
perceptivo

Release 0.0.1

Jonny L Saunders, Avinash Singh Bala

Mar 08, 2022

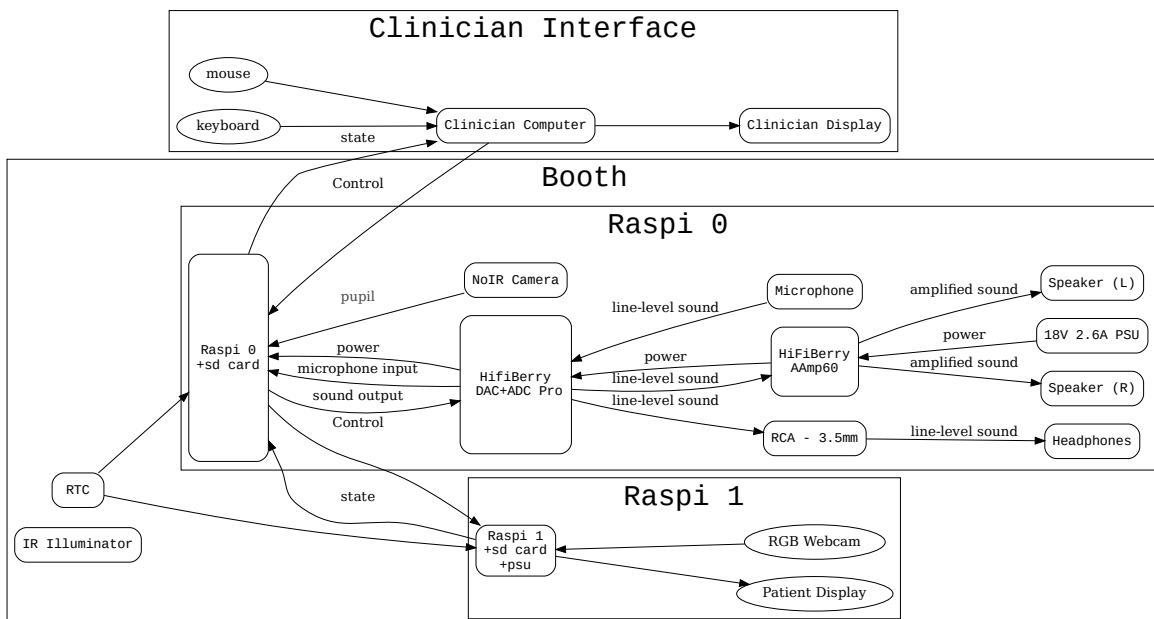
HARDWARE

1	Overview	3
2	Speaker Candidates	7
3	Software Overview	11
4	Installation	15
5	runtimes	19
6	data	23
7	types	25
8	gui	39
9	networking	43
10	psychophys	47
11	sound	53
12	stim	55
13	video	57
14	util	63
15	prefs	65
16	root	69
17	TODO	71
18	References	73
	Bibliography	75
	Python Module Index	77
	Index	79

A hardware/software package to perform audiology exams by measuring pupil dilation.

OVERVIEW

1.1 Hardware Block Diagram



1.2 Parts List

1.2.1 Clinician Interface

Name	Distributor	Number	Link	Datasheet
Raspberry Pi 400	Adafruit	1		
Touchscreen		1		
Mouse	Newegg	1		

1.2.2 Raspis

Name	Distribu- tor	Num- ber	Link	Datasheet
Raspberry Pi 4B - 8GB	Adafruit	2		- brief - mechanical - circuits - bcm2711
USB-C 5.1V 3A PSU	Adafruit	2		
18V 2.6A PSU	Digikey	1		
Samsung 64GB Pro Endurance mi- croSD	Newegg	2		
Enclosures?		2		
RTC?		1		

1.2.3 Audio

Name	Distributor	Number	Link	Datasheet
AAmp60	HiFiBerry	1		
DAC+ADC Pro	HiFiBerry	1		
Tang Band W3-1878	Parts Express	2		
Condenser Microphone		1		
Phantom Power Supply		1		
Sennheiser HDA 300		1		
RCA -> 3.5mm Adapter		1		

1.2.4 Video

Name	Distributor	Number	Link	Datasheet
NoIR Camera	Adafruit	1		
Raspberry Pi Camera V2	Adafruit	1		
Touchscreen				
Light source?				

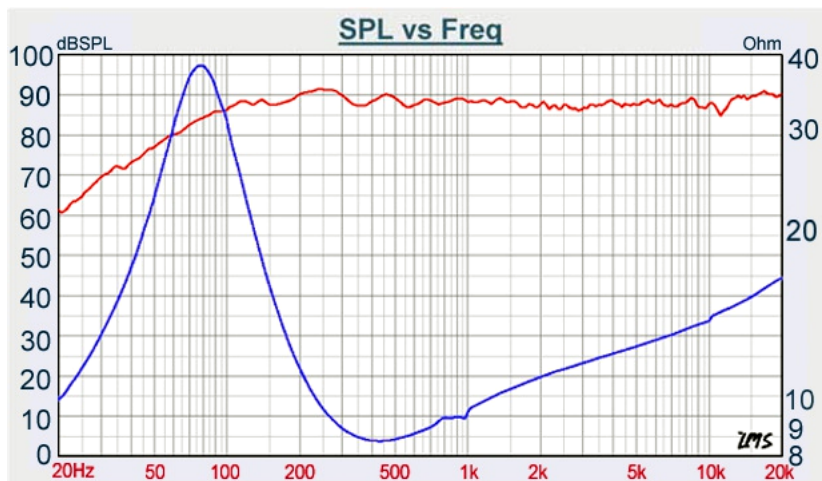
1.2.5 Etc

Name	Distributor	Number	Link	Datasheet
Netgear 5-Port Switch (GS105NA)	Newegg	1		
Ethernet Cables (10ft)	Newegg	3		
Perixx Periboard-706plus with trackball	Newegg	1		
Speaker Wire				

SPEAKER CANDIDATES

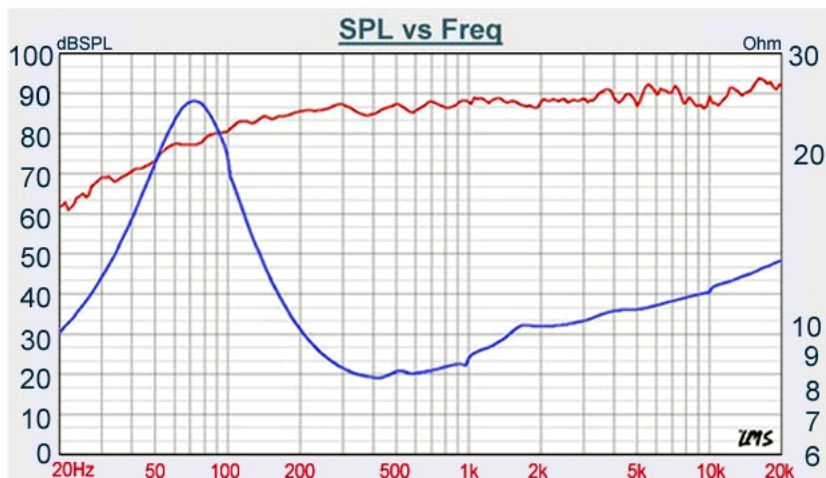
2.1 Tang Band W3-1878

\$87.49, 3"



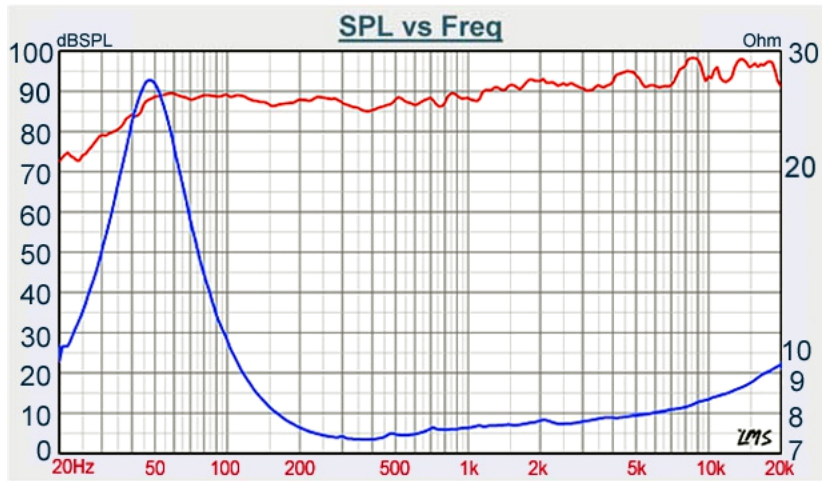
2.2 Tang Band W4-1320SJ

\$69.40, 4"



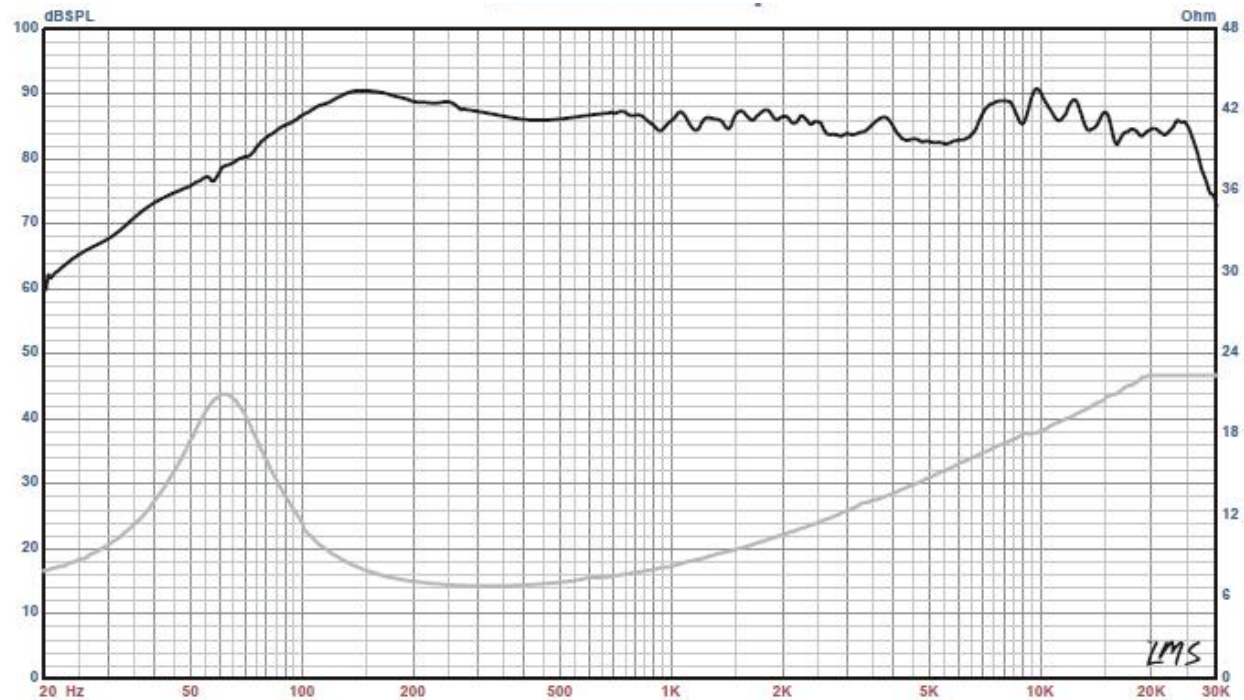
2.3 Tang Band W6-2144

\$91.10, 6.5"



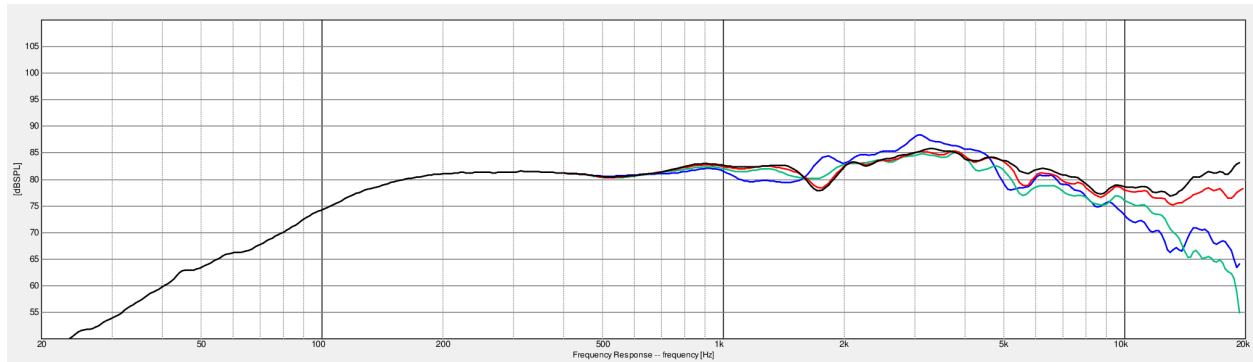
2.4 Fountek FR135EX

\$99.98, 5.5"



2.5 Dayton RS75T-8

\$25.98, 3"



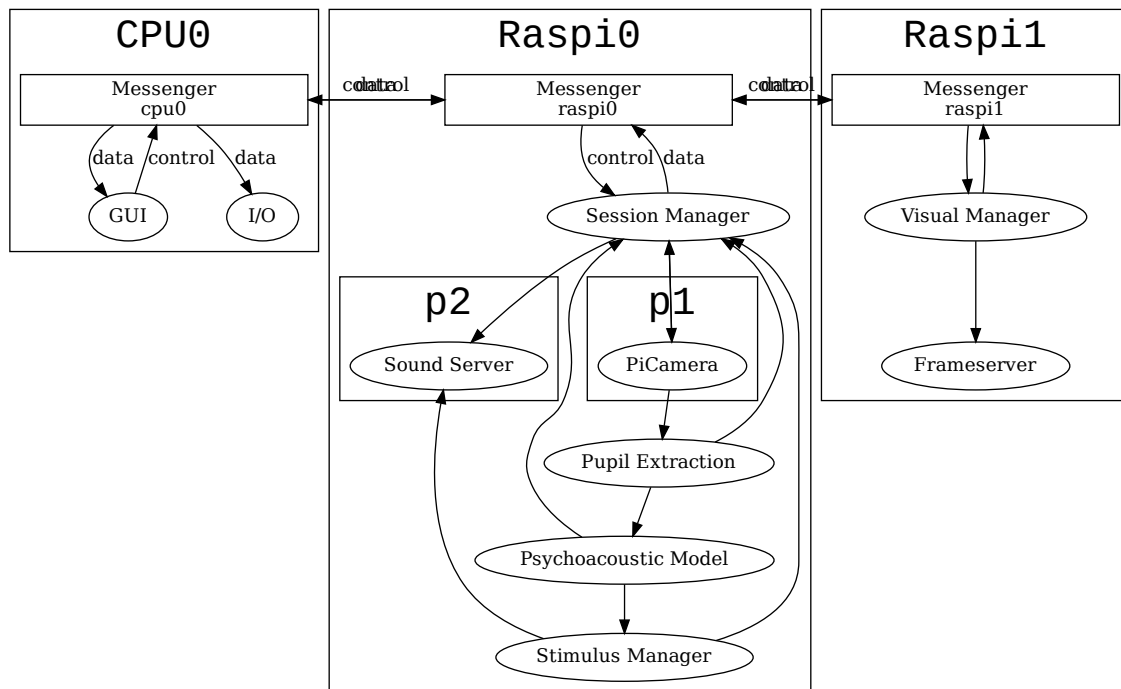
2.6 Additional References

- <http://feleppa.com.au/speakermeasmid.html>

.. _SoftwareOverview:

SOFTWARE OVERVIEW

3.1 Block Diagram



3.2 Draft Description

3.2.1 Message-based architecture

- The system will consist of separable modules that communicate with a globally-defined enum of message types.
- The system will consist of independent computers networked with an [ad-hoc or hosted? wireless network].
- Messages will be serialized and sent as tcp packets between computers, and as inter-and intra-process packets within a computer.

3.2.2 Distribution of Labor

- One computer will serve as the clinician-facing interface that controls the operation of the examination
- One computer will serve as the primary examination device, delivering auditory stimuli, measuring pupil dilation, and controlling the stepping algorithm
- One computer will serve as the patient-facing interface, presenting visual stimuli (as well as physical enclosure for cameras and speakers)

3.2.3 Synchronization

- The two computers operating the examination will be tightly synchronized with a shared real-time clock
- The examination computers will communicate asynchronously with the clinician-facing computer

3.2.4 GUI

- The GUI will be made with Qt6, written in PySide6
- [*control requirements*]
- [*display requirements*]
- [*utility requirements*]

3.2.5 Image Capture

- Pupil images will be captured with a PiCam NoIR camera as single-channel luminance images from a YUV-encoded frame
- Raw images will only be saved if explicitly requested, otherwise they will be shown in the GUI as a diagnostic

3.2.6 Pupil Processing

- Pupil diameter will be extracted as the shorter diameter of an ellipse fit on the edges of a tracked pupil to account for eccentricity/occlusion
- [*filtering and signal conditioning*]
- [*preservation of provenance*]

3.2.7 Psychoacoustic Model

this'll be a bit of work, ya!?

3.2.8 Sound

Generation

talk to avinash about stimuli

Presentation

- Sound will be presented with jack audio
- Sound will be prebuffered in the presenting process, prioritizing continuity over low-latency

Recording & Calibration

3.2.9 Video

Generation

Presentation

3.2.10 Patient Data

3.2.11 Maintenance, Logging, Debugging

INSTALLATION

4.1 Imaging RaspiOS

- Download the RaspiOS image from the [Download Page](#)
 - For Patient, download Raspi OS Lite - [HTTP](#), [.torrent](#)
 - For Clinician, download Raspi OS with Desktop - [HTTP](#), [.torrent](#)

4.1.1 Mac

- Find the name of the SD card using `sudo diskutil list`, will be something like `/dev/disk2`
- Unmount disk with `sudo diskutil unmountDisk /dev/disk2` (but replacing the name/number of your disk)
- Use `dd` to copy the image, note the use of `rdisk` (with same number) rather than `disk`. You can check the status of the transfer with `ctrl+t`:

```
sudo dd if=/path/to/raspios-image.img of=/dev/rdisk2 bs=1m
```

4.1.2 Imager

You can also use the raspberry pi imager, available for windows, mac, and ubuntu:

<https://www.raspberrypi.org/downloads.../>

4.2 Shared

On all raspberry pis, after installing the operating system you should...

4.2.1 Basic Configuration

- Change the password using `passwd`
- Update and upgrade system packages with `sudo apt update && sudo apt upgrade -y`
- Use `sudo raspi-config` to configure
 - localization settings and timezone
 - enable SSH access
 - enable WiFi access (if needed)
- If enabling SSH, install an RSA key and disable password access - see <https://wiki.auto-pi-lot.com/index.php/SSH>

4.2.2 Install system packages

Install the following system packages from apt:

```
sudo apt install -y \  
git \  
python3-pip \  
openssl \  
build-essential \  
libssl-dev \  
libffi-dev \  
libjpeg-dev \  
zlib1g-dev \  
libatlas-base-dev \  
gfortran \  
libhdf5-dev \  
cmake \  
ninja-build \  
libopenjp2-7 \  
libtiff5
```

Install rust (needed to install poetry):

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh  
source ~/.bashrc
```

You may need to add the local bin folder to PATH

```
echo 'export PATH=/home/pi/.local/bin:$PATH' >> ~/.bashrc  
source ~/.bashrc
```

Install poetry:

```
pip install --upgrade pip  
pip install poetry
```

4.2.3 Install perceptivo

Clone the repository

```
git clone https://github.com/perceptivo/perceptivo
```

install perceptivo

```
cd perceptivo
poetry shell
```

Depending on which raspi this is, you need to specify some additional, optional packages:

patient

```
poetry install -E patient
```

Clinician

```
poetry install -E clinician
```

4.3 Patient

Install additional post-install dependencies using perceptivo & autopilot scripts

- Install jackd audio from source using `jackd_source` script
- Do performance-enhancing tweaks using `performance`
- Enable hifiberry DAC / ADC Pro

Call

```
python -m perceptivo.setup --patient
```

and then restart.

4.3.1 Audio

Depending on the raspberry pi, you might need some additional configuration to tell alsa which sound card to use. By default, autopilot creates an alsa configuration file that points to the 0th card.

To tell which card to use...

```
>>> aplay -l

**** List of PLAYBACK Hardware Devices ****
card 0: vc4hdmi0 [vc4-hdmi-0], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: vc4hdmi1 [vc4-hdmi-1], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: sndrpihifiberry [snd_rpi_hifiberry_dacplusadcpro], device 0: HiFiBerry DAC+ADC
↳ Pro HiFi multicodec-0 [HiFiBerry DAC+ADC Pro HiFi multicodec-0]
```

(continues on next page)

(continued from previous page)

```
Subdevices: 0/1
Subdevice #0: subdevice #0
```

in this case we want to use card 2, so we replace that number in `/etc/asound.conf`

RUNTIMES

```
class perceptivo.runtimes.runtime.Runtime(**kwargs)
    Bases: perceptivo.root.Perceptivo_Object
    Root object for the various perceptivo runtime objects, Patient, and clinician.Clinician.
    (at the moment empty, but kept as a scaffold for shared functionality)
    property procs: List[subprocess.Popen]
        List of processes opened by this runtime agent!
        Returns typing.List[subprocess.Popen]
    abstract property prefs_class: Type[perceptivo.prefs.Prefs]
    load_prefs(prefs_file: Optional[pathlib.Path] = None) → Union[perceptivo.prefs.Prefs,
        perceptivo.prefs.Patient_Prefs, perceptivo.prefs.Clinician_Prefs]
        Load prefs file. If defaults haven't already been dumped to a prefs.json file, do so.
        Parameters prefs_file (Path) – prefs.json file to load. if None, use Runtime.prefs_file
        Returns perceptivo.prefs.Prefs a subtype of prefs, specified by Runtime.prefs_class
perceptivo.runtimes.runtime.base_args(parser: argparse.ArgumentParser) → argparse.ArgumentParser
```

5.1 clinician

entrypoint for clinician interface

```
perceptivo.runtimes.clinician.clinician_parser(manual_args: Optional[List[str]] = None) →
    argparse.Namespace
```

```
class perceptivo.runtimes.clinician.Clinician(networking: Op-
    tional[perceptivo.types.networking.Clinician_Networking]
    = None, prefs_file: pathlib.Path =
    PosixPath('/home/docs/.perceptivo/prefs.json'),
    **kwargs)
```

Bases: *perceptivo.runtimes.runtime.Runtime*

prefs_class
alias of *perceptivo.prefs.Clinician_Prefs*

init_gui()

perceptivo.runtimes.clinician.main()

5.2 patient

entrypoint for patient interface

```
class perceptivo.runtimes.patient.Patient(audio_config: perceptivo.types.sound.Audio_Config =  
    Audio_Config(fs=44100), audiogram_model: Op-  
    tional[perceptivo.types.psychophys.Psychoacoustic_Model] =  
    None, picamera_params:  
    Optional[perceptivo.types.video.Picamera_Params] = None,  
    oracle: Optional[callable] = None, pupil_extractor:  
    Optional[perceptivo.video.pupil.Pupil_Extractors] = None,  
    pupil_extractor_params:  
    Optional[perceptivo.video.pupil.EllipseExtractor_Params] =  
    None, networking:  
    Optional[perceptivo.types.networking.Patient_Networking] =  
    None, prefs_file: pathlib.Path =  
    PosixPath('/home/docs/.perceptivo/prefs.json'), **kwargs)
```

Bases: `perceptivo.runtimes.runtime.Runtime`

Runtime agent for the patient-facing Pi (see SoftwareOverview).

Runs the

- Sound Server
- PiCamera
- Processing stages including pupil extraction, psychoacoustic model, and stimulus manager

On initialization, boot the sound server and rehydrate the psychoacoustic model from the parameterization passed in `audiogram_model`. The patient runtime is parameterized by the `perceptivo.prefs.Patient_Prefs` object, which creates and reads from a `prefs.json` file (located at `perceptivo.prefs.Directories.prefs_file`).

The basic operation of the Patient runtime is encapsulated in the `trial()` method, see that for further documentation.

Parameters

- **audio_config** (*Jackd_Config*) – Configuration used to boot the jackd server
- **audiogram_model** (*Psychoacoustic_Model*) – Model parameterization used to model the audiogram as well as generate optimal stimuli to sample
- **oracle** (*callable*) – Optional, if present use an oracle to generate responses to stimuli rather than getting them from the pupil extraction method. Mostly for testing, takes a function that accepts a *Sound* object and returns a boolean response, typically generated by functions in *oracle* like `reference_audiogram()`

prefs_class

alias of `perceptivo.prefs.Patient_Prefs`

trial() → `perceptivo.types.psychophys.Sample`

One complete loop through a probe cycle. In order:

- check if a previous trial is still running using the `_trial_active` event, if so, return, logging an exception
- clear the lists that collect pupil samples: `_frames` and `_pupils`
- `next_sound()` to parameterize the next sound, returning a `types.sound.Sound` object, based on the output of the `Audiogram_Model.next()` method

- `probe()` to deliver the sound and collect the response. Within the probe method:
 - the `Picamera_Process.collecting` flag is set to indicate that it should dump frames into its queue
 - the sound is played with `play_sound()`
 - the `await_response()` method spawns a `_collecting_thread`, which calls `_collect_frames()` to pull frames from `Picamera_Process.q` and process them with `pupil_extractor` until the queue is empty. `types.video.Frame`s and `types.pupil.Pupil`s are appended to the `_frames` and `_pupils` collectors
 - once the thread finishes, the picamera's collection event is cleared, and the `types.pupil.Pupil_Params`, which set the threshold of dilation that constitutes a positive response to the sound is updated with `_update_pupil_params()`
 - The `Pupil_Params`, `Sound`, and list of `Pupil` objects are collected into a `Dilation` object and returned
- the `probe()` method then combines the `Sound` and `Dilation` objects into a `Sample` object, which is then appended to the `samples` attr
- Finally, the model is updated with the `update_model()`

Stores the `Samples` in `samples`, which also include the parameterizations and timestamps of the presented sounds

next_sound() → `perceptivo.types.sound.Sound`

Generate the next sound using the psychoacoustic model

Returns `Sound` to play

probe(*sound*: `perceptivo.types.sound.Sound`) → `Optional[perceptivo.types.psychophys.Sample]`

One loop of

- Presenting a sound stimulus
- Signaling to the other Pi to present a visual stimulus
- Estimating the Pupil Response

Returns `perceptivo.types.psychophys.Sample`

play_sound(*sound*: `perceptivo.types.sound.Sound`) → `perceptivo.types.sound.Sound`

Play a parameterized sound

Parameters `sound()`

Returns `Sound`

await_response(*sound*: `perceptivo.types.sound.Sound`) → `Optional[perceptivo.types.pupil.Dilation]`

Wait until we are given a pupil from the picamera process

Returns `bool`

_collect_frames(*start_time*: `datetime.datetime`)

Collect frames from the picamera for one sample

handle_message(*message*)

Handle a message by calling some method according to its key attribute

Parameters `message` (*bytes*) – a serialized `networking.messages.Message` object

cb_control(*control*: *Union*[*perceptivo.types.gui.GUI_Control*, *Dict*[*str*, *perceptivo.types.gui.GUI_Control*]])

Handle GUI Control messages.

Parameters **control** ()

Returns:

cb_start(*params*: *Union*[*perceptivo.types.exam.Exam_Params*, *Dict*[*str*, *perceptivo.types.exam.Exam_Params*]])

Start the exam!

Parameters **params** (*types.exam.Exam_Params*) – Parameters to run the exam!

cb_stop(*value=**None*)

Stop the exam :Parameters: **value** ()

Returns:

_update_pupil_params(*pupils*: *List*[*perceptivo.types.pupil.Pupil*]) → *perceptivo.types.pupil.Pupil_Params*

Parameters **pupils** ()

Returns:

_init_audio() → *Union*[*autopilot.stim.sound.jackclient.JackClient*, *soundcard.pulseaudio._Speaker*]

Start the jackd process, connect a client to it!

Returns *autopilot.stim.sound.jackclient.JackClient* - A booted jack client!

perceptivo.runtimes.patient.main()

Data

6.1 logging

Logging and debugging tools

`perceptivo.data.logging._LOGGERS: list = []`

List of instantiated loggers, used in `init_logger()` to return existing loggers without modification

`perceptivo.data.logging.init_logger(instance=None, module_name=None, class_name=None, object_name=None, loglevel: Optional[str] = None) → logging.Logger`

Initialize a logger

Loggers are created such that...

- There is one logger per module (eg. all gpio objects will log to hardware.gpio)
- If the passed object has a `name` attribute, that name will be prefixed to its log messages in the file
- The loglevel for the file handler and the stdout is determined by `prefs.get('LOGLEVEL')`, and if none is provided `WARNING` is used by default
- logs are rotated according to `prefs.get('LOGSIZE')` (in bytes) and `prefs.get('LOGNUM')` (number of backups of `prefs.get('LOGSIZE')` to cycle through)

Logs are stored in `prefs.get('LOGDIR')`, and are formatted like:

`"%(asctime)s - %(name)s - %(levelname)s : %(message)s"`

Loggers can be initialized either by passing an object to the first `instance` argument, or by specifying any of `module_name`, `class_name`, or `object_name` (at least one must be specified) which are combined with periods like `module.class_name.object_name`

Parameters

- **instance** – The object that we are creating a logger for! if `None`, at least one of `module`, `class_name`, or `object_name` must be passed
- **module_name** (*None, str*) – If no `instance` passed, the module name to create a logger for
- **class_name** (*None, str*) – If no `instance` passed, the class name to create a logger for
- **object_name** (*None, str*) – If no `instance` passed, the object name/id to create a logger for

Returns `logging.logger`

6.2 patient

```
class perceptivo.data.patient.Patient_Data(name: str, dob: datetime.date)
```

```
    Bases: object
```

```
    Container for patient-specific data
```

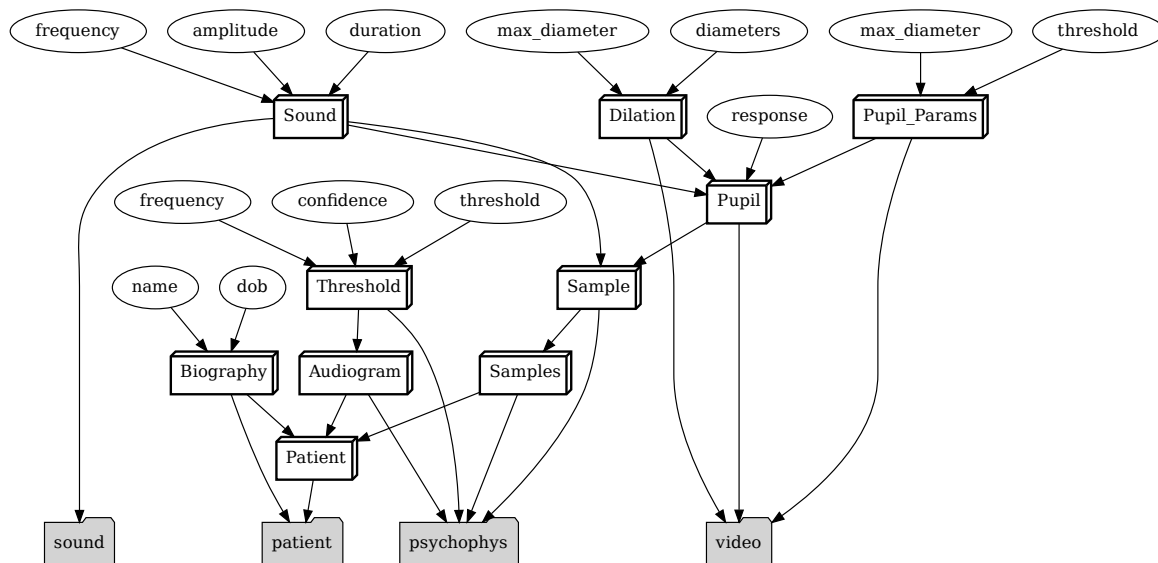
```
    name: str
```

```
    dob: datetime.date
```

TYPES

Representations of data and parameters used throughout the system.

Here are some of the basic relationships between the basic types mapped to show how they relate in order to derive a patient's audiogram



Data types to keep inter-module communication consistent

7.1 exam

Types for controlling the administration of the examination

```
class perceptivo.types.exam.Completion_Metric(*, log_likelihood: float = - 70, n_trials: int = None,  
                                              duration: float = None, use: str = 'any')
```

Bases: perceptivo.types.root.PerceptivoType

A means of deciding whether the exam is completed or not

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

log_likelihood: `Optional[float]`

End exam when log likelihood of model is below this value

n_trials: `Optional[int]`

End exam after n trials

duration: `Optional[float]`

End exam after n minutes

use: `str`

Name of which (non-None) metric to use. Default `any` for ending exam if any of the criteria are met

```
class perceptivo.types.exam.Exam_Params(*, frequencies: Tuple[float], amplitudes: Tuple[float], iti: float,
                                         iti_jitter: float = 0.1, completion_metric:
                                         perceptivo.types.exam.Completion_Metric =
                                         Completion_Metric(log_likelihood=- 70, n_trials=None,
                                         duration=None, use='any'), allow_repeats: bool = False)
```

Bases: `perceptivo.types.root.PerceptivoType`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

frequencies: `Tuple[float]`

Frequencies (Hz) to test in exam

amplitudes: `Tuple[float]`

Amplitudes (dbSPL) to test in exam

iti: `float`

Seconds between each trial

iti_jitter: `float`

Amount to jitter trials as a proportion of the ITI (eg. 0.1 for an iti of 5s would be maximum 0.5s of jitter)

completion_metric: `perceptivo.types.exam.Completion_Metric`

Metric that decides when the exam is over.

allow_repeats: `bool`

Allow repeated sounds

7.2 gui

`perceptivo.types.gui.GUI_PARAM_KEY`

Possible keys for GUI parameters.

- `frequencies` - a tuple of frequencies to test
- `amplitudes` - a tuple of amplitudes to test
- `log_x` - boolean indicating whether an x-axis should be log scaled (True) or linearly scaled
- `log_y` - boolean indicating whether a y-axis should be log scaled (True) or linearly scaled
- `extra_amplitude` - boolean indicating whether an additional, suprathreshold amplitude should be tested as a confirmation
- `amplitude_step` - Step size of amplitudes to test in dB

alias of `Literal`['frequencies', 'amplitudes', 'log_x', 'log_y', 'extra_amplitude', 'amplitude_step', 'amplitude_range', 'max_amplitude', 'frequency_step', 'frequency_range', 'iti', 'iti_jitter']

perceptivo.types.gui.GUI_WIDGET_TYPE

Widget types that correspond to particular Qt Widgets

- int, float - PySide.QtWidgets.QSpinBox and PySide.QtWidgets.QDoubleSpinBox
- range - *widgets.components.Range_Setter*
- tuple - PySide.QtWidgets.QLineEdit evaluated by `ast.literal_eval`
- bool - PySide.QtWidgets.QCheckBox

alias of `Literal['int', 'float', 'range', 'tuple', 'bool']`

```
class perceptivo.types.gui.GUI_Control(*, key: Literal['frequencies', 'amplitudes', 'log_x', 'log_y',
                                                    'extra_amplitude', 'amplitude_step', 'amplitude_range',
                                                    'max_amplitude', 'frequency_step', 'frequency_range', 'iti',
                                                    'iti_jitter'], value: Union[str, float, tuple])
```

Bases: `perceptivo.types.root.PerceptivoType`

Container for GUI_Params in transit

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

key: `Literal['frequencies', 'amplitudes', 'log_x', 'log_y', 'extra_amplitude', 'amplitude_step', 'amplitude_range', 'max_amplitude', 'frequency_step', 'frequency_range', 'iti', 'iti_jitter']`

value: `Union[str, float, tuple]`

```
class perceptivo.types.gui.GUI_Range(*, min: float, max: float, n: int)
```

Bases: `perceptivo.types.root.PerceptivoType`

Range for *widgets.components.Range_Setter*

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

min: `float`

max: `float`

n: `int`

```
class perceptivo.types.gui.GUI_Param(*, key: Literal['frequencies', 'amplitudes', 'log_x', 'log_y',
                                                    'extra_amplitude', 'amplitude_step', 'amplitude_range',
                                                    'max_amplitude', 'frequency_step', 'frequency_range', 'iti',
                                                    'iti_jitter'], name: str, widget_type: Literal['int', 'float', 'range',
                                                    'tuple', 'bool'], default: Optional[Union[float, int,
                                                    perceptivo.types.gui.GUI_Range, tuple]] = None, args: list = None,
                                                    kwargs: dict = None)
```

Bases: `perceptivo.types.root.PerceptivoType`

Parameterization for a GUI Parameter itself. ie. How a particular parameter should be represented.

Params: key (GUI_PARAMS): the key used for the parameter name (str): A human readable name for the parameter widget_type (GUI_WIDGETS): A string that indicates the type of widget that should be used.

Different `widget_type`s may use different widgets, combinations of widgets, and validators, and are thus not strictly isomorphic to a single widget type.

default (any): the default value to be set, must correspond to widget type args (list): args to pass to the widget kwargs (dict): kwargs to pass to the widget

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
key: Literal['frequencies', 'amplitudes', 'log_x', 'log_y', 'extra_amplitude',
'amplitude_step', 'amplitude_range', 'max_amplitude', 'frequency_step',
'frequency_range', 'iti', 'iti_jitter']
```

```
name: str
```

```
widget_type: Literal['int', 'float', 'range', 'tuple', 'bool']
```

```
default: Optional[Union[float, int, perceptivo.types.gui.GUI_Range, tuple]]
```

```
args: list
```

```
kwargs: dict
```

```
class perceptivo.types.gui.Control_Panel_Params(*, amplitude_range: perceptivo.types.gui.GUI_Param
= GUI_Param(key='amplitude_range',
name='Amplitude Range (dBSPL)',
widget_type='range', default=GUI_Range(min=0.0,
max=80.0, n=8), args=[], kwargs={'limits': (0,
100)}), frequency_range:
perceptivo.types.gui.GUI_Param =
GUI_Param(key='frequency_range',
name='Frequency Range (Hz)', widget_type='range',
default=GUI_Range(min=0.0, max=8000.0, n=17),
args=[], kwargs={'limits': (0, 20000)}), iti:
perceptivo.types.gui.GUI_Param =
GUI_Param(key='iti', name='Inter-Trial Interval (s)',
widget_type='float', default=5.0, args=[],
kwargs={}), iti_jitter:
perceptivo.types.gui.GUI_Param =
GUI_Param(key='iti_jitter', name='Inter-Trial Jitter
(proportion of ITI)', widget_type='float', default=0.1,
args=[], kwargs={}))
```

Bases: `perceptivo.types.root.PerceptivoType`

Defaults and parameters for `perceptivo.gui.widgets.Control_Panel`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class perceptivo.types.gui.GUI_Params(*, control_panel: perceptivo.types.gui.Control_Panel_Params =
Con-
trol_Panel_Params(amplitude_range=GUI_Param(key='amplitude_range',
name='Amplitude Range (dBSPL)', widget_type='range',
default=GUI_Range(min=0.0, max=80.0, n=8), args=[],
kwargs={'limits': (0, 100)}),
frequency_range=GUI_Param(key='frequency_range',
name='Frequency Range (Hz)', widget_type='range',
default=GUI_Range(min=0.0, max=8000.0, n=17), args=[],
kwargs={'limits': (0, 20000)}), iti=GUI_Param(key='iti',
name='Inter-Trial Interval (s)', widget_type='float', default=5.0,
args=[], kwargs={}), iti_jitter=GUI_Param(key='iti_jitter',
name='Inter-Trial Jitter (proportion of ITI)', widget_type='float',
default=0.1, args=[], kwargs={})))
```

Bases: `perceptivo.types.root.PerceptivoType`

Container for all parameters to be given to the GUI on init

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

control_panel: `perceptivo.types.gui.Control_Panel_Params`

7.3 networking

```
class perceptivo.types.networking.Socket(id: str, socket_type: Literal['REQ', 'REP', 'PUB', 'SUB', 'PAIR',
                                                                    'DEALER', 'ROUTER', 'PULL', 'PUSH'], protocol:
                                                                    Literal['tcp', 'ipc', 'inproc'], mode: Literal['connect', 'bind'],
                                                                    port: int, ip: str = '*', to: Optional[str] = None)
```

Bases: `object`

id: `str`

socket_type: `Literal['REQ', 'REP', 'PUB', 'SUB', 'PAIR', 'DEALER', 'ROUTER', 'PULL', 'PUSH']`

protocol: `Literal['tcp', 'ipc', 'inproc']`

mode: `Literal['connect', 'bind']`

port: `int`

ip: `str = '*'`

to: `Optional[str] = None`

```
class perceptivo.types.networking.Clinician_Networking(*, ip: str = "", patient_ip: str = "", eyecam:
                                                                    perceptivo.types.networking.Socket =
                                                                    Socket(id='clinician:eyecam',
                                                                    socket_type='PULL', protocol='tcp',
                                                                    mode='bind', port=5500, ip='*', to=None),
                                                                    control: perceptivo.types.networking.Socket
                                                                    = Socket(id='clinician:control',
                                                                    socket_type='ROUTER', protocol='tcp',
                                                                    mode='bind', port=5600, ip='*', to=None))
```

Bases: `pydantic.main.BaseModel`

Default networking properties for the Clinician computer

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

ip: `str`

patient_ip: `str`

eyecam: `perceptivo.types.networking.Socket`

control: `perceptivo.types.networking.Socket`

```
class perceptivo.types.networking.Patient_Networking(*, ip: str = "", clinician_ip: str = "", eyecam:
                                                    perceptivo.types.networking.Socket =
                                                    Socket(id='patient:eyecam',
                                                    socket_type='PUSH', protocol='tcp',
                                                    mode='connect', port=5500, ip="", to=None),
                                                    control: perceptivo.types.networking.Socket =
                                                    Socket(id='patient:control',
                                                    socket_type='DEALER', protocol='tcp',
                                                    mode='connect', port=5600, ip="",
                                                    to='clinician:control'))
```

Bases: `pydantic.main.BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

ip: `str`

clinician_ip: `str`

eyecam: `perceptivo.types.networking.Socket`

control: `perceptivo.types.networking.Socket`

7.4 patient

```
class perceptivo.types.patient.Biography(name: str, dob: datetime.date)
```

Bases: `object`

Biographical details for a patient

name: `str`

dob: `datetime.date`

```
class perceptivo.types.patient.Patient(biography: perceptivo.types.patient.Biography, samples:
                                         perceptivo.types.psychophys.Samples, audiogram:
                                         perceptivo.types.psychophys.Audiogram)
```

Bases: `object`

Data for a given patient

biography: `perceptivo.types.patient.Biography`

samples: `perceptivo.types.psychophys.Samples`

audiogram: `perceptivo.types.psychophys.Audiogram`

```
class perceptivo.types.patient.Collection_Params(*, collection_wait: float = 5)
```

Bases: `pydantic.main.BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

collection_wait: `float`

Total duration to wait to collect pupil frames, starting when the sound does.

7.5 psychophys

```
class perceptivo.types.psychophys.Sample(sound: perceptivo.types.sound.Sound, dilation:
                                         typing.Optional[perceptivo.types.pupil.Dilation] = None,
                                         timestamp: datetime.datetime = <factory>, response:
                                         dataclasses.InitVar[bool] = <property object>)
```

Bases: `object`

A single sample of a psychophysical response to a sound

Variables

- `~Sample.dilation` (`types.pupil.Dilation`) – Pupil object storing dilation for a given sample
- `~Sample.sound` (`types.sound.Sound`) – Sound presented to elicit Pupil response
- `~Sample.timestamp` (`datetime.datetime`) – Timestamp at which the response was elicited

Properties: `response` (bool): Sub/Subrathreshold response from `Pupil.response`

sound: `perceptivo.types.sound.Sound`

dilation: `Optional[perceptivo.types.pupil.Dilation]` = `None`

timestamp: `datetime.datetime`

property response: `bool`

```
class perceptivo.types.psychophys.Samples(samples:
                                         Optional[List[perceptivo.types.psychophys.Sample]] = None,
                                         dilations: Optional[List[perceptivo.types.pupil.Dilation]] =
                                         None, frequencies: Optional[List[float]] = None, amplitudes:
                                         Optional[List[float]] = None, responses:
                                         Optional[List[bool]] = None)
```

Bases: `object`

Multiple Samples!

Convenience class to init samples from numpy arrays and convert to pandas dataframe

samples: `List[perceptivo.types.psychophys.Sample]`

responses: `List[bool]`

frequencies: `List[float]`

amplitudes: `List[float]`

append(sample: `perceptivo.types.psychophys.Sample`)

Add a sample to the collection

Parameters `sample` (`Sample`) – A New Sample!

to_df() → `pandas.core.frame.DataFrame`

Make a dataframe with sound parameterization flattened out

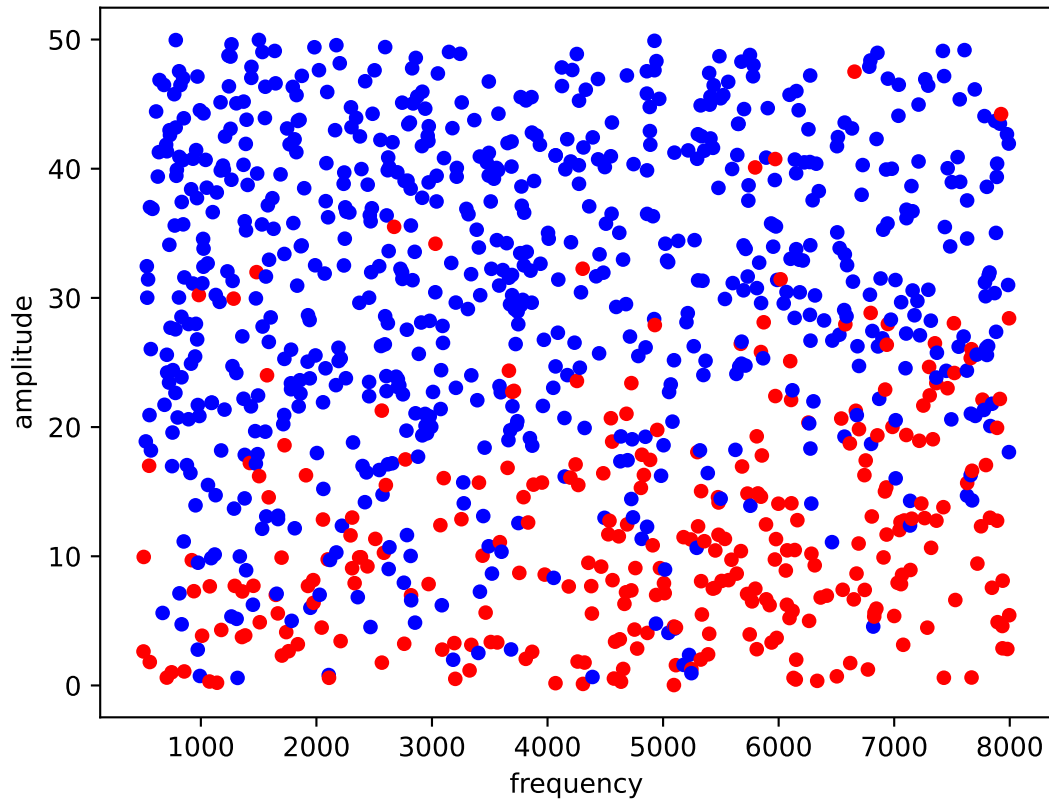
plot(show=True)

Plot a collection of samples as points, with blue meaning the sample was audible and red meaning inaudible

Examples

```
from perceptivo.psychophys.oracle import generate_samples

samples = generate_samples(n_samples=1000, scale=10)
samples.plot()
```



Parameters `show` (*bool*) – If True (default), call `plt.show()`

class `perceptivo.types.psychophys.Threshold`(*frequency: float, threshold: float, confidence: float = 0*)
 Bases: `object`

The audible threshold for a particular frequency

Parameters

- **frequency** (*float*) – Frequency of threshold in Hz
- **threshold** (*float*) – Audible threshold in dbSPL
- **confidence** (*float*) – Confidence of threshold, units vary depending on estimation type

frequency: `float`

threshold: `float`

confidence: `float = 0`

class `perceptivo.types.psychophys.Audiogram`(*thresholds*: `List[perceptivo.types.psychophys.Threshold]`)
 Bases: `object`

A collection of `Threshold`'s that represent a patient's audiogram.

Thresholds can be accessed like a dictionary, using frequencies as keys, eg:

```
>>> agram = Audiogram([Threshold(1000, 10), Threshold(2000, 20)])
>>> agram[1000]
Threshold(frequency=1000, threshold=10, confidence=0)
>>> agram[3000] = Threshold(3000, 30)
>>> agram[3000]
Threshold(frequency=1000, threshold=10, confidence=0)
```

thresholds: `List[perceptivo.types.psychophys.Threshold]`

property frequencies: `List[float]`

List of frequencies in *thresholds*

to_dict() → `Dict[float, float]`

Return audiogram thresholds as a {frequency:threshold} dictionary, eg.:

```
>>> agram = Audiogram([Threshold(1000, 10), Threshold(2000, 20)])
>>> agram.to_dict()
{1000: 10, 2000: 20}
```

class `perceptivo.types.psychophys.Kernel`(*, *length_scale*: `Tuple[float, float] = (100.0, 200.0)`,
length_scale_bounds: `Tuple[float, float] = (1, 100000.0)`)

Bases: `pydantic.main.BaseModel`

Default kernel to use with *psychophys.model.Gaussian_Process*

Uses a kernel with a short length scale for frequency, but a longer length scale for amplitude, which should be smoother/monotonic where frequency can have an unpredictable shape

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

length_scale: `Tuple[float, float]`

length_scale_bounds: `Tuple[float, float]`

property kernel: `sklearn.gaussian_process.kernels.RBF`

class `Config`

Bases: `object`

arbitrary_types_allowed = `True`

class `perceptivo.types.psychophys.Psychoacoustic_Model`(*model_type*:
`typing.Literal['Gaussian_Process'] = 'Gaussian_Process'`, *args*:
`typing.Optional[list] = <factory>`, *kwargs*:
`typing.Optional[typing.Dict[str, perceptivo.types.psychophys.Kernel]] = <factory>`)

Bases: `object`

Parameterization of a psychoacoustic model to use to estimate audiograms and control the presentation of stimuli

```
model_type: Literal['Gaussian_Process'] = 'Gaussian_Process'
args: Optional[list]
kwargs: Optional[Dict[str, perceptivo.types.psychophys.Kernel]]
```

7.6 pupil

Types specifically for carrying and manipulating pupil measurements

```
class perceptivo.types.pupil.Pupil(ellipse: perceptivo.types.units.Ellipse, frame:
                                   perceptivo.types.video.Frame)
```

Bases: `object`

A single-frame measurement of a pupil

Variables

- `~Pupil.ellipse` (*Ellipse*) – Fit ellipse given frame
- `~Pupil.params` (*Pupil_Params*) – Pupil parameterization!

`ellipse:` *perceptivo.types.units.Ellipse*

`frame:` *perceptivo.types.video.Frame*

```
class perceptivo.types.pupil.Pupil_Params(threshold: float, max_diameter: float)
```

Bases: `object`

Parameters to use with *video.pupil.PupilExtractor* classes to parameterize

Variables

- `~Pupil_Params.threshold` (*float*) – Diameter threshold as a fraction of maximum diameter to consider a positive response to a stimulus
- `~Pupil_Params.max_diameter` (*float*) – Maximum diameter of pupil in pixels

`threshold:` *float*

`max_diameter:` *float*

```
class perceptivo.types.pupil.Dilation(params: perceptivo.types.pupil.Pupil_Params, pupils:
                                       List[perceptivo.types.pupil.Pupil], timestamps:
                                       List[datetime.datetime])
```

Bases: `object`

A timeseries of pupil diameters and timestamps corresponding to a pupil dilation event

Variables

- `~Dilation.ellipses` (*List[Pupil]*) – List of ellipses from a pupil measurement
- `~Dilation.timestamps` (*List[datetime.datetime]*) – List of timestamps of equal length to ellipses
- `~Dilation.sound` (*types.sound.Sound*) – Sound that was presented for this pupil response

Properties: `max_diameter` (*float*): maximum diameter reached during a given sample diameters (*typing.List[float]*): List of diameters in pixels of equal length to `timestamps` response (*bool*): True/False whether the sound was heard, calculated by dividing

the maximum measured pupil dilation in pixels / maximum possible dilation in pixels and comparing to the detection threshold. Aka (*Dilation.max_diameter* / *Pupil_Params.max_diameter*) > *Pupil_Params.threshold*

params: *perceptivo.types.pupil.Pupil_Params*

pupils: List[*perceptivo.types.pupil.Pupil*]

timestamps: List[*datetime.datetime*]

property diameters: List[float]

Extract major axes from ellipses

Returns List of major axes in pixels

Return type List[float]

property max_diameter: float

property response: bool

7.7 sound

class *perceptivo.types.sound.Audio_Config*(fs: int = 44100)

Bases: *object*

Base class for audio configuration

Params: fs (int): Sampling rate in Hz, default 44100

fs: int = 44100

class *perceptivo.types.sound.Jackd_Config*(fs: int = 44100, bin: *pathlib.Path* = <factory>, priority: int = 75, driver: str = 'alsa', device_name: typing.Union[str, int] = 'hw:sndrpihifiberry', nperiods: int = 3, period: int = 1024, playback_only: bool = True, outchannels: list = <factory>)

Bases: *perceptivo.types.sound.Audio_Config*

Configure the jackd daemon used by the sound server, see <https://linux.die.net/man/1/jackd>

Params: bin (*pathlib.Path*): Path to the jackd binary priority (int): Priority to run the process (higher is better), default 75 driver (str): Driver to use, default 'alsa' device_name (str, int): Device to use in alsa's parlance, default 'hw:sndrpihifiberry'.

Also accepts ints for use with coreaudio

nperiods (int): Number of periods per buffer cycle, default 3 period (int): size of period, default 1024 samples. launch_str (str): launch string with arguments compiled from the other arguments

bin: *pathlib.Path*

priority: int = 75

driver: str = 'alsa'

device_name: Union[str, int] = 'hw:sndrpihifiberry'

nperiods: int = 3

period: int = 1024

playback_only: bool = True

outchannels: list

property launch_str: `str`

class `perceptivo.types.sound.Sound`(**frequency: float, amplitude: float, duration: float = 0.5, sound_type: Literal['Gammatone'] = 'Gammatone', timestamp: datetime.datetime = None, jack_client: JackClient = None, uuid: str = None*)

Bases: `pydantic.main.BaseModel`

Parameterization of an abstract probe sound

Parameters

- **frequency** (*float*) – Frequency in Hz
- **amplitude** (*float*) – Amplitude in dbSPL
- **duration** (*float*) – Duration of sound in seconds

Variables `~Sound.uuid` (*str*) – Unique UUID to identify sounds

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

frequency: `float`

amplitude: `float`

duration: `float`

sound_type: `Literal['Gammatone']`

timestamp: `Optional[datetime.datetime]`

jack_client: `Optional[JackClient]`

uuid: `str`

class `Config`

Bases: `object`

arbitrary_types_allowed: `bool = True`

stamp_time()

Record the time that the sound is played in `Sound.timestamp`

property sound_kwargs: `dict`

Sound kwargs that the sound class accepts

(ie. filtering out `sound_type` and others the sound class doesn't take)

Returns `dict` of arguments

property sound_class: `autopilot.stim.sound.base.Sound`

The sound class that corresponds to the `sound_type` retrieved from the `perceptivo.sound.sounds` module.

Returns `autopilot.stim.sound.sounds.Jack_Sound` - The sound class!

7.8 units

Very basic units or unit-like things

class `perceptivo.types.units.Ellipse`(*x: int, y: int, a: float, b: float, t: float*)

Bases: `object`

Parameterization of an ellipse corresponding to a Pupil

Attrs: *x* (int): Ellipse center in pixels *y* (int): Ellipse center in pixels *a* (float): Major axis in pixels *b* (float): Minor axis in pixels *t* (float): Orientation in radians, clockwise from vertical

x: `int`

y: `int`

a: `float`

b: `float`

t: `float`

mask(*scale: float = 1*) → `Tuple[numpy.ndarray, numpy.ndarray]`

Coordinates for a boolean mask, created with `skimage.draw.ellipse()`

Note: When calling `ellipse`, *y* is used as the 0th dimension and *x* as the 1st, since rows in a frame (*y*) are typically the 0th dimension.

Parameters *scale* (`float`) – Scale the major and minor axes by this much!

Returns tuple of two ndarrays, coordinates in the 0th and 1st axis of the mask points

7.9 video

class `perceptivo.types.video.Frame`(*, *frame: numpy.ndarray, timestamp: datetime.datetime = None, color: bool = None, cropped: Frame = None, dtype: numpy.dtype = None*)

Bases: `perceptivo.types.root.PerceptivoType`

Single video frame container

Variables

- **~Frame.frame** (`numpy.ndarray`) – Frame!
- **~Frame.timestamp** (`datetime.datetime`) – Time of acquisition
- **~Frame.color** (`bool`) – If `False`, grayscale (frame should be 2 dimensional or 3rd axis should be `len == 1`). if `True`, RGB Color.

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

frame: `numpy.ndarray`

timestamp: `datetime.datetime`

cropped: `Optional[perceptivo.types.video.Frame]`

dtype: `Optional[numpy.dtype]`

color: `Optional[bool]`

set_color(*color*)

norm()

make frame 0-1

property gray: `numpy.ndarray`

Grayscale version of the frame, if color

crop(*bbox*: `List[int]`)

Crop with a bounding box (top, bottom, left, right), assign to self.cropped

Parameters *bbox* ()

Returns new Frame image with cropped image as its frame

class `Config`

Bases: `object`

arbitrary_types_allowed: `bool = True`

```
class perceptivo.types.video.Picamera_Params(*, sensor_mode: int = 0, resolution: Tuple[int, int] =
(1280, 720), fps: int = 30, format: Literal['rgb',
'grayscale'] = 'grayscale', output_file: pathlib.Path =
None)
```

Bases: `pydantic.main.BaseModel`

Configuration for a `perceptivo.video.cameras.PiCamera`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

sensor_mode: `int`

resolution: `Tuple[int, int]`

fps: `int`

format: `Literal['rgb', 'grayscale']`

output_file: `Optional[pathlib.Path]`

```
class perceptivo.types.video.Camera_Calibration(picam: perceptivo.types.video.Picamera_Params,
distance: float, mm_per_px: float)
```

Bases: `object`

Parameters that define the conditions of use for a camera

Parameters

- **picam** (*Picamera_Params*) – Parameterization of the PiCamera
- **distance** (*float*) – distance from camera to subject in mm
- **mm_per_px** (*float*) – approximate number of mm per pixel at a given distance

picam: `perceptivo.types.video.Picamera_Params`

distance: `float`

mm_per_px: `float`

8.1 main

Main Gui container for Perceptivo Clinician interface

```
class perceptivo.gui.main.Perceptivo_Clinician(*args: Any, **kwargs: Any)
    Bases: PySide6.QtWidgets.QMainWindow
    GUI container for the Perceptivo clinician interface
    controlChanged
        alias of perceptivo.types.gui.GUI_Control
    changeControl(value: perceptivo.types.gui.GUI_Control)
        Receive changed control settings from widgets, etc. and emit them to the patient.
        Also emits from the controlChanged signal
        Parameters value (types.gui.GUI_Control) – Control value that changed
    setStarted(value: bool)
    property exam_params: perceptivo.types.exam.Exam_Params
    property settings
        Load and return PySide6.QtCore.QSettings
    drawFrame(frame)
    receive_messages()
    cb_connect(msg: perceptivo.networking.messages.Message)
        Parameters msg ()
        Returns:
    cb_data(msg: perceptivo.networking.messages.Message)
        Receive data from the patient during an exam
        Message that contains a types.psychophys.Sample
    closeEvent(event)
class Frame_Receiver(*args: Any, **kwargs: Any)
    Bases: PySide6.QtCore.QThread
    Thread to launch a networking node, receive threads, and emit new frames
```

frame
alias of `perceptivo.types.video.Frame`

run()

quitting()

8.2 params

8.3 styles

8.3.1 Styles

Values and stylesheets that control the appearance of the GUI

8.4 widgets

8.4.1 audiogram

Plot displaying audiogram options, current estimate of audiogram

```
class perceptivo.gui.widgets.audiogram.Audiogram(*args: Any, **kwargs: Any)
    Bases: PySide6.QtWidgets.QGroupBox

    gridChanged(value: perceptivo.types.gui.GUI_Control)
    scaleChanged(value: perceptivo.types.gui.GUI_Control)
```

8.4.2 components

Subcomponents for larger GUI widgets

```
class perceptivo.gui.widgets.components.Range_Setter(*args: Any, **kwargs: Any)
    Bases: PySide6.QtWidgets.QWidget
```

Buttons and text fields to parameterize a linearly or logarithmically spaced array of values

Parameters

- **key** (*str*) – key of value that is set by this widget, likely one of `types.GUI_PARAM_KEY`
- **name** (*str*) – human-readable name of parameter
- **round** (*int*) – Digits to round generated values to (default 0)
- **limits** (*tuple*) – Absolute allowable maximum and minimum
- **step** (*float*) – Step size of the spinboxes
- ***args, **kwargs** – passed to `PySide6.QtWidgets.QWidget`

valueChanged
alias of `perceptivo.types.gui.GUI_Control`

buttonClicked
alias of `perceptivo.types.gui.GUI_Control`

scaleChangedalias of `str`

__init__(key: *Literal*['frequencies', 'amplitudes', 'log_x', 'log_y', 'extra_amplitude', 'amplitude_step', 'amplitude_range', 'max_amplitude', 'frequency_step', 'frequency_range', 'iti', 'iti_jitter'], name: *str*, round: *int* = 0, limits: *Tuple*[*int*, *int*] = (0, 100), default: `perceptivo.types.gui.GUI_Range` = `GUI_Range`(min=0.0, max=100.0, n=10), *args, **kwargs)

Parameters

- **key** (*str*) – key of value that is set by this widget, likely one of `types.GUI_PARAM_KEY`
- **name** (*str*) – human-readable name of parameter
- **round** (*int*) – Digits to round generated values to (default 0)
- **limits** (*tuple*) – Absolute allowable maximum and minimum
- **step** (*float*) – Step size of the spinboxes
- ***args, **kwargs** – passed to `PySide6.QtWidgets.QWidget`

value() → *Tuple*[*float*]**setMaximum**(value: *float*)**setMinimum**(value: *float*)

8.4.3 control_panel

Control operation of perceptivo, set audiogram params

class `perceptivo.gui.widgets.control_panel.Control_Panel`(*args: *Any*, **kwargs: *Any*)Bases: `PySide6.QtWidgets.QGroupBox`**valueChanged**alias of `perceptivo.types.gui.GUI_Control`**scaleChanged**alias of `perceptivo.types.gui.GUI_Control`**startToggled**alias of `perceptivo.types.gui.GUI_Control`**setValue**(value: `perceptivo.types.gui.GUI_Control`)

8.4.4 patient

Dialog popup to set patient params

8.4.5 pupil

Timeseries of pupil diameter, audio/stimulus presentation info

```
class perceptivo.gui.widgets.pupil.Pupil(*args: Any, **kwargs: Any)  
    Bases: PySide6.QtWidgets.QGroupBox
```

8.4.6 video

<https://stackoverflow.com/a/35316662/13113166>

```
class perceptivo.gui.widgets.video.Video(*args: Any, **kwargs: Any)  
    Bases: pyqtgraph.ImageView
```

NETWORKING

Networking between objects and computers

9.1 messages

Message classes for explicit typing and the sanity of clear expectations

```
class perceptivo.networking.messages.Message(message_number: Optional[int] = None, timestamp:
                                           Optional[datetime.datetime] = None, key: str = "",
                                           **kwargs)
```

Bases: *perceptivo.root.Perceptivo_Object*

Message container implementing msgpack-based numpy array de/serialization.

Subclass this to make specific message types!

Parameters ****kwargs** (*dict*) – key/value pairs stored in `Message.value`

Attrs: `value` (*dict*): (deserialized) dictionary of values passed from ****kwargs**

counter = `count(0)`

```
__init__(message_number: Optional[int] = None, timestamp: Optional[datetime.datetime] = None, key: str
         = "", **kwargs)
```

Parameters ****kwargs** (*dict*) – key/value pairs stored in `Message.value`

Attrs: `value` (*dict*): (deserialized) dictionary of values passed from ****kwargs**

serialize(*msg: Optional[dict] = None*) → *bytes*

classmethod from_serialized(*msg: bytes*) → *perceptivo.networking.messages.Message*

Create an instance of `Message` from a msgpack serialized bytestring

9.2 node

Messenger objects for communication intra, interprocess and intercomputer

```
class perceptivo.networking.node.Node(socket: perceptivo.types.networking.Socket, poll_mode:
                                     perceptivo.networking.node.Node.Poll_Mode =
                                     Poll_Mode.IOLOOP, callback: Optional[Callable] = None, to:
                                     Optional[str] = None, deque_size: int = 256)
```

Bases: `perceptivo.root.Perceptivo_Object`

Wrapper around zmq sockets to send and receive messages

Parameters

- **socket** (`types.Socket`) – Socket descriptor (see `Socket`)
- **poll_mode** (`Poll_Mode`) – Strategy for polling messages.
 - **IOLOOP** - uses tornado's **IOloop** and **ZMQStreams** to poll for messages. Needs to be given callback as well, which will be called with the received message as the only argument
 - **DEQUE** - a thread is spawned to poll the socket and add any message to deque
 - **NONE** - interact with the socket manually
- **callback** (`typing.Callable`) – A callable object that will be called with a received message as its only argument if `poll_mode == IOLOOP`

```
class Poll_Mode(value)
```

Bases: `enum.Enum`

An enumeration.

IOLOOP = 1

DEQUE = 2

NONE = 3

```
__init__(socket: perceptivo.types.networking.Socket, poll_mode:
         perceptivo.networking.node.Node.Poll_Mode = Poll_Mode.IOLOOP, callback:
         Optional[Callable] = None, to: Optional[str] = None, deque_size: int = 256)
```

Wrapper around zmq sockets to send and receive messages

Parameters

- **socket** (`types.Socket`) – Socket descriptor (see `Socket`)
- **poll_mode** (`Poll_Mode`) – Strategy for polling messages.
 - **IOLOOP** - uses tornado's **IOloop** and **ZMQStreams** to poll for messages. Needs to be given callback as well, which will be called with the received message as the only argument
 - **DEQUE** - a thread is spawned to poll the socket and add any message to deque
 - **NONE** - interact with the socket manually
- **callback** (`typing.Callable`) – A callable object that will be called with a received message as its only argument if `poll_mode == IOLOOP`

property address: `str`

The full address, including protocol, ip, port, or endpoint, depending on the protocol

Returns str

send(msg: *Optional*[perceptivo.networking.messages.Message] = None, to: *Optional*[str] = None, ***kwargs*)
for now just wrapping the socket

_start_ioloop(loop: tornado.ioloop.IOLoop)
spawn a tornado ioloop

_start_polling()
spawn a thread to poll the socket and add incoming messages to the queue

release()

9.3 sockets

Abstract description of socket topology.

Each entry in each dict is a set of sockets to be run in an independent process.

```
perceptivo.networking.sockets.CLINICIAN = {'gui': (Socket(id='clinician.command',
socket_type='PUB', protocol='tcp', mode='bind', port=5000, ip='*', to=None),
Socket(id='clinician.data', socket_type='ROUTER', protocol='tcp', mode='bind', port=5001,
ip='*', to=None))}
```

Sockets used by the clinician object

- clinician.command - PUB socket for dispersing control commands to subordinate computers
- clinician.data - ROUTER for receiving data from subordinate computers

```
perceptivo.networking.sockets.EXAMINER = {'data': (Socket(id='examiner.data_in',
socket_type='ROUTER', protocol='ipc', mode='bind', port=5003, ip='*', to=None),
Socket(id='examiner.data_out', socket_type='DEALER', protocol='tcp', mode='connect',
port=5001, ip='192.168.0.100', to=None)), 'manager':
(Socket(id='examiner.manager.command', socket_type='SUB', protocol='tcp', mode='connect',
port=5000, ip='192.168.0.100', to=None), Socket(id='examiner.manager.process',
socket_type='ROUTER', protocol='ipc', mode='bind', port=5002, ip='*', to=None),
Socket(id='examiner.manager.data_out', socket_type='DEALER', protocol='ipc',
mode='connect', port=5003, ip='localhost', to=None)), 'picamera':
(Socket(id='examiner.picamera.data_out', socket_type='DEALER', protocol='ipc',
mode='connect', port=5002, ip='*', to=None),)}
```

Sockets used by the 'examiner' machine responsible for managing the exam - measuring the pupil, presenting sounds, and maintaining the psychoacoustic model

Processes:

manager

- examiner.manager.command - SUB - subscriber to clinician commands
- examiner.manager.process - ROUTER - receives data from picamera
- examiner.manager.data_out - DEALER - sends data to the data process to forward to clinician

picamera

- examiner.picamera.data_out - DEALER - sends frames to the process socket

data

- examiner.data.data_in - ROUTER - receives data from process
- examiner.data.data_out - DEALER - sends data to clinician

```
perceptivo.networking.sockets.STIM = {'stim': (Socket(id='stim.command',
socket_type='SUB', protocol='tcp', mode='connect', port=5000, ip='192.168.0.100',
to=None), Socket(id='stim.manager', socket_type='DEALER', protocol='tcp', mode='connect',
port=5002, ip='192.168.0.101', to=None))}
```

Sockets used by the stimulus delivery machine

- `stim.command` - SUB - subscriber to clinician commands
- `stim.manager` - DEALER - subscriber to the process socket of the examiner, receives commands to present stimuli, etc.

PSYCHOPHYS

10.1 gaussian

Refactoring of `sklearn.gaussian_process.GaussianProcessClassifier` to allow for iterative training

```
class perceptivo.psychophys.gaussian._IterativeBinaryGPLaplace(kernel=None, *,  
                                                             optimizer='fmin_l_bfgs_b',  
                                                             n_restarts_optimizer=0,  
                                                             max_iter_predict=100,  
                                                             warm_start=False,  
                                                             copy_X_train=True,  
                                                             random_state=None)
```

Bases: `sklearn.gaussian_process._gpc._BinaryGaussianProcessClassifierLaplace`

Reclassing to allow for fitting without needing a sample with ≥ 2 categories

fit(*X, y*)

Fit Gaussian process classification model.

Parameters

- **X** – array-like of shape (n_samples, n_features) or list of object Feature vectors or other representations of training data.
- **y** – array-like of shape (n_samples,) Target values, must be binary.

Returns self

Return type returns an instance of self.

```
class perceptivo.psychophys.gaussian.IterativeGPC(kernel=None, *, optimizer='fmin_l_bfgs_b',  
                                                  n_restarts_optimizer=0, max_iter_predict=100,  
                                                  warm_start=False, copy_X_train=True,  
                                                  random_state=None, multi_class='one_vs_rest',  
                                                  n_jobs=None)
```

Bases: `sklearn.gaussian_process._gpc.GaussianProcessClassifier`

Reclassified to use patched `_IterativeBinaryGPLaplace` instead of original model

fit(*X, y*)

Fit Gaussian process classification model.

Parameters

- **X** – array-like of shape (n_samples, n_features) or list of object Feature vectors or other representations of training data.
- **y** – array-like of shape (n_samples,) Target values, must be binary.

Returns self

`clone_kernel()` → `sklearn.gaussian_process.kernels.Kernel`

10.2 models

`perceptivo.psychophys.model.f_to_bark(frequency: float) → float`
Convert frequency to Bark using [WSG91]

Parameters `frequency` (*float*) – Frequency to convert

Returns (*float*) Bark

`perceptivo.psychophys.model.bark_to_f(bark: float) → float`
Convert bark to frequency using inverted [WSG91]

Parameters `bark` (*float*) – bark to convert

Returns (*float*) frequency

`class perceptivo.psychophys.model.Audiogram_Model(freq_range: Tuple[float, float] = (125, 8500),
amplitude_range: Tuple[float, float] = (5, 60),
exam_params:
Optional[perceptivo.types.exam.Exam_Params] =
None, *args, **kwargs)`

Bases: `perceptivo.root.Perceptivo_Object`

Metaclass for Audiogram models and estimators.

These classes are used to estimate the audiogram, as well as control the order of the presentation of probe sounds.

Note: This class may be split into an experimental runner class and an audiogram model, but since the choice of the next stimulus should ideally be based on the current audiogram model, they are built together for now.

Parameters

- **freq_range** (*tuple*) – Tuple of two floats indicating min/max frequency (default: (125, 8500))
- **amplitude_range** (*tuple*) – Tuple of two floats indicating min/max amplitude in dbSPL (default: (5,60))

Variables

- **~Audiogram_Model.audiogram** (`types.psychophys.Audiogram`) – Audiogram of model
- **~Audiogram_Model.samples** (`types.psychophys.Samples`) – Individual samples of frequency/amplitude and whether a sound was detected.

abstract update(*sample*: `perceptivo.types.psychophys.Sample`)
Update the model with a new :class:`~.types.psychophys.Sample`

abstract next() → `perceptivo.types.sound.Sound`
Generate parameters for the next *Sound* to be presented

Next should generate samples that respect the frequencies and amplitudes set in *exam_params*, if present.
As well as *allow_repeats*

```
class perceptivo.psychophys.model.Gaussian_Process(kernel: Op-
                                                    tional[Union[sklearn.gaussian_process.kernels.Kernel,
                                                    perceptivo.types.psychophys.Kernel]] = None,
                                                    *args, **kwargs)
```

Bases: *perceptivo.psychophys.model.Audiogram_Model*

Gaussian process model based on [CdeVries16]

Model: * Bayesian Process Classifier, predicting binary audibility as a function of frequency and amplitude *

Kernel: * Covariance Function: Squared Exponent (RBF)

Process: * Convert sampled frequency to bark with *f_to_bark()* * Update model * Generate next stimulus *
Convert back to freq

Examples

```
from perceptivo.psychophys.oracle import reference_audiogram
from perceptivo.psychophys.model import Gaussian_Process
from perceptivo.types.psychophys import Sample

oracle = reference_audiogram(scale=3)
model = Gaussian_Process(amplitude_range=(5, 35))

for i in range(100):
    sound = model.next()
    sample = Sample(response=oracle(sound), sound=sound)
    model.update(sample)

model.plot()
```

References

- [CdeVries16]
- [GMG+15]
- [MSG16]
- []

property kernel: *sklearn.gaussian_process.kernels.Kernel*

Kernel used in the gaussian process model. If None is given on init, use the *types.psychophys.Kernel*

Returns *sklearn.gaussian_process.kernels.Kernel*

property samples: *perceptivo.types.psychophys.Samples*

Stored samples from updates

Returns *Samples*

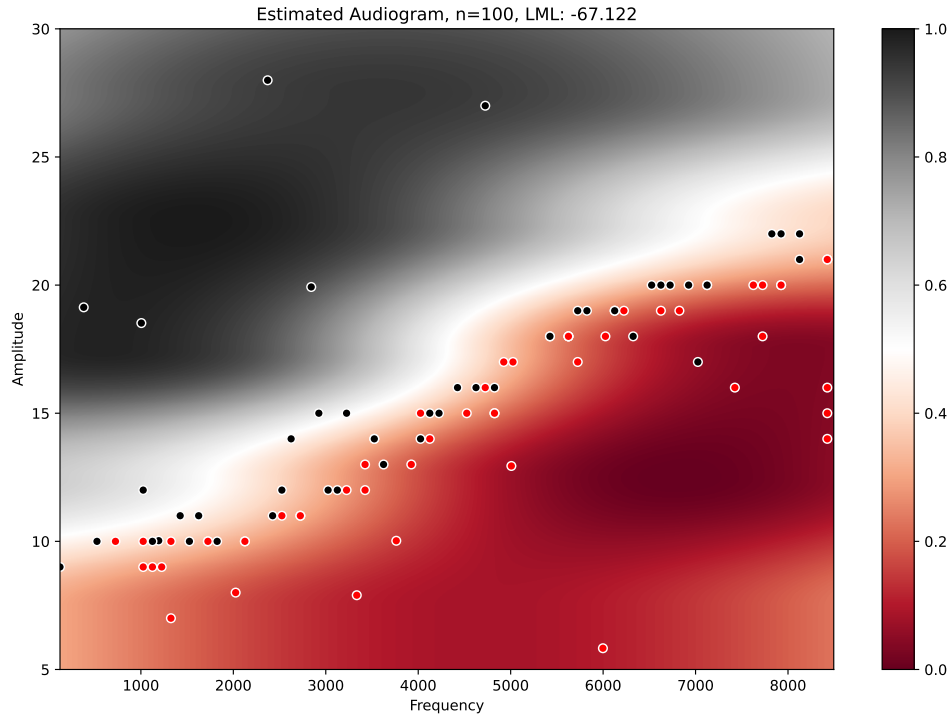
update(*sample*: *perceptivo.types.psychophys.Sample*)

Update the model with a new sample!

Parameters *sample* ()

_get_params() → *Tuple[float, float]*

Generate sound params



Returns a tuple of freq, amp

next() → *perceptivo.types.sound.Sound*
Generate parameters for the next sound to present

Returns *Sound*

plot(*mesh_resolution: int = 5*)

References

https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpc_iris.html#sphx-glr-auto-examples-gaussian-process-plot-gpc-iris-py

Returns:

10.3 oracle

Oracle functions for testing model

perceptivo.psychophys.oracle.piecewise_probabilistic(*points: numpy.ndarray, scale: float = 5*) → callable

Make a piecewise function along a series of (frequency, amplitude) points with some gaussian error

Parameters

- **points** (*np.ndarray*) – n x 2 array of x/y (frequency, amplitude) coordinates that make up an audiogram
- **scale** (*float*) – Scale parameter of noise in amplitude domain to get answers “wrong”

Returns:

`perceptivo.psychophys.oracle.reference_audiogram(scale: float = 2)` → callable

Generate fake audiometry samples using median threshold values obtained from the NHANES dataset: https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/AUX_I.htm

The median rates make a piecewise linear function:

Frequency	Threshold
500	10
1000	10
2000	10
3000	10
4000	15
6000	20
8000	20

Parameters *scale* (*float*) – amount of randomness to multiply the noise of the pseudo-response threshold by

Returns callable made by `piecewise_probabilistic()` that works as an oracle function

Parameters *scale* ()

Returns A numpy piecewise function that returns Sample objects for a given input frequency and amplitude

`perceptivo.psychophys.oracle.generate_samples(n_samples: int, scale: float = 2, freqs=None, amplitudes=None, randomize=False, freq_range=(500, 8000), amplitude_range=(0, 50), oracle: Optional[callable] = None)` → `perceptivo.types.psychophys.Samples`

Generate fake audiometry samples using median threshold values obtained from the NHANES dataset: https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/AUX_I.htm

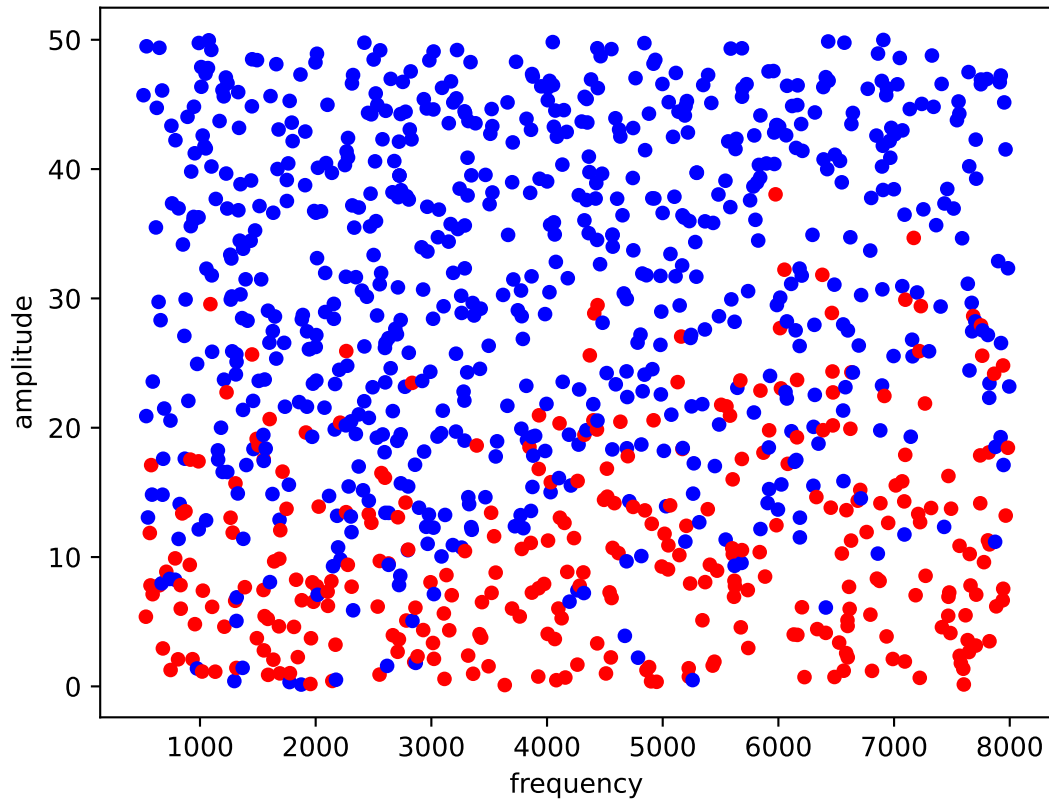
The median rates make a piecewise linear function:

Frequency	Threshold
500	10
1000	10
2000	10
3000	10
4000	15
6000	20
8000	20

Examples

```
from perceptivo.psychophys.oracle import generate_samples

samples = generate_samples(n_samples=1000, scale=10)
samples.plot()
```



Parameters

- **n_samples** (*int*) – number of samples to generate
- **scale** (*float*) – amount of randomness to multiply the noise of the pseudo-response threshold by
- **freqs** (*arraylike*) – (Optional) - predetermined array of frequencies (of length n_samples) to test
- **amplitudes** (*arraylike*) – (Optional) - predetermined array of amplitudes (of length n_samples) to test
- **randomize** (*bool*) – Randomize order of samples before returning, (default False)

Returns *types.psychophys.Samples*

11.1 server

Wrapper around autopilot's `JackClient`

- boot and kill the jackd daemon
- references to jackclient module

`perceptivo.sound.server.boot_jackd(config: perceptivo.types.sound.Jackd_Config) → subprocess.Popen`

Boot the jackd server given the configuration given by `types.Jackd_Config`

Launches with `shell = True` and `preexec_fn=os.setsid` so that the process can be killed later.

Registers the jackd sound server to be killed at exit.

Thanks to <https://stackoverflow.com/a/4791612/13113166> for information about how to kill a process with `shell = True`

Returns `subprocess.Popen` - opened subprocess

`perceptivo.sound.server.kill_jackd(proc: Optional[subprocess.Popen] = None)`

11.2 sounds

Sound synthesis

CHAPTER
TWELVE

STIM

13.1 cameras

Picamera capture. Easy enough with Autopilot

```
class perceptivo.video.cameras.Picamera_Process(params: perceptivo.types.video.Picamera_Params =  
                                                Picamera_Params(sensor_mode=0,  
                                                resolution=(1280, 720), fps=30, format='grayscale',  
                                                output_file=None), networking:  
                                                Optional[perceptivo.types.networking.Socket] =  
                                                None, queue_size: int = 1024, **kwargs)
```

Bases: `multiprocessing.context.Process`, `perceptivo.root.Perceptivo_Object`

Separate process for the picamera

q

Queue for the parent runtime to grab frames from the picamera.

collecting

Event set when collecting a sample. when set, dump frames into `Picamera_Process.q`

run()

Method to be run in sub-process; can be overridden in sub-class

release()

Stop running and release picamera resources

13.2 pupil

Process and extract pupil

Sketch of default strategy:

- Track to find approximate position of eyes with `processors.Haar_Tracker`
- Mask image around both eyes, split processing in parallel L/R (if present)
- Use white of eyes to mask cornea and pupil
- Sigmoid filter images to separate cornea and pupil
- Blob detection to find center mass of pupil
- Compare blob vs. edge detection of pupil.
- Use Kalman filter on `perceptivo.types.units.Ellipse` properties to avoid jumps and all that

```
class perceptivo.video.pupil.PupilExtractor(preprocessor:
    Optional[perceptivo.video.pupil.Preprocessor] = None,
    filter: Optional[perceptivo.video.pupil.PupilFilter] = None,
    **kwargs)

Bases: perceptivo.root.Perceptivo_Object

Base class for pupil extraction strategies.

process(frame: perceptivo.types.video.Frame) → Optional[perceptivo.types.pupil.Pupil]
    Call preprocess() and then _process(), returning a Pupil estimate

    Parameters frame (types.video.Frame) – Frame to process

    Returns types.pupil.Pupil Pupil Estimate

abstract _process(frame: perceptivo.types.video.Frame) → perceptivo.types.pupil.Pupil
    Given a frame, extract a pupil estimate

    Parameters frame (types.video.Frame) – Frame to process!

    Returns Estimated Pupil!

    Return type types.pupil.Pupil

class perceptivo.video.pupil.PupilFilter(**kwargs)
    Bases: perceptivo.root.Perceptivo_Object

    Base class for filtering pupil tracks – for example by using a Kalman filter to filter erroneous pupil detections
    from a PupilExtractor .

    PupilFilters should be given to the PupilExtractor as its filter argument, and should be called last in the
    PupilExtractor.process() method.

    Each subclass should implement a _process method that takes and returns a Pupil object.

    process(pupil: perceptivo.types.pupil.Pupil) → perceptivo.types.pupil.Pupil

class perceptivo.video.pupil.Preprocessor(**kwargs)
    Bases: perceptivo.root.Perceptivo_Object

    Base class for preprocessing images before they reach the main PupilExtractor.process() method.

    Each subclass should implement a process method that takes and returns a Frame object.

    abstract process(frame: perceptivo.types.video.Frame) → perceptivo.types.video.Frame

class perceptivo.video.pupil.EllipseExtractor_Params(*, footprint_size: int = 5, search_scale: float =
    1.5)

    Bases: pydantic.main.BaseModel

    Create a new model by parsing and validating input data from keyword arguments.

    Raises ValidationError if the input data cannot be parsed to form a valid model.

    footprint_size: int

    search_scale: float

class perceptivo.video.pupil.EllipseExtractor(footprint_size: int = 5, search_scale: float = 1.5,
    **kwargs)

    Bases: perceptivo.video.pupil.PupilExtractor

    Very simple extractor that estimates an ellipse from the edges of a pupil. This extractor assumes that the Frame
    given to it has very high contrast, ie. that the rest of the face is essentially white and the pupil and cornea are the
    only colored things in the image.
```

In order

- Median Filter to smooth image - `skimage.filters.rank.median()`
- Scharr Filter to detect edges - `skimage.filters.scharr()`
- Get the otsu threshold on the scharr filtered image - `skimage.filters.threshold_otsu()`
- Skeletonize the pixels above the threshold - `skimage.morphology.skeletonize()`
- Label the independent edges - `skimage.measure.label()`
- If present, use the `types.units.Ellipse` from `PupilExtractor.filter.last_pupil` to select only those edges within the ellipse (scaled by `search_scale`)
- Estimate ellipses from remaining edges - `skimage.measure.EllipseModel`
- Keep the ellipses with the lowest median pixel value (presumably the pupil is dark)
- Return a `Pupil` object.

References

- Read a few years ago, might be worth revisiting: https://cdn.intechopen.com/pdfs/33559/InTech-Methods_for_ellipse_detection_from_edge_maps_of_real_images.pdf

Parameters

- **footprint_size** (*int*) – Diameter of footprint (a `skimage.morphology.disk()`) used in the median filter `skimage.filters.rank.median()`. This should be roughly the size of the pupil.
- **search_scale** (*float*) – If present, how much to scale the `PupilExtractor.filter.last_pupil` to select edges before fitting ellipses. Eg. 1.5 enlarges the last ellipse by 1.5 and rejects all edges outside of that radius.
- ****kwargs** ()

`__init__(footprint_size: int = 5, search_scale: float = 1.5, **kwargs)`

Parameters

- **footprint_size** (*int*) – Diameter of footprint (a `skimage.morphology.disk()`) used in the median filter `skimage.filters.rank.median()`. This should be roughly the size of the pupil.
- **search_scale** (*float*) – If present, how much to scale the `PupilExtractor.filter.last_pupil` to select edges before fitting ellipses. Eg. 1.5 enlarges the last ellipse by 1.5 and rejects all edges outside of that radius.
- ****kwargs** ()

property footprint_size: `int`

As described in the attr docstring. When setting a footprint size, remake the footprint

Returns `int`

filter_edges(*edges: numpy.ndarray*) → `numpy.ndarray`

Set all edges outside of a search radius, given our previous ellipse, to zero

choose_ellipse(*edges: numpy.ndarray, frame: numpy.ndarray*) →
Optional[skimage.measure.fit.EllipseModel]

Given an array of edge labels (from `skimage.morphology.label()`), usually from `_process()`, return an `skimage.measure.EllipseModel`, choose the one that is the pupil

Parameters

- **edges** () – An array of image labels, ie. an array of ints where background == 0, edge 1 == 1, and so on.
- **frame** () – The original or filtered image frame (the array, not the `Frame` object)

Returns the most pupil-like ellipse

Return type `skimage.measure.EllipseModel`

class `perceptivo.video.pupil.EnsembleExtractor_NonIR`(*sigmoid=(0.5, 5), canny_kwargs: Optional[dict] = None, hough_kwargs: Optional[dict] = None, *args, **kwargs*)

Bases: `perceptivo.video.pupil.PupilExtractor`

Extractor that uses an ensemble of techniques to track a pupil.

Written before realizing the tracking problem was much easier using IR! Kept to mine parts from before discontinuing

- Track to find approximate position of eyes with `processors.Haar_Tracker`
- Mask image around both eyes, split processing in parallel L/R (if present)
- Use white of eyes to mask cornea and pupil
- Sigmoid filter images to separate cornea and pupil
- Blob detection to find center mass of pupil
- Compare blob vs. edge detection of pupil.
- Use Kalman filter on `perceptivo.types.units.Ellipse` properties to avoid jumps and all that

preprocess(*frame: perceptivo.types.video.Frame*) → `perceptivo.types.video.Frame`

_bbox_from_circle(*circle: List[int]*)
convert opencv's circles to a bounding box (top, bottom, left, right)

class `perceptivo.video.pupil.Pupil_Extractors`(*value*)
Bases: `enum.Enum`

An enumeration.

simple = <class 'perceptivo.video.pupil.EllipseExtractor'>

`perceptivo.video.pupil.get_extractor`(*extractor=<enum 'Pupil_Extractors'>*) →
`Type[perceptivo.video.pupil.EllipseExtractor]`

Parameters **extractor** (str, `Pupil_Extractors`) – str corresponding to one of the entries in `Pupil_Extractors`, eg 'simple'

Returns:

13.3 processors

Individual transformation operations for video frames.

To be used with the `perceptivo.video.pupil.PupilExtractor` subclasses

class `perceptivo.video.processors.Processor(**kwargs)`

Bases: `perceptivo.root.Perceptivo_Object`

Individual processing stage, can be added together to make a processing chain

`__add__` method based on `autopilot.transform.transforms.Transform`

property parent: `Optional[perceptivo.video.processors.Processor]`

If this Transform is in a chain of transforms, the transform that precedes it

Returns Transform, None if no parent.

abstract process(*input*: `Union[perceptivo.types.video.Frame, perceptivo.types.units.Ellipse]`) → `Union[perceptivo.types.video.Frame, perceptivo.types.units.Ellipse]`

Process a frame!

Typically you want a chain of processors to end up outputting an Ellipse, but this is not enforced

Returns:

`__add__`(*other*)

Add another Transformation in the chain to make a processing pipeline :Parameters: **other** (Transformation) – The transformation to be chained

class `perceptivo.video.processors.Canny(blur_sigma: float = 1, low_thresh: float = 0.2, high_thresh: float = 0.5)`

Bases: `perceptivo.video.processors.Processor`

Canny edge detection.

Slight modification of `skimage.feature.canny()`, but using `opencv` and Scharr kernel rather than `sobel` for better orientation invariance, and also using the eigenvectors of the structure tensor rather than the simple hypotenuse

The source for this class is really blippy because it is optimized for speed!

Variables

- `~Canny.blur_sigma` (*float*) – Amount of blurring to use in the initial smoothing step
- `~Canny.low_thresh` (*float*) – Low threshold for canny edge detection
- `~Canny.high_thresh` (*float*) – High threshold for canny edge detection

References

- [WS02]
- [BY15]
- [ZFD+16]

process(*frame*: `perceptivo.types.video.Frame`) → `perceptivo.types.video.Frame`

Process a frame!

Typically you want a chain of processors to end up outputting an Ellipse, but this is not enforced

Returns:

```
class perceptivo.video.processors.Haar_Tracker(tracker_type: str = 'eye', min_neighbors: int = 20,
                                              adaptive_neighbors: bool = True, **kwargs)
```

Bases: `perceptivo.video.processors.Processor`

Download and use a haar cascade to track.

Many trained cascade .xml files are available at <https://github.com/opencv/opencv/tree/master/data/haarcascades>

References

- [OpenCV Cascade Classifier Tutorial](#)

```
XML_URLS = {'eye': 'https://raw.githubusercontent.com/opencv/opencv/415a42f327104653604fc99314eb215cd938d6d7/data/haarcascades/haarcascade_eye.xml',
            'face_default': 'https://raw.githubusercontent.com/opencv/opencv/415a42f327104653604fc99314eb215cd938d6d7/data/haarcascades/haarcascade_frontalface_default.xml'}
```

property url: `str`

property filename: `pathlib.Path`

process(frame: `perceptivo.types.video.Frame`) → `Tuple[List[tuple], List[int]]`

Parameters frame (*Frame*)

Returns list of tuples corresponding to (x,y,width,height)

```
class perceptivo.video.processors.Filtered_Hough(radii=(15, 30, 100), max_considered=3,
                                              peaks_kwargs: Optional[dict] = None)
```

Bases: `perceptivo.video.processors.Processor`

A hough transform to detect circles, returning the one that bounds the darkest area in the image

process(edges: `numpy.ndarray`)

Frame to process, along with edges from canny edge detection

```
class perceptivo.video.processors.Filter_Circles(prior_bias=0.5)
```

Bases: `perceptivo.video.processors.Processor`

Filter Circles!

Parameters prior_bias (*float*) – how strongly to weight the similarity to the prior circle, if given

`__init__`(prior_bias=0.5)

Parameters prior_bias (*float*) – how strongly to weight the similarity to the prior circle, if given

process(frame: `perceptivo.types.video.Frame`, circles, prev_eye=None)

Process a frame!

Typically you want a chain of processors to end up outputting an Ellipse, but this is not enforced

Returns:

`perceptivo.video.processors.circle_to_mask`(frame, ix, iy, rad)

UTIL

Utility functions! everyone's favorite!

`perceptivo.util.download(url: str, file_name: Union[pathlib.Path, str]) → bool`

Download a file with a progress bar

Returns True if nothing happened and its probs good, False otherwise

Return type bool

References

<https://gist.github.com/yanqd0/c13ed29e29432e3cf3e7c38467f42f51>

`perceptivo.util.pack_array(array) → dict`

`perceptivo.util.unpack_array(shape: tuple, dtype: numpy.dtype, array: bytes) → numpy.ndarray`

`perceptivo.util.serialize(array: Union[numpy.ndarray, Any]) → Union[dict, Any]`

Serialization for use with `msgpack.packb` as default

Returns

dict like:

```
{
  '__numpy__': True,
  'shape': array.shape,
  'dtype': str(array.dtype),
  'array': array.data
}
```

`perceptivo.util.deserialize(obj)`

`perceptivo.util.msgpack_loads(msg)`

Wrapper of the `msgpack`

`perceptivo.util.msgpack_dumps(msg, *, default=None)`

PREFS

Preferences and configuration shared throughout the program.

Saves and loads to a prefs file (default is `~/.perceptivo/prefs.json`)

Each runtime has its own set of preferences. When first run, if there is not prefs file detected it populates with defaults (though defaults can be populated at any time by instantiating the object with no arguments and using save, eg.:

```
prefs = Patient_Prefs()
prefs.save()
```

```
class perceptivo.prefs.Runtimes(value)
```

Bases: `enum.Enum`

An enumeration.

```
patient = 'patient'
```

```
clinician = 'clinician'
```

```
stimuli = 'stimuli'
```

```
perceptivo.prefs.json_dumps_pretty(v, *, default)
```

```
class perceptivo.prefs.Prefs(*, loglevel: Literal['DEBUG', 'INFO', 'WARNING', 'ERROR'] = 'DEBUG')
```

Bases: `pydantic.main.BaseModel`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
loglevel: Literal['DEBUG', 'INFO', 'WARNING', 'ERROR']
```

```
class Config
```

Bases: `object`

```
json_dumps(*, default)
```

```
save(file: pathlib.Path = PosixPath('/home/docs/.perceptivo/prefs.json'))
```

```
classmethod load(file: pathlib.Path = PosixPath('/home/docs/.perceptivo/prefs.json')) →  
perceptivo.prefs.Prefs
```

```
classmethod get_runtime_prefs(runtime: perceptivo.prefs.Runtimes) → perceptivo.prefs.Patient_Prefs
```

```
class perceptivo.prefs.Patient_Prefs(*, loglevel: Literal['DEBUG', 'INFO', 'WARNING', 'ERROR'] =
    'DEBUG', runtime: perceptivo.prefs.Runtimes = 'patient',
    Audio_Config: perceptivo.types.sound.Audio_Config =
    Audio_Config(fs=44100), Audiogram_Model:
    perceptivo.types.psychophys.Psychoacoustic_Model =
    Psychoacoustic_Model(model_type='Gaussian_Process', args=[],
    kwargs={'kernel': Kernel(length_scale=(100.0, 200.0),
    length_scale_bounds=(1, 100000.0))}), Picamera_Params:
    perceptivo.types.video.Picamera_Params =
    Picamera_Params(sensor_mode=0, resolution=(1280, 720),
    fps=30, format='grayscale', output_file=None), picamera_process:
    bool = True, picam_queue_size: int = 1024, pupil_extractor: str =
    'simple', pupil_extractor_params:
    perceptivo.video.pupil.EllipseExtractor_Params =
    EllipseExtractor_Params(footprint_size=5, search_scale=1.5),
    collection_params: perceptivo.types.patient.Collection_Params =
    Collection_Params(collection_wait=5), networking:
    perceptivo.types.networking.Patient_Networking =
    Patient_Networking(ip="", clinician_ip="",
    eyecam=Socket(id='patient:eyecam', socket_type='PUSH',
    protocol='tcp', mode='connect', port=5500, ip="", to=None),
    control=Socket(id='patient:control', socket_type='DEALER',
    protocol='tcp', mode='connect', port=5600, ip="",
    to='clinician:control'))
```

Bases: `perceptivo.prefs.Prefs`

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

runtime: `perceptivo.prefs.Runtimes`

Audio_Config: `perceptivo.types.sound.Audio_Config`

Audiogram_Model: `perceptivo.types.psychophys.Psychoacoustic_Model`

Picamera_Params: `perceptivo.types.video.Picamera_Params`

picamera_process: `bool`

Run the picamera in a separate Process (using `cameras.Picamera_Process` . Only True supported for now!

picam_queue_size: `int`

pupil_extractor: `str`

pupil_extractor_params: `perceptivo.video.pupil.EllipseExtractor_Params`

collection_params: `perceptivo.types.patient.Collection_Params`

networking: `perceptivo.types.networking.Patient_Networking`

class Config

Bases: `object`

`use_enum_values = True`

```

class perceptivo.prefs.Clinician_Prefs(*, loglevel: Literal['DEBUG', 'INFO', 'WARNING', 'ERROR'] =
    'DEBUG', networking:
    perceptivo.types.networking.Clinician_Networking =
    Clinician_Networking(ip="", patient_ip="",
    eyecam=Socket(id='clinician:eyecam', socket_type='PULL',
    protocol='tcp', mode='bind', port=5500, ip='*', to=None),
    control=Socket(id='clinician:control', socket_type='ROUTER',
    protocol='tcp', mode='bind', port=5600, ip='*', to=None)), gui:
    perceptivo.types.gui.GUI_Params =
    GUI_Params(control_panel=Control_Panel_Params(amplitude_range=GUI_Param
    name='Amplitude Range (dBSPL)', widget_type='range',
    default=GUI_Range(min=0.0, max=80.0, n=8), args=[],
    kwargs={'limits': (0, 100)}),
    frequency_range=GUI_Param(key='frequency_range',
    name='Frequency Range (Hz)', widget_type='range',
    default=GUI_Range(min=0.0, max=8000.0, n=17), args=[],
    kwargs={'limits': (0, 20000)}), iti=GUI_Param(key='iti',
    name='Inter-Trial Interval (s)', widget_type='float', default=5.0,
    args=[], kwargs={}), iti_jitter=GUI_Param(key='iti_jitter',
    name='Inter-Trial Jitter (proportion of ITI)', widget_type='float',
    default=0.1, args=[], kwargs={})), update_period: float = 0.05)

```

Bases: `perceptivo.prefs.Prefs`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

networking: `perceptivo.types.networking.Clinician_Networking`

gui: `perceptivo.types.gui.GUI_Params`

update_period: `float`

`perceptivo.prefs.get(field: str, file: pathlib.Path = PosixPath('/home/docs/.perceptivo/prefs.json'))`

`perceptivo.prefs.set_global(prefs: perceptivo.prefs.Prefs)`

`perceptivo.prefs.get_global()` → `perceptivo.prefs.Prefs`

ROOT

Root Perceptivo Object from which others inherit

```
class perceptivo.root.Perceptivo_Object
    Bases: abc.ABC
    property logger: logging.Logger
```


TODO

17.1 Sound

- Calibrate sound output to relate amplitudes 0-1 to dbSPL
- Translation back and forth between Bark and Hz

17.2 GUI

- Connect rest of exam params to gui elements

17.3 Code Cleanup

- So much.
- Relate *perceptivo.types.gui.GUI_Params* to *perceptivo.types.exam.Exam_Params* so that GUI elements can be derived from control parameters

17.4 Documentation

-

CHAPTER
EIGHTEEN

REFERENCES

BIBLIOGRAPHY

- [BY15] Ahmadreza Baghaie and Zeyun Yu. Structure Tensor Based Image Interpolation Method. *AEU - International Journal of Electronics and Communications*, 69(2):515–522, February 2015. [arXiv:1402.5564](#), [doi:10.1016/j.aeue.2014.10.022](#).
- [CdeVries16] Marco Cox and Bert de Vries. A Bayesian binary classification approach to pure tone audiometry. *arXiv:1511.08670 [stat]*, March 2016. [arXiv:1511.08670](#).
- [GMG+15] Jacob Gardner, Gustavo Malkomes, Roman Garnett, Kilian Q Weinberger, Dennis Barbour, and John P Cunningham. Bayesian Active Model Selection with an Application to Automated Audiometry. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [MSG16] Gustavo Malkomes, Charles Schaff, and Roman Garnett. Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [WSG91] S. Wang, A. Sekey, and A. Gersho. Auditory distortion measure for speech coding. In *Acoustics, Speech, and Signal Processing, IEEE International Conference On*, 493–496. IEEE Computer Society, April 1991. [doi:10.1109/ICASSP.1991.150384](#).
- [WS02] Joachim Weickert and Hanno Scharr. A Scheme for Coherence-Enhancing Diffusion Filtering with Optimized Rotation Invariance. *Journal of Visual Communication and Image Representation*, 13(1):103–118, March 2002. [doi:10.1006/jvci.2001.0495](#).
- [ZFD+16] Wenxing Zhang, Jérôme Fehrenbach, Annaïck Desmaison, Valérie Lobjois, Bernard Ducommun, and Pierre Weiss. Structure Tensor Based Analysis of Cells and Nuclei Organization in Tissues. *IEEE transactions on medical imaging*, 35(1):294–306, January 2016. [doi:10.1109/TMI.2015.2470093](#).

PYTHON MODULE INDEX

p

- `perceptivo.data`, 23
- `perceptivo.data.logging`, 23
- `perceptivo.data.patient`, 24
- `perceptivo.gui`, 39
- `perceptivo.gui.main`, 39
- `perceptivo.gui.styles`, 40
- `perceptivo.gui.widgets`, 40
- `perceptivo.gui.widgets.audiogram`, 40
- `perceptivo.gui.widgets.components`, 40
- `perceptivo.gui.widgets.control_panel`, 41
- `perceptivo.gui.widgets.patient`, 41
- `perceptivo.gui.widgets.pupil`, 42
- `perceptivo.gui.widgets.video`, 42
- `perceptivo.networking`, 43
- `perceptivo.networking.messages`, 43
- `perceptivo.networking.node`, 44
- `perceptivo.networking.sockets`, 45
- `perceptivo.prefs`, 65
- `perceptivo.psychophys`, 47
- `perceptivo.psychophys.gaussian`, 47
- `perceptivo.psychophys.model`, 48
- `perceptivo.psychophys.oracle`, 50
- `perceptivo.root`, 69
- `perceptivo.runtimes.clinician`, 19
- `perceptivo.runtimes.patient`, 20
- `perceptivo.runtimes.runtime`, 19
- `perceptivo.sound`, 53
- `perceptivo.sound.server`, 53
- `perceptivo.sound.sounds`, 53
- `perceptivo.stim`, 55
- `perceptivo.types`, 25
- `perceptivo.types.exam`, 25
- `perceptivo.types.gui`, 26
- `perceptivo.types.networking`, 29
- `perceptivo.types.patient`, 30
- `perceptivo.types.psychophys`, 31
- `perceptivo.types.pupil`, 34
- `perceptivo.types.sound`, 35
- `perceptivo.types.units`, 37
- `perceptivo.types.video`, 37
- `perceptivo.util`, 63
- `perceptivo.video`, 57
- `perceptivo.video.cameras`, 57
- `perceptivo.video.processors`, 61
- `perceptivo.video.pupil`, 57

Symbols

- `_IterativeBinaryGPCLaplace` (class in `perceptivo.psychophys.gaussian`), 47
- `_LOGGERS` (in module `perceptivo.data.logging`), 23
- `__add__()` (`perceptivo.video.processors.Processor` method), 61
- `__init__()` (`perceptivo.gui.widgets.components.Range_Slider` method), 41
- `__init__()` (`perceptivo.networking.messages.Message` method), 43
- `__init__()` (`perceptivo.networking.node.Node` method), 44
- `__init__()` (`perceptivo.video.processors.Filter_Circles` method), 62
- `__init__()` (`perceptivo.video.pupil.EllipseExtractor` method), 59
- `_bbox_from_circle()` (`perceptivo.video.pupil.EnsembleExtractor_NonIR` method), 60
- `_collect_frames()` (`perceptivo.runtimes.patient.Patient` method), 21
- `_get_params()` (`perceptivo.psychophys.model.Gaussian_Process` method), 49
- `_init_audio()` (`perceptivo.runtimes.patient.Patient` method), 22
- `_process()` (`perceptivo.video.pupil.PupilExtractor` method), 58
- `_start_ioloop()` (`perceptivo.networking.node.Node` method), 45
- `_start_polling()` (`perceptivo.networking.node.Node` method), 45
- `_update_pupil_params()` (`perceptivo.runtimes.patient.Patient` method), 22
- A**
 - `a` (`perceptivo.types.units.Ellipse` attribute), 37
 - `address` (`perceptivo.networking.node.Node` property), 44
 - `allow_repeats` (`perceptivo.types.exam.Exam_Params` attribute), 26
 - `amplitude` (`perceptivo.types.sound.Sound` attribute), 36
 - `amplitudes` (`perceptivo.types.exam.Exam_Params` attribute), 26
 - `amplitudes` (`perceptivo.types.psychophys.Samples` attribute), 31
 - `append()` (`perceptivo.types.psychophys.Samples` method), 31
 - `arbitrary_types_allowed` (`perceptivo.types.psychophys.Kernel.Config` attribute), 33
 - `arbitrary_types_allowed` (`perceptivo.types.sound.Sound.Config` attribute), 36
 - `arbitrary_types_allowed` (`perceptivo.types.video.Frame.Config` attribute), 38
 - `args` (`perceptivo.types.gui.GUI_Param` attribute), 28
 - `args` (`perceptivo.types.psychophys.Psychoacoustic_Model` attribute), 34
 - `Audio_Config` (class in `perceptivo.types.sound`), 35
 - `Audio_Config` (`perceptivo.prefs.Patient_Prefs` attribute), 66
 - `Audiogram` (class in `perceptivo.gui.widgets.audiogram`), 40
 - `Audiogram` (class in `perceptivo.types.psychophys`), 33
 - `audiogram` (`perceptivo.types.patient.Patient` attribute), 30
 - `Audiogram_Model` (class in `perceptivo.psychophys.model`), 48
 - `Audiogram_Model` (`perceptivo.prefs.Patient_Prefs` attribute), 66
 - `await_response()` (`perceptivo.runtimes.patient.Patient` method), 21
- B**
 - `b` (`perceptivo.types.units.Ellipse` attribute), 37
 - `bark_to_f()` (in module `perceptivo.psychophys.model`), 48
 - `base_args()` (in module `perceptivo.runtimes.runtime`), 19
 - `bin` (`perceptivo.types.sound.Jackd_Config` attribute), 35
 - `Biography` (class in `perceptivo.types.patient`), 30
 - `biography` (`perceptivo.types.patient.Patient` attribute),

- 30
 boot_jackd() (in module *perceptivo.sound.server*), 53
 buttonClicked (perceptivo.gui.widgets.components.Range_Setter attribute), 40
- ## C
- Camera_Calibration (class in *perceptivo.types.video*), 38
 Canny (class in *perceptivo.video.processors*), 61
 cb_connect() (perceptivo.gui.main.Perceptivo_Clinician method), 39
 cb_control() (perceptivo.runtimes.patient.Patient method), 21
 cb_data() (perceptivo.gui.main.Perceptivo_Clinician method), 39
 cb_start() (perceptivo.runtimes.patient.Patient method), 22
 cb_stop() (perceptivo.runtimes.patient.Patient method), 22
 changeControl() (perceptivo.gui.main.Perceptivo_Clinician method), 39
 choose_ellipse() (perceptivo.video.pupil.EllipseExtractor method), 59
 circle_to_mask() (in module *perceptivo.video.processors*), 62
 Clinician (class in *perceptivo.runtimes.clinician*), 19
 CLINICIAN (in module *perceptivo.networking.sockets*), 45
 clinician (perceptivo.prefs.Runtimes attribute), 65
 clinician_ip (perceptivo.types.networking.Patient_Networking attribute), 30
 Clinician_Networking (class in *perceptivo.types.networking*), 29
 clinician_parser() (in module *perceptivo.runtimes.clinician*), 19
 Clinician_Prefs (class in *perceptivo.prefs*), 66
 clone_kernel() (perceptivo.psychophys.gaussian.IterativeGPC method), 48
 closeEvent() (perceptivo.gui.main.Perceptivo_Clinician method), 39
 collecting (perceptivo.video.cameras.Picamera_Process attribute), 57
 Collection_Params (class in *perceptivo.types.patient*), 30
 collection_params (perceptivo.prefs.Patient_Prefs attribute), 66
 collection_wait (perceptivo.types.patient.Collection_Params attribute), 30
 color (perceptivo.types.video.Frame attribute), 37
 Completion_Metric (class in *perceptivo.types.exam*), 25
 completion_metric (perceptivo.types.exam.Exam_Params attribute), 26
 confidence (perceptivo.types.psychophys.Threshold attribute), 32
 control (perceptivo.types.networking.Clinician_Networking attribute), 29
 control (perceptivo.types.networking.Patient_Networking attribute), 30
 Control_Panel (class in *perceptivo.gui.widgets.control_panel*), 41
 control_panel (perceptivo.types.gui.GUI_Params attribute), 29
 Control_Panel_Params (class in *perceptivo.types.gui*), 28
 controlChanged (perceptivo.gui.main.Perceptivo_Clinician attribute), 39
 counter (perceptivo.networking.messages.Message attribute), 43
 crop() (perceptivo.types.video.Frame method), 38
 cropped (perceptivo.types.video.Frame attribute), 37
- ## D
- default (perceptivo.types.gui.GUI_Param attribute), 28
 DEQUE (perceptivo.networking.node.Node.Poll_Mode attribute), 44
 deserialize() (in module *perceptivo.util*), 63
 device_name (perceptivo.types.sound.Jackd_Config attribute), 35
 diameters (perceptivo.types.pupil.Dilation property), 35
 Dilation (class in *perceptivo.types.pupil*), 34
 dilation (perceptivo.types.psychophys.Sample attribute), 31
 distance (perceptivo.types.video.Camera_Calibration attribute), 38
 dob (perceptivo.data.patient.Patient_Data attribute), 24
 dob (perceptivo.types.patient.Biography attribute), 30
 download() (in module *perceptivo.util*), 63
 drawFrame() (perceptivo.gui.main.Perceptivo_Clinician method), 39
 driver (perceptivo.types.sound.Jackd_Config attribute), 35
 dtype (perceptivo.types.video.Frame attribute), 37
 duration (perceptivo.types.exam.Completion_Metric attribute), 26
 duration (perceptivo.types.sound.Sound attribute), 36

E

Ellipse (class in *perceptivo.types.units*), 37
 ellipse (*perceptivo.types.pupil.Pupil* attribute), 34
 EllipseExtractor (class in *perceptivo.video.pupil*), 58
 EllipseExtractor_Params (class in *perceptivo.video.pupil*), 58
 EnsembleExtractor_NonIR (class in *perceptivo.video.pupil*), 60
 Exam_Params (class in *perceptivo.types.exam*), 26
 exam_params (*perceptivo.gui.main.Perceptivo_Clinician* property), 39
 EXAMINER (in module *perceptivo.networking.sockets*), 45
 eyecam (*perceptivo.types.networking.Clinician_Networking* attribute), 29
 eyecam (*perceptivo.types.networking.Patient_Networking* attribute), 30

F

f_to_bark() (in module *perceptivo.psychophys.model*), 48
 filename (*perceptivo.video.processors.Haar_Tracker* property), 62
 Filter_Circles (class in *perceptivo.video.processors*), 62
 filter_edges() (*perceptivo.video.pupil.EllipseExtractor* method), 59
 Filtered_Hough (class in *perceptivo.video.processors*), 62
 fit() (*perceptivo.psychophys.gaussian._IterativeBinaryGPGC_Laplace* method), 47
 fit() (*perceptivo.psychophys.gaussian.IterativeGPC_Laplace* method), 47
 footprint_size (*perceptivo.video.pupil.EllipseExtractor* property), 59
 footprint_size (*perceptivo.video.pupil.EllipseExtractor_Params* attribute), 58
 format (*perceptivo.types.video.Picamera_Params* attribute), 38
 fps (*perceptivo.types.video.Picamera_Params* attribute), 38
 Frame (class in *perceptivo.types.video*), 37
 frame (*perceptivo.gui.main.Perceptivo_Clinician.Frame_Receiver* attribute), 39
 frame (*perceptivo.types.pupil.Pupil* attribute), 34
 frame (*perceptivo.types.video.Frame* attribute), 37
 Frame.Config (class in *perceptivo.types.video*), 38
 frequencies (*perceptivo.types.exam.Exam_Params* attribute), 26
 frequencies (*perceptivo.types.psychophys.Audiogram* property), 33

frequencies (*perceptivo.types.psychophys.Samples* attribute), 31
 frequency (*perceptivo.types.psychophys.Threshold* attribute), 32
 frequency (*perceptivo.types.sound.Sound* attribute), 36
 from_serialized() (*perceptivo.networking.messages.Message* class method), 43
 fs (*perceptivo.types.sound.Audio_Config* attribute), 35

G

Gaussian_Process (class in *perceptivo.psychophys.model*), 48
 generate_samples() (in module *perceptivo.psychophys.oracle*), 51
 get() (in module *perceptivo.prefs*), 67
 get_extractor() (in module *perceptivo.video.pupil*), 60
 get_global() (in module *perceptivo.prefs*), 67
 get_runtime_prefs() (*perceptivo.prefs.Prefs* class method), 65
 gray (*perceptivo.types.video.Frame* property), 38
 gridChanged() (*perceptivo.gui.widgets.audiogram.Audiogram* method), 40
 gui (*perceptivo.prefs.Clinician_Prefs* attribute), 67
 GUI_Control (class in *perceptivo.types.gui*), 27
 GUI_Param (class in *perceptivo.types.gui*), 27
 GUI_PARAM_KEY (in module *perceptivo.types.gui*), 26
 GUI_Params (class in *perceptivo.types.gui*), 28
 GUI_Range (class in *perceptivo.types.gui*), 27
 GUI_WIDGET_TYPE (in module *perceptivo.types.gui*), 26

H

Haar_Tracker (class in *perceptivo.video.processors*), 61
 handle_message() (*perceptivo.runtimes.patient.Patient* method), 21

I

id (*perceptivo.types.networking.Socket* attribute), 29
 init_gui() (*perceptivo.runtimes.clinician.Clinician* method), 19
 init_logger() (in module *perceptivo.data.logging*), 23
 IOLOOP (*perceptivo.networking.node.Node.Poll_Mode* attribute), 44
 ip (*perceptivo.types.networking.Clinician_Networking* attribute), 29
 ip (*perceptivo.types.networking.Patient_Networking* attribute), 30
 ip (*perceptivo.types.networking.Socket* attribute), 29
 IterativeGPC (class in *perceptivo.psychophys.gaussian*), 47
 iti (*perceptivo.types.exam.Exam_Params* attribute), 26

`iti_jitter` (*perceptivo.types.exam.Exam_Params* attribute), 26

J

`jack_client` (*perceptivo.types.sound.Sound* attribute), 36

`Jackd_Config` (class in *perceptivo.types.sound*), 35

`json_dumps()` (*perceptivo.prefs.Prefs.Config* method), 65

`json_dumps_pretty()` (in module *perceptivo.prefs*), 65

K

`Kernel` (class in *perceptivo.types.psychophys*), 33

`kernel` (*perceptivo.psychophys.model.Gaussian_Process* property), 49

`kernel` (*perceptivo.types.psychophys.Kernel* property), 33

`Kernel.Config` (class in *perceptivo.types.psychophys*), 33

`key` (*perceptivo.types.gui.GUI_Control* attribute), 27

`key` (*perceptivo.types.gui.GUI_Param* attribute), 28

`kill_jackd()` (in module *perceptivo.sound.server*), 53

`kwargs` (*perceptivo.types.gui.GUI_Param* attribute), 28

`kwargs` (*perceptivo.types.psychophys.Psychoacoustic_Model* attribute), 34

L

`launch_str` (*perceptivo.types.sound.Jackd_Config* property), 36

`length_scale` (*perceptivo.types.psychophys.Kernel* attribute), 33

`length_scale_bounds` (*perceptivo.types.psychophys.Kernel* attribute), 33

`load()` (*perceptivo.prefs.Prefs* class method), 65

`load_prefs()` (*perceptivo.runtimes.runtime.Runtime* method), 19

`log_likelihood` (*perceptivo.types.exam.Completion_Metric* attribute), 25

`logger` (*perceptivo.root.Perceptivo_Object* property), 69

`loglevel` (*perceptivo.prefs.Prefs* attribute), 65

M

`main()` (in module *perceptivo.runtimes.clinician*), 19

`main()` (in module *perceptivo.runtimes.patient*), 22

`mask()` (*perceptivo.types.units.Ellipse* method), 37

`max` (*perceptivo.types.gui.GUI_Range* attribute), 27

`max_diameter` (*perceptivo.types.pupil.Dilation* property), 35

`max_diameter` (*perceptivo.types.pupil.Pupil_Params* attribute), 34

`Message` (class in *perceptivo.networking.messages*), 43

`min` (*perceptivo.types.gui.GUI_Range* attribute), 27

`mm_per_px` (*perceptivo.types.video.Camera_Calibration* attribute), 38

`mode` (*perceptivo.types.networking.Socket* attribute), 29

`model_type` (*perceptivo.types.psychophys.Psychoacoustic_Model* attribute), 33

module

`perceptivo.data`, 23

`perceptivo.data.logging`, 23

`perceptivo.data.patient`, 24

`perceptivo.gui`, 39

`perceptivo.gui.main`, 39

`perceptivo.gui.styles`, 40

`perceptivo.gui.widgets`, 40

`perceptivo.gui.widgets.audiogram`, 40

`perceptivo.gui.widgets.components`, 40

`perceptivo.gui.widgets.control_panel`, 41

`perceptivo.gui.widgets.patient`, 41

`perceptivo.gui.widgets.pupil`, 42

`perceptivo.gui.widgets.video`, 42

`perceptivo.networking`, 43

`perceptivo.networking.messages`, 43

`perceptivo.networking.node`, 44

`perceptivo.networking.sockets`, 45

`perceptivo.prefs`, 65

`perceptivo.psychophys`, 47

`perceptivo.psychophys.gaussian`, 47

`perceptivo.psychophys.model`, 48

`perceptivo.psychophys.oracle`, 50

`perceptivo.root`, 69

`perceptivo.runtimes.clinician`, 19

`perceptivo.runtimes.patient`, 20

`perceptivo.runtimes.runtime`, 19

`perceptivo.sound`, 53

`perceptivo.sound.server`, 53

`perceptivo.sound.sounds`, 53

`perceptivo.stim`, 55

`perceptivo.types`, 25

`perceptivo.types.exam`, 25

`perceptivo.types.gui`, 26

`perceptivo.types.networking`, 29

`perceptivo.types.patient`, 30

`perceptivo.types.psychophys`, 31

`perceptivo.types.pupil`, 34

`perceptivo.types.sound`, 35

`perceptivo.types.units`, 37

`perceptivo.types.video`, 37

`perceptivo.util`, 63

`perceptivo.video`, 57

`perceptivo.video.cameras`, 57

`perceptivo.video.processors`, 61

`perceptivo.video.pupil`, 57

`msgpack_dumps()` (in module *perceptivo.util*), 63

`msgpack_loads()` (in module *perceptivo.util*), 63

N

`n` (*perceptivo.types.gui.GUI_Range* attribute), 27

`n_trials` (*perceptivo.types.exam.Completion_Metric* attribute), 26

`name` (*perceptivo.data.patient.Patient_Data* attribute), 24

`name` (*perceptivo.types.gui.GUI_Param* attribute), 28

`name` (*perceptivo.types.patient.Biography* attribute), 30

`networking` (*perceptivo.prefs.Clinician_Prefs* attribute), 67

`networking` (*perceptivo.prefs.Patient_Prefs* attribute), 66

`next()` (*perceptivo.psychophys.model.Audiogram_Model* method), 48

`next()` (*perceptivo.psychophys.model.Gaussian_Process* method), 50

`next_sound()` (*perceptivo.runtimes.patient.Patient* method), 21

`Node` (class in *perceptivo.networking.node*), 44

`Node.Poll_Mode` (class in *perceptivo.networking.node*), 44

`NONE` (*perceptivo.networking.node.Node.Poll_Mode* attribute), 44

`norm()` (*perceptivo.types.video.Frame* method), 38

`nperiods` (*perceptivo.types.sound.Jackd_Config* attribute), 35

O

`outchannels` (*perceptivo.types.sound.Jackd_Config* attribute), 35

`output_file` (*perceptivo.types.video.Picamera_Params* attribute), 38

P

`pack_array()` (in module *perceptivo.util*), 63

`params` (*perceptivo.types.pupil.Dilation* attribute), 35

`parent` (*perceptivo.video.processors.Processor* property), 61

`Patient` (class in *perceptivo.runtimes.patient*), 20

`Patient` (class in *perceptivo.types.patient*), 30

`patient` (*perceptivo.prefs.Runtimes* attribute), 65

`Patient_Data` (class in *perceptivo.data.patient*), 24

`patient_ip` (*perceptivo.types.networking.Clinician_Networking* attribute), 29

`Patient_Networking` (class in *perceptivo.types.networking*), 29

`Patient_Prefs` (class in *perceptivo.prefs*), 65

`Patient_Prefs.Config` (class in *perceptivo.prefs*), 66

`perceptivo.data`
module, 23

`perceptivo.data.logging`
module, 23

`perceptivo.data.patient`
module, 24

`perceptivo.gui`
module, 39

`perceptivo.gui.main`
module, 39

`perceptivo.gui.styles`
module, 40

`perceptivo.gui.widgets`
module, 40

`perceptivo.gui.widgets.audiogram`
module, 40

`perceptivo.gui.widgets.components`
module, 40

`perceptivo.gui.widgets.control_panel`
module, 41

`perceptivo.gui.widgets.patient`
module, 41

`perceptivo.gui.widgets.pupil`
module, 42

`perceptivo.gui.widgets.video`
module, 42

`perceptivo.networking`
module, 43

`perceptivo.networking.messages`
module, 43

`perceptivo.networking.node`
module, 44

`perceptivo.networking.sockets`
module, 45

`perceptivo.prefs`
module, 65

`perceptivo.psychophys`
module, 47

`perceptivo.psychophys.gaussian`
module, 47

`perceptivo.psychophys.model`
module, 48

`perceptivo.psychophys.oracle`
module, 50

`perceptivo.root`
module, 69

`perceptivo.runtimes.clinician`
module, 19

`perceptivo.runtimes.patient`
module, 20

`perceptivo.runtimes.runtime`
module, 19

`perceptivo.sound`
module, 53

`perceptivo.sound.server`
module, 53

`perceptivo.sound.sounds`
module, 53

`perceptivo.stim`
module, 55

perceptivo.types
 module, 25
 perceptivo.types.exam
 module, 25
 perceptivo.types.gui
 module, 26
 perceptivo.types.networking
 module, 29
 perceptivo.types.patient
 module, 30
 perceptivo.types.psychophys
 module, 31
 perceptivo.types.pupil
 module, 34
 perceptivo.types.sound
 module, 35
 perceptivo.types.units
 module, 37
 perceptivo.types.video
 module, 37
 perceptivo.util
 module, 63
 perceptivo.video
 module, 57
 perceptivo.video.cameras
 module, 57
 perceptivo.video.processors
 module, 61
 perceptivo.video.pupil
 module, 57
 Perceptivo_Clinician (class in perceptivo.gui.main), 39
 Perceptivo_Clinician.Frame_Receiver (class in perceptivo.gui.main), 39
 Perceptivo_Object (class in perceptivo.root), 69
 period (perceptivo.types.sound.Jackd_Config attribute), 35
 picam (perceptivo.types.video.Camera_Calibration attribute), 38
 picam_queue_size (perceptivo.prefs.Patient_Prefs attribute), 66
 Picamera_Params (class in perceptivo.types.video), 38
 Picamera_Params (perceptivo.prefs.Patient_Prefs attribute), 66
 Picamera_Process (class in perceptivo.video.cameras), 57
 picamera_process (perceptivo.prefs.Patient_Prefs attribute), 66
 piecewise_probabilistic() (in module perceptivo.psychophys.oracle), 50
 play_sound() (perceptivo.runtimes.patient.Patient method), 21
 playback_only (perceptivo.types.sound.Jackd_Config attribute), 35
 plot() (perceptivo.psychophys.model.Gaussian_Process method), 50
 plot() (perceptivo.types.psychophys.Samples method), 31
 port (perceptivo.types.networking.Socket attribute), 29
 Prefs (class in perceptivo.prefs), 65
 Prefs.Config (class in perceptivo.prefs), 65
 prefs_class (perceptivo.runtimes.clinician.Clinician attribute), 19
 prefs_class (perceptivo.runtimes.patient.Patient attribute), 20
 prefs_class (perceptivo.runtimes.runtime.Runtime property), 19
 preprocess() (perceptivo.video.pupil.EnsembleExtractor_NonIR method), 60
 Preprocessor (class in perceptivo.video.pupil), 58
 priority (perceptivo.types.sound.Jackd_Config attribute), 35
 probe() (perceptivo.runtimes.patient.Patient method), 21
 process() (perceptivo.video.processors.Canny method), 61
 process() (perceptivo.video.processors.Filter_Circles method), 62
 process() (perceptivo.video.processors.Filtered_Hough method), 62
 process() (perceptivo.video.processors.Haar_Tracker method), 62
 process() (perceptivo.video.processors.Processor method), 61
 process() (perceptivo.video.pupil.Preprocessor method), 58
 process() (perceptivo.video.pupil.PupilExtractor method), 58
 process() (perceptivo.video.pupil.PupilFilter method), 58
 Processor (class in perceptivo.video.processors), 61
 procs (perceptivo.runtimes.runtime.Runtime property), 19
 protocol (perceptivo.types.networking.Socket attribute), 29
 Psychoacoustic_Model (class in perceptivo.types.psychophys), 33
 Pupil (class in perceptivo.gui.widgets.pupil), 42
 Pupil (class in perceptivo.types.pupil), 34
 pupil_extractor (perceptivo.prefs.Patient_Prefs attribute), 66
 pupil_extractor_params (perceptivo.prefs.Patient_Prefs attribute), 66
 Pupil_Extractors (class in perceptivo.video.pupil), 60
 Pupil_Params (class in perceptivo.types.pupil), 34
 PupilExtractor (class in perceptivo.video.pupil), 57
 PupilFilter (class in perceptivo.video.pupil), 58

pupils (*perceptivo.types.pupil.Dilation* attribute), 35

Q

q (*perceptivo.video.cameras.Picamera_Process* attribute), 57

quitting() (*perceptivo.gui.main.Perceptivo_Clinician.Frame_Receiver* method), 40

R

Range_Setter (class in *perceptivo.gui.widgets.components*), 40

receive_messages() (*perceptivo.gui.main.Perceptivo_Clinician* method), 39

reference_audiogram() (in module *perceptivo.psychophys.oracle*), 51

release() (*perceptivo.networking.node.Node* method), 45

release() (*perceptivo.video.cameras.Picamera_Process* method), 57

resolution (*perceptivo.types.video.Picamera_Params* attribute), 38

response (*perceptivo.types.psychophys.Sample* property), 31

response (*perceptivo.types.pupil.Dilation* property), 35

responses (*perceptivo.types.psychophys.Samples* attribute), 31

run() (*perceptivo.gui.main.Perceptivo_Clinician.Frame_Receiver* method), 40

run() (*perceptivo.video.cameras.Picamera_Process* method), 57

Runtime (class in *perceptivo.runtimes.runtime*), 19

runtime (*perceptivo.prefs.Patient_Prefs* attribute), 66

Runtimes (class in *perceptivo.prefs*), 65

S

Sample (class in *perceptivo.types.psychophys*), 31

Samples (class in *perceptivo.types.psychophys*), 31

samples (*perceptivo.psychophys.model.Gaussian_Process* property), 49

samples (*perceptivo.types.patient.Patient* attribute), 30

samples (*perceptivo.types.psychophys.Samples* attribute), 31

save() (*perceptivo.prefs.Prefs* method), 65

scaleChanged (*perceptivo.gui.widgets.components.Range_Setter* attribute), 40

scaleChanged (*perceptivo.gui.widgets.control_panel.Control_Panel* attribute), 41

scaleChanged() (*perceptivo.gui.widgets.audiogram.Audiogram* method), 40

search_scale (*perceptivo.video.pupil.EllipseExtractor_Params* attribute), 58

send() (*perceptivo.networking.node.Node* method), 45

sensor_mode (*perceptivo.types.video.Picamera_Params* attribute), 38

serialize() (in module *perceptivo.util*), 63

serialize() (*perceptivo.networking.messages.Message* method), 43

set_color() (*perceptivo.types.video.Frame* method), 38

set_global() (in module *perceptivo.prefs*), 67

setMaximum() (*perceptivo.gui.widgets.components.Range_Setter* method), 41

setMinimum() (*perceptivo.gui.widgets.components.Range_Setter* method), 41

setStarted() (*perceptivo.gui.main.Perceptivo_Clinician* method), 39

settings (*perceptivo.gui.main.Perceptivo_Clinician* property), 39

setValue() (*perceptivo.gui.widgets.control_panel.Control_Panel* method), 41

simple (*perceptivo.video.pupil.Pupil_Extractors* attribute), 60

Socket (class in *perceptivo.types.networking*), 29

socket_type (*perceptivo.types.networking.Socket* attribute), 29

Sound (class in *perceptivo.types.sound*), 36

sound (*perceptivo.types.psychophys.Sample* attribute), 31

Sound.Config (class in *perceptivo.types.sound*), 36

sound_class (*perceptivo.types.sound.Sound* property), 36

sound_kwargs (*perceptivo.types.sound.Sound* property), 36

sound_type (*perceptivo.types.sound.Sound* attribute), 36

stamp_time() (*perceptivo.types.sound.Sound* method), 36

startToggled (*perceptivo.gui.widgets.control_panel.Control_Panel* attribute), 41

STIM (in module *perceptivo.networking.sockets*), 45

stimuli (*perceptivo.prefs.Runtimes* attribute), 65

T

t (*perceptivo.types.units.Ellipse* attribute), 37

Threshold (class in *perceptivo.types.psychophys*), 32

threshold (*perceptivo.types.psychophys.Threshold* attribute), 32

threshold (*perceptivo.types.pupil.Pupil_Params* attribute), 34

thresholds (*perceptivo.types.psychophys.Audiogram* attribute), 33

timestamp (*perceptivo.types.psychophys.Sample attribute*), 31
 timestamp (*perceptivo.types.sound.Sound attribute*), 36
 timestamp (*perceptivo.types.video.Frame attribute*), 37
 timestamps (*perceptivo.types.pupil.Dilation attribute*), 35
 to (*perceptivo.types.networking.Socket attribute*), 29
 to_df() (*perceptivo.types.psychophys.Samples method*), 31
 to_dict() (*perceptivo.types.psychophys.Audiogram method*), 33
 trial() (*perceptivo.runtimes.patient.Patient method*), 20

U

unpack_array() (*in module perceptivo.util*), 63
 update() (*perceptivo.psychophys.model.Audiogram_Model method*), 48
 update() (*perceptivo.psychophys.model.Gaussian_Process method*), 49
 update_period (*perceptivo.prefs.Clinician_Prefs attribute*), 67
 url (*perceptivo.video.processors.Haar_Tracker property*), 62
 use (*perceptivo.types.exam.Completion_Metric attribute*), 26
 use_enum_values (*perceptivo.prefs.Patient_Prefs.Config attribute*), 66
 uuid (*perceptivo.types.sound.Sound attribute*), 36

V

value (*perceptivo.types.gui.GUI_Control attribute*), 27
 value() (*perceptivo.gui.widgets.components.Range_Setter method*), 41
 valueChanged (*perceptivo.gui.widgets.components.Range_Setter attribute*), 40
 valueChanged (*perceptivo.gui.widgets.control_panel.Control_Panel attribute*), 41
 Video (*class in perceptivo.gui.widgets.video*), 42

W

widget_type (*perceptivo.types.gui.GUI_Param attribute*), 28

X

x (*perceptivo.types.units.Ellipse attribute*), 37
 XML_URLS (*perceptivo.video.processors.Haar_Tracker attribute*), 62

Y

y (*perceptivo.types.units.Ellipse attribute*), 37