 – A participant in position for testing with the custom Ganzfeld stimulation and measurement system

Table of Contents

[1. Introduction 2](#_Toc57219179)

[2. Equipment location 2](#_Toc57219180)

[3. Operation: Checking connections and powering up 2](#_Toc57219181)

[4. Operation: Running a protocol 2](#_Toc57219182)

[5. Operation: Finishing up 3](#_Toc57219183)

[Appendix 1 - Components 4](#_Toc57219184)

## Introduction

PyPlr is a versatile, integrated system of hardware and custom Python software for researching the human pupillary light reflex. It is developed against the Pupil Core (Pupil Labs, GmbH) eye tracking platform, which affords accurate and reliable measurements of pupil size and has a synchronised forward-facing ‘World’ camera that we exploit to timestamp the onset of light stimuli with good precision. This method of timestamping light stimuli opens the door to integration with virtually any light source and stimulus geometry, but in our configuration, we use a Spectra Tune Lab (Ledmotive Technologies, LLC) 10-primary light engine and a low-cost integrating sphere to deliver a full-field ‘Ganzfeld’ stimulus. See Appendix 1 for further details on the various components of this system.

## Equipment location

The equipment is currently housed in one of the small psychophysics testing rooms (616.00.107) in the Anna Watts building.

## Operation: Checking connections and powering up

All of the equipment is powered via a single extension lead located behind the integrating sphere. Make sure the laptop, the light engine and the LIGHTHUB (a small orange and grey computer that connects the light source to the laptop via USB) are plugged into the extension lead and then turn the extension lead on at the wall. Check that the individual switches for each of the active plug sockets on the extension lead are turned on. Now, boot the laptop, turn on the light engine by flipping the switch at the back – you should hear the fan start to whir – and check that the leds on the LIGHTHUB have lit up to show that it is receiving power. Make sure the USB cable coming out of the LIGHTHUB is connected to the USB port on the right side of the laptop and that the ethernet cable is connected to the light source. Plug the eye tracker into the USB port on the left side of the laptop and ask the participant to put it on like they would a pair of glasses.

## Operation: Running a protocol

1. Press Ctrl+Alt+Del on the laptop and enter the following login credentials:

Username: engs2242

Password: TempPa55!

Active testing protocols are kept in a ‘protocols’ folder on the desktop. Open this folder and locate the protocol that you wish to run. Read carefully through the README file for a detailed description of the protocol and what is required on the behalf of the participant. If necessary, review the script as well.

1. Open an Anaconda Prompt by clicking on the relevant icon in the task bar. Type conda activate pupil and press Enter. This will ensure that you run the experiment in the correct environment where all of the required dependencies are installed. If you are using your own laptop you will first need to clone the repo and create a Python environment using the requirements.txt / .yml file.
2. Now type cd Desktop/protocols/<protocol\_name> and press Enter to change the current working directory in the Anaconda prompt to the directory of the protocol that you will be running. You are now ready to run the protocol by typing python <protocol\_name>.py, but don’t do that just yet.
3. Start Pupil Capture by clicking on the blue circular icon in the task bar. If the participant is wearing the eye tracker the software will automatically attempt to detect their pupil, but it may not be successful depending on the camera angle. Adjust the eye camera so that the pupil is in the centre of the image. Now check the following settings:
4. Annotation Capture plugin is enabled
5. Network API is enabled, and the Frame Publishing Mode is set to ‘bgr’
6. Correct eye(s) is selected for recording
7. Frame rate of the eye and world cameras is set to 120
8. Exposure mode for the World camera is set to ‘manual’
9. Seat the participant on the chair in front of the integrating sphere and adjust the height of the chair and/or the chin rest so that they are sat comfortably with their eyes at the vertical centre of the viewing port. Tell the participant not to tilt their head forwards or backwards, as this can cause problems for the eye tracking.
10. Type python <protocol\_name>.py at the Anaconda Prompt and press Enter to begin the protocol. The participant must now remain in position and follow the relevant instructions.

## Operation: Finishing up

When you have finished, quit the Pupil Capture software and then unplug the eye tracker. Turn off the light engine using the power switch on the top of it and shutdown the laptop. Turn off the extension lead leading to the light source at the wall.

## Appendix 1 - Components

The system includes the following components:

1. **Pupil Core**

<https://pupil-labs.com/products/core/>

Pupil Core (Pupil Labs, GmbH) is a versatile eye tracking system with open-source software. The device is modular, durable and lightweight, and it can be used for monocular and binocular eye tracking and pupillometry. The system has a forward-facing ‘World’ camera which records the observer’s field of view and additional infrared cameras to record the eyes. It connects to a laptop, PC or mobile device via USB/USB-C.

Graphical user interface, application

Description automatically generated

1. **Pupil capture and the Network API**

<https://docs.pupil-labs.com/core/software/pupil-capture/>

Pupil Capture is the software that allows you to view and record real-time gaze and pupil data with the Pupil Core system. The Pupil Labs Network API allows Pupil Core to be controlled via Pupil Capture with the Python programming language using ZeroMQ and MessagePack for fast and reliable communication. ZeroMQ (<https://zeromq.org/>) is an open-source universal messaging library and MessagePack (<https://msgpack.org/index.html>) is a binary format for computer data interchange, like JSON but faster and more efficient. The *pupil.py* module has a class for the Pupil Core device which encapsulates most routine operations.

Graphical user interface, text, application

Description automatically generated

1. **Pupil Player**

<https://docs.pupil-labs.com/core/software/pupil-player/>

This is the software that allows you to visualise data and media and to export data for further analysis.

Graphical user interface, website

Description automatically generated

1. **Integrating sphere**

<https://www.projectplastics.co.uk/>

<https://aviantechnologies.com/product/avian-b-white-reflectance-coating/>

The integrating sphere was constructed from two flanged 45-cm diameter acrylic half domes (Project Plastics Ltd). The front dome has a 28 cm circular opening to serve as a viewing port and the rear dome has a 7 cm circular opening which can be used to introduce another stimulus (e.g. a fixation target on a monitor) or to exclude the foveal macular pigment from stimulation.

The inside of the integrating sphere is coated with Avian-B (Avian Technologies LLC) – a highly Lambertian water-based barium sulphate coating which exhibits reflectance of >97% over 350-850 nm and greater than 92% from 250-1300 nm. This coating ensures that light from the light source is scattered homogenously on the inside surface of the sphere and therefore that participants experience a full and uniform field of illumination (i.e. Ganzfeld viewing conditions). The entry port for the light source is angled such that the light source cannot be seen directly when looking straight ahead from the plane of the viewing port.

1. **Spectra Tune Lab**

<https://ledmotive.com/product/spectratune-lab/>

A picture containing text

Description automatically generatedThe STLAB (Ledmotive Technologies, LLC) is a spectrally tuneable light engine with 10 LED colour channels, capable of generating a broad range of spectral compositions. It connects via ethernet cable to a small Linux-based computer (called the LIGHT HUB), which in turn connects to a computer via USB.

The STLAB can be controlled programmatically with most languages using its RESTFUL\_API, which supports communication via generic HTTP requests. The API includes commands to set a specific spectrum, turn the light off, get readouts from the onboard spectrometer, etc. The *PyPlr* Python library includes an *stlab.p*y module which uses the Python *requests* library to wrap the entire RESTFUL\_API, and which also includes other useful routines for working with the device.

The charts below shows the spectral power distributions for each of the 10 LEDs at their maximum intensity setting (left) and the 10 coordinates in CIE chromaticity space (right). The data for these plots were obtained using the STLAB’s on-board spectrometer. Spectrums are defined by passing a list of 10 values—one for each LED channel—between 0 and 4095 (12-bit resolution depth), corresponding to the maximum and minimum input.

**Chart

Description automatically generated**

The STLAB’s default mode of operation is *synchronous*, meaning it acknowledges receipt of all the commands sent by the LIGHT HUB before it accepts a new instruction. This enables “collisions” between messages to be detected and corrected. According to the manual, typical response times of this operation mode are approximately 250 ms. My own (J.M.) measurements using the python requests library suggest an average time of 180 – 220 ms (depending on the command being processed), but on rarer and seemingly random occasions it can be as high as 5 s. Most of the commands in the STLAB are programmed to work in synchronous mode.

The device can also operate in an *asynchronous* mode which permits real-time streaming of light spectra at approximately 1 spectrum every 10 ms. We leverage this mode of operation to attain maximum control over the temporal characteristics of our light stimuli. This mode requires the creation of *dynamic sequence files* (basically json files) which specify the spectrum to be played at each time point. The *stlab.py* module contains a set of tools for creating video files.

1. **PyPlr**

<https://github.com/spitschan/cvd_pupillometry.git>

This is the custom Python software that integrates hardware and streamlines everything. There is a pupil.py module which contains classes and methods for interfacing with the PupilCore system and for timestamping light stimuli. There’s also a module for the STLAB which contains a class for the device and methods to encapsulate the full range of functionality available in its RESTFUL\_API, which is based on generic http requests. In addition to this there is support for integration with an oceanoptics spectrometer, and tools for a fast and efficient pupillometry data processing pipeline, as well as support for pupillometer-style parametrization of PLRs. See Jupyter notebooks for more information and examples.