**Standard Operating Procedure for Pupillometry**

***PyPlr* system overview**

*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) infrared video-based eye tracking system, which yields accurate and reliable measurements of pupil size and has a synchronised forward-facing ‘World’ camera that can be exploited 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. In our configuration we use a Spectra Tune Lab (Ledmotive Technologies, LLC) 10-primary light engine and a low-cost integrating sphere to deliver full-field ‘Ganzfeld’ stimuli.

A person standing in front of a television

Description automatically generated

**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.

**Pupil capture and the Network API**

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

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.

**Integrating sphere**

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

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

We constructed our integrating sphere 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 perceive 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.

**Spectra Tune Lab**

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

The STLAB (Ledmotive Technologies, LLC) is a spectrally tuneable light engine with 10 LED colour channels that is 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 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 in order to attain the highest level of 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.