ESCHER

a Matlab-based app for generating visual stimuli

and

HERMES

an Arduino-based platform to synchronize visual stimuli and recordings

Table of Contents

[1. Introduction and setup overview 3](#_Toc161666324)

[1.1. Components 3](#_Toc161666325)

[2. Interaction with Hermes 4](#_Toc161666326)

[3. Control of wide field camera (PCO Edge4.2) 4](#_Toc161666327)

[4. Master-Slave mode with UDP connection 4](#_Toc161666328)

[5. Luminance calibration 4](#_Toc161666329)

[6. Visual stimulation 4](#_Toc161666330)

[6.1. Stimulation pipeline 4](#_Toc161666331)

[6.2. Stand-alone vs Master-Slave mode 4](#_Toc161666332)

[6.3. New visual stimulus: step-by-step guide 4](#_Toc161666333)

1. **Introduction and setup overview**

Escher + Hermes is a comprehensive tool to deliver visual stimuli during experimental recordings. Its purposes include:

* Generation of visual stimuli of different kinds:
  + Flashes.
  + Alternating checkerboards.
  + Moving grids or dots.
* Synchronization of visual stimulation with electrophysiology and/or imaging.
* Acquisition of movies with a wide field camera (PCO Edge4.2).
  1. Components
* **Stimulation PC (Escher).** Visual stimuli are generated with a Matlab App called Escher, which relies on Psychtoolbox 3 (PTB), a free collection of functions for vision research. PTB offers functions for generating a wide range of visual stimuli and to obtain accurate timestamps of their presentation. However, the timing reliability decreased drastically with new Windows OSs (from Vista on), so that the best way to keep track of stimulation time is to physically record the stimulation monitor with a light sensor. This is why we developed Hermes, an Arduino-based that monitors the luminosity of a dedicated section of the monitor and provides TTL [[1]](#footnote-1) signals with sub-millisecond precision.
* **Temporization hardware (Hermes).** Hermes was the messenger of Gods in ancient Greek mythology. With his winged boots, he reported messages very rapidly. Here, Hermes is an Arduino provided with:
  + A phototransistor to detect the luminance of a dedicated region of the stimulation monitor. The detection of a visual stimulus triggers one or more TTL signals.
  + A USB cable connected to the stimulation PC. This allows serial communication with Escher for sensor calibration and for the customization of TTL output(s).
  + Up to two BNC cables for TTL outputs.

TTL outputs can be recorded with the same acquisition system used for your experiment (e.g. the digitizer of your electrophysiology setup) to report visual stimulation time. Also, they can synchronize the acquisition of one or more frames with a camera locked on the visual stimuli.

* **Imaging device (optional).** Escher is programmed to take care of the acquisition of images with a PCO Edge4.2 CMOS camera. Acquiring images with the same app that provides the stimuli makes it easy to control the acquisition parameters (number of frames, framerate, exposure… ). Escher code is specific for the abovementioned camera model, but it shouldn’t be too hard to adapt it to other cameras/acquisition systems. See **3.** Control of wide field camera (PCO Edge4.2) more info.

1. **Interaction with Hermes**
2. **Control of wide field camera (PCO Edge4.2)**
3. **Master-Slave mode with UDP connection**
4. **Luminance calibration**
5. **Visual stimulation**
   1. **General stimulation pipeline**

Here is the general pipeline of how visual stimuli are generated:

1. A **button** on the GUI is **pressed** (e.g. “Checkerboard GO!”) and its callback is triggered. Here:
   1. Some parameters relevant to the stimulus are saved in a cell array called “ParameterVector”.
   2. A structure with general info about the screen and specific info about each stimulus presentation is prepared and store in the app property “CurrentLogStruct”. Once the stimulation is complete, this will be complemented with stimuli timestamps and saved as an xml log file.
   3. The total stimulation time is predicted (relevant for Master-Slave mode).
   4. If a movie has to be acquired, parameters relevant to the camera (like number of frames per stimulus, total number of frames…) are computed with the function “PcoSetForStimulation”. Relevant parameters are sent to Hermes.
   5. The function “callVisStim” is called.
2. The function “**callVisStim**” performs some action that are common to all visual stimuli:
   1. Some general parameters are added at the end of “ParameterVector”.
   2. If a movie has to be acquired, it starts the camera videoinput object. From now on, the camera will wait for TTLs generated by Hermes, which will be triggered by the visual stimuli themselves.
   3. Visual stimuli are generated by calling its specific function contained in the folder named “VisualStimulation” (e.g. “CheckPresent.m”). During the stimulation, Matlab will be busy inside that function and won’t be able to do anything else. Visual stimulation can be interrupted by the user pressing the keys “s”+”t”+”o”+”p” on the keyboard \*\*\*migliora usando  [ListenChar](http://psychtoolbox.org/docs/ListenChar)/[GetChar](http://psychtoolbox.org/docs/GetChar) di PTB o generando una figura con un apposito KeyPressFcn o KeyReleaseFcn.
   4. Once the stimulation is done, the timestamps of each stimulus are returned. “CurrentLogStruct” is filled with stimulus-specific timestamps and is saved as an xml file.
   5. **Master-Slave mode**

If Master-Slave mode is active, the Master sets all the stimulation parameters and writes the log file, but who perform the actual stimulation is the Slave.

1. The pipeline starts the same way up to a point in the function “callVisStim”. After point 2.2 of the section above, if Master-Slave mode is active, “ParameterVector” is converted to a string and sent as a UDP datagram to the Slave, and “callVisStim” ends.
2. The Slave receives and parse “ParameterVector” back into a cell array, and sends a UDP message to the master to inform that a stimulation is going to happen, and how long it will take. The Master stats a timeout callback: if no message from the Slave is received within the expected stimulation time, it assumes that UDP connection has been lost.
3. The Slave calls the specific stimulation function within a try-catch. If no error occurs, the slave Sends the collected timestamps to the Master, otherwise it sends a warning message to the master and reports the error.
4. When the Master receives the slave response, it writes the timestamps into “CurrentLogStruct” and save the xml log file and stops the acquisition with the camera, if necessary.
   1. **New visual stimulus: step-by-step guide**

Here is a step-by-step guide on how to add a new custom visual stimulation function to Escher, to make it compatible with its architecture.

1. **Create a specific function** for visual stimulation and save it as an ‘\*.m’ file into yourPath\EscherUDP\VisualStimulation. The function must accept two input variables: app, the handle to either EscherMasterApp or EscherSlaveApp; ParameterVector, a cell array of scalars. Also, it must return two outputs: timestamps, a 1D vector of double with stimuli timestamps; interrupted, a Boolean that reports if the stimulation has been interrupted by pressing “stop” on the keyboard. The file MyFunction.mat is an example of how to design such a function, and can be found at yourPath\EscherUDP\VisualStimulation.
2. **Modify EscherAppMaster GUI.** Add to the GUI any graphic element needed to specify additional parameters for your stimulus. Also, add a button to start the stimulation. Traditionally, these buttons are called “Stimulus\_name GO!”. Its ButtonPressedCallback should be organized as follows:

function MyStimulusGO\_Callback(app, event)

% load parameters in local variables

param1 = app.param1EditField.Value;

param2 = app.param2EditField.Value;

…

paramN = app.paramNEditField.Value;

ParameterVector = num2cell([param1, param2 … paramN]);

% Write the log file.

% General log structure using the function PrepareLogStruct. This function

% creates fields of CurrentLogStructures with general parameters like screen

% measures, distance, calibration function… Its input arguments are app and

% a character array with the stimulus name.

app.CurrentLogStruct = PrepareLogStruct(app,'MyStimulus');

% Here create the stimulus-specific part of CurrentLogStruct. For more details

% see the section of the User Guide dedicated to the xml log file

% If you want to record a movie while stimulating, setup the camera and

% its metafile.

if app.PcoWhileStimulusCk.Value == 1

% Based on the selected frame rate, the function PcoSetForStimulation

% computes the correct number of frames to acquire.

% PcoSetForStimulation is a public method of EscherMasterApp.

% Read the function for additional details.

PcoSetForStimulation(app,stimul\_period,n\_of\_trials,initial\_delay)

% Send the new parameters to Hermes

HermesMsgMake(app,'mod','camAc','p',…

app.PcoNumberFramesEditField.Value/n)

app.TriggerDropDown.Value = 'phototransistor';

% Populate the fields of PcoMeta.StimInfo with relevant parameters.

% PcoMeta.StimInfo fields should be: ‘StimType’, ‘nTrials‘,

% ‘FramesPerTrial’, ‘Orientation’, ‘StimF’, ‘Baseline\_bool’

% and ‘DeadTimeBetweenTrials’.

end

% Call the visual stimulus. Inputs to the function callVisStim must be:

% - app: the handle to EscherMasterApp

% - stimulusName: character array. It will define the name of the log file

% - TotStimDur: total duration of the stimulus

callVisStim( app, stimulusName, ParameterVector, TotStimDur)

end

1. Update the function callVisStim.

1. TTL (Transistor-Transistor Logic) is a common protocol for digital data. Standard TTL circuits operate with a 5-volt power supply. A TTL signal is defined as "low" when between 0 V and 0.8 V with respect to the ground terminal, and "high" when between 2 V and VCC (5 V). [↑](#footnote-ref-1)