Package ‘OPI’

May 7, 2026

Type Package

Title Open Perimetry Interface

Version 3.1.0

Date 2026-02-26

Description Implementation of the Open Perimetry Interface (OPI) for simulating and controlling visual field machines using R. The OPI is a standard for interfacing with visual field testing machines (perimeters) first started as an open source project with support of Haag-Streit in 2010. It specifies basic functions that allow many visual field tests to be constructed. As of February 2022 it is fully implemented on the Haag-Streit Octopus 900 and 'CrewT ImoVifa' ('Topcon Tempo') with partial implementations on the Centervue Compass, Kowa AP 7000 and Android phones. It also has a cousin: the R package 'visualFields', which has tools for analysing and manipulating visual field data.

License Apache License (>= 2)

URL https://opi.lei.org.au/

Imports jsonlite, Rfast, abind, openssl

LazyData true

Encoding UTF-8

RoxygenNote 7.3.3

Collate opi.r Display.r PicoVR.r PhoneHMD.r ImoVifa.r dbTocd.r data-RtDbUnits.r data-RtSigmaUnits.r SimYes.r SimNo.r SimHenson.r SimGaussian.r zest.r mocs.r fourTwo.r full_threshold.r pix2deg.r QUESTP.r KTPsi.r Compass.r Octopus900.r Octopus600.r OPI-package.r SimHensonRT.r KowaAP7000Client.r MAIA.r Envision.r

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Andrew Turpin [cre, aut, cph] (ORCID: 0000-0003-2559-8769),
David Lawson [ctb, cph],
Ivan Marin-Franch [ctb, cph],
Matthias Muller [ctb],

Jonathan Denniss [ctb, cph],
Astrid Zeman [ctb],
Giovanni Montesano [ctb]

Maintainer Andrew Turpin <andrew.turpin@lei.org.au>

Repository CRAN

Date/Publication 2026-03-04 04:40:02 UTC

Contents

.opi_env . . . 4
cdTodb . . . 5
chooseOPI . . . 5
dbTocd . . . 6
degTopix . . . 6
fourTwo.start . . . 7
FT . . . 9
kowa.presentKinetic . . . 13
kowa.presentStatic . . . 14
kowa.presentTemporal . . . 15
KTPsi . . . 15
MOCS . . . 19
octo900.presentKinetic . . . 24
octo900.presentStatic . . . 25
octo900.presentTemporal . . . 26
open_socket . . . 27
opiClose . . . 27
opiClose_for_Compass . . . 28
opiClose_for_Display . . . 28
opiClose_for_Envision . . . 29
opiClose_for_ImoVifa . . . 30
opiClose_for_KowaAP7000 . . . 30
opiClose_for_MAIA . . . 31
opiClose_for_O600 . . . 31
opiClose_for_Octopus900 . . . 31
opiClose_for_PhoneHMD . . . 32
opiClose_for_PicoVR . . . 32
opiClose_for_SimGaussian . . . 33
opiClose_for_SimHenson . . . 33
opiClose_for_SimHensonRT . . . 34
opiClose_for_SimNo . . . 34
opiClose_for_SimYes . . . 34
opiInitialise . . . 35
opiInitialise_for_Compass . . . 35
opiInitialise_for_Display . . . 36
opiInitialise_for_Envision . . . 37
opiInitialise_for_ImoVifa . . . 38
opiInitialise_for_KowaAP7000 . . . 39

opiInitialise_for_MAIA . . . 39
opiInitialise_for_O600 . . . 40
opiInitialise_for_Octopus900 . . . 41
opiInitialise_for_PhoneHMD . . . 42
opiInitialise_for_PicoVR . . . 43
opiInitialise_for_SimGaussian . . . 44
opiInitialise_for_SimHenson . . . 45
opiInitialise_for_SimHensonRT . . . 46
opiInitialise_for_SimNo . . . 47
opiInitialise_for_SimYes . . . 47
opiKineticStimulus . . . 48
opiPresent . . . 48
opiPresent_for_Compass . . . 49
opiPresent_for_Display . . . 50
opiPresent_for_Envision . . . 53
opiPresent_for_ImoVifa . . . 54
opiPresent_for_KowaAP7000 . . . 57
opiPresent_for_MAIA . . . 58
opiPresent_for_O600 . . . 59
opiPresent_for_Octopus900 . . . 60
opiPresent_for_PhoneHMD . . . 61
opiPresent_for_PicoVR . . . 64
opiPresent_for_SimGaussian . . . 66
opiPresent_for_SimHenson . . . 67
opiPresent_for_SimHensonRT . . . 68
opiPresent_for_SimNo . . . 70
opiPresent_for_SimYes . . . 70
opiQueryDevice . . . 71
opiQueryDevice_for_Compass . . . 71
opiQueryDevice_for_Display . . . 72
opiQueryDevice_for_Envision . . . 72
opiQueryDevice_for_ImoVifa . . . 74
opiQueryDevice_for_KowaAP7000 . . . 75
opiQueryDevice_for_MAIA . . . 75
opiQueryDevice_for_O600 . . . 76
opiQueryDevice_for_Octopus900 . . . 76
opiQueryDevice_for_PhoneHMD . . . 77
opiQueryDevice_for_PicoVR . . . 77
opiQueryDevice_for_SimGaussian . . . 78
opiQueryDevice_for_SimHenson . . . 78
opiQueryDevice_for_SimHensonRT . . . 79
opiQueryDevice_for_SimNo . . . 79
opiQueryDevice_for_SimYes . . . 79
opiSetBackground . . . 80
opiSetup . . . 80
opiSetup_for_Compass . . . 81
opiSetup_for_Display . . . 81
opiSetup_for_Envision . . . 83

opiSetup_for_ImoVifa 84
opiSetup_for_KowaAP7000 86
opiSetup_for_MAIA 86
opiSetup_for_O600 87
opiSetup_for_Octopus900 88
opiSetup_for_PhoneHMD 89
opiSetup_for_PicoVR 91
opiSetup_for_SimGaussian 93
opiSetup_for_SimHenson 93
opiSetup_for_SimHensonRT 94
opiSetup_for_SimNo 94
opiSetup_for_SimYes 95
opiStaticStimulus 95
opiTemporalStimulus 95
pixTodeg 96
QUESTP 96
RtDbUnits 105
RtSigmaUnits 106
ZEST 107

Index **112**

`.opi_env` *Global environment for OPI to hold machine specific constants, etc.*

Description

Global environment for OPI to hold machine specific constants, etc.

Usage

```
.opi_env
```

Format

An object of class `environment` of length 12.

cdTodb — *Convert cd/m^2 to perimetric dB.*

Description

Given a value in cd/m^2, return the equivalent dB value. Default is to use HFA units, so maximum stimulus is 10000 apostilbs.

Usage

```
cdTodb(cd, maxStim = 10000/pi)
```

Arguments

cd	Value to convert to dB in cd/m^2. Must be > 0.
maxStim	Stimulus value for 0dB in cd/m^2.

Value

A dB value for cd cd/m^2.

Examples

```
# candela to decibels
dB <- cdTodb(10000/pi)  # 0 dB
dB <- cdTodb(1000/pi)   # 10 dB
dB <- cdTodb(100/pi)    # 20 dB
dB <- cdTodb(10/pi)     # 30 dB
dB <- cdTodb(1/pi)      # 40 dB
dB <- cdTodb(0.1/pi)    # 50 dB
```

chooseOPI — *chooseOPI selects an OPI machine to use.*

Description

It should be called before any other OPI functions.

Usage

```
chooseOPI(machine = NULL)

chooseOpi(machine = NULL)
```

Arguments

machine	Machine name to use. Set to NULL to get a list.

Value

NULL on success or list of machines otherwise.

`dbTocd` *Convert perimetric dB to cd/m^2*

Description

Given a value in dB, return the cd/m^2 equivalent. Default is to use HFA units, so maximum stimulus is 10000 apostilbs.

Usage

```
dbTocd(db, maxStim = 10000/pi)
```

Arguments

`db`	Value to convert to cd/m^2.
`maxStim`	Stimulus value for 0dB in cd/m^2.

Value

cd/m^2 value for `db` dB.

Examples

```
# decibels to candela
cd <- dbTocd(0)   # 10000/pi
cd <- dbTocd(10)  # 1000/pi
cd <- dbTocd(20)  # 100/pi
cd <- dbTocd(30)  # 10/pi
cd <- dbTocd(40)  # 1/pi
```

`degTopix` *Convert degrees to pixels for machine 'machine'*

Description

Convert degrees to pixels for machine 'machine'

Usage

```
degTopix(xy, machine = "compass")
```

Arguments

xy	a 2 element vector c(x,y) where x and y are in pixels
machine	"compass" or ...?

Value

xy converted to pixels (top-left is (0,0)) for the machine or NA if machine is unknown

Examples

```
degTopix(c(0, 0), machine="compass")  # c(960, 960) pixels
degTopix(c(-15, 2)) # c(495, 898) pixels
```

---

fourTwo.start *4-2 Staircase*

Description

fourTwo is a 4-2 dB staircase beginning at level est terminating after two reversals. The final estimate is the average of the last two presentations. It also terminates if the minStimulus is not seen twice, or the maxStimulus is seen twice.

Usage

```
fourTwo.start(est = 25, instRange = c(0, 40), verbose = FALSE, makeStim, ...)

fourTwo.step(state, nextStim = NULL)

fourTwo.stop(state)

fourTwo.final(state)
```

Arguments

est	Starting estimate in dB
instRange	Dynamic range of the instrument c(min,max) in dB
verbose	True if you want each presentation printed
makeStim	A function that takes a dB value and numPresentations and returns an OPI datatype ready for passing to opiPresent
...	Extra parameters to pass to the opiPresent function
state	Current state of the fourTwo returned by fourTwo.start and fourTwo.step
nextStim	A valid object for opiPresent to use as its nextStim.

Details

This is an implementation of a 4-2 1-up 1-down staircase. The initial staircase starts at `est` and proceeds in steps of 4 dB until the first reversal, and 2dB until the next reversal. The mean of the last two presentations is taken as the threshold value. Note this function will repeatedly call `opiPresent` for a stimulus until `opiPresent` returns `NULL` (ie no error occurred). If more than one fourTwo is to be interleaved (for example, testing multiple locations), then the `fourTwo.start`, `fourTwo.step`, `fourTwo.stop` and `fourTwo.final` calls can maintain the state of the fourTwo after each presentation, and should be used. See examples below.

Value

Multiple locations:

`fourTwo.start` returns a list that can be passed to `fourTwo.step`, `fourTwo.stop`, and `fourTwo.final`. It represents the state of a fourTwo at a single location at a point in time and contains the following.

- `name` `fourTwo`
- A copy of all of the parameters supplied to fourTwo.start: `startingEstimate=est`, `minStimulus=instRange[1]`, `maxStimulus=instRange[2]`, `makeStim`, and `opiParams=list(...)`
- `currentLevel` The next stimulus to present.
- `lastSeen` The last seen stimulus.
- `lastResponse` The last response given.
- `stairResult` The final result if finished (initially `NA`).
- `finished` `"Not"` if staircase has not finished, or one of `"Rev"` (finished due to 2 reversals), `"Max"` (finished due to 2 `maxStimulus` seen), `"Min"` (finished due to 2 `minStimulus` not seen)
- `numberOfReversals` Number of reversals so far.
- `currSeenLimit` Number of times `maxStimulus` has been seen.
- `currNotSeenLimit` Number of times `minStimulus` not seen.
- `numPresentations` Number of presentations so far.
- `stimuli` Vector of stimuli shown at each call to `fourTwo.step`.
- `responses` Vector of responses received (1 seen, 0 not) received at each call to `fourTwo.step`.
- `responseTimes` Vector of response times received at each call to `fourTwo.step`.

`fourTwo.step` returns a list containing

- `state` The new state after presenting a stimuli and getting a response.
- `resp` The return from the `opiPresent` call that was made.

`fourTwo.stop` returns `TRUE` if the staircase is finished (2 reversals, or `maxStimulus` is seen twice or `minStimulus` is not seen twice).

`fourTwo.final` returns the final estimate of threshold (mean of last two reversals). This issues a warning if called before the staircase has finished, but still returns a value.

See Also

`dbTocd`, `opiPresent`, `FT`

Examples

```
# Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500)
  class(s) <- "opiStaticStimulus"
  return(s)
}
chooseOpi("SimHenson")
if (!is.null(opiInitialize(type="C", cap=6)$err))
  stop("opiInitialize failed")

##############################################
# This section is for multiple fourTwos
##############################################
makeStimHelper <- function(db,n, x, y) {  # returns a function of (db,n)
  ff <- function(db, n) db+n
  body(ff) <- substitute({
    s <- list(x=x, y=y, level=dbTocd(db), size=0.43, color="white",
              duration=200, responseWindow=1500)
    class(s) <- "opiStaticStimulus"
    return(s)}, list(x=x,y=y))
  return(ff)
}
# List of (x, y, true threshold) triples
locations <- list(c(9,9,30), c(-9,-9,32), c(9,-9,31), c(-9,9,33))

# Setup starting states for each location
states <- lapply(locations, function(loc) {
  fourTwo.start(makeStim=makeStimHelper(db,n,loc[1],loc[2]),
                tt=loc[3], fpr=0.03, fnr=0.01)})

# Loop through until all states are "stop"
while(!all(st <- unlist(lapply(states, fourTwo.stop)))) {
  i <- which(!st)                        # choose a random,
  i <- i[runif(1, min=1, max=length(i))]  # unstopped state
  r <- fourTwo.step(states[[i]])               # step it
  states[[i]] <- r$state                 # update the states
}

finals <- lapply(states, fourTwo.final)    # get final estimates of threshold
for(i in 1:length(locations)) {
  cat(sprintf("Location (%+2d,%+2d) ",locations[[i]][1], locations[[i]][2]))
      cat(sprintf("has threshold %4.2f\n", finals[[i]]))
}

if (!is.null(opiClose()$err))
  warning("opiClose() failed")
```

FT *Full Threshold*

Description

FT begins with a 4-2dB staircase beginning at level `est`. If the final estimate (last seen) is more than 4dB away from `est`, a second 4-2 staircase is completed beginning at the estimate returned from the first

Usage

```
FT(est = 25, instRange = c(0, 40), verbose = FALSE, makeStim, ...)

FT.start(est = 25, instRange = c(0, 40), makeStim, ...)

FT.step(state, nextStim = NULL)

FT.stop(state)

FT.final(state)
```

Arguments

`est`	Starting estimate in dB
`instRange`	Dynamic range of the instrument c(min,max) in dB
`verbose`	True if you want each presentation printed
`makeStim`	A function that takes a dB value and numPresentations and returns an OPI datatype ready for passing to opiPresent
`...`	Extra parameters to pass to the opiPresent function
`state`	Current state of the FT returned by `FT.start` and `FT.step`
`nextStim`	A valid object for `opiPresent` to use as its `nextStim`

Details

This is an implementation of a 4-2 1-up 1-down staircase as implemented in the first Humphrey Field Analyzer. The initial staircase starts at `est` and proceeds in steps of 4 dB until the first reversal, and 2dB until the next reversal. The last seen stimulus is taken as the threshold value. If, after the first staircase, the threshold is more than 4 dB away from the starting point, then a second staircase is initiated with a starting point equal to the threshold found with the first staircase.

Note this function will repeatedly call `opiPresent` for a stimulus until `opiPresent` returns `NULL` (ie no error occurred)

If more than one FT is to be interleaved (for example, testing multiple locations), then the `FT.start`, `FT.step`, `FT.stop` and `FT.final` calls can maintain the state of the FT after each presentation, and should be used. If only a single FT is required, then the simpler `FT` can be used. See examples below

Value

Single location:

Returns a list containing

- `npres` Total number of presentations
- `respSeq` Response sequence stored as a list of (seen,dB) pairs
- `first` First staircase estimate in dB
- `final` Final threshold estimate in dB

Multiple locations:

`FT.start` returns a list that can be passed to `FT.step`, `FT.stop`, and `FT.final`. It represents the state of a FT at a single location at a point in time and contains the following.

- `name` `FT`
- A copy of all of the parameters supplied to FT.start: `startingEstimate=est`, `minStimulus=instRange[1]`, `maxStimulus=instRange[2]`, `makeStim`, and `opiParams=list(...)`.
- `currentLevel` The next stimulus to present.
- `lastSeen` The last seen stimulus.
- `lastResponse` The last response given.
- `firstStairResult` The result of the first staircase (initially `NA`).
- `secondStairResult` The result of the first staircase (initially `NA`, and could remain `NA`).
- `finished` `TRUE` if staircase has finished (2 reversals, or max/min seen/not-seen twice).
- `numberOfReversals` Number of reversals so far.
- `currSeenLimit` Number of times `maxStimulus` has been seen.
- `currNotSeenLimit` Number of times `minStimulus` not seen.
- `numPresentations` Number of presentations so far.
- `stimuli` Vector of stimuli shown at each call to `FT.step`.
- `responses` Vector of responses received (1 seen, 0 not) received at each call to `FT.step`.
- `responseTimes` Vector of response times received at each call to`FT.step`.

`FT.step` returns a list containing

- `state` The new state after presenting a stimuli and getting a response.
- `resp` The return from the `opiPresent` call that was made.

`FT.stop` returns `TRUE` if the first staircase has had 2 reversals, or `maxStimulus` is seen twice or `minStimulus` is not seen twice and the final estimate is within 4 dB of the starting stimulus. Returns `TRUE` if the second staircase has had 2 reversals, or `maxStimulus` is seen twice or `minStimulus` is not seen twice

`FT.final` returns the final estimate of threshold based on state, which is the last seen in the second staircase, if it ran, or the first staircase otherwise

`FT.final.details` returns a list containing

- `final` The final threshold.
- `first` The threshold determined by the first staircase (might be different from final).
- `stopReason` Either `Reversals`, `Max`, or `Min` which are the three ways in which FT can terminate.
- `np` Number of presentation for the whole procedure (including both staircases if run).

References

A. Turpin, P.H. Artes and A.M. McKendrick. "The Open Perimetry Interface: An enabling tool for clinical visual psychophysics", Journal of Vision 12(11) 2012.

H. Bebie, F. Fankhauser and J. Spahr. "Static perimetry: strategies", Acta Ophthalmology 54 1976.

C.A. Johnson, B.C. Chauhan, and L.R. Shapiro. "Properties of staircase procedures for estimating thresholds in automated perimetry", Investigative Ophthalmology and Vision Science 33 1993.

See Also

dbTocd, opiPresent, fourTwo.start

Examples

```
# Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500)
  class(s) <- "opiStaticStimulus"
  return(s)
}
chooseOpi("SimHenson")
if (!is.null(opiInitialize(type="C", cap=6)$err))
  stop("opiInitialize failed")

result <- FT(makeStim=makeStim, tt=30, fpr=0.15, fnr=0.01)
if (!is.null(opiClose()$err))
  warning("opiClose() failed")

##############################################
# This section is for multiple FTs
##############################################
chooseOpi("SimHenson")
if (!is.null(opiInitialize(type="C", cap=6)$err))
  stop("opiInitialize failed")

makeStimHelper <- function(db,n, x, y) {  # returns a function of (db,n)
  ff <- function(db, n) db+n
  body(ff) <- substitute({
    s <- list(x=x, y=y, level=dbTocd(db), size=0.43, color="white",
              duration=200, responseWindow=1500)
    class(s) <- "opiStaticStimulus"
    return(s)
  }, list(x=x,y=y))
  return(ff)
}

# List of (x, y, true threshold) triples
locations <- list(c(9,9,30), c(-9,-9,32), c(9,-9,31), c(-9,9,33))
# Setup starting states for each location
states <- lapply(locations, function(loc) {
  FT.start(makeStim=makeStimHelper(db,n,loc[1],loc[2]),
```

```
            tt=loc[3], fpr=0.03, fnr=0.01)})

# Loop through until all states are "stop"
while(!all(st <- unlist(lapply(states, FT.stop)))) {
  i <- which(!st)                        # choose a random,
  i <- i[runif(1, min=1, max=length(i))]  # unstopped state
  r <- FT.step(states[[i]])              # step it
  states[[i]] <- r$state                 # update the states
}

finals <- lapply(states, FT.final)    # get final estimates of threshold
for(i in 1:length(locations)) {
  cat(sprintf("Location (%+2d,%+2d) ",locations[[i]][1], locations[[i]][2]))
  cat(sprintf("has threshold %4.2f\n", finals[[i]]))
}

if(!is.null(opiClose()$err))
  warning("opiClose() failed")
```

`kowa.presentKinetic` *Present kinetic on Kowa AP7000 (internal use)*

Description

Implementation of opiPresent for the Kowa AP7000 machine. Version for opiKineticStimulus.

This is for internal use only. Use `opiPresent()` with `stim` as class `opiStaticStimulus` and you will get the Value back.

Arguments

`stim`	Stimulus to present (a list, see details).

Details

`stim` is a list containing at least the following 3 elements:

- `path`, a list containing `x` 2 x-coordinate in degrees (floating point) (range $[-80,80]$) and `y`, list of 2 y-coordinate in degrees (floating point) (range $[-70,65]$).
- `sizes` list of 1 size; one of `.opi_env$KowaAP7000$SIZES_DEGREES`.
- `colors` list of 1 color; one of `.opi_env$KowaAP7000$COLOR_WHITE`, `.opi_env$KowaAP7000$COLOR_GREEN`, `.opi_env$KowaAP7000$COLOR_BLUE`, or `.opi_env$KowaAP7000$COLOR_RED`
- `levels` list of 1 level; luminance in cd/m^2, and is rounded to the nearest whole dB for display (range 0 to 50). 0dB is 10000aps.
- `speeds` list of 1 speed; degrees per second range $[3, 5]$.
- `duration` of stimulus on in milliseconds (range $[100, 1200]$).
- `responseWindow` from start of stimulus presentation in milliseconds (max 5000).

Value

A list containing

- `err` String message or NULL for no error.
- `seen` 1 if seen, 0 otherwise.
- `time` Reaction time (if seen).
- `x` Location of button press in degrees.
- `y` Location of button press in degrees.

`kowa.presentStatic` *Present static on Kowa AP7000 (internal use)*

Description

Implementation of opiPresent for the Kowa AP7000 machine. Version for opiStaticStimulus.

This is for internal use only. Use `opiPresent()` with `stim` as class `opiStaticStimulus` and you will get the Value back.

Arguments

`stim`	Stimulus to present (a list, see details).
`nextStim`	The stimulus to present after stim (it is not presented, but projector can move to it during response window)

Details

`stim` is a list containing at least the following 3 elements:

- `x`, x-coordinate in degrees (floating point) (range $[-80,80]$).
- `y`, y-coordinate in degrees (floating point) (range $[-70,65]$).
- `level` is luminance in cd/m^2, and is rounded to the nearest whole dB for display (range 0 to 50). 0dB is 10000aps.
- `duration` of stimulus on in milliseconds (range $[100, 1200]$).
- `responseWindow` from start of stimulus presentation in milliseconds (max 5000).
- `size` one of `.opi_env$KowaAP7000$SIZES_DEGREES`.
- `color` one of `.opi_env$KowaAP7000$COLOR_WHITE`, `.opi_env$KowaAP7000$COLOR_GREEN`, `.opi_env$KowaAP7000$COLOR_BLUE`, or `.opi_env$KowaAP7000$COLOR_RED`

Value

A list containing

- `err` String message or NULL for no error.
- `seen` 1 if seen, 0 otherwise.
- `time` Reaction time (if seen).
- `pupilX`
- `pupilY`
- `purkinjeX`
- `purkinjeY`

`kowa.presentTemporal` *Present temporal stim on Kowa AP7000 (internal use)*

Description

Implementation of opiPresent for the Kowa AP7000 machine. Version for opiKineticStimulus.

This is for internal use only. Use `opiPresent()` with `stim` as class `opiStaticStimulus` and you will get the Value back.

`KTPsi` *An implementation of Kontsevich and Tyler Ψ algorithm.*

Description

An implementation of Kontsevich and Tyler (Vis Res 39 (1999) pages 2729–2737 default parameterised for Standard Automated Perimetry. based on A. Turpin, D. Jankovic and A.M. McKendrick, "Identifying Steep Psychometric Function Slope Quickly in Clinical Applications", Vision Research, 50(23). November 2010. Pages 2476-2485

Usage

```
KTPsi(
 domains = list(slopes = 1:5, thresholds = 20:40, fps = c(0, 0.025, 0.05, 0.1, 0.2), fns
    = c(0, 0.025, 0.05, 0.1, 0.2)),
  priors = list(slopes = rep(1, length(domains$slopes))/length(domains$slopes),
   thresholds = rep(1, length(domains$thresholds))/length(domains$thresholds), fps =
    rep(1, length(domains$fps))/length(domains$fps), fns = rep(1,
    length(domains$fns))/length(domains$fns)),
  stimValues = 17:40,
  stopType = "N",
  stopValue = 140,
  maxPresentations = 200,
```

```
  minInterStimInterval = NA,
  maxInterStimInterval = NA,
  verbose = 0,
  makeStim,
  ...
)

KTPsi.start(
 domains = list(slopes = 1:5, thresholds = 20:40, fps = c(0, 0.025, 0.05, 0.1, 0.2), fns
    = c(0, 0.025, 0.05, 0.1, 0.2)),
  priors = list(slopes = rep(1, length(domains$slopes))/length(domains$slopes),
   thresholds = rep(1, length(domains$thresholds))/length(domains$thresholds), fps =
    rep(1, length(domains$fps))/length(domains$fps), fns = rep(1,
    length(domains$fns))/length(domains$fns)),
  stimValues = 17:40,
  stopType = "N",
  stopValue = 140,
  maxPresentations = 200,
  minInterStimInterval = NA,
  maxInterStimInterval = NA,
  verbose = 0,
  makeStim,
  ...
)

KTPsi.step(state, nextStim = NULL, fixedStimValue = NA)

KTPsi.final(state, method = "expectation")

KTPsi.stop(state)
```

Arguments

`domains` A list of 4 vectors:

- `slopes` The valid slopes in the domain of psychometric functions.
- `thresholds` The valid thresholds in the domain of psychometric functions.
- `fps` The valid upper asymptotes (false positives) in the domain of psychometric functions.
- `fns` The valid lower asymptotes (false negatives) in the domain of psychometric functions.

`priors` A list of 4 vectors:

- `slopes` The prior probability vector for `domains$slopes`.
- `thresholds` The prior probability vector for `domains$thresholds`.
- `fps` The prior probability vector for `domains$fps`.
- `fns` The prior probability vector for `domains$fns`.

Each prior should the same length as its `domains` counterpart and sum to 1.

`stimValues` Vector of allowable stimulus values.

`stopType` `N`, for number of presentations and `H`, for the entropy of the pdf.

`stopValue` Value for number of presentations (`stopType=N`), or Entropy (`stopType=H`).

`maxPresentations`
Maximum number of presentations regardless of `stopType`.

`minInterStimInterval`
If both `minInterStimInterval` and `maxInterStimInterval` are not `NA`, then between each stimuli there is a random wait period drawn uniformly between `minInterStimInterval` and `maxInterStimInterval`.

`maxInterStimInterval`
`minInterStimInterval`.

`verbose` `verbose=0` does nothing, `verbose=1` stores pdfs for returning, and `verbose=2` stores pdfs and also prints each presentation.

`makeStim` A function that takes a stimulus value and numPresentations and returns an OPI datatype ready for passing to opiPresent. See examples.

`...` Extra parameters to pass to the opiPresent function

`state` Current state of the KTPsi as returned by (eg) `KTPsi.start`.

`nextStim` The next stimulus to present in a suitable format for passing to `opiPresent`

`fixedStimValue` Currently ignored.

`method` Either `"expectation"` or `"MAP"`.

Details

The assumed psychometric function is the cumulative Gaussian:

$$\text{fp} + (1 - \text{fp} - \text{fn})(1 - \text{pnorm}(x, \text{threshold}, \text{slope})$$

hence `domain$slopes` are standard deviations and `domain$thresholds` are the mean.

While it is assumed that `domains$thresholds` and `stimValues` are in dB, this need not be the case. As long as the `makeStim` function converts `stimValues` into cd/m^2 for the `opiPresent` function, then any units should work.

The `checkFixationOK` function is called (if present in stim made from `makeStim`) after each presentation, and if it returns FALSE, the pdf for that state is not changed (ie the presentation is ignored), but the stim, number of presentations etc is recorded in the state.

If more than one KTPsi is to be interleaved (for example, testing multiple locations), then thePsi `KTPsi.start`, `KTPsi.step`, `KTPsi.stop` and `KTPsi.final` calls can maintain the state of the KTPsi after each presentation, and should be used. If only a single KTPsi is required, then the simpler `KTPsi` function can be used, which is a wrapper for the four functions that maintain state. See examples below.

Value

Single location:

`KTPsi` returns a list containing

- `npres` Total number of presentations used.

- `respSeq` Response sequence stored as a matrix: row 1 is dB values of stimuli, row 2 is 1/0 for seen/not-seen, row 3 is fixated 1/0 (always 1 if `checkFixationOK` not present in stim objects returned from `makeStim`).
- `pdfs` If `verbose` is bigger than 0, then this is a list of the pdfs used for each presentation, otherwise NULL.
- `final` The mean/median/mode of the final pdf, depending on `stimChoice`, which is the determined threshold.
- `opiResp` A list of responses received from each successful call to `opiPresent` within `KTPsi`.

Multilple locations:

`KTPsi.start` returns a list that can be passed to `KTPsi.step`, `KTPsi.stop`, and `KTPsi.final`. It represents the state of a KTPsi at a single location at a point in time and contains the following.

- `name` `KTPsi`
- A copy of all of the parameters supplied to KTPsi.start: `domains`, `priors`, `stimValues`, `stopType`, `stopValue`, `maxPresentations`, `makeStim` and `opiParams`.
- `psi` A matrix where `psi[domain_index, stim]` is the probability of seeing `stim` assuming the psychometric function for the domain index `domain_index`.
- `labels` A text representation of `psi[domain_index, ]`, or the the psychometric function for the domain index `domain_index`.
- `pdf` Current pdf: vector of probabilities the same length as product of lengths of `domain` elements.
- `numPresentations` The number of times `KTPsi.step` has been called on this state.
- `stimuli` A vector containing the stimuli used at each call of `KTPsi.step`.
- `responses` A vector containing the responses received at each call of `KTPsi.step`.
- `responseTimes` A vector containing the response times received at each call of `KTPsi.step`.
- `fixated` A vector containing TRUE/FALSE if fixation was OK according to `checkFixationOK` for each call of `KTPsi.step` (defaults to TRUE if `checkFixationOK` not present).
- `opiResp` A list of responses received from each call to `opiPresent` within `KTPsi.step`.

`KTPsi.step` returns a list containing

- `stat`: The new state after presenting a stimuli and getting a response.
- `resp` The return from the `opiPresent` call that was made.

`KTPsi.stop` returns `TRUE` if the KTPsi has reached its stopping criteria, and `FALSE` otherwise.

`KTPsi.final` returns an estimate of threshold based on state based on its parameter.

TRUE if the `state` has reached its stopping criteria, and FALSE otherwise.

References

Kontsevich and Tyler. Vision Research 39 (1999) pages 2729–2737.

A. Turpin, D. Jankovic and A.M. McKendrick, "Identifying Steep Psychometric Function Slope Quickly in Clinical Applications", Vision Research, 50(23). November 2010. Pages 2476-2485

A. Turpin, P.H. Artes and A.M. McKendrick "The Open Perimetry Interface: An enabling tool for clinical visual psychophysics", Journal of Vision 12(11) 2012.

See Also

dbTocd, opiPresent

Examples

```
chooseOpi("SimGaussian")
if(!is.null(opiInitialize(sd = 2)$err))
  stop("opiInitialize failed")

     # This section is for single location KTPsi
     # Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500, checkFixationOK=NULL)
  class(s) <- "opiStaticStimulus"
  return(s)
}

KTPsi(makeStim = makeStim, stopType="H", stopValue=  3, tt=30, fpr=0.03)
KTPsi(makeStim = makeStim, stopType="N", stopValue= 27, verbose = 0, tt=30, fpr=0.03)

     # For multiple locations...
## Not run:
states <- lapply(1:10, function(loc) KTPsi.start(makeStim = makeStim))
unfinished <- 1:10
while (length(unfinished) > 0) {
     loc <- unfinished[[1]]
     states[[loc]] <- KTPsi.step(states[[loc]])$state
     if (KTPsi.stop(states[[loc]]))
         unfinished <- unfinished[-1]
}

## End(Not run)
```

MOCS *Method of Constant Stimuli (MOCS)*

Description

MOCS performs either a yes/no or n-interval-forced-choice Method of Constant Stimuli test

Usage

```
MOCS(
  params = NA,
  order = "random",
  responseWindowMeth = "constant",
  responseFloor = 1500,
```

```
  responseHistory = 5,
 keyHandler = function(correct, ret) return(list(seen = TRUE, time = 0, err = NULL)),
  interStimMin = 200,
  interStimMax = 500,
  beep_function,
  makeStim,
  stim_print,
  ...
)
```

Arguments

`params` A matrix where each row is `x y i n correct_n ll1 ll2 ... llm` where

- `x` is X coordinate of location
- `y` is Y coordinate of location
- `i` is a location number (assigned by caller)
- `n` is Number of times this location/luminance(s) should be repeated
- `correct_n` is the index i of the luminance level (`lli`) that should be treated as a "correct" response (the correct interval). For a standard MOCS, this will be 1; for a 2AFC, this will be 1 or 2. This number will be in the range `[1,m]`.
- `lli` is the i'th luminance level to be used at this location for interval i of the presentation in cd/m^2. For a standard MOCS, i=1, and the `params` matrix will have 5 columns. For a 2AFC, there will be two lli's, and `params` will have 6 columns.

`order` Control the order in which the stimuli are presented.

- `"random"` Randomise the order of trials/locations.
- `"fixed"` Present each row of `params` in order of `1:nrow(params)`, ignoring the `n` (4th) column in `params`.

`responseWindowMeth`
Control time perimeter waits for response.

- `"speed"` After an average of the last `speedHistory` response times, with a minimum of `responseFloor`. Initially #' `responseFloor`.
- `"constant"` Always use `responseFloor`.
- `"forceKey"` Wait for a keyboard input.

`responseFloor` Minimum response window (for any `responseWindowMeth` except `"forceKey"`).

`responseHistory`
Number of past yeses to average to get response window (only used if `responseWindowMeth` is `"speed"`).

`keyHandler` Function to get a keyboard input and returns as for `opiPresent`: `err`, `seen` and `time`. The parameters passed to the function are the correct interval number (column 4 of `params`), and the result of `opiPresent`. See Examples.

`interStimMin` Regardless of response, wait `runif(interStimMin, interStimMax)` ms.

`interStimMax` Regardless of response, wait `runif(interStimMin, interStimMax)` ms.

`beep_function`	A function that takes the string `'correct'`, the string `'incorrect'`, or a stimulus number and plays an appropriate sound. See examples.
`makeStim`	A helper function to take a row of `params` and a response window length in ms, and create a list of OPI stimuli types for passing to opiPresent. This may include a `checkFixationOK` function. See Example.
`stim_print`	A function that takes an `opiStaticStimulus` and return list from `opiPresent` and returns a string to print for each presentation. It is called immediately after each `opiPresent`, and the string is prepended with the (x,y) coordinates of the presentation and ends with a newline.
`...`	Extra parameters to pass to the opiPresent function.

Details

Whether the test is yes/no or forced-choice is determined by the number of columns in `params`. The code simply presents all columns from 5 onwards and collects a response at the end. So if there is only 5 columns, it is a yes/no task. If there are 6 columns it is a 2-interval-forced-choice. Generally, an nIFC experiment has 4+n columns in `params`.

Note that when the `order` is `"random"`, the number of trials in the test will be the sum of the 3rd column of `params`. When the `order` is `"fixed"`, there is only one presentation per row, regardless of the value in the 3rd column of `params`.

If a response is received before the final trial in a nIFC experiment, it is ignored.

If the `checkFixationOK` function is present in a stimulus, then it is called after each presentation, and the result is "anded" with each stimulus in a trial to get a TRUE/FALSE for fixating on all stimuli in a trial.

Value

Returns a data.frame with one row per stimulus copied from params with extra columns appended: checkFixation checks, and the return values from `opiPresent()` (see example). These last values will differ depending on which machine/simulation you are running (as chosen with `chooseOpi()`.

- column 1: x
- column 2: y
- column 3: location number
- column 4: number of times to repeat this stim
- column 5: correct stimulus index
- column 6: TRUE/FALSE was fixating for all presentations in this trial according to `checkFixationOK`
- column 7...: columns from params
- ...: columns from opiPresent return

References

A. Turpin, P.H. Artes and A.M. McKendrick. "The Open Perimetry Interface: An enabling tool for clinical visual psychophysics", Journal of Vision 12(11) 2012.

See Also

dbTocd, opiPresent

Examples

```
# For the Octopus 900
# Check if pupil centre is within 10 pixels of (160,140)
checkFixationOK <- function(ret) return(sqrt((ret$pupilX - 160)^2 + (ret$pupilY - 140)^2) < 10)

# Return a list of opi stim objects (list of class opiStaticStimulus) for each level (dB) in
# p[5:length(p)]. Each stim has responseWindow BETWEEN_FLASH_TIME, except the last which has
# rwin. This one assumes p is on old Octopus 900 dB scale (0dB == 4000 cd/m^2).
makeStim <- function(p, rwin) {
  BETWEEN_FLASH_TIME <- 750   # ms
  res <- NULL
  for(i in 5:length(p)) {
    s <- list(x=p[1], y=p[2], level=dbTocd(p[i],4000/pi), size=0.43, duration=200,
              responseWindow=ifelse(i < length(p), BETWEEN_FLASH_TIME, rwin),
              checkFixationOK=NULL)
    class(s) <- "opiStaticStimulus"
    res <- c(res, list(s))
  }
  return(res)
}

##################################################################
# Read in a key press 'z' is correct==1, 'm' otherwise
#    correct is either 1 or 2, whichever is the correct interval
#
# Return list(seen={TRUE|FALSE}, time=time, err=NULL))
#        seen is TRUE if correct key pressed
##################################################################
## Not run:
  if (length(dir(".", "getKeyPress.py")) < 1)
    stop('Python script getKeyPress.py missing?')

## End(Not run)

keyHandler <- function(correct, ret) {
  return(list(seen=TRUE, time=0, err=NULL))
  ONE <- "b'z'"
  TWO <- "b'm'"
  time <- Sys.time()
  key <- 'q'
  while (key != ONE && key != TWO) {
    a <- system('python getKeyPress.py', intern=TRUE)
    key <- a # substr(a, nchar(a), nchar(a))
    print(paste('Key pressed: ',key,'from',a))
    if (key == "b'8'")
      stop('Key 8 pressed')
  }
  time <- Sys.time() - time
```

```
  if ((key == ONE && correct == 1) || (key == TWO && correct == 2))
    return(list(seen=TRUE, time=time, err=NULL))
  else
    return(list(seen=FALSE, time=time, err=NULL))
}

################################################################
# Read in return value from opipresent with F310 controller.
# First param is correct, next is 1 for left button, 2 for right button
# Left button (LB) is correct for interval 1, RB for interval 2
#    correct is either 1 or 2, whichever is the correct interval
#
# Return list(seen={TRUE|FALSE}, time=time, err=NULL))
#        seen is TRUE if correct key pressed
################################################################
F310Handler <- function(correct, opiResult) {
  z <- opiResult$seen == correct
  opiResult$seen <- z
  return(opiResult)
}

################################################################
# 2 example beep_function
################################################################
## Not run:
  require(beepr)
  myBeep <- function(type='None') {
    if (type == 'correct') {
      beepr::beep(2)  # coin noise
      Sys.sleep(0.5)
    }
    if (type == 'incorrect') {
      beepr::beep(1) # system("rundll32 user32.dll,MessageBeep -1") # system beep
      #Sys.sleep(0.0)
    }
  }
  require(audio)
  myBeep <- function(type="None") {
    if (type == 'correct') {
      wait(audio::play(sin(1:10000/10)))
    }
    if (type == 'incorrect') {
      wait(audio::play(sin(1:10000/20)))
    }
  }

## End(Not run)

################################################################
# An example stim_print function
################################################################
## Not run:
  stim_print <- function(s, ret) {
```

```
    sprintf("%4.1f %2.0f",cdTodb(s$level,10000/pi), ret$seen)
  }

## End(Not run)
```

`octo900.presentKinetic`

Present Kinetic stimuli on O900 (internal use)

Description

Implementation of opiPresent for Kinetic stimuli on the Octopus090 machine.

This is for internal use only. Use `opiPresent()` with these Arguments and `stim` as class `opiKineticStimulus` and you will get the Value back.

Arguments

`stim`	Stimulus to present (a list, see details).
`nextStim`	Ignored.

Details

`stim` is a list containing at least the following 3 elements:

- `path`, A list of (x,y) coordinates in degrees that is usable by `xy.coords()`.
- `sizes`, A list where `sizes[i]` is the size of stimulus (diameter in degrees) to use for the section of path specified by `path[i]..path[i+1]`. Rounded to nearest Goldmann size.
- `levels` A list where `levels[i]` is the stimulus luminance in cd/m^2 to use for the section of path specified by `path[i]..path[i+1]`.
- `speeds` A list where `speeds[i]` is the speed in degrees per second to use for the section of path specified by `path[i]..path[i+1]`.

Value

A list containing

- `err` String message or NULL for no error.
- `seen` 1 if seen, 0 otherwise.
- `time` Reaction time (if seen).
- `x` Coordinate where button was pressed (degrees - i guess).
- `y` Coordinate where button was pressed (degrees - i guess).

octo900.presentStatic *Present static on O900 (internal use)*

Description

Implementation of opiPresent for the Octopus090 machine. Version for opiStaticStimulus.

This is for internal use only. Use `opiPresent()` with `stim` as class `opiStaticStimulus` and you will get the Value back.

Arguments

`stim`	Stimulus to present (a list, see details).
`nextStim`	The stimulus to present after stim (it is not presented, but projector can move to it during response window)
`F310`	If `F310` is `FALSE`, response is taken from internal button. If `F310` is `TRUE` , response is taken from external controller

Details

`stim` is a list containing at least the following 3 elements:

- `x`, x-coordinate in degrees (floating point) (range $[-30,30]$).
- `y`, y-coordinate in degrees (floating point) (range $[-30,30]$).
- `level` is luminance in cd/m^2, and is rounded to the nearest whole dB for display (range 0 to 50). 0dB is 10000aps.

It can also contain:

- `responseWindow` from start of stimulus presentation in milliseconds (default is 1500).
- `duration` of stimulus on in milliseconds (default 200).
- `color` one of `.opi_env$O900$STIM_WHITE`, `.opi_env$O900$STIM_BLUE` or `.opi_env$O900$STIM_RED`. It must be same as that initialised by `opiSetup()` or `opiInitialize()` (default `.opi_env$O900$STIM_WHITE`).
- `size` of stimulus diameter in degrees (default Size III == 0.43). This is rounded to the nearest support Goldmann size.

If responses are taken from the F310 Controller then

- If the L button is pressed, `seen` is set to 1.
- If the R button is pressed, `seen` is set to 2.
- If no button is pressed within `responseWindow`, then `seen` is set to 0.

If stim is null, always return err = NULL status.

Value

A list containing

- `err` String message or NULL for no error.
- `seen` 1 if seen, 0 otherwise. (See details for F310)
- `time` Reaction time (if seen).

`octo900.presentTemporal` *Present Temporal stimuli on O900 (internal use)*

Description

Implementation of opiPresent for Temporal stimuli on the Octopus090 machine.

This is for internal use only. Use `opiPresent()` with these Arguments (`stim` as class `opiTemporalStimulus`) and you will get the Value back.

Arguments

`stim`	Stimulus to present (a list, see details).
`nextStim`	The stimulus to present after stim (it is not presented, but projector can move to it during response window)

Details

`stim` is a list containing at least the following 3 elements:

- `x`, x-coordinate in degrees (floating point) (range $[-30,30]$).
- `y`, y-coordinate in degrees (floating point) (range $[-30,30]$).
- `rate` is frequency in Hz.

It can also contain:

- `responseWindow` from start of stimulus presentation in milliseconds (default is 1500).
- `duration` of stimulus on in milliseconds (default 200).
- `size` of stimulus diameter in degrees (default Size III == 0.43). This is rounded to the nearest support Goldmann size.

If stim is null, always return `err = NULL` status.

Value

A list containing

- `err` String message or NULL for no error.
- `seen` 1 if seen, 0 otherwise.
- `time` Reaction time (if seen).

open_socket — *Open a socket on ip and port.*

Description

Internal use only.

Usage

```
open_socket(ip, port, machineName)
```

Arguments

ip	IP address of socket
port	TCP port of socket
machineName	Machine name for error message

Value

Socket or NULL on error

opiClose — *Calls opiClose_for_MACHINE as appropriate.*

Description

Specific parameters and return values can be seen in the machine specific versions listed below in the 'See Also'.

Usage

```
opiClose()
```

Value

Each implementation should(!) return a list with at least the following elements:

- `err` `NULL` if no error, otherwise a string describing the error.

See Also

opiClose_for_Compass(), opiClose_for_Octopus900(), opiClose_for_ImoVifa(), opiClose_for_PhoneHMD(), opiClose_for_Display(), opiClose_for_PicoVR(), opiClose_for_SimNo(), opiClose_for_SimYes(), opiClose_for_SimHenson(), opiClose_for_SimHensonRT(), opiClose_for_SimGaussian(), opiClose_for_Envision()

opiClose_for_Compass *Implementation of opiClose for the Compass machine.*

Description

This is for internal use only. Use `opiClose()` with these Arguments and you will get the Value back.

Details

WARNING: as at Feb 2026, this will only return data once per Exam initiated on the Compass machine. If you want data returned after one opiClose call, "Stop Exam" on the Compass machine and create a new Exam. `opiInitialise` etc will still work as normal if you start a second session with the same Exam, but no data will be returned by `opiClose`.

Value

A list with elements

- `err`, which is an error code, NULL for no error
- `fixations`, which is a matrix one row per fixation and three columns:
  - `time` (same as `time_hw` in `opiPresent`)
  - `x` (degrees relative to the centre of the image returned by `opiInitialise` - not the PRL)
  - `y` (as for x)

opiClose_for_Display *Implementation of opiClose for the Display machine.*

Description

This is for internal use only. Use `opiClose()` after `chooseOPI("Display")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiClose()`

Examples

```
## Not run:
chooseOpi("Display")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiClose()

## End(Not run)
```

`opiClose_for_Envision` *Implementation of opiClose for the Envision HMP.*

Description

This is for internal use only. Use `opiClose()` after `chooseOPI("Envision")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiClose()`

Examples

```
## Not run:
chooseOpi("Envision")
opiInitialise(address = list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiClose()

## End(Not run)
```

opiClose_for_ImoVifa *Implementation of opiClose for the ImoVifa machine.*

Description

This is for internal use only. Use `opiClose()` after `chooseOPI("ImoVifa")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiClose()`

Examples

```
## Not run:
chooseOpi("ImoVifa")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiClose()

## End(Not run)
```

opiClose_for_KowaAP7000 *Implementation of opiClose for the Kowa AP7000 machine.*

Description

This is for internal use only. Use `opiClose()` with the same parameters.

Value

Returns `list(err = NULL)`.

opiClose_for_MAIA *Implementation of opiClose for the MAIA machine.*

Description

This is for internal use only. Use `opiClose()` with these Arguments and you will get the Value back.

Value

A list with elements

- `err`, which is an error code, NULL for no error
- `fixations`, which is a matrix one row per fixation and four columns:
  - `time` (same as `time_hw` in `opiPresent`)
  - `flags` Bit 0(same as `time_hw` in `opiPresent`)
    - bit 0: 1 if the tracking event is valid (quality>threshold), 0 otherwise (in this case, x and y shall be =0)
    - bit 2: 1 if a stimulus was being presented during the tracking event, 0 otherwise
    - bit 3: 1 if the tracking was enabled during the tracking event, 0 otherwise
    - bits 16..31: ID of the stimulus being presented (if bit 2 == 1)
  - `x` (degrees relative to the centre of the image returned by `opiInitialise` - not the PRL)
  - `y` (as for x)

opiClose_for_O600 *Implementation of opiClose for the O600 machine.*

Description

This is for internal use only. Use `opiClose()` with the same parameters.

Value

Returns `list(err = NULL)`.

opiClose_for_Octopus900 *Implementation of opiClose for the Octopus900 machine.*

Description

This is for internal use only. Use `opiClose()` with the same parameters.

Value

Returns `list(err = NULL)`.

opiClose_for_PhoneHMD *Implementation of opiClose for the PhoneHMD machine.*

Description

This is for internal use only. Use `opiClose()` after `chooseOPI("PhoneHMD")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiClose()`

Examples

```
## Not run:
chooseOpi("PhoneHMD")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiClose()

## End(Not run)
```

opiClose_for_PicoVR *Implementation of opiClose for the PicoVR machine.*

Description

This is for internal use only. Use `opiClose()` after `chooseOPI("PicoVR")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiClose()`

Examples

```
## Not run:
chooseOpi("PicoVR")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiClose()

## End(Not run)
```

`opiClose_for_SimGaussian`
opiClose_for_SimGaussian

Description

Does nothing.

Value

A list with elements:

- `error` Always `FALSE`.
- `msg` A string "Close OK".

`opiClose_for_SimHenson`
opiClose_for_SimHenson

Description

Does nothing.

Value

A list with elements:

- `err` Always `NULL`.

opiClose_for_SimHensonRT *opiClose_for_SimHensonRT*

Description

Does nothing.

Value

A list with elements:

- `err` Always `NULL`.

opiClose_for_SimNo *opiClose_for_SimNo*

Description

Does nothing.

Value

A list with elements:

- `error` Always `FALSE`.
- `msg` A string "Close OK".

opiClose_for_SimYes *opiClose_for_SimYes*

Description

Does nothing.

Value

A list with elements:

- `err` Always `NULL`.

opiInitialise — *Calls opiInitialise_for_MACHINE as appropriate.*

Description

Establishes connection with the device and a Monitor (aka Server) if appropriate. Sends any startup parameters that might be needed by the machine. Specific parameters and return values can be seen in the machine specific versions listed below in the 'See Also'.

Usage

```
opiInitialise(...)

opiInitialize(...)
```

Arguments

...	Parameters specific to each machine as described in the 'See Also' functions.

Value

A list containing at least the following elements:

- `err` `NULL` if no error, otherwise a string describing the error.

See Also

`opiInitialise_for_ImoVifa()`, `opiInitialise_for_PhoneHMD()`, `opiInitialise_for_Display()`, `opiInitialise_for_PicoVR()`, `opiInitialise_for_Octopus900()`, `opiInitialise_for_Compass()`, `opiInitialise_for_SimNo()`, `opiInitialise_for_SimYes()`, `opiInitialise_for_SimHenson()`, `opiInitialise_for_SimHensonRT()`, `opiInitialise_for_SimGaussian()` `opiInitialise_for_Envision()`

opiInitialise_for_Compass — *Implementation of opiInitialise for the Compass machine.*

Description

This is for internal use only. Use `opiInitialise()` with these Arguments and you will get the Value back.

Arguments

ip	IP address on which server is listening as a string
port	Port number on which server is listening
...	Could be used for fake compass, simulations, etc

Details

Warning: this returns a list, not a single error code.

Value

A list with elements:

- `err` NULL if successful, not otherwise.
- `prl` A pair giving the (x,y) in degrees of the Preferred Retinal Locus detected in the initial alignment.
- `onh` a pair giving the (x,y) in degrees of the ONH as selected by the user.
- `image` raw bytes being the JPEG compressed infra-red image acquired during alignment.

Examples

```
## Not run:
  # Set up the Compass
  chooseOpi("Compass")
  result <- opiInitialize(ip = "192.168.1.7", port = 50008)
  if (is.null(result$err))
    print(result$prl)

## End(Not run)
```

`opiInitialise_for_Display` *Implementation of opiInitialise for the Display machine.*

Description

This is for internal use only. Use `opiInitialise()` after `chooseOPI("Display")` to call this function.

Arguments

`address`	A list containing: • `port` TCP port of the OPI Monitor. • `ip` IP Address of the OPI Monitor.

Details

`port` can take on values in the range `[0, 65535]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

opiInitialise()

Examples

```
## Not run:
chooseOpi("Display")
result <- opiInitialise(address = list(port = 50001, ip = "localhost"))

## End(Not run)
```

---

opiInitialise_for_Envision

Implementation of opiInitialise for the Envision HMP.

Description

This is for internal use only. Use opiInitialise() after chooseOPI("Envision") to call this function.

Arguments

address	A list containing: • port TCP port of the Envision's OPI. • ip IP Address of the Envision's OPI.

Details

port can take on values in the range [0, 65535].

Value

A list containing:

- err NULL if there was no error, a string message if there is an error.

See Also

opiInitialise()

Examples

```
## Not run:
chooseOpi("Envision")
result <- opiInitialise(address = list(port = 50001, ip = "localhost"))

## End(Not run)
```

`opiInitialise_for_ImoVifa`
Implementation of opiInitialise for the ImoVifa machine.

Description

This is for internal use only. Use `opiInitialise()` after `chooseOPI("ImoVifa")` to call this function.

Arguments

`address` A list containing:

- `port` TCP port of the OPI Monitor.
- `ip` IP Address of the OPI Monitor.

Details

`port` can take on values in the range `[0, 65535]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiInitialise()`

Examples

```
## Not run:
chooseOpi("ImoVifa")
result <- opiInitialise(address = list(port = 50001, ip = "localhost"))

## End(Not run)
```

opiInitialise_for_KowaAP7000

Implementation of opiInitialise for the Octopus900 machine.

Description

This is for internal use only. Use `opiInitialise()` with these Arguments and you will get the Value back.

Arguments

`ip`	IP address of AP7000 machine (as string)
`port`	Port number on which AP7000 server is listening

Details

If the chosen OPI implementation is `KowaAP7000`, then you must specify the IP address and port of the AP-7000 server.

Value

`list(err = NULL)`

Examples

```
## Not run:
  # Set up the Kowa AP-7000
  chooseOpi("KowaAP7000")
  opiInitialize(ip="192.168.1.7", port=44965)

## End(Not run)
```

opiInitialise_for_MAIA

Implementation of opiInitialise for the MAIA machine.

Description

This is for internal use only. Use `opiInitialise()` with these Arguments and you will get the Value back.

Arguments

`ip`	IP address on which server is listening as a string
`port`	Port number on which server is listening
`...`	Could be used for fake MAIA, simulations, etc

Details

Establishes socket connection but sends not messages (see `opiSetup_for_MAIA`).

Value

A list with elements:

- `err` NULL if successful, not otherwise.

Examples

```
## Not run:
  # Set up the MAIA
  chooseOpi("MAIA")
  result <- opiInitialize(ip = "192.168.1.7", port = 5555)
  if (is.null(result$err))
    print(result$err)

## End(Not run)
```

---

opiInitialise_for_O600

Implementation of opiInitialise for the O600 machine.

Description

This is for internal use only. Use `opiInitialise()` with these Arguments and you will get the Value back.

Arguments

`ipAddress`	The IP address of the O600 as a string.
`eye`	Either "left" or "right".
`pupilTracking`	TRUE to turn on IR illumination and set pupil black level (which happens at the first stimulus presentation).
`pulsar`	TRUE for pulsar stimulus, FALSE for size III white-on-white.

`eyeControl`	One of
	`* 0 is off`
	`* 1 is eye blink`
	`* 2 is eye blink, forehead rest, fixation control`
	`* 3 is eye blink, forehead rest, fixation control, fast eye movements`

Details

The default background and stimulus setup is to white-on-white perimetry.

Uses port 50000 on the O600.

Examples

```
## Not run:
  # Set up the O600
  chooseOpi("O600")
  opiInitialize(ip="192.168.1.7", eye = "left")

## End(Not run)
```

---

opiInitialise_for_Octopus900

Implementation of opiInitialise for the Octopus900 machine.

Description

This is for internal use only. Use `opiInitialise()` with these Arguments and you will get the Value back.

Arguments

`serverPort`	port number on which server is listening for "Octopus900"
`eyeSuiteSettingsLocation`	dir name containing EyeSuite settings for "Octopus900"
`eye`	eye; "right" or "left" for "Octopus900", "Octopus600"
`gazeFeed`	NA or a folder name for "Octopus900"
`bigWheel`	FALSE (standard machine), TRUE for modified aperture wheel for "Octopus900"
`pres_buzzer`	0 (no buzzer),1, 2, 3 (max volume) for "Octopus900"
`resp_buzzer`	0 (no buzzer),1, 2, 3 (max volume) for "Octopus900"
`zero_dB_is_10000_asb`	Is 0 dB 10000 apostilb (TRUE) or or 4000 (FALSE) for "Octopus900"

Details

If the chosen OPI implementation is `Octopus900`, then you must specify a directory and the eye to be tested.

`serverPort` is the TCP/IP port on which the server is listening (on localhost).

`eyeSuiteSettingsLocation` is the folder name containing the EyeSuite setting files, and should include the trailing slash.

`eye` must be either "left" or "right".

`gazeFeed` is the name of an existing folder into which the video frames of eye tracker are recorded. Set to `NA` for no recording.

`bigWheel` is `FALSE` for a standard Octopus 900 machine. Some research machines are fitted with an alternate aperture wheel that has 24 sizes, which are accessed with `bigWheel` is `TRUE`. The mapping from size to 'hole on wheel' is hard coded; see code for details.

If `pres_buzzer` is greater than zero, a buzzer will sound with each stimuli presented.

If `resp_buzzer` is greater than zero, a buzzer will sound with each button press (response). The volume can be one of 0 (no buzzer), 1, 2, or 3 (max volume). If both buzzers are more than zero, the maximum of the two will be used as the volume.

If `zero_dB_is_10000_asb` is `TRUE` then 0 dB is taken as 10000 apostilbs, otherwise 0 dB is taken as 4000 apostilbs.

Value

A list containing `err` which is

- NULL if successful
- 1 if Octopus900 is already initialised by a previous call to `opiInitialize`
- 2 if some error occurred that prevented initialisation.

Examples

```
## Not run:
  chooseOpi("Octopus900")
  res <- opiInitialize(serverPort = 50001,
                       eyeSuiteSettingsLocation = "C:/ProgramData/Haag-Streit/EyeSuite/",
                        eye = "", gazeFeed = "", bigWheel = FALSE,
                        pres_buzzer = 0, resp_buzzer = 0, zero_dB_is_10000_asb = TRUE)
  if (!is.null(res$err))
      stop("opiInitialize failed")

## End(Not run)
```

---

opiInitialise_for_PhoneHMD

Implementation of opiInitialise for the PhoneHMD machine.

Description

This is for internal use only. Use `opiInitialise()` after `chooseOPI("PhoneHMD")` to call this function.

Arguments

`address` A list containing:

- `port` TCP port of the OPI Monitor.
- `ip` IP Address of the OPI Monitor.

Details

`port` can take on values in the range `[0, 65535]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiInitialise()`

Examples

```
## Not run:
chooseOpi("PhoneHMD")
result <- opiInitialise(address = list(port = 50001, ip = "localhost"))

## End(Not run)
```

---

opiInitialise_for_PicoVR

Implementation of opiInitialise for the PicoVR machine.

Description

This is for internal use only. Use `opiInitialise()` after `chooseOPI("PicoVR")` to call this function.

Arguments

`address`	A list containing: • `port` TCP port of the OPI Monitor. • `ip` IP Address of the OPI Monitor.

Details

`port` can take on values in the range `[0, 65535]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiInitialise()`

Examples

```
## Not run:
chooseOpi("PicoVR")
result <- opiInitialise(address = list(port = 50001, ip = "localhost"))

## End(Not run)
```

---

opiInitialise_for_SimGaussian *opiInitialize_for_SimGaussian*

Description

Simulates responses using a Frequency of Seeing (FoS) curve.

The FoS is modelled as a cumulative Gaussian function with standard deviation equal to `sd` as provided and the mean as the true threshold given as `tt` opiPresent. All values are in dB relative to `maxStim`.

This is for internal use only, use `opiInitialize()`.

Arguments

`sd`	Standard deviation of Cumulative Gaussian.
`maxStim`	The maximum stimuls value (0 dB) in cd/m^2.
`...`	Any other parameters you like, they are ignored.

Value

A list with elements:

- `err` NULL if initialised, a message otherwise

Examples

```
    # Set up a simple simulation for white-on-white perimetry
chooseOpi("SimGaussian")
res <- opiInitialize(sd = 2.5)
if (!is.null(res$err))
  stop(paste("opiInitialize() failed:", res$msg))
```

opiInitialise_for_SimHenson

opiInitialise_for_SimHenson

Description

Simulates responses using a Frequency of Seeing (FoS) curve.

For internal use only, use `opiInitialize()`.

The FoS is modelled as a cumulative Gaussian function over dB with standard deviation equal to `min(cap, exp( A * t + B))`, where `t` is the threshold/mean of the FoS in dB. All values are in dB relative to `maxStim`.

Arguments

`type`	A single character that is: • `N` for using the A and B values from the Normals in Henson et al (2000) • `G` for using the A and B values from the Glaucomas in Henson et al (2000) • `C` for using the A and B values from the Combined in Henson et al (2000) • `X` to specify your own A and B values as parameters
`A`	Coefficient of `t` in the formula (ignored if `type != 'X'`).
`B`	Addend of `t` in the formula (ignored if `type != 'X'`).
`cap`	Maximum dB value for the stdev of the FoS curve.
`maxStim`	The maximum stimulus value (0 dB) in cd/m^2.
`response_time`	A function of that is called to generate a response time for each presentation
`...`	Any other parameters you like, they are ignored.

Value

A list with elements:

- `err` NULL if initialised, msg otherwise

Examples

```
    # Set up a simple simulation for white-on-white perimetry
chooseOpi("SimHenson")
res <- opiInitialize(type = "C", cap = 6)
if (!is.null(res$err))
  stop(paste("opiInitialize() failed:", res$err))
```

opiInitialise_for_SimHensonRT

opiInitialise_for_SimHensonRT

Description

Simulates responses using a Frequency of Seeing (FoS) curve.

For internal use only, use `opiInitialize()`.

The FoS is modelled as a cumulative Gaussian function over dB with standard deviation equal to `min(cap, exp( A * t + B))`, where `t` is the threshold/mean of the FoS in dB. All values are in dB relative to `maxStim`.

Arguments

`type`	A single character that is: • `N` for using the A and B values from the Normals in Henson et al (2000) • `G` for using the A and B values from the Glaucomas in Henson et al (2000) • `C` for using the A and B values from the Combined in Henson et al (2000) • `X` to specify your own A and B values as parameters
`A`	Coefficient of `t` in the formula (ignored if `type != 'X'`).
`B`	Addend of `t` in the formula (ignored if `type != 'X'`).
`cap`	Maximum dB value for the stdev of the FoS curve.
`maxStim`	The maximum stimulus value (0 dB) in cd/m^2.
`rtData`	A data.frame with colnames == "Rt", "Dist", "Person" (or NULL for default).
`rtFP`	A response time for false positives ??? for "SimHensonRT"
`...`	Any other parameters you like, they are ignored.

Details

If the chosen OPI implementation is `SimHensonRT`, then the first six parameters are as in `SimHenson`, and `rtData` is a data frame with at least 2 columns: `"Rt"`, response time; and `"Dist"`, signifying that distance between assumed threshold and stimulus value in your units.

This package contains `RtSigmaUnits` or `RtDbUnits` that can be loaded with the commands `data(RtSigmaUnits)` or `data(RtDbUnits)`, and are suitable to pass as values for `rtData`.

`rtFp` gives the vector of values in milliseconds from which a response time for a false positive response is randomly sampled.

Value

A list with elements:

- `err` NULL if initialised, string msg otherwise

Examples

```
# Set up a simple simulation for white-on-white perimetry
# and display the stimuli in a plot region and simulate response times
chooseOpi("SimHensonRT")
data(RtSigmaUnits)
oi <- opiInitialize(type="C", cap=6, display=NA, rtData=RtSigmaUnits, rtFP=1:100)
if (!is.null(oi$err))
  stop("opiInitialize failed")

    # Set up a simple simulation for white-on-white perimetry
chooseOpi("SimHenson")
res <- opiInitialize(type = "C", cap = 6)
if (!is.null(res$err))
  stop(paste("opiInitialize() failed:", res$err))
```

`opiInitialise_for_SimNo`

opiInitialise_for_SimNo

Description

Does nothing.

Arguments

...	Any object you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

`opiInitialise_for_SimYes`

opiInitialise_for_SimYes

Description

Does nothing.

Arguments

...	Any object you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

---

opiKineticStimulus *For backwards compatibility. Used by Octopus900 and KowaAP7000.*

Description

For backwards compatibility. Used by Octopus900 and KowaAP7000.

Usage

```
opiKineticStimulus()
```

---

opiPresent *Calls opiPresent_for_MACHINE as appropriate.*

Description

Specific parameters and return values can be seen in the machine specific versions listed below in the 'See Also'.

Usage

```
opiPresent(stim, ...)
```

Arguments

`stim`	A stimulus object or list as described for each machine in the 'See Also' methods.
`...`	Other arguments that might be needed by each machine in the 'See Also' methods.

Value

Each implementation should(!) return a list with at least the following elements:

- `err` `NULL` if no error, otherwise a string describing the error.
- `seen` `TRUE` if stimulus seen, `FALSE` otherwise
- `time` Response time from onset of stimulus in milliseconds.

See Also

opiPresent_for_Compass(), opiPresent_for_Octopus900(), opiPresent_for_ImoVifa(), opiPresent_for_PhoneHM
opiPresent_for_Display(), opiPresent_for_PicoVR(), opiPresent_for_SimNo(), opiPresent_for_SimYes(),
opiPresent_for_SimHenson(), opiPresent_for_SimHensonRT(), opiPresent_for_SimGaussian(),
opiPresent_for_Envision(),

---

opiPresent_for_Compass

Implementation of opiPresent for the Compass machine.

Description

This is for internal use only. Use opiSetup() with these Arguments and you will get the Value back.

Arguments

stim	A list of stimulus parameters (see Details).
nextStim	Unused - included for compliance with OPI standard.

Details

If the chosen OPI implementation is Compass, then nextStim is ignored. Note that the dB level is rounded to the nearest integer.

If tracking is on, then this will block until the tracking is obtained, and the stimulus presented.

stim is a list containing some or all of the following elements:

- x, x-coordinate in degrees (floating point) (range $[-30,30]$).
- y, y-coordinate in degrees (floating point) (range $[-30,30]$).
- level is luminance in cd/m^2, and is rounded to the nearest whole dB for display (range 0 to 50). 0dB is 10000aps.
- responseWindow is in milliseconds (range 0 to 2680).

Stimulus duration is assumed to be 200ms, and size is assumed to be Goldmann III (0.43 degrees diameter), color is assumed to be white. These cannot be changed.

Value

A list containing:

- err 0 all clear, >= 1 some error codes (eg cannot track, etc) (integer)
- seen FALSE for not seen, TRUE for seen (button pressed in response window)
- time Response time in ms (integer) since stimulus onset, -1 for not seen
- time_rec Time since epoch when command was received at Compass (integer ms)
- time_pres Time since epoch that stimulus was presented (integer ms)

- num_track_events Number of tracking events that occurred during presentation (integer)
- num_motor_fails Number of times motor could not follow fixation movement during presentation (integer)
- pupil_diam Pupil diameter in mm (float)
- loc_x Pixels integer, location in image of presentation (integer)
- loc_y Pixels integer, location in image of presentation (integer)

Examples

```
## Not run:
  # Set up the Compass
  chooseOpi("Compass")
  result <- opiInitialize(ip = "192.168.1.7", port = 44965)
  if (!is.null(result$err))
    stop("Initialisation failed")

   #' @param x X location of stim in degrees
   #' @param y Y location of stim in degrees
   #' @param size If 3, Goldmann III, else V
   #' @param db Value in dB
   #' @return stim object ready for opiPresent
   makeStim <- function(x, y, size, db) {
       s <- list(x = x, y = y, level = dbTocd(db, 10000 / pi),
           size = ifelse(size == 3, 0.43, 1.77),
           duration = 200, responseWindow = 1500)
       class(s) <- "opiStaticStimulus"
       return(s)
   }

   result <- opiPresent(makeStim(9, 9, 3, 10))

## End(Not run)
```

---

opiPresent_for_Display

Implementation of opiPresent for the Display machine.

Description

This is for internal use only. Use opiPresent() after chooseOPI("Display") to call this function.

Arguments

stim — A list containing:

- lum List of stimuli luminances (cd/m^2).
- stim.length The number of elements in this stimuli.

- `color1` List of stimulus colors for FLAT shapes and patterns.
- `sx` List of diameters along major axis of ellipse (degrees).
- `sy` List of diameters along minor axis of ellipse (degrees).
- `eye` The eye for which to apply the settings.
- `t` List of stimuli presentation times (ms). If 0, then the next stim list element will be shown simultaneously.
- `w` Time to wait for response including presentation time (ms).
- `x` List of x co-ordinates of stimuli (degrees).
- `y` List of y co-ordinates of stimuli (degrees).
- `envSdx` (Optional) List of envelope sd in x direction in degrees. Only useful if envType != NONE
- `envSdy` (Optional) List of envelope sd in y direction in degrees. Only useful if envType != NONE
- `envRotation` (Optional) List of envelope rotations in degrees. Only useful if envType != NONE
- `type` (Optional) Stimulus type. Values include FLAT, SINE, CHECKERBOARD, SQUARESINE, G1, G2, G3, IMAGE
- `frequency` (Optional) List of frequencies (in cycles per degrees) for generation of spatial patterns. Only useful if type != FLAT
- `color2` (Optional) List of second colors for non-FLAT shapes
- `fullFoV` (Optional) If !0 fullFoV scales image to full field of view and sx/sy are ignored.
- `phase` (Optional) List of phases (in degrees) for generation of spatial patterns. Only useful if type != FLAT
- `imageFilename` (Optional) If type == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use
- `shape` (Optional) Stimulus shape. Values include CROSS, TRIANGLE, CIRCLE, SQUARE, OPTOTYPE.
- `rotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if sx != sy specified.
- `texRotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if type != FLAT
- `defocus` (Optional) List of defocus values in Diopters for stimulus post-processing.
- `envType` (Optional) List of envelope types to apply to the stims). Only useful if type != FLAT
- `contrast` (Optional) List of stimulus contrasts (from 0 to 1). Only useful if type != FLAT.
- `optotype` (Optional) If shape == OPTOTYPE, the letter A to Z to use

`...` Parameters for other opiPresent implementations that are ignored here.

Details

Elements in `lum` can take on values in the range `[0.0, 1.0E10]`.

`stim.length` can take on values in the range `[1, 2147483647]`.

Elements in `color1` can take on values in the range `[0.0, 1.0]`.

Elements in `sx` can take on values in the range `[0.0, 180.0]`.

Elements in `sy` can take on values in the range `[0.0, 180.0]`.

Elements in `eye` can take on values in the set `{"left", "right", "both", "none"}`.

Elements in `t` can take on values in the range `[0.0, 1.0E10]`.

`w` can take on values in the range `[0.0, 1.0E10]`.

Elements in `x` can take on values in the range `[-90.0, 90.0]`.

Elements in `y` can take on values in the range `[-90.0, 90.0]`.

Elements in `envSdx` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envSdy` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envRotation` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `type` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

Elements in `frequency` can take on values in the range `[0.0, 300.0]`.

Elements in `color2` can take on values in the range `[0.0, 1.0]`.

Elements in `fullFoV` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `phase` can take on values in the range `[0.0, 1.0E10]`.

Elements in `shape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

Elements in `rotation` can take on values in the range `[0.0, 360.0]`.

Elements in `texRotation` can take on values in the range `[0.0, 360.0]`.

Elements in `defocus` can take on values in the range `[0.0, 1.0E10]`.

Elements in `envType` can take on values in the set `{"none", "square", "circle", "gaussian"}`.

Elements in `contrast` can take on values in the range `[0.0, 1.0]`.

Elements in `optotype` can take on values in the set `{"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"}`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.
- `time` Response time from stimulus onset if button pressed (ms).
- `seen` '1' if seen, '0' if not.

See Also

`opiPresent()`

Examples

```
## Not run:
chooseOpi("Display")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiPresent(stim = list(lum = list(300.0), stim.length = 1, color1 = list(list(1.0,
                  1.0, 1.0)), sx = list(1.72), sy = list(1.72),
             eye = list("LEFT"), t = list(200.0), w = 1500.0, x = list(0.0), y = list(0.0)))

## End(Not run)
```

`opiPresent_for_Envision`

Implementation of opiPresent for the Envision HMP.

Description

This is for internal use only. Use `opiPresent()` after `chooseOPI("Envision")` to call this function.

Arguments

`stim`	A list containing: • `eye` Eye where to present the stimulus • `x` x-coordinate • `y` y-coordinate • `sx` Size in the x-axis (diameter for circles). • `sy` (Optional) Size in the y-axis (diameter for circles). Defaults to sx • `level` Stimulus level • `theta` (Optional) Stimulus rotation from 0 to 360 degrees. Defaults to 0 • `t` (Optional) Stimulus presentation time. Defaults to 200 ms • `window` (Optional) Response window. Defaults to 1500 ms
`...`	Parameters for other opiPresent implementations that are ignored here.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error. If there was no error the the following fields are also returned.
- `stamp` Time stamp of stimulus onset.
- `time` Response time from stimulus onset if button pressed (ms).
- `seen` `TRUE` if seen, `FALSE` if not.
- `lpupil`, `rpupil` Fixation positions (`x`, `y`), and diameter (`d`) for both left (`lpupil`) and right pupils (`rpupil`).

See Also

opiPresent()

Examples

```
## Not run:
chooseOpi("Envision")
opiInitialise(address = list(port = 50001, ip = "localhost"))
result <- opiPresent(stim = list(eye = "os", x = 5, y = 4, sx = 0.43, level = 0))

## End(Not run)
```

---

opiPresent_for_ImoVifa

Implementation of opiPresent for the ImoVifa machine.

Description

This is for internal use only. Use opiPresent() after chooseOPI("ImoVifa") to call this function.

Arguments

stim — A list containing:

- lum List of stimuli luminances (cd/m^2).
- stim.length The number of elements in this stimuli.
- color1 List of stimulus colors for FLAT shapes and patterns.
- sx List of diameters along major axis of ellipse (degrees).
- sy List of diameters along minor axis of ellipse (degrees).
- eye The eye for which to apply the settings.
- t List of stimuli presentation times (ms). If 0, then the next stim list element will be shown simultaneously.
- w Time to wait for response including presentation time (ms).
- x List of x co-ordinates of stimuli (degrees).
- y List of y co-ordinates of stimuli (degrees).
- envSdx (Optional) List of envelope sd in x direction in degrees. Only useful if envType != NONE
- envSdy (Optional) List of envelope sd in y direction in degrees. Only useful if envType != NONE
- envRotation (Optional) List of envelope rotations in degrees. Only useful if envType != NONE
- type (Optional) Stimulus type. Values include FLAT, SINE, CHECKERBOARD, SQUARESINE, G1, G2, G3, IMAGE
- frequency (Optional) List of frequencies (in cycles per degrees) for generation of spatial patterns. Only useful if type != FLAT

- `color2` (Optional) List of second colors for non-FLAT shapes
- `fullFoV` (Optional) If !0 fullFoV scales image to full field of view and sx/sy are ignored.
- `phase` (Optional) List of phases (in degrees) for generation of spatial patterns. Only useful if type != FLAT
- `imageFilename` (Optional) If type == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use
- `shape` (Optional) Stimulus shape. Values include CROSS, TRIANGLE, CIRCLE, SQUARE, OPTOTYPE.
- `rotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if sx != sy specified.
- `texRotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if type != FLAT
- `defocus` (Optional) List of defocus values in Diopters for stimulus post-processing.
- `envType` (Optional) List of envelope types to apply to the stims). Only useful if type != FLAT
- `contrast` (Optional) List of stimulus contrasts (from 0 to 1). Only useful if type != FLAT.
- `optotype` (Optional) If shape == OPTOTYPE, the letter A to Z to use

`...` Parameters for other opiPresent implementations that are ignored here.

Details

Elements in `lum` can take on values in the range `[0.0, 1.0E10]`.

`stim.length` can take on values in the range `[1, 2147483647]`.

Elements in `color1` can take on values in the range `[0.0, 1.0]`.

Elements in `sx` can take on values in the range `[0.0, 180.0]`.

Elements in `sy` can take on values in the range `[0.0, 180.0]`.

Elements in `eye` can take on values in the set `{"left", "right", "both", "none"}`.

Elements in `t` can take on values in the range `[0.0, 1.0E10]`.

`w` can take on values in the range `[0.0, 1.0E10]`.

Elements in `x` can take on values in the range `[-90.0, 90.0]`.

Elements in `y` can take on values in the range `[-90.0, 90.0]`.

Elements in `envSdx` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envSdy` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envRotation` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `type` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

Elements in `frequency` can take on values in the range `[0.0, 300.0]`.

Elements in `color2` can take on values in the range `[0.0, 1.0]`.

Elements in `fullFoV` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in phase can take on values in the range [0.0, 1.0E10].

Elements in shape can take on values in the set {"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}.

Elements in rotation can take on values in the range [0.0, 360.0].

Elements in texRotation can take on values in the range [0.0, 360.0].

Elements in defocus can take on values in the range [0.0, 1.0E10].

Elements in envType can take on values in the set {"none", "square", "circle", "gaussian"}.

Elements in contrast can take on values in the range [0.0, 1.0].

Elements in optotype can take on values in the set {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"}.

Value

A list containing:

- err NULL if there was no error, a string message if there is an error.
- eyedStart Diameter of pupil at stimulus onset (pixels).
- eyexEnd x co-ordinate of pupil at button press (pixels from image centre. Image is 640x480, left < 0). Note that for multi-part stimuli (t=0), the eye taken is the *last* eye in the list of components. Not valid for not-seen response.
- eyeyEnd y co-ordinate (pixels). See eyexEnd for more details. (up > 0). Not valid for not-seen response.
- eyedEnd Diameter of pupil at button press or response window expiry (pixels). Not valid for not-seen response.
- eyexStart x co-ordinates of pupil at stimulus onset (pixels from image centre. Image is 640x480, left < 0). For a multi-part stimulus (t=0), the eye taken is the *first* eye in the list of components.
- time Response time from stimulus onset if button pressed (ms).
- seen '1' if seen, '0' if not.
- eyeyStart y co-ordinates (pixels from image centre). See eyexStart for more details. (up > 0)

See Also

opiPresent()

Examples

```
## Not run:
chooseOpi("ImoVifa")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiPresent(stim = list(lum = list(300.0), stim.length = 1, color1 = list(list(1.0,
                  1.0, 1.0)), sx = list(1.72), sy = list(1.72),
```

```
              eye = list("LEFT"), t = list(200.0), w = 1500.0, x = list(0.0), y = list(0.0)))

## End(Not run)
```

---

opiPresent_for_KowaAP7000

Implementation of opiPresent for the KowaAP7000 machine.

Description

This is for internal use only. Use `opiPresent()` with the same arguments and the class of `stim` as one of `opiStaticStimulus`, `opiTemporalStimulus`, or `opiKineticStimulus`.

Arguments

`stim`	Stimulus to present (a list, see `kowa.*` for details).
`nextStim`	The stimulus to present after stim (it is not presented, but projector can move to it during response window)

Value

See kowa.presentStatic, kowa.presentTemporal, or kowa.presentKinetic.

See Also

kowa.presentStatic, kowa.presentTemporal, kowa.presentKinetic

Examples

```
## Not run:
  chooseOpi("KowaAP7000")
  if (!is.null(opiInitialize()$err))
      stop("opiInitialize failed")
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500, checkFixationOK=NULL)
  class(s) <- "opiStaticStimulus"
  print(opiPresent(s, NULL))

## End(Not run)
```

`opiPresent_for_MAIA` *Implementation of opiPresent for the MAIA machine.*

Description

This is for internal use only. Use `opiSetup()` with these Arguments and you will get the Value back.

Arguments

`stim`	A list of stimulus parameters (see Details).
`nextStim`	Unused - included for compliance with OPI standard.

Details

If the chosen OPI implementation is `MAIA`, then `nextStim` is ignored. Note that the dB level is rounded to the nearest integer.

If tracking is on, then this will block until the tracking is obtained, and the stimulus presented.

`stim` is a list containing some or all of the following elements:

- `x`, x-coordinate in degrees (floating point) (range $[-30,30]$).
- `y`, y-coordinate in degrees (floating point) (range $[-30,30]$).
- `level` is luminance in cd/m^2, and is rounded to the nearest whole dB for display (range 0 to 50). 0dB is 10000aps.
- `responseWindow` is in milliseconds (range 0 to 2680).

Stimulus duration is assumed to be 200ms, and size is assumed to be Goldmann III (0.43 degrees diameter), color is assumed to be white. These cannot be changed.

Value

A list containing:

- `err` 0 all clear, >= 1 some error codes (eg cannot track, etc) (integer)
- `seen` `FALSE` for not seen, `TRUE` for seen (button pressed in response window)
- `time` Response time in ms (integer) since stimulus onset, -1 for not seen
- `time_rec` Time since epoch when command was received at MAIA (integer ms)
- `time_pres` Time since epoch that stimulus was presented (integer ms)
- `num_track_events` Number of tracking events that occurred during presentation (integer)
- `num_motor_fails` Number of times motor could not follow fixation movement during presentation (integer)
- `pupil_diam` Pupil diameter in mm (float)
- `loc_x` X location in image of presentation (pixels, integer)
- `loc_y` Y location in image of presentation (pixels, integer)

Examples

```
## Not run:
  # Set up the MAIA
  chooseOpi("MAIA")
  result <- opiInitialize(ip = "192.168.1.7", port = 44965)
  if (!is.null(result$err))
    stop("Initialisation failed")

   #' @param x X location of stim in degrees
   #' @param y Y location of stim in degrees
   #' @param size If 3, Goldmann III, else V
   #' @param db Value in dB
   #' @return stim object ready for opiPresent
   makeStim <- function(x, y, size, db) {
       s <- list(x = x, y = y, level = dbTocd(db, 10000 / pi),
           size = ifelse(size == 3, 0.43, 1.77),
           duration = 200, responseWindow = 1500)
       class(s) <- "opiStaticStimulus"
       return(s)
   }

   result <- opiPresent(makeStim(9, 9, 3, 10))

## End(Not run)
```

`opiPresent_for_O600` *Implementation of opiPresent for the O600 machine.*

Description

This is for internal use only. Use `opiPresent()` with the same arguments.

Arguments

`stim` Stimulus to present which is a list with the following elements:

- `x` positionX (in 1/10deg)
- `y` positionY (in 1/10deg)
- `level` dLog (intensity; 0dB is 4000 apostilbs) (in 1/10 dB)
- `duration` Stimulus presentation duration in ms, for W/W 100ms, for pulsar 500ms
- `responseWindow` Maximal allowed reaction time in ms, >=500ms and <4s
- `sound` # Bit 1 = sound for patient response button ON; Bit2=1 sound for fixation lost ON

`nextStim` The stimulus to present after stim (it is not presented, but projector can move to it during response window)

Value

A list containing

- `err` String message or NULL for no error.
- `seen` 1 if seen, 0 otherwise.
- `time` Reaction time (if seen).

Examples

```
## Not run:
  chooseOpi("O600")
  if (!is.null(opiInitialize()$err))
      stop("opiInitialize failed")
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500, checkFixationOK=NULL)
  print(opiPresent(s, NULL))

## End(Not run)
```

---

opiPresent_for_Octopus900

Implementation of opiPresent for the Octopus090 machine.

Description

This is for internal use only. Use `opiPresent()` with the same arguments and the class of `stim` as one of `opiStaticStimulus`, `opiTemporalStimulus`, or `opiKineticStimulus`.

Arguments

`stim`	Stimulus to present (a list, see `octo900.present*` for details).
`nextStim`	The stimulus to present after stim (it is not presented, but projector can move to it during response window)
`F310`	If `F310` is `FALSE`, response is taken from O900 button. If `F310` is `TRUE`, response is taken from external controller for (static stimuli only).

See Also

octo900.presentStatic, octo900.presentTemporal, octo900.presentKinetic

Examples

```
## Not run:
  chooseOpi("Octopus900")
  if (!is.null(opiInitialize()$err))
      stop("opiInitialize failed")
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500, checkFixationOK=NULL)
  class(s) <- "opiStaticStimulus"
  print(opiPresent(s, NULL))

## End(Not run)
```

`opiPresent_for_PhoneHMD`

Implementation of opiPresent for the PhoneHMD machine.

Description

This is for internal use only. Use `opiPresent()` after `chooseOPI("PhoneHMD")` to call this function.

Arguments

`stim` A list containing:

- `lum` List of stimuli luminances (cd/m^2).
- `stim.length` The number of elements in this stimuli.
- `color1` List of stimulus colors for FLAT shapes and patterns.
- `sx` List of diameters along major axis of ellipse (degrees).
- `sy` List of diameters along minor axis of ellipse (degrees).
- `eye` The eye for which to apply the settings.
- `t` List of stimuli presentation times (ms). If 0, then the next stim list element will be shown simultaneously.
- `w` Time to wait for response including presentation time (ms).
- `x` List of x co-ordinates of stimuli (degrees).
- `y` List of y co-ordinates of stimuli (degrees).
- `envSdx` (Optional) List of envelope sd in x direction in degrees. Only useful if envType != NONE
- `envSdy` (Optional) List of envelope sd in y direction in degrees. Only useful if envType != NONE
- `envRotation` (Optional) List of envelope rotations in degrees. Only useful if envType != NONE
- `type` (Optional) Stimulus type. Values include FLAT, SINE, CHECKERBOARD, SQUARESINE, G1, G2, G3, IMAGE

- `frequency` (Optional) List of frequencies (in cycles per degrees) for generation of spatial patterns. Only useful if type != FLAT
- `color2` (Optional) List of second colors for non-FLAT shapes
- `fullFoV` (Optional) If !0 fullFoV scales image to full field of view and sx/sy are ignored.
- `phase` (Optional) List of phases (in degrees) for generation of spatial patterns. Only useful if type != FLAT
- `imageFilename` (Optional) If type == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use
- `shape` (Optional) Stimulus shape. Values include CROSS, TRIANGLE, CIRCLE, SQUARE, OPTOTYPE.
- `rotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if sx != sy specified.
- `texRotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if type != FLAT
- `defocus` (Optional) List of defocus values in Diopters for stimulus post-processing.
- `envType` (Optional) List of envelope types to apply to the stims). Only useful if type != FLAT
- `contrast` (Optional) List of stimulus contrasts (from 0 to 1). Only useful if type != FLAT.
- `optotype` (Optional) If shape == OPTOTYPE, the letter A to Z to use

... Parameters for other opiPresent implementations that are ignored here.

Details

Elements in `lum` can take on values in the range `[0.0, 1.0E10]`.

`stim.length` can take on values in the range `[1, 2147483647]`.

Elements in `color1` can take on values in the range `[0.0, 1.0]`.

Elements in `sx` can take on values in the range `[0.0, 180.0]`.

Elements in `sy` can take on values in the range `[0.0, 180.0]`.

Elements in `eye` can take on values in the set `{"left", "right", "both", "none"}`.

Elements in `t` can take on values in the range `[0.0, 1.0E10]`.

`w` can take on values in the range `[0.0, 1.0E10]`.

Elements in `x` can take on values in the range `[-90.0, 90.0]`.

Elements in `y` can take on values in the range `[-90.0, 90.0]`.

Elements in `envSdx` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envSdy` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envRotation` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `type` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

Elements in `frequency` can take on values in the range `[0.0, 300.0]`.

Elements in `color2` can take on values in the range `[0.0, 1.0]`.

Elements in `fullFoV` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `phase` can take on values in the range `[0.0, 1.0E10]`.

Elements in `shape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

Elements in `rotation` can take on values in the range `[0.0, 360.0]`.

Elements in `texRotation` can take on values in the range `[0.0, 360.0]`.

Elements in `defocus` can take on values in the range `[0.0, 1.0E10]`.

Elements in `envType` can take on values in the set `{"none", "square", "circle", "gaussian"}`.

Elements in `contrast` can take on values in the range `[0.0, 1.0]`.

Elements in `optotype` can take on values in the set `{"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"}`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.
- `time` Response time from stimulus onset if button pressed (ms).
- `seen` '1' if seen, '0' if not.

See Also

`opiPresent()`

Examples

```
## Not run:
chooseOpi("PhoneHMD")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiPresent(stim = list(lum = list(300.0), stim.length = 1, color1 = list(list(1.0,
                  1.0, 1.0)), sx = list(1.72), sy = list(1.72),
             eye = list("LEFT"), t = list(200.0), w = 1500.0, x = list(0.0), y = list(0.0)))

## End(Not run)
```

`opiPresent_for_PicoVR` *Implementation of opiPresent for the PicoVR machine.*

Description

This is for internal use only. Use `opiPresent()` after `chooseOPI("PicoVR")` to call this function.

Arguments

`stim` A list containing:

- `lum` List of stimuli luminances (cd/m^2).
- `stim.length` The number of elements in this stimuli.
- `color1` List of stimulus colors for FLAT shapes and patterns.
- `sx` List of diameters along major axis of ellipse (degrees).
- `sy` List of diameters along minor axis of ellipse (degrees).
- `eye` The eye for which to apply the settings.
- `t` List of stimuli presentation times (ms). If 0, then the next stim list element will be shown simultaneously.
- `w` Time to wait for response including presentation time (ms).
- `x` List of x co-ordinates of stimuli (degrees).
- `y` List of y co-ordinates of stimuli (degrees).
- `envSdx` (Optional) List of envelope sd in x direction in degrees. Only useful if envType != NONE
- `envSdy` (Optional) List of envelope sd in y direction in degrees. Only useful if envType != NONE
- `envRotation` (Optional) List of envelope rotations in degrees. Only useful if envType != NONE
- `type` (Optional) Stimulus type. Values include FLAT, SINE, CHECKERBOARD, SQUARESINE, G1, G2, G3, IMAGE
- `frequency` (Optional) List of frequencies (in cycles per degrees) for generation of spatial patterns. Only useful if type != FLAT
- `color2` (Optional) List of second colors for non-FLAT shapes
- `fullFoV` (Optional) If !0 fullFoV scales image to full field of view and sx/sy are ignored.
- `phase` (Optional) List of phases (in degrees) for generation of spatial patterns. Only useful if type != FLAT
- `imageFilename` (Optional) If type == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use
- `shape` (Optional) Stimulus shape. Values include CROSS, TRIANGLE, CIRCLE, SQUARE, OPTOTYPE.
- `rotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if sx != sy specified.
- `texRotation` (Optional) List of angles of rotation of stimuli (degrees). Only useful if type != FLAT

- `defocus` (Optional) List of defocus values in Diopters for stimulus post-processing.
- `envType` (Optional) List of envelope types to apply to the stims). Only useful if type != FLAT
- `contrast` (Optional) List of stimulus contrasts (from 0 to 1). Only useful if type != FLAT.
- `optotype` (Optional) If shape == OPTOTYPE, the letter A to Z to use

... Parameters for other opiPresent implementations that are ignored here.

Details

Elements in `lum` can take on values in the range `[0.0, 1.0E10]`.

`stim.length` can take on values in the range `[1, 2147483647]`.

Elements in `color1` can take on values in the range `[0.0, 1.0]`.

Elements in `sx` can take on values in the range `[0.0, 180.0]`.

Elements in `sy` can take on values in the range `[0.0, 180.0]`.

Elements in `eye` can take on values in the set `{"left", "right", "both", "none"}`.

Elements in `t` can take on values in the range `[0.0, 1.0E10]`.

`w` can take on values in the range `[0.0, 1.0E10]`.

Elements in `x` can take on values in the range `[-90.0, 90.0]`.

Elements in `y` can take on values in the range `[-90.0, 90.0]`.

Elements in `envSdx` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envSdy` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `envRotation` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `type` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

Elements in `frequency` can take on values in the range `[0.0, 300.0]`.

Elements in `color2` can take on values in the range `[0.0, 1.0]`.

Elements in `fullFoV` can take on values in the range `[-1.0E10, 1.0E10]`.

Elements in `phase` can take on values in the range `[0.0, 1.0E10]`.

Elements in `shape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

Elements in `rotation` can take on values in the range `[0.0, 360.0]`.

Elements in `texRotation` can take on values in the range `[0.0, 360.0]`.

Elements in `defocus` can take on values in the range `[0.0, 1.0E10]`.

Elements in `envType` can take on values in the set `{"none", "square", "circle", "gaussian"}`.

Elements in `contrast` can take on values in the range `[0.0, 1.0]`.

Elements in `optotype` can take on values in the set `{"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"}`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.
- `time` Response time from stimulus onset if button pressed (ms).
- `seen` '1' if seen, '0' if not.

See Also

`opiPresent()`

Examples

```
## Not run:
chooseOpi("PicoVR")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiPresent(stim = list(lum = list(300.0), stim.length = 1, color1 = list(list(1.0,
                  1.0, 1.0)), sx = list(1.72), sy = list(1.72),
             eye = list("LEFT"), t = list(200.0), w = 1500.0, x = list(0.0), y = list(0.0)))

## End(Not run)
```

---

opiPresent_for_SimGaussian

opiPresent_for_SimGaussian

Description

Determine the response to a stimuli by sampling from a cumulative Gaussian Frequency-of-Seeing (FoS) curve (also known as the psychometric function).

The FoS has formula

$$\text{fpr} + (1 - \text{fpr} - \text{fnr})(1 - \text{pnorm}(x, \text{tt}, \text{sd})$$

where x is the stimulus value in dB, and `sd` is set by `opiInitialize` and `tt`, `fpr` and `fnr` are parameters.

This is for internal use only, use `opiPresent()`.

Arguments

`stim`	A list that contains at least: • `level` which is the stim value in cd/m^2.
`fpr`	false positive rate for the FoS curve (range 0..1).
`fnr`	false negative rate for the FoS curve (range 0..1).
`tt`	mean of the assumed FoS curve in dB.
`...`	Any other parameters you like, they are ignored.

Value

A list with elements:

- `err` `NULL` if no error, a string message otherwise.
- `seen` `TRUE` or `FALSE`.
- `time` Always `NA`.

Examples

```
    # Stimulus is Size III white-on-white as in the HFA
chooseOpi("SimGaussian")
res <- opiInitialize(sd = 1.6)
if (!is.null(res$err))
  stop(paste("opiInitialize() failed:", res$err))

result <- opiPresent(stim = list(level = dbTocd(20)), tt = 30, fpr = 0.15, fnr = 0.01)
print(paste("Seen:", result$seen, quote = FALSE))

res <- opiClose()
if (!is.null(res$err))
  warning(paste("opiClose() failed:", res$err))
```

---

opiPresent_for_SimHenson

opiPresent_for_SimHenson

Description

Determine the response to a stimuli by sampling from a cumulative Gaussian Frequency-of-Seeing (FoS) curve (also known as the psychometric function).

For internal use only, use `opiPresent()`.

Arguments

`stim`	A list that contains at least: • `level` which is the stim value in cd/m^2.
`fpr`	false positive rate for the FoS curve (range 0..1).
`fnr`	false negative rate for the FoS curve (range 0..1).
`tt`	mean of the assumed FoS curve in dB.
`...`	Any other parameters you like, they are ignored.

Details

The FoS formula is

$$\text{fpr} + (1 - \text{fpr} - \text{fnr})(1 - \text{pnorm}(x, \text{tt}, \text{pxVar})$$

where `x` is the stimulus value in dB, and `pxVar` is

$$\min(\text{cap}, e^{A \times \text{tt} + B}).$$

The ceiling `cap` is set with the call to `opiInitialize`, and `A` and `B` are from Table 1 in Henson et al (2000), also set in the call to `opiInitialise` using the `type` parameter.

Value

A list with elements:

- `err` `NULL` if no error, a string message otherwise.
- `seen` `TRUE` or `FALSE`.
- `time` Always `NA`.

Examples

```
    # Stimulus is Size III white-on-white as in the HFA
chooseOpi("SimHenson")
res <- opiInitialize(type = "C", cap = 6)
if (!is.null(res$err))
  stop(paste("opiInitialize() failed:", res$err))

result <- opiPresent(stim = list(level = dbTocd(20)), tt = 30, fpr = 0.15, fnr = 0.01)
print(paste("Seen:", result$seen, quote = FALSE))

res <- opiClose()
if (!is.null(res$err))
  stop(paste("opiClose() failed:", res$err))
```

`opiPresent_for_SimHensonRT`

opiPresent_for_SimHensonRT

Description

Determine the response to a stimuli by sampling from a cumulative Gaussian Frequency-of-Seeing (FoS) curve (also known as the psychometric function).

For internal use only, use `opiPresent()`.

Arguments

`stim`	A list that contains at least: • `level` which is the stim value in cd/m^2.
`fpr`	false positive rate for the FoS curve (range 0..1).
`fnr`	false negative rate for the FoS curve (range 0..1).
`tt`	mean of the assumed FoS curve in dB.
`dist`	The distance of the stimulus level from `tt` in appropriate units (same as `rtData$Dist`).
`...`	Any other parameters you like, they are ignored.

Details

As the response time returned for a false positive is determined separately from a positive response, we first check for a false response. If there is no false response, we use the FoS formula

$$1 - \text{pnorm}(x, \text{tt}, \text{pxVar})$$

where `x` is the stimulus value in dB, and `pxVar` is

$$\min(\text{cap}, e^{A \times \text{tt} + B}).$$

The ceiling `cap` is set with the call to `opiInitialize`, and `A` and `B` are from Table 1 in Henson et al (2000), also set in the call to `opiInitialise` using the `type` parameter.

Thus, this function is the same as for `SimHenson`, but reaction times are determined by sampling from `rtData` as passed to `opiInitialize`. The `dist` parameter is the distance of the stimulus level from the true threshold, and should be in the same units as the `Dist` column of `rtData`. The default is just the straight difference between the stimulus level and the true threshold, but you might want it scaled somehow to match `rtData`.

Value

A list with elements:

- `err` `NULL` if no error, a string message otherwise.
- `seen` `TRUE` or `FALSE`.
- `time` The response time.

Examples

```
    # Stimulus is Size III white-on-white as in the HFA
chooseOpi("SimHensonRT")
data(RtSigmaUnits)
res <- opiInitialize(type = "C", cap = 6, rtData = RtSigmaUnits)
if (!is.null(res$err))
  stop(paste("opiInitialize() failed:", res$err))

dist <- (10 - 30) / min(exp(-0.098 * 30 + 3.62), 6)
result <- opiPresent(stim = list(level = dbTocd(20)), tt = 30, fpr = 0.15, fnr = 0.01, dist=dist)
print(result, quote = FALSE)
```

```
if (!is.null(opiClose()))
  warning("opiClose() failed")
```

---

opiPresent_for_SimNo *opiPresent_for_SimNo*

Description

Always respond 'not seen' to any parameter. No checking is done on the validity of `stim`.

Arguments

`stim`	Anything you like, it is ignored.
`...`	Any parameters you like, they are ignored.

Value

A list with elements:

- `err` Always `NULL`.
- `time` Always `NA`.

---

opiPresent_for_SimYes *opiPresent_for_SimYes*

Description

Always respond 'yes' immediately to any parameter.

Arguments

`stim`	Anything you like, it is ignored.
`...`	Any parameters you like, they are ignored.

Value

A list with elements:

- `err` Always `FALSE`.
- `seen` Always `TRUE`.
- `time` Always `NA`.

opiQueryDevice — *Calls opiQueryDevice_for_MACHINE as appropriate.*

Description

Returns a list that describes the current state of the machine. Specific parameters and return values can be seen in the machine specific versions listed below in the 'See Also'.

Usage

```
opiQueryDevice()
```

Value

A list specific to each machine.

See Also

opiQueryDevice_for_ImoVifa(), opiQueryDevice_for_Compass(), opiQueryDevice_for_Envision(), opiQueryDevice_for_Octopus900(), opiQueryDevice_for_PhoneHMD(), opiQueryDevice_for_Display(), opiQueryDevice_for_PicoVR(), opiQueryDevice_for_SimNo(), opiQueryDevice_for_SimYes(), opiQueryDevice_for_SimHenson(), opiQueryDevice_for_SimHensonRT(), opiQueryDevice_for_SimGaussian()

opiQueryDevice_for_Compass — *Implementation of opiQueryDevice for the Compass machine.*

Description

This is for internal use only. Use opiQueryDevice() with these Arguments and you will get the Value back.

Value

A list containing constants and their values used in the OPI Compass module.

opiQueryDevice_for_Display

Implementation of opiQueryDevice for the Display machine.

Description

This is for internal use only. Use `opiQueryDevice()` after `chooseOPI("Display")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiQueryDevice()`

Examples

```
## Not run:
chooseOpi("Display")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiQueryDevice()

## End(Not run)
```

opiQueryDevice_for_Envision

Implementation of opiQueryDevice for the Envision HMP.

Description

This is for internal use only. Use `opiQueryDevice()` after `chooseOPI("Envision")` to call this function.

Details

`specs` is a list containing:

- `hardware` String with the Hardware name.
- `screens` Whether the hardware has 1 screen (for both eyes) or 2 (1 for each).
- `fov` A list with the field of view for x-axis (`x`) and y-axis (`y`).

- `resolution` A list with the resolution for width (`w`) and height (`h`).
- `ipd` A list with min (`min`) and max (`max`) possible IPDs and the current value (`val`).
- `depth` Display's bit depth.
- `refresh` Display's refresh rate.
- `domain` An array of feasible dB values by the hardware.

`settings` is a list containing:

- `bgeye` The eye where the background is shown ("OD", "OS", "OU").
- `bglum` Background luminance in cd/m2.
- `fixeye` The eye where a fixation target is shown ("OD", "OS", "OU").
- `fixlum` Fixation luminance in cd/m2.
- `fixtarget` Fixation target.
- `fixcx` Fixation x-coordinate.
- `fixcy` Fixation y-coordinate.
- `fixdx` Fixation size in the x-axis (diameter for circles).
- `fixdy` Fixation size in the y-axis (diameter for circles).
- `fixtheta` Fixation rotation in degrees.
- `tracking` Whether active correction is active.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.
- `specs` A list with specifications (see Details) if there was no error.
- `settings` A list with the settings (see Details) if there was no error.

See Also

`opiQueryDevice()`

Examples

```
## Not run:
chooseOpi("Envision")
opiInitialise(address = list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiQueryDevice()

## End(Not run)
```

`opiQueryDevice_for_ImoVifa`
Implementation of opiQueryDevice for the ImoVifa machine.

Description

This is for internal use only. Use `opiQueryDevice()` after `chooseOPI("ImoVifa")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.
- `rightEyex` x co-ordinates of right pupil (pixels from image centre, left < 0)
- `rightEyey` y co-ordinates of right pupil (pixels from image centre, up > 0)
- `leftEyex` x co-ordinates of left pupil (pixels from image centre, left < 0)
- `leftEyey` y co-ordinates of left pupil (pixels from image centre, up > 0)
- `leftEyed` Diameter of left pupil (pixels)
- `rightEyed` Diameter of right pupil (pixels)

See Also

`opiQueryDevice()`

Examples

```
## Not run:
chooseOpi("ImoVifa")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiQueryDevice()

## End(Not run)
```

opiQueryDevice_for_KowaAP7000

Implementation of opiQueryDevice for the Kowa AP7000 machine.

Description

This is for internal use only. Use `opiQueryDevice()` with these Arguments and you will get the Value back.

Details

Prints defined constants in OPI package pertaining to Kowa AP7000.

Value

List containing

- `isSim` = `FALSE`.
- `pupilX`, the x-coordinate of the pupil position in pixels.
- `pupilY`, the y-coordinate of the pupil position in pixels.
- `purkinjeX`, the x-coordinate of the purkinje position in pixels.
- `purkinjeY`, the y-coordinate of the purkinje position in pixels.

opiQueryDevice_for_MAIA

Implementation of opiQueryDevice for the MAIA machine.

Description

This is for internal use only. Use `opiQueryDevice()` with these Arguments and you will get the Value back.

Value

A list containing constants and their values used in the OPI MAIA module.

opiQueryDevice_for_O600

Implementation of opiQueryDevice for the O600 machine.

Description

This is for internal use only. Use `opiQueryDevice()` with these Arguments and you will get the Value back.

Details

Prints defined constants in OPI package pertaining to O600.

Value

Returns a list of 10 items:

```
* \code{answerButton} 0 = not pressed, 1 = pressed
* \code{headSensor} 0 = no forehead detected, 1 = forehead detected
* \code{eyeLidClosureLeft} 0 = eye is open, 1 = eye is closed
* \code{eyeLidClosureRight} 0 = eye is open, 1 = eye is closed
* \code{fixationLostLeft} 1 = eye pos lost, 0 = eye pos ok
* \code{fixationLostRight} 1 = eye pos lost, 0 = eye pos ok
* \code{pupilPositionXLeft} (in px)
* \code{pupilPositionYLeft} (in px)
* \code{pupilPositionXRight} (in px)
* \code{pupilPositionYRight} (in px)
```

opiQueryDevice_for_Octopus900

Implementation of opiQueryDevice for the Octopus900 machine.

Description

This is for internal use only. Use `opiQueryDevice()` with these Arguments and you will get the Value back.

Details

Prints defined constants in OPI package pertaining to Octopus 900.

Value

List containing `isSim = FALSE`.

opiQueryDevice_for_PhoneHMD

Implementation of opiQueryDevice for the PhoneHMD machine.

Description

This is for internal use only. Use `opiQueryDevice()` after `chooseOPI("PhoneHMD")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiQueryDevice()`

Examples

```
## Not run:
chooseOpi("PhoneHMD")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiQueryDevice()

## End(Not run)
```

opiQueryDevice_for_PicoVR

Implementation of opiQueryDevice for the PicoVR machine.

Description

This is for internal use only. Use `opiQueryDevice()` after `chooseOPI("PicoVR")` to call this function.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiQueryDevice()`

Examples

```
## Not run:
chooseOpi("PicoVR")
opiInitialise(list(port = 50001, ip = "localhost"))
opiSetup(list(eye = "BOTH"))
result <- opiQueryDevice()

## End(Not run)
```

---

opiQueryDevice_for_SimGaussian

opiQueryDevice_for_SimGaussian

Description

Returns a simple list.

Value

A list with elements:

- `err` Always `NULL`
- `machine` that is set to `"SimGaussian"`.

---

opiQueryDevice_for_SimHenson

opiQueryDevice_for_SimHenson

Description

Returns name of the machine.

Value

A list with elements:

- `isSim` Always `TRUE`.
- `machine` that is set to `"SimHenson"`.

opiQueryDevice_for_SimHensonRT

opiQueryDevice_for_SimHensonRT

Description

Returns name of the machine.

Value

A list with elements:

- `isSim` Always `TRUE`.
- `machine` that is set to `"SimHensonRT"`.

opiQueryDevice_for_SimNo

opiQueryDevice_for_SimNo

Description

Returns name of the machine.

Value

A list with elements:

- `machine` that is set to `"SimNo"`.
- `isSim` that is set to TRUE.

opiQueryDevice_for_SimYes

opiQueryDevice_for_SimYes

Description

Returns name of the machine.

Value

A list with elements:

- `isSim` Always `TRUE`.
- `machine` that is set to `"SimYes"`.

opiSetBackground *Deprecated. Use* opiSetup().

Description

In older OPIs it set background color and luminance in both eyes. Deprecated for OPI >= v3.0.0 and replaced with opiSetup().

See Also

opiSetup()

opiSetup *Calls opiSetup_for_MACHINE as appropriate.*

Description

Specific parameters and return values can be seen in the machine specific versions listed below in the 'See Also'.

Usage

```
opiSetup(settings)
```

Arguments

settings	A list containing specific settings for a machine.

Value

Each implementation should(!) return a list with at least the following elements:

- err NULL if no error, otherwise a string describing the error.

See Also

opiSetup_for_Compass(), opiSetup_for_Octopus900(), opiSetup_for_ImoVifa(), opiSetup_for_PhoneHMD(), opiSetup_for_Display(), opiSetup_for_PicoVR(), opiSetup_for_SimNo(), opiSetup_for_SimYes(), opiSetup_for_SimHenson(), opiSetup_for_SimHensonRT(), opiSetup_for_SimGaussian(), opiSetup_for_Envision()

opiSetup_for_Compass *Implementation of opiSetup for the Compass machine.*

Description

This is for internal use only. Use `opiSetup()` with these Arguments and you will get the Value back.

Arguments

`settings` is a list that could contain:

- `fixation` `c(x,y,t)` where
  - `x` is one of -20, -6, -3, 0, 3, 6, 20 degrees.
  - `y` is 0 degrees.
  - `t` is 0 for a spot fixation marker at `c(x,y)`, or 1 for a square centred on one of `(-3,0)`, `(0,0)`, `(+3,0)`.
- `tracking_on` `TRUE` for tracking on, `FALSE` for off

Details

Note: tracking will be relative to the PRL established with the fixation marker used at setup (call to OPI-OPEN), so when tracking is on you should use the same fixation location as in the setup.

Value

A list containing `err` which is `NULL` for success, or some string description for fail.

opiSetup_for_Display *Implementation of opiSetup for the Display machine.*

Description

This is for internal use only. Use `opiSetup()` after `chooseOPI("Display")` to call this function.

Arguments

`settings` A list containing:

- `eye` The eye for which to apply the settings.
- `bgImageFilename` (Optional) If present, display the image in the background for eye (scaled to fill fov, bgLum and bgCol ignored)
- `fixShape` (Optional) Fixation target type for eye.
- `fixLum` (Optional) Fixation target luminance for eye.
- `fixType` (Optional) Fixation target texture for eye.
- `fixCx` (Optional) x-coordinate of fixation target (degrees).

- `fixCy` (Optional) y-coordinate of fixation target (degrees).
- `fixCol` (Optional) Fixation target color for eye.
- `bgLum` (Optional) Background luminance for eye (cd/m^2).
- `tracking` (Optional) Whether to correct stimulus location based on eye position.
- `bgCol` (Optional) Background color for eye (rgb).
- `fixSx` (Optional) diameter along major axis of ellipse (degrees). 0 to hide fixation marker.
- `fixSy` (Optional) diameter along minor axis of ellipse (degrees). If not received, then sy = sx.
- `fixRotation` (Optional) Angles of rotation of fixation target (degrees). Only useful if sx != sy specified.
- `fixImageFilename` (Optional) If fixType == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use

Details

`eye` can take on values in the set `{"left", "right", "both", "none"}`.

`fixShape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

`fixLum` can take on values in the range `[0.0, 1.0E10]`.

`fixType` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

`fixCx` can take on values in the range `[-90.0, 90.0]`.

`fixCy` can take on values in the range `[-90.0, 90.0]`.

Elements in `fixCol` can take on values in the range `[0.0, 1.0]`.

`bgLum` can take on values in the range `[0.0, 1.0E10]`.

`tracking` can take on values in the range `[0, 1]`.

Elements in `bgCol` can take on values in the range `[0.0, 1.0]`.

`fixSx` can take on values in the range `[0.0, 1.0E10]`.

`fixSy` can take on values in the range `[0.0, 1.0E10]`.

`fixRotation` can take on values in the range `[0.0, 360.0]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiSetup()`

Examples

```
## Not run:
chooseOpi("Display")
opiInitialise(list(port = 50001, ip = "localhost"))
result <- opiSetup(settings = list(eye = "BOTH"))

## End(Not run)
```

---

opiSetup_for_Envision *Implementation of opiSetup for the Envision HMP.*

Description

This is for internal use only. Use `opiSetup()` after `chooseOPI("Envision")` to call this function.

Arguments

`settings` A list containing:

- `bgeye` The eye where the background is shown ("OD", "OS", "OU"). Default is "OU".
- `bglum` Background luminance in cd/m2. Default is 10.
- `fixeye` The eye where a fixation target is shown ("OD", "OS", "OU"). Default is "OU".
- `fixlum` Fixation luminance in cd/m2. Default is 20.
- `fixtarget` Fixation target. Default is "CROSS".
- `fixcx` Fixation x-coordinate. Default is 0.
- `fixcy` Fixation y-coordinate. Default is 0.
- `fixdx` Fixation size in the x-axis (diameter for circles). Default is 2.
- `fixdy` Fixation size in the y-axis (diameter for circles). Default is 2.
- `fixtheta` Fixation rotation in degrees. Default is 0.
- `tracking` Whether active correction is active. Default is False.

Details

All fields are optional. Missing fields will be assigned their Default value. `opiSetup()` sets all to default.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiSetup()`

Examples

```
## Not run:
chooseOpi("Envision")
opiInitialise(address = list(port = 50001, ip = "localhost"))
result <- opiSetup(settings = list(eye = "BOTH"))

## End(Not run)
```

`opiSetup_for_ImoVifa` *Implementation of opiSetup for the ImoVifa machine.*

Description

This is for internal use only. Use `opiSetup()` after `chooseOPI("ImoVifa")` to call this function.

Arguments

`settings` A list containing:

- `eye` The eye for which to apply the settings.
- `bgImageFilename` (Optional) If present, display the image in the background for eye (scaled to fill fov, bgLum and bgCol ignored)
- `fixShape` (Optional) Fixation target type for eye.
- `fixLum` (Optional) Fixation target luminance for eye.
- `fixType` (Optional) Fixation target texture for eye.
- `fixCx` (Optional) x-coordinate of fixation target (degrees).
- `fixCy` (Optional) y-coordinate of fixation target (degrees).
- `fixCol` (Optional) Fixation target color for eye.
- `bgLum` (Optional) Background luminance for eye (cd/m^2).
- `tracking` (Optional) Whether to correct stimulus location based on eye position.
- `bgCol` (Optional) Background color for eye (rgb).
- `fixSx` (Optional) diameter along major axis of ellipse (degrees). 0 to hide fixation marker.
- `fixSy` (Optional) diameter along minor axis of ellipse (degrees). If not received, then sy = sx.
- `fixRotation` (Optional) Angles of rotation of fixation target (degrees). Only useful if sx != sy specified.
- `fixImageFilename` (Optional) If fixType == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use

Details

`eye` can take on values in the set `{"left", "right", "both", "none"}`.

`fixShape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

`fixLum` can take on values in the range `[0.0, 1.0E10]`.

`fixType` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

`fixCx` can take on values in the range `[-90.0, 90.0]`.

`fixCy` can take on values in the range `[-90.0, 90.0]`.

Elements in `fixCol` can take on values in the range `[0.0, 1.0]`.

`bgLum` can take on values in the range `[0.0, 1.0E10]`.

`tracking` can take on values in the range `[0, 1]`.

Elements in `bgCol` can take on values in the range `[0.0, 1.0]`.

`fixSx` can take on values in the range `[0.0, 1.0E10]`.

`fixSy` can take on values in the range `[0.0, 1.0E10]`.

`fixRotation` can take on values in the range `[0.0, 360.0]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiSetup()`

Examples

```
## Not run:
chooseOpi("ImoVifa")
opiInitialise(list(port = 50001, ip = "localhost"))
result <- opiSetup(settings = list(eye = "BOTH"))

## End(Not run)
```

opiSetup_for_KowaAP7000 *opiSetBackground*

Description

Implementation of opiSetup for the Kowa AP7000 machine.

This is for internal use only. Use `opiSetup()` with these Arguments.

Usage

```
opiSetup_for_KowaAP7000(lum = NA, color = NA, fixation = NA)
```

Arguments

`lum`	Must be 10 for a white background and 100 for a yellow.
`color`	One of `.opi_env$KowaAP7000$BACKGROUND_WHITE` or `.opi_env$KowaAP7000$BACKGROUND_YELLOW`.
`fixation`	One of * `.opi_env$KowaAP7000$FIX_CENTER`, fixation marker in the centre. * `.opi_env$KowaAP7000$FIX_CENTRE`, fixation marker in the centre. * `.opi_env$KowaAP7000$FIX_AUX`, fixation marker is ???. * `.opi_env$KowaAP7000$FIX_MACULA`, fixation marker is a circle(?). * `.opi_env$KowaAP7000$FIX_AUX_LEFT`, fixation marker is as for AUX but only lower left.

Details

If `lum` is 10 and `color` is not set, then `.opi_env$KowaAP7000$BACKGROUND_WHITE` is assumed.

If `lum` is 100 and `color` is not set, then `.opi_env$KowaAP7000$BACKGROUND_YELLOW` is assumed.

If both `lum` and `color` is set, then `lum` is ignored (a warning will be generated

If `lum` is incompatible with `color`).

Value

Always returns `list(err = NULL)`

opiSetup_for_MAIA *Implementation of opiSetup for the MAIA machine.*

Description

This is for internal use only. Use `opiSetup()` with these Arguments and you will get the Value back.

Arguments

`settings` is a list that could contain:

- `fixation` `list(x,y,t,c)` where
  - `x` is one of 0, +-2.4, +-4.8 degrees.
  - `y` is one of 0, +-2.4, +-4.8 degrees.
  - `t` is 0 for big circle off, 1 for big circle on.
  - `p` power integer in range 0..1023.
- `bg` which is 0 for no background and other for some background
- `tracking` which is 0 for no tracking and other for some tracking
- `open` If present, will send OPI-OPEN to MAIA and look for prl, onh, fundus image

Details

Note that corner locations are missing, so both `x` and `y` cannot be +-4.8.

Value

A list containing * `err` which is `NULL` for success * `message` for some messages from MAIA (probably "0") For `open` list might (but might not) contain:

- `prl` A pair giving the (x,y) in degrees of the Preferred Retinal Locus detected in the initial alignment.
- `onh` a pair giving the (x,y) in degrees of the ONH as selected by the user.
- `image` raw bytes being the JPEG compressed infra-red image acquired during alignment.

`opiSetup_for_O600` *opiSetup for the O600 machine.*

Description

Implementation of opiSetup for the O600 machine.

This is for internal use only. Use `opiSetup()` with these Arguments.

Arguments

`bgColor`	Background color 0 to 255.
`fixType`	fixation type 1, 2, 3 or 4.
`fixColor`	fixation color 0, 2, 4 or 5.
`fixIntensity`	fixation point intensity 0 to 255.

Value

A list containing one of the following

- `err` = `-1` to be implemented
- `err` = `-2` O600 sent back an error; bad background parameters
- `err` = `-3` O600 sent back an error; bad fixation parameters
- `err` = `NULL` Success

---

opiSetup_for_Octopus900

Implementation of opiSetup for the Octopus900 machine.

Description

This is for internal use only. Use `opiSetup()` with the same parameters.

Arguments

`lum`	Luminance level in cd/m^2
`color`	Stimulus color (see details)
`fixation`	fixation target
`fixIntensity`	fixation point intensity

Details

Allowable `lum` and `color` are defined in the `.opi_env$O900` environment.

- `lum` is intensity of the background and can be one of
  - `.opi_env$O900$BG_OFF`, which turns background off.
  - `.opi_env$O900$BG_1`, background of 1.27 cd/m^2.
  - `.opi_env$O900$BG_10`, background of 10 cd/m^2.
  - `.opi_env$O900$BG_100`, background of 100 cd/m^2.
- `color` can be one of the following choices.
  - `.opi_env$O900$MET_COL_WW` for white-on-white
  - `.opi_env$O900$MET_COL_RW` for red-on-white
  - `.opi_env$O900$MET_COL_BW` for blue-on-white
  - `.opi_env$O900$MET_COL_WY` for white-on-yellow
  - `.opi_env$O900$MET_COL_RY` for red-on-yellow
  - `.opi_env$O900$MET_COL_BY` for blue-on-yellow
- `fixation` is one of
  - `.opi_env$O900$FIX_CENTRE` or `.opi_env$O900$FIX_CENTER`
  - `.opi_env$O900$FIX_CROSS`
  - `.opi_env$O900$FIX_RING`
- `fixIntensity` is a percentage between 0 and 100. 0 is off, 100 the brightest.

Note if you specify `fixation` you also have to specify `fixIntensity`.

Value

A list with element `err` which is

- NULL on success
- -1 indicates `opiInitialize` has not been called.
- -2 indicates could not set the background color.
- -3 indicates could not set the fixation marker.
- or a string message about bad parameters

Examples

```
## Not run:
  chooseOpi("Octopus900")
  oi <- opiInitialize(eyeSuiteJarLocation="c:/EyeSuite/",
              eyeSuiteSettingsLocation="c:/Documents and Settings/All Users/Haag-Streit/",
                     eye="left")
  if(!is.null(oi$err))
      stop("opiInitialize failed")
  if(!is.null(opiSetup(fixation=.opi_env$O900$FIX_CENTRE)$err))
      stop("opiSetup failed")
  if(!is.null(opiSetup(fixation=.opi_env$O900$FIX_RING, fixIntensity=0)$err))
      stop("opiSetup failed")
  if(!is.null(opiSetup(color=.opi_env$O900$MET_COL_BY)$err))
      stop("opiSetup failed")
  if(!is.null(opiSetup(lum=.opi_env$O900$BG_100, color=.opi_env$O900$MET_COL_RW)$err))
      stop("opiSetup failed")
  opiClose()

## End(Not run)
```

`opiSetup_for_PhoneHMD` *Implementation of opiSetup for the PhoneHMD machine.*

Description

This is for internal use only. Use `opiSetup()` after `chooseOPI("PhoneHMD")` to call this function.

Arguments

`settings` A list containing:

- `eye` The eye for which to apply the settings.
- `bgImageFilename` (Optional) If present, display the image in the background for eye (scaled to fill fov, bgLum and bgCol ignored)
- `fixShape` (Optional) Fixation target type for eye.
- `fixLum` (Optional) Fixation target luminance for eye.
- `fixType` (Optional) Fixation target texture for eye.

- `fixCx` (Optional) x-coordinate of fixation target (degrees).
- `fixCy` (Optional) y-coordinate of fixation target (degrees).
- `fixCol` (Optional) Fixation target color for eye.
- `bgLum` (Optional) Background luminance for eye (cd/m^2).
- `tracking` (Optional) Whether to correct stimulus location based on eye position.
- `bgCol` (Optional) Background color for eye (rgb).
- `fixSx` (Optional) diameter along major axis of ellipse (degrees). 0 to hide fixation marker.
- `fixSy` (Optional) diameter along minor axis of ellipse (degrees). If not received, then sy = sx.
- `fixRotation` (Optional) Angles of rotation of fixation target (degrees). Only useful if sx != sy specified.
- `fixImageFilename` (Optional) If fixType == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use

Details

`eye` can take on values in the set `{"left", "right", "both", "none"}`.

`fixShape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

`fixLum` can take on values in the range `[0.0, 1.0E10]`.

`fixType` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

`fixCx` can take on values in the range `[-90.0, 90.0]`.

`fixCy` can take on values in the range `[-90.0, 90.0]`.

Elements in `fixCol` can take on values in the range `[0.0, 1.0]`.

`bgLum` can take on values in the range `[0.0, 1.0E10]`.

`tracking` can take on values in the range `[0, 1]`.

Elements in `bgCol` can take on values in the range `[0.0, 1.0]`.

`fixSx` can take on values in the range `[0.0, 1.0E10]`.

`fixSy` can take on values in the range `[0.0, 1.0E10]`.

`fixRotation` can take on values in the range `[0.0, 360.0]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiSetup()`

Examples

```
## Not run:
chooseOpi("PhoneHMD")
opiInitialise(list(port = 50001, ip = "localhost"))
result <- opiSetup(settings = list(eye = "BOTH"))

## End(Not run)
```

`opiSetup_for_PicoVR` *Implementation of opiSetup for the PicoVR machine.*

Description

This is for internal use only. Use `opiSetup()` after `chooseOPI("PicoVR")` to call this function.

Arguments

`settings` A list containing:

- `eye` The eye for which to apply the settings.
- `bgImageFilename` (Optional) If present, display the image in the background for eye (scaled to fill fov, bgLum and bgCol ignored)
- `fixShape` (Optional) Fixation target type for eye.
- `fixLum` (Optional) Fixation target luminance for eye.
- `fixType` (Optional) Fixation target texture for eye.
- `fixCx` (Optional) x-coordinate of fixation target (degrees).
- `fixCy` (Optional) y-coordinate of fixation target (degrees).
- `fixCol` (Optional) Fixation target color for eye.
- `bgLum` (Optional) Background luminance for eye (cd/m^2).
- `tracking` (Optional) Whether to correct stimulus location based on eye position.
- `bgCol` (Optional) Background color for eye (rgb).
- `fixSx` (Optional) diameter along major axis of ellipse (degrees). 0 to hide fixation marker.
- `fixSy` (Optional) diameter along minor axis of ellipse (degrees). If not received, then sy = sx.
- `fixRotation` (Optional) Angles of rotation of fixation target (degrees). Only useful if sx != sy specified.
- `fixImageFilename` (Optional) If fixType == IMAGE, the filename on the local filesystem of the machine running JOVP of the image to use

Details

`eye` can take on values in the set `{"left", "right", "both", "none"}`.

`fixShape` can take on values in the set `{"triangle", "square", "polygon", "hollow_triangle", "hollow_square", "hollow_polygon", "cross", "maltese", "circle", "annulus", "optotype", "text", "model"}`.

`fixLum` can take on values in the range `[0.0, 1.0E10]`.

`fixType` can take on values in the set `{"flat", "checkerboard", "sine", "squaresine", "g1", "g2", "g3", "text", "image"}`.

`fixCx` can take on values in the range `[-90.0, 90.0]`.

`fixCy` can take on values in the range `[-90.0, 90.0]`.

Elements in `fixCol` can take on values in the range `[0.0, 1.0]`.

`bgLum` can take on values in the range `[0.0, 1.0E10]`.

`tracking` can take on values in the range `[0, 1]`.

Elements in `bgCol` can take on values in the range `[0.0, 1.0]`.

`fixSx` can take on values in the range `[0.0, 1.0E10]`.

`fixSy` can take on values in the range `[0.0, 1.0E10]`.

`fixRotation` can take on values in the range `[0.0, 360.0]`.

Value

A list containing:

- `err` `NULL` if there was no error, a string message if there is an error.

See Also

`opiSetup()`

Examples

```
## Not run:
chooseOpi("PicoVR")
opiInitialise(list(port = 50001, ip = "localhost"))
result <- opiSetup(settings = list(eye = "BOTH"))

## End(Not run)
```

opiSetup_for_SimGaussian

opiSetup_for_SimGaussian

Description

Does nothing.

Arguments

...	Any object you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

opiSetup_for_SimHenson

opiSetup_for_SimHenson

Description

Does nothing. For internal use only, use `opiSetup()`.

Arguments

...	Any object you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

opiSetup_for_SimHensonRT *opiSetup_for_SimHensonRT*

Description

Does nothing. For internal use only, use `opiSetup()`.

Arguments

...	Any object you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

opiSetup_for_SimNo *opiSetup_for_SimNo*

Description

Does nothing.

Arguments

`settings`	Anything you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

opiSetup_for_SimYes — *opiSetup_for_SimYes*

Description

Does nothing.

Arguments

`settings`	Any object you like, it is ignored.

Value

A list with elements:

- `err` Always `NULL`.

opiStaticStimulus — *For backwards compatibility. Used by Octopus900 and KowaAP7000.*

Description

For backwards compatibility. Used by Octopus900 and KowaAP7000.

Usage

```
opiStaticStimulus()
```

opiTemporalStimulus — *For backwards compatibility. Used by Octopus900 and KowaAP7000.*

Description

For backwards compatibility. Used by Octopus900 and KowaAP7000.

Usage

```
opiTemporalStimulus()
```

pixTodeg — *Convert pixels to degrees for machine 'machine'*

Description

Convert pixels to degrees for machine 'machine'

Usage

```
pixTodeg(xy, machine = "compass")
```

Arguments

xy	a 2 element vector c(x,y) where x and y are in pixels
machine	"compass" or ...?

Value

xy converted to degrees of visual field with the usual conventions or NA if machine is unknown

Examples

```
pixTodeg(c(1000, 200), machine="compass") # c(1.290323, 24.516129) degrees
pixTodeg(c(1920/2, 1920/2)) # c(0,0) degrees
```

QUESTP — *QUEST+*

Description

An implementation of the Bayesian test procedure QUEST+ by AB Watson. This is mostly a translation of the MATLAB implementation by P Jones (see References). Its use is similar to ZEST. The objective is to estimate parameters of a function that defines the probability of responding stimuli. The steps are optimized based on entropy rather than the mean or the mode of the current pdfs.

Usage

```
QUESTP(
  Fun,
  stimDomain,
  paramDomain,
  likelihoods = NULL,
  priors = NULL,
  stopType = "H",
  stopValue = 4,
```

```
  maxSeenLimit = 2,
  minNotSeenLimit = 2,
  minPresentations = 1,
  maxPresentations = 100,
  minInterStimInterval = NA,
  maxInterStimInterval = NA,
  verbose = 0,
  makeStim,
  ...
)

QUESTP.Prior(state, priors = NULL)

QUESTP.Likelihood(state)

QUESTP.start(
  Fun,
  stimDomain,
  paramDomain,
  likelihoods = NULL,
  priors = NULL,
  stopType = "H",
  stopValue = 4,
  maxSeenLimit = 2,
  minNotSeenLimit = 2,
  minPresentations = 1,
  maxPresentations = 100,
  makeStim,
  ...
)

getTargetStim(state)

QUESTP.step(state, nextStim = NULL)

QUESTP.stop(state)

QUESTP.final(state, Choice = "mean")

QUESTP.stdev(state, WhichP = NULL)

QUESTP.entropy(state)
```

Arguments

`Fun`	Function to be evaluated, of the form `pseen = function(stim, param){...}`. Outputs a probability of seen.
`stimDomain`	Domain of values for the stimulus. Can be multi-dimensional (list, one element

per dimension)

`paramDomain` Domain of values for pdfs of the parameters in Fun. Can be multi-parametric (list, one element per parameter).

`likelihoods` Pre-computed likelihoods if available (for QUESTP.start)

`priors` Starting probability distributions for the parameter domains (list, one element per parameter)

`stopType` `N`, for number of presentations; `S`, for standard deviation of the pdf; and `H`, for the entropy of the pdf (default).

`stopValue` Value for number of presentations (`stopType=N`), standard deviation (`stopType=S`) or Entropy (`stopType=H`).

`maxSeenLimit` Will terminate if `maxStimulus` value is seen this many times.

`minNotSeenLimit` Will terminate if `minStimulus` value is not seen this many times.

`minPresentations` Minimum number of presentations

`maxPresentations` Maximum number of presentations regarless of `stopType`.

`minInterStimInterval` If both `minInterStimInterval` and `maxInterStimInterval` are not `NA`, then between each stimuli there is a random wait period drawn uniformly between `minInterStimInterval` and `maxInterStimInterval`.

`maxInterStimInterval` `minInterStimInterval`.

`verbose` `verbose=0` does nothing, `verbose=1` stores pdfs for returning, and `verbose=2` stores pdfs and also prints each presentaion.

`makeStim` A function that takes a dB value and numPresentations and returns an OPI datatype ready for passing to opiPresent. See examples.

`...` Extra parameters to pass to the opiPresent function

`state` Current state of the QUESTP returned by `QUESTP.start` and `QUESTP.step`.

`nextStim` A valid object for `opiPresent` to use as its `nextStim`.

`Choice` How to compute final values in QUESTP.final ("mean","mode","median")

`WhichP` Which parameter (numeric index) to monitor when calling QUESTP.stdev directly (returns max(stdev) if unspecified)

Details

An implementation of the Bayesian test procedure QUEST+ by AB Watson. This is mostly a translation of the MATLAB implementation by P Jones (see References). Its use is similar to ZEST. The objective is to estimate parameters of a function that defines the probability of responding to stimuli. The steps are optimized based on entropy rather than the mean or the mode of the current pdfs.

The stimulus, parameter and response domain are separate and can be multidimensional. Each parameter has its own pdf. For evaluation, the pdfs are chained as a long vector (so no co-variances

are considered). More complex functions will require larger combined pdfs and longer computation time for likelihoods and updating at each step. In these cases, it is recommended to pre-calculate the likelihoods using QUESTP.Likelihood and store them.

The function to be fitted needs to output a probability of seen (i.e. `pseen = function(stim, param){...}`) and must take `stim` and `param` as inputs. `stim` is a vector with length = number of stimulus dimensions (in simple one-dimensional cases, the intensity in dB). `param` is a vector with length = number of parameters to be fitted in Fun.

For example, QUEST+ can fit a Gaussian psychometric function with stimDomain = `{0, 1,..., 39, 40}` dB and paramDomain = `({0, 1,..., 39, 40}; {0.5, 1,..., 5.5, 6})` dB for the mean and standard deviation respectively. A standard ZEST procedure can be replicated by setting stimDomain = `{0, 1,..., 39, 40}` dB and paramDomain = `({0, 1,..., 39, 40}; {1})` dB, i.e. by setting the stimDomain = paramDomain for the mean and by having a static standard deviation = 1 dB. Note however that the stimulus selection is always based on entropy and not on the mean/mode of the current pdf. See examples below

Note this function will repeatedly call `opiPresent` for a stimulus until `opiPresent` returns `NULL` (ie no error occurred).

The `checkFixationOK` function is called (if present in stim made from `makeStim`) after each presentation, and if it returns FALSE, the pdf for that location is not changed (ie the presentation is ignored), but the stim, number of presentations etc is recorded in the state.

If more than one QUESTP is to be interleaved (for example, testing multiple locations), then the `QUESTP.start`, `QUESTP.step`, `QUESTP.stop` and `QUESTP.final` calls can maintain the state of the QUESTP after each presentation, and should be used. If only a single QUESTP is required, then the simpler `QUESTP` can be used, which is a wrapper for the four functions that maintain state. See examples below.

Value

##Single location `QUESTP` returns a list containing

- `npres` Total number of presentations used.
- `respSeq` Response sequence stored as a data frame: column 1 is a string identified of a (potentially) multidimensional stimulus values of stimuli (dimensions chained into a string), column 2 is 1/0 for seen/not-seen, column 3 is fixated 1/0 (always 1 if `checkFixationOK` not present in stim objects returned from `makeStim`). All additional columns report each stimulus dimension, one for each row.
- `pdfs` If `verbose` is bigger than 0, then this is a list of the pdfs used for each presentation, otherwise NULL.
- `final` The mean (default, strongly suggested)/median/mode of the parameters' pdf, depending on `Choice`.
- `opiResp`A list of responses received from each successful call to `opiPresent` within `QUESTP`.

Multiple locations:

`QUESTP.start` returns a list that can be passed to `QUESTP.step`, `QUESTP.stop`, and `QUESTP.final`. It represents the state of a QUESTP at a single location at a point in time and contains the following.

- `name` `QUESTP`

- A copy of all of the parameters supplied to QUESTP.start: `stimDomain`, `paramDomain`, `likelihoods`, `priors`, `stopType`, `stopValue`, `maxSeenLimit`, `minNotSeenLimit`, `minPresentations`, `maxPresentations`, `makeStim`, and `opiParams`.
- `pdf` Current pdf: vector of probabilities, collating all parameter domains.
- `priorsP` List of starting pdfs, one for each parameter.
- `numPresentations` The number of times `QUESTP.step` has been called on this state.
- `stimuli` A vector containing the stimuli used at each call of `QUESTP.step`.
- `responses` A vector containing the responses received at each call of `QUESTP.step`.
- `responseTimes` A vector containing the response times received at each call of `QUESTP.step`.
- `fixated` A vector containing TRUE/FALSE if fixation was OK according to `checkFixationOK` for each call of `QUESTP.step` (defaults to TRUE if `checkFixationOK` not present).
- `opiResp` A list of responses received from each call to `opiPresent` within `QUESTP.step`.

`QUESTP.step` returns a list containing

- `state` The new state after presenting a stimuli and getting a response.
- `resp` The return from the `opiPresent` call that was made.

`QUESTP.stop` returns `TRUE` if the QUESTP has reached its stopping criteria, and `FALSE` otherwise.

`QUESTP.final` returns an estimate of parameters based on state. If `state$Choice` is `mean` then the mean is returned (the only one that really makes sense for QUESTP). If `state$Choice` is `mode` then the mode is returned. If `state$Choice` is `median` then the median is returned.

References

Andrew B. Watson; QUEST+: A general multidimensional Bayesian adaptive psychometric method. Journal of Vision 2017;17(3):10. doi: https://doi.org/10.1167/17.3.10.

Jones, P. R. (2018). QuestPlus: a MATLAB implementation of the QUEST+ adaptive psychometric method, Journal of Open Research Software, 6(1):27. doi: http://doi.org/10.5334/jors.195

A. Turpin, P.H. Artes and A.M. McKendrick "The Open Perimetry Interface: An enabling tool for clinical visual psychophysics", Journal of Vision 12(11) 2012.

See Also

`dbTocd`, `opiPresent`

Examples

```
chooseOpi("SimHenson")
if(!is.null(opiInitialize(type="C", cap=6)$err))
    stop("opiInitialize failed")

#######################################################
# This section is for single location QUESTP
# This example fits a FoS curve
# Note: only fitting threshold and slope,
# modify the domain for FPR and FNR to fit those as well
#######################################################
# Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
```

```
    s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
              duration=200, responseWindow=1500, checkFixationOK=NULL)
    class(s) <- "opiStaticStimulus"
    return(s)
}

#True parameters (variability is determined according to Henson et al. based on threshold)
loc <- list(threshold = 20, fpr = 0.05, fnr = 0.05)

#Function to fit (Gaussian psychometric function)
pSeen <- function(x, params){return(params[3] +
                                        (1 - params[3] - params[4]) *
                                        (1 - pnorm(x, params[1], params[2])))}
#QUEST+
QP <- QUESTP(Fun = pSeen,
             stimDomain = list(0:50),
             paramDomain = list(seq(0, 40, 1), #Domain for the 50% threshold (Mean)
                                seq(.5, 8, .5), #Domain for the slope (SD)
                                seq(0.05, 0.05, 0.05), #Domain for the FPR (static)
                                seq(0.05, 0.05, 0.05)), #Domain for the FNR (static)
             stopType="H", stopValue=4, maxPresentations=500,
             makeStim = makeStim,
             tt=loc$threshold, fpr=loc$fpr, fnr=loc$fnr,
             verbose = 2)

#Plots results
#Henson's FoS function (as implemented in OPI - ground truth)
HensFunction <- function(Th){
    SD <- exp(-0.081*Th + 3.27)
    SD[SD > 6] <- 6
    return(SD)
}

#Stimulus domain
dB_Domain <- 0:50
FoS <- pSeen(dB_Domain, params = QP$final) # Estimated FoS
FoS_GT <- pSeen(dB_Domain, params = c(loc$threshold, HensFunction(loc$threshold),
                            loc$fpr, loc$fnr)) #Ground truth FoS (based on Henson et al.)

#Plot (seen stimuli at the top, unseen stimuli at the bottom)
plot(dB_Domain, FoS_GT, type = "l", ylim = c(0, 1), xlab = "dB", ylab = "% seen", col = "blue")
lines(dB_Domain, FoS, col = "red")
points(QP$respSeq$stimuli, QP$respSeq$responses, pch = 16,
       col = rgb(red = 1, green = 0, blue = 0, alpha = 0.1))
legend("top", inset = c(0, -.2),legend = c("True","Estimated","Stimuli"),
       col=c("blue", "red","red"), lty=c(1,1,0),
       pch = c(16, 16, 16), pt.cex = c(0, 0, 1),
       horiz = TRUE, xpd = TRUE, xjust = 0)

if (!is.null(opiClose()$err))
  warning("opiClose() failed")
```

```
chooseOpi("SimHenson")
if(!is.null(opiInitialize(type="C", cap=6)$err))
    stop("opiInitialize failed")

#####################################################################
# This section is for single location QUESTP
# This example shows that QUEST+ can replicate a ZEST procedure
# by fitting a FoS curve with fixed Slope, FPR and FNR
# Compared with ZEST
# Note that QUEST+ should be  marginally more efficient in selecting
# the most informative stimulus
#####################################################################
# Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
    s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
              duration=200, responseWindow=1500, checkFixationOK=NULL)
    class(s) <- "opiStaticStimulus"
    return(s)
}

#True parameters (variability is determined according to Henson et al. based on threshold)
loc <- list(threshold = 30, fpr = 0.03, fnr = 0.03)

#Function to fit (Gaussian psychometric function - Fixed slope (same as default likelihood in ZEST))
pSeen <- function(domain, tt){{0.03+(1-0.03-0.03)*(1-pnorm(domain,tt,1))}}

# ZEST-like QUEST+ procedure
QP <- QUESTP(Fun = pSeen,
             stimDomain = list(0:40),
             paramDomain = list(seq(0, 40, 1)),
             stopType="S", stopValue=1.5, maxPresentations=500,
             makeStim = makeStim,
             tt=loc$threshold, fpr=loc$fpr, fnr=loc$fnr,
             verbose = 2)

# ZEST
ZE <- ZEST(domain = 0:40,
           stopType="S", stopValue=1.5, maxPresentations=500,
           makeStim = makeStim,
           tt=loc$threshold, fpr=loc$fpr, fnr=loc$fnr,
           verbose = 2)

#Plots results
#Henson's FoS function (as implemented in OPI - ground truth)
HensFunction <- function(Th){
    SD <- exp(-0.081*Th + 3.27)
    SD[SD > 6] <- 6
    return(SD)
}

#Stimulus domain
dB_Domain <- 0:50
```

```
FoS_QP <- pSeen(domain = dB_Domain, tt = QP$final) # Estimated FoS
FoS_ZE <- pSeen(domain = dB_Domain, tt = ZE$final) # Estimated FoS

#Plot (seen stimuli at the top, unseen stimuli at the bottom)
plot(dB_Domain, FoS_QP, type = "l", ylim = c(0, 1), xlab = "dB", ylab = "% seen", col = "blue")
lines(dB_Domain, FoS_ZE, col = "red")
points(QP$respSeq$stimuli, QP$respSeq$responses, pch = 16,
       col = rgb(red = 0, green = 0, blue = 1, alpha = 0.5))
points(ZE$respSeq[1,], ZE$respSeq[2,], pch = 16,
       col = rgb(red = 1, green = 0, blue = 0, alpha = 0.5))
legend("bottomleft", legend = c("QUEST+","ZEST","Stimuli QUEST+", "Stimuli ZEST"),
       col=c("blue", "red","blue","red"), lty=c(1,1,0,0),
       pch = c(16, 16, 16, 16), pt.cex = c(0, 0, 1, 1),
       horiz = FALSE, xpd = TRUE, xjust = 0)
abline(v = loc$threshold, lty = "dashed")

if (!is.null(opiClose()$err))
  warning("opiClose() failed")


chooseOpi("SimHenson")
if(!is.null(opiInitialize(type="C", cap=6)$err))
  stop("opiInitialize failed")

#######################################################
# This section is for single location QUESTP
# This example fits a broken stick spatial summation function
# with a multi-dimensional stimulus (varying in size and intensity).
# Stimulus sizes are limited to GI, GII, GIII, GIV and GV.
# The example also shows how to use a helper function to
# simulate responses to multi-dimensional stimuli
# (here, the simulated threshold varies based on stimulus size)
#######################################################
makeStim <- function(stim, n) {
  s <- list(x=9, y=9, level=dbTocd(stim[1]), size=stim[2], color="white",
            duration=200, responseWindow=1500, checkFixationOK=NULL)
  class(s) <- "opiStaticStimulus"
  return(s)
}

# Helper function for true threshold (depends on log10(stimulus size),
# diameter assumed to be the second element of stim vector)
ttHelper_SS <- function(location) {  # returns a function of (stim)
  ff <- function(stim) stim

  body(ff) <- substitute(
   {return(SensF(log10(pi*(stim[2]/2)^2), c(location$Int1, location$Int2, location$Slo2)))}
  )
  return(ff)
}

# Function of sensivity vs SSize (log10(stimulus area))
SensF <- function(SSize, params){
```

```
  Sens <- numeric(length(SSize))
  for (i in 1:length(SSize)){
    Sens[i] <- min(c(params[1] + 10*SSize[i], params[2] + params[3]*SSize[i]))
  }
  Sens[Sens < 0] <- 0
  return(Sens)
}

Sizes <- c(0.1, 0.21, 0.43, 0.86, 1.72)

#True parameters (variability is determined according to Henson et al. based on threshold)
loc <- list(Int1 = 32, Int2 = 28, Slo2 = 2.5, fpr = 0.05, fnr = 0.05, x = 9, y = 9)


# Function to fit (probability of seen given a certain stimulus intensity and size,
# for different parameters)
pSeen <- function(stim, params){

  Th <- SensF(log10(pi*(stim[2]/2)^2), params)

  return(0.03 +
           (1 - 0.03 - 0.03) *
           (1 - pnorm(stim[1], Th, 1)))
}


## Not run:
set.seed(111)
#QUEST+ - takes some time to calculate likelihoods
QP <- QUESTP(Fun = pSeen,
             stimDomain = list(0:50, Sizes),
             paramDomain = list(seq(0, 40, 1), # Domain for total summation intercept
                                seq(0, 40, 1), # Domain for partial summation intercept
                                seq(0, 3, 1)), # Domain for partial summation slope
             stopType="H", stopValue=1, maxPresentations=500,
             makeStim = makeStim,
             ttHelper=ttHelper_SS(loc), tt = 30,
             fpr=loc$fpr, fnr=loc$fnr,
             verbose = 2)
#Stimulus sizes
G <- log10(c(pi*(0.1/2)^2, pi*(0.21/2)^2, pi*(0.43/2)^2, pi*(0.86/2)^2, pi*(1.72/2)^2));
SizesP <- seq(min(G), max(G), .05)

# True and estimated response
Estim_Summation <- SensF(SizesP, params = QP$final) # Estimated spatial summation
GT_Summation <- SensF(SizesP, params = c(loc$Int1, loc$Int2, loc$Slo2)) # True spatial summation

#Plot
plot(10^SizesP, GT_Summation, type = "l", ylim = c(0, 40), log = "x",
     xlab = "Stimulus area (deg^2)", ylab = "Sensitivity (dB)", col = "blue")
lines(10^SizesP, Estim_Summation, col = "red")
points(pi*(QP$respSeq$stimuli.2/2)^2, QP$respSeq$stimuli.1, pch = 16,
       col = rgb(red = 1, green = 0, blue = 0, alpha = 0.3))
```

```
legend("top", inset = c(0, -.2),legend = c("True","Estimated","Stimuli"),
       col=c("blue", "red","red"), lty=c(1,1,0),
       pch = c(16, 16, 16), pt.cex = c(0, 0, 1),
       horiz = TRUE, xpd = TRUE, xjust = 0)


## End(Not run)
if (!is.null(opiClose()$err))
  warning("opiClose() failed")
```

RtDbUnits *Response times to white-on-white Goldmann Size III targets for 12 subjects in dB units*

Description

Response times to white-on-white Goldmann Size III targets for 12 subjects. The second column is the distance of the stimuli from measured threshold in HFA dB units. The threshold was determined by post-hoc fit of FoS curves to the data.

Usage

```
RtDbUnits
```

Format

An object of class `data.frame` with 30620 rows and 3 columns.

Details

A data frame with 30620 observations on the following 3 variables.

`Rt` Reaction time in ms.

`Dist` Distance of stimuli from threshold in dB.

`Person` Identifier of each subject.

References

A.M. McKendrick, J. Denniss and A. Turpin. "Response times across the visual field: empirical observations and application to threshold determination". Vision Research 101 2014.

See Also

`RtSigmaUnits`

`RtSigmaUnits` *Response times to white-on-white Goldmann Size III targets for 12 subjects in sigma units*

Description

Response times to white-on-white Goldmann Size III targets for 12 subjects. The second column is the distance of the stimuli from measured threshold in 'sigma' units. The threshold was determined by post-hoc fit of a cummulative gaussian FoS curve to the data for each location and subject. Sigma is the standard deviation of the fitted FoS.

Usage

```
RtSigmaUnits
```

Format

An object of class `data.frame` with 30620 rows and 3 columns.

Details

A data frame with 30620 observations on the following 3 variables.

`Rt` Reaction time in ms.

`Dist` Distance of stimuli from threshold in sigma units.

`Person` Identifier of each subject.

References

A.M. McKendrick, J. Denniss and A. Turpin. "Response times across the visual field: empirical observations and application to threshold determination". Vision Research 101 2014.

See Also

`RtDbUnits`

ZEST *ZEST*

Description

An implementation of the Bayesian test procedures of King-Smith et al. and Watson and Pelli. Note that we use the term pdf throughout as in the original paper, even though they are discrete probability functions in this implementation.

Usage

```
ZEST(
  domain = 0:40,
  prior = rep(1/length(domain), length(domain)),
  likelihood = sapply(domain, function(tt) 0.03 + (1 - 0.03 - 0.03) * (1 -
    stats::pnorm(domain, tt, 1))),
  stopType = "S",
  stopValue = 1.5,
  minStimulus = utils::head(domain, 1),
  maxStimulus = utils::tail(domain, 1),
  maxSeenLimit = 2,
  minNotSeenLimit = 2,
  maxPresentations = 100,
  minInterStimInterval = NA,
  maxInterStimInterval = NA,
  verbose = 0,
  makeStim,
  stimChoice = "mean",
  ...
)

ZEST.start(
  domain = 0:40,
  prior = rep(1/length(domain), length(domain)),
  likelihood = sapply(domain, function(tt) 0.03 + (1 - 0.03 - 0.03) * (1 -
    stats::pnorm(domain, tt, 1))),
  stopType = "S",
  stopValue = 1.5,
  minStimulus = utils::head(domain, 1),
  maxStimulus = utils::tail(domain, 1),
  maxSeenLimit = 2,
  minNotSeenLimit = 2,
  maxPresentations = 100,
  makeStim,
  stimChoice = "mean",
  ...
)
```

```
ZEST.step(state, nextStim = NULL, fixedStimValue = NA, fixedResponse = NA)

ZEST.stop(state)

ZEST.final(state)
```

Arguments

domain	Vector of values over which pdf is kept.
prior	Starting probability distribution over domain. Same length as domain.
likelihood	Matrix where likelihood[s,t] is likelihood of seeing s given t is the true threshold. That is, Pr(s|t) where s and t are indexes into domain.
stopType	N, for number of presentations; S, for standard deviation of the pdf; and H, for the entropy of the pdf.
stopValue	Value for number of presentations (stopType=N), standard deviation (stopType=S) or Entropy (stopType=H).
minStimulus	The smallest stimuli that will be presented. Could be different from domain[1].
maxStimulus	The largest stimuli that will be presented. Could be different from tail(domain,1).
maxSeenLimit	Will terminate if maxStimulus value is seen this many times.
minNotSeenLimit	Will terminate if minStimulus value is not seen this many times.
maxPresentations	Maximum number of presentations regardless of stopType.
minInterStimInterval	If both minInterStimInterval and maxInterStimInterval are not NA, then between each stimuli there is a random wait period drawn uniformly between minInterStimInterval and maxInterStimInterval.
maxInterStimInterval	minInterStimInterval.
verbose	verbose=0 does nothing, verbose=1 stores pdfs for returning, and verbose=2 stores pdfs and also prints each presentation.
makeStim	A function that takes a dB value and numPresentations and returns an OPI datatype ready for passing to opiPresent. See examples.
stimChoice	A true ZEST procedure uses the "mean" of the current pdf as the stimulus, but "median" and "mode" (as used in a QUEST procedure) are provided for your enjoyment.
...	Extra parameters to pass to the opiPresent function
state	Current state of the ZEST returned by ZEST.start and ZEST.step.
nextStim	A valid object for opiPresent to use as its nextStim.
fixedStimValue	A number in state$domain that, is !is.na, will be used as the stimulus value overriding state$minStimulus, state$maxStimulus and state$stimChoice.
fixedResponse	Ignored if !is.na otherwise used as the response to the stim shown.

Details

This is an implementation of King-Smith et al.'s ZEST procedure and Watson and Pelli's QUEST procedure. All presentations are rounded to an element of the supplied domain.

Note this function will repeatedly call `opiPresent` for a stimulus until `opiPresent` returns `NULL` (ie no error occurred).

The `checkFixationOK` function is called (if present in stim made from `makeStim`) after each presentation, and if it returns FALSE, the pdf for that location is not changed (ie the presentation is ignored), but the stim, number of presentations etc is recorded in the state.

If more than one ZEST is to be interleaved (for example, testing multiple locations), then the `ZEST.start`, `ZEST.step`, `ZEST.stop` and `ZEST.final` calls can maintain the state of the ZEST after each presentation, and should be used. If only a single ZEST is required, then the simpler `ZEST` can be used, which is a wrapper for the four functions that maintain state. See examples below.

Value

Single location:

`ZEST` returns a list containing

- `npres` Total number of presentations used.
- `respSeq` Response sequence stored as a matrix: row 1 is dB values of stimuli, row 2 is 1/0 for seen/not-seen, row 3 is fixated 1/0 (always 1 if `checkFixationOK` not present in stim objects returned from `makeStim`).
- `pdfs` If `verbose` is bigger than 0, then this is a list of the pdfs used for each presentation, otherwise NULL.
- `final` The mean/median/mode of the final pdf, depending on `stimChoice`, which is the determined threshold.
- `opiResp` A list of responses received from each successful call to `opiPresent` within `ZEST`.

Multilple locations:

`ZEST.start` returns a list that can be passed to `ZEST.step`, `ZEST.stop`, and `ZEST.final`. It represents the state of a ZEST at a single location at a point in time and contains the following.

- `name` `ZEST`.
- `pdf` Current pdf: vector of probabilities the same length as `domain`.
- `numPresentations` The number of times `ZEST.step` has been called on this state.
- `stimuli` A vector containing the stimuli used at each call of `ZEST.step`.
- `responses` A vector containing the responses received at each call of `ZEST.step`.
- `responseTimes` A vector containing the response times received at each call of `ZEST.step`.
- `fixated` A vector containing TRUE/FALSE if fixation was OK according to `checkFixationOK` for each call of `ZEST.step` (defaults to TRUE if `checkFixationOK` not present).
- `opiResp` A list of responses received from each call to `opiPresent` within `ZEST.step`.
- A copy of all of the parameters supplied to ZEST.start: `domain`, `likelihood`, `stopType`, `stopValue`, `minStimulus`, `maxStimulus`, `maxSeenLimit`, `minNotSeenLimit`, `maxPresentations`, `makeStim`, `stimChoice`, `currSeenLimit`, `currNotSeenLimit`, and `opiParams`.

ZEST.step returns a list containing * state The new state after presenting a stimuli and getting a response. * resp The return from the opiPresent call that was made.

ZEST.stop returns TRUE if the ZEST has reached its stopping criteria, and FALSE otherwise.

ZEST.final returns an estimate of threshold based on state. If state$stimChoice is mean then the mean is returned. If state$stimChoice is mode then the mode is returned. If state$stimChoice is median then the median is returned.

A list containing * state The new state after presenting a stimuli and getting a response. * resp The return from the opiPresent call that was made.

References

P.E. King-Smith, S.S. Grigsny, A.J. Vingrys, S.C. Benes, and A. Supowit. "Efficient and Unbiased Modifications of the QUEST Threshold Method: Theory, Simulations, Experimental Evaluation and Practical Implementation", Vision Research 34(7) 1994. Pages 885-912.

A.B. Watson and D.G. Pelli. "QUEST: A Bayesian adaptive psychophysical method", Perception and Psychophysics 33 1983. Pages 113-l20.

A. Turpin, P.H. Artes and A.M. McKendrick "The Open Perimetry Interface: An enabling tool for clinical visual psychophysics", Journal of Vision 12(11) 2012.

See Also

dbTocd, opiPresent

Examples

```
chooseOpi("SimHenson")
if(!is.null(opiInitialize(type="C", cap=6)$err))
  stop("opiInitialize failed")

##############################################
# This section is for single location ZESTs
##############################################
# Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
  s <- list(x=9, y=9, level=dbTocd(db), size=0.43, color="white",
            duration=200, responseWindow=1500, checkFixationOK=NULL)
  class(s) <- "opiStaticStimulus"
  return(s)
}

repp <- function(...) sapply(1:50, function(i) ZEST(makeStim=makeStim, ...))
a <- repp(stopType="H", stopValue=  3, verbose=0, tt=30, fpr=0.03)
b <- repp(stopType="S", stopValue=1.5, verbose=0, tt=30, fpr=0.03)
c <- repp(stopType="S", stopValue=2.0, verbose=0, tt=30, fpr=0.03)
d <- repp(stopType="N", stopValue= 50, verbose=0, tt=30, fpr=0.03)
e <- repp(prior=dnorm(0:40,m=0,s=5), tt=30, fpr=0.03)
f <- repp(prior=dnorm(0:40,m=10,s=5), tt=30, fpr=0.03)
g <- repp(prior=dnorm(0:40,m=20,s=5), tt=30, fpr=0.03)
h <- repp(prior=dnorm(0:40,m=30,s=5), tt=30, fpr=0.03)
```

```
layout(matrix(1:2,1,2))
boxplot(lapply(list(a,b,c,d,e,f,g,h), function(x) unlist(x["final",])))
boxplot(lapply(list(a,b,c,d,e,f,g,h), function(x) unlist(x["npres",])))

##############################################
# This section is for multiple ZESTs
##############################################
makeStimHelper <- function(db,n, x, y) {  # returns a function of (db,n)
  ff <- function(db, n) db+n
  body(ff) <- substitute({
    s <- list(x=x, y=y, level=dbTocd(db), size=0.43, color="white",
              duration=200, responseWindow=1500, checkFixationOK=NULL)
    class(s) <- "opiStaticStimulus"
    return(s)
  }, list(x=x,y=y))
  return(ff)
}

# List of (x, y, true threshold) triples
locations <- list(c(9,9,30), c(-9,-9,32), c(9,-9,31), c(-9,9,33))

# Setup starting states for each location
states <- lapply(locations, function(loc) {
  ZEST.start(
    domain=-5:45,
    minStimulus=0,
    maxStimulus=40,
    makeStim=makeStimHelper(db,n,loc[1],loc[2]),
    stopType="S", stopValue= 1.5, tt=loc[3], fpr=0.03, fnr=0.01)})

# Loop through until all states are "stop"
while(!all(st <- unlist(lapply(states, ZEST.stop)))) {
  i <- which(!st)                        # choose a random,
  i <- i[runif(1, min=1, max=length(i))]  # unstopped state
  r <- ZEST.step(states[[i]])            # step it
  states[[i]] <- r$state                 # update the states
}

finals <- lapply(states, ZEST.final)    # get final estimates of threshold
for(i in 1:length(locations)) {
  #cat(sprintf("Location (%+2d,%+2d) ",locations[[i]][1], locations[[i]][2]))
  #cat(sprintf("has threshold %4.2f\n", finals[[i]]))
}

if (!is.null(opiClose()$err))
  warning("opiClose() failed")
```

Index

∗ **datasets**
.opi_env, 4
∗ **dataset**
RtDbUnits, 105
RtSigmaUnits, 106
.opi_env, 4

cdTodb, 5
chooseOPI, 5
chooseOpi *(chooseOPI)*, 5

dbTocd, 6, *8*, *12*, *19*, *22*, *100*, *110*
degTopix, 6

fourTwo.final *(fourTwo.start)*, 7
fourTwo.start, 7, *12*
fourTwo.step *(fourTwo.start)*, 7
fourTwo.stop *(fourTwo.start)*, 7
FT, *8*, 9

getTargetStim *(QUESTP)*, 96

kowa.presentKinetic, 13, *57*
kowa.presentStatic, 14, *57*
kowa.presentTemporal, 15, *57*
KTPsi, 15

MOCS, 19

octo900.presentKinetic, 24, *60*
octo900.presentStatic, 25, *60*
octo900.presentTemporal, 26, *60*
open_socket, 27
opiClose, 27
opiClose(), *28–32*
opiClose_for_Compass, 28
opiClose_for_Compass(), *27*
opiClose_for_Display, 28
opiClose_for_Display(), *27*
opiClose_for_Envision, 29
opiClose_for_Envision(), *27*
opiClose_for_ImoVifa, 30
opiClose_for_ImoVifa(), *27*
opiClose_for_KowaAP7000, 30
opiClose_for_MAIA, 31
opiClose_for_O600, 31
opiClose_for_Octopus900, 31
opiClose_for_Octopus900(), *27*
opiClose_for_PhoneHMD, 32
opiClose_for_PhoneHMD(), *27*
opiClose_for_PicoVR, 32
opiClose_for_PicoVR(), *27*
opiClose_for_SimGaussian, 33
opiClose_for_SimGaussian(), *27*
opiClose_for_SimHenson, 33
opiClose_for_SimHenson(), *27*
opiClose_for_SimHensonRT, 34
opiClose_for_SimHensonRT(), *27*
opiClose_for_SimNo, 34
opiClose_for_SimNo(), *27*
opiClose_for_SimYes, 34
opiClose_for_SimYes(), *27*
opiInitialise, 35
opiInitialise(), *35–43*
opiInitialise_for_Compass, 35
opiInitialise_for_Compass(), *35*
opiInitialise_for_Display, 36
opiInitialise_for_Display(), *35*
opiInitialise_for_Envision, 37
opiInitialise_for_Envision(), *35*
opiInitialise_for_ImoVifa, 38
opiInitialise_for_ImoVifa(), *35*
opiInitialise_for_KowaAP7000, 39
opiInitialise_for_MAIA, 39
opiInitialise_for_O600, 40
opiInitialise_for_Octopus900, 41
opiInitialise_for_Octopus900(), *35*
opiInitialise_for_PhoneHMD, 42
opiInitialise_for_PhoneHMD(), *35*
opiInitialise_for_PicoVR, 43

opiInitialise_for_PicoVR(), *35*
opiInitialise_for_SimGaussian, 44
opiInitialise_for_SimGaussian(), *35*
opiInitialise_for_SimHenson, 45
opiInitialise_for_SimHenson(), *35*
opiInitialise_for_SimHensonRT, 46
opiInitialise_for_SimHensonRT(), *35*
opiInitialise_for_SimNo, 47
opiInitialise_for_SimNo(), *35*
opiInitialise_for_SimYes, 47
opiInitialise_for_SimYes(), *35*
opiInitialize *(*opiInitialise*)*, 35
opiInitialize(), *25*, *44*
opiKineticStimulus, 48
opiPresent, *8*, *12*, *17*, *19*, *22*, *44*, 48, *100*, *110*
opiPresent(), *13–15*, *24–26*, *50*, *52–54*, *56*, *57*, *59–61*, *63*, *64*, *66*
opiPresent_for_Compass, 49
opiPresent_for_Compass(), *49*
opiPresent_for_Display, 50
opiPresent_for_Display(), *49*
opiPresent_for_Envision, 53
opiPresent_for_Envision(), *49*
opiPresent_for_ImoVifa, 54
opiPresent_for_ImoVifa(), *49*
opiPresent_for_KowaAP7000, 57
opiPresent_for_MAIA, 58
opiPresent_for_O600, 59
opiPresent_for_Octopus900, 60
opiPresent_for_Octopus900(), *49*
opiPresent_for_PhoneHMD, 61
opiPresent_for_PhoneHMD(), *49*
opiPresent_for_PicoVR, 64
opiPresent_for_PicoVR(), *49*
opiPresent_for_SimGaussian, 66
opiPresent_for_SimGaussian(), *49*
opiPresent_for_SimHenson, 67
opiPresent_for_SimHenson(), *49*
opiPresent_for_SimHensonRT, 68
opiPresent_for_SimHensonRT(), *49*
opiPresent_for_SimNo, 70
opiPresent_for_SimNo(), *49*
opiPresent_for_SimYes, 70
opiPresent_for_SimYes(), *49*
opiQueryDevice, 71
opiQueryDevice(), *71–77*
opiQueryDevice_for_Compass, 71
opiQueryDevice_for_Compass(), *71*
opiQueryDevice_for_Display, 72
opiQueryDevice_for_Display(), *71*
opiQueryDevice_for_Envision, 72
opiQueryDevice_for_Envision(), *71*
opiQueryDevice_for_ImoVifa, 74
opiQueryDevice_for_ImoVifa(), *71*
opiQueryDevice_for_KowaAP7000, 75
opiQueryDevice_for_MAIA, 75
opiQueryDevice_for_O600, 76
opiQueryDevice_for_Octopus900, 76
opiQueryDevice_for_Octopus900(), *71*
opiQueryDevice_for_PhoneHMD, 77
opiQueryDevice_for_PhoneHMD(), *71*
opiQueryDevice_for_PicoVR, 77
opiQueryDevice_for_PicoVR(), *71*
opiQueryDevice_for_SimGaussian, 78
opiQueryDevice_for_SimGaussian(), *71*
opiQueryDevice_for_SimHenson, 78
opiQueryDevice_for_SimHenson(), *71*
opiQueryDevice_for_SimHensonRT, 79
opiQueryDevice_for_SimHensonRT(), *71*
opiQueryDevice_for_SimNo, 79
opiQueryDevice_for_SimNo(), *71*
opiQueryDevice_for_SimYes, 79
opiQueryDevice_for_SimYes(), *71*
opiSetBackground, 80
opiSetup, 80
opiSetup(), *25*, *49*, *58*, *80–92*
opiSetup_for_Compass, 81
opiSetup_for_Compass(), *80*
opiSetup_for_Display, 81
opiSetup_for_Display(), *80*
opiSetup_for_Envision, 83
opiSetup_for_Envision(), *80*
opiSetup_for_ImoVifa, 84
opiSetup_for_ImoVifa(), *80*
opiSetup_for_KowaAP7000, 86
opiSetup_for_MAIA, 86
opiSetup_for_O600, 87
opiSetup_for_Octopus900, 88
opiSetup_for_Octopus900(), *80*
opiSetup_for_PhoneHMD, 89
opiSetup_for_PhoneHMD(), *80*
opiSetup_for_PicoVR, 91
opiSetup_for_PicoVR(), *80*
opiSetup_for_SimGaussian, 93
opiSetup_for_SimGaussian(), *80*
opiSetup_for_SimHenson, 93

opiSetup_for_SimHenson(), *80*
opiSetup_for_SimHensonRT, 94
opiSetup_for_SimHensonRT(), *80*
opiSetup_for_SimNo, 94
opiSetup_for_SimNo(), *80*
opiSetup_for_SimYes, 95
opiSetup_for_SimYes(), *80*
opiStaticStimulus, 95
opiTemporalStimulus, 95

pixTodeg, 96

QUESTP, 96

RtDbUnits, 105, *106*
RtSigmaUnits, *105*, 106

ZEST, 107