Package 'OPI'

September 19, 2019

Type Package

Title Open Perimetry Interface

Version 2.9

Date 2019-08-23

Author Andrew Turpin [cre, aut, cph],
David Lawson [ctb, cph],
Matthias Muller [ctb],
Jonathan Dennis [ctb, cph],
Astrid Zeman [ctb],
Ivan Marin-Franch [ctb]

Maintainer Andrew Turpin <aturpin@unimelb.edu.au>

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). It specifies basic functions that allow many visual field tests to be constructed. As of October 2017 it is fully implemented on the Octopus 900 and partially on the Heidelberg Edge Perimeter, the Kowa AP 7000, the CrewT imo and the Centervue Compass. It also has a cousin: the R package 'visualFields', which has tools for analysing and manipulating visual field data.

License GPL-3

URL http://people.eng.unimelb.edu.au/aturpin/opi/index.html

Depends methods

RoxygenNote 6.1.1

Collate opi.r compassClient.r daydreamClient.r dbTocd.r fourTwo.r full_threshold.r full_threshold_state.r imoClient.r kowaAP7000Client.r mocs.r multiLocation.r octopus600.r octopus900Client.r pix2deg.r simDisplay.r simG.r simH.r simH_RT.r simNo.r simYes.r zest.r

NeedsCompilation no

Repository CRAN

Date/Publication 2019-09-19 04:10:02 UTC

R topics documented:

`OPI-package` *Open Perimetry Interface*

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). It specifies basic functions that allow many visual field tests to be constructed. As of October 2017 it is fully implemented on the Octopus 900 and partially on the Heidelberg Edge Perimeter, the Kowa AP 7000, the CrewT imo and the Centervue Compass. It also has a cousin: the R package 'visualFields', which has tools for analysing and manipulating visual field data. License: GPL-3

Details

Package:	OPI
Type:	Package
Version:	2.9
Date:	2012-10-26
License:	GPL-3

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au> Maintainer: Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

See Also

visualFields

---

cdTodb *Convert cd/m^2 to dB*

Description

Given a value in cd/m^2, return the dB equivalent. The default is HFA dB scale (maxumum stimulus 10000 apostilbs).

Usage

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

Arguments

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

Value

Returns dB value.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

See Also

`dbTocd`

Examples

```
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` *Choose an implementation of the OPI.*

Description

Chooses an implementation of the OPI to use.

Usage

```
chooseOpi(opiImplementation)
```

Arguments

`opiImplementation`

A character string that is one of the following.

- `"SimNo"` for a simulator that always doesn't see.
- `"SimYes"` for a simulator that always does see.
- `"SimHenson"` for a simulator that uses a cummulative gaussian psychometric function with standard deviation according to Henson et al (2000) where variability increases as true threshold (Humphrey dB) value decreases.
- `"SimHensonRT"` as for SimHenson, but response times in ms are sampled from a supplied response time data set for each true positive response.
- `"SimGaussian"` for a simulator that uses a cummulative gaussian psychometric function with standard deviation supplied in opiInitialize().
- `"Octopus900"` for interfacing with the Octopus 900.
- `"Octopus900F310"` for interfacing with the Octopus 900 using Logitech F310 controller.
- `"Octopus600"` for interfacing with the Octopus 600.
- `"HEP"` not working so well in new HEPs.
- `"KowaAP7000"` for interfacing with Kowa AP-7000.
- `NULL` print a list of available OPI implementations.

Value

Returns TRUE if successful, FALSE otherwise.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

David B. Henson, Shaila Chaudry, Paul H. Artes, E. Brian Faragher, and Alec Ansons. Response Variability in the Visual Field: Comparison of Optic Neuritis, Glaucoma, Ocular Hypertension, and Normal Eyes. Investigative Ophthalmology & Visual Science, February 2000, Vol. 41(2).

A.M. McKendrick, J. Denniss and A. Turpin "Response times across the visual field: empirical observations and application to threshold determination". In submission, August 2013.

Please cite: 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.

http://perimetry.org/OPI

Examples

```
if(!chooseOpi("SimHenson"))
    warnings()
```

`dbTocd` *Convert 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 dB
`maxStim`	Stimulus value for 0dB in cd/m^2

Value

Returns cd/m^2 value.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

See Also

cdTodb

Examples

```
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 degrees on a visual field with the usual conventions
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 *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 4dB 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 occured).

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

Multilple 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 staircae has not finished, or one of `"Rev"` (finished due to 2 reversas), `"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) receieved at each call to `fourTwo.step`.
- responseTimes: Vector of response times receieved 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 (mena of last two reversals). This issues a warning if called before the staircase has finished, but still returns a value.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

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)))
    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()))
    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)
FT.final.details(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 4dB 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 occured).

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

Multilple 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 staircae 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) receieved at each call to `FT.step`.
- responseTimes: Vector of response times receieved 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 (indcluding both staircases if run).

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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", Investagative Ophthalmology and Vision Science 33 1993.

http://perimetry.org/OPI

See Also

dbTocd, opiPresent, fourTwo

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)))
    stop("opiInitialize failed")

result <- FT(makeStim=makeStim, tt=30, fpr=0.15, fnr=0.01)

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

    ###############################################
    # This section is for multiple FTs
    ###############################################
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()))
    warning("opiClose() failed")
```

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(TRUE, 0, 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 (3rd) 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.

responseHistory

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

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

keyHandler Function to get a keyboard input and returns as for opiPresent: list(seen=TRUE|FALSE, response time (in ms), error code) 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` funciton 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 that are location number in the first column, and the return values from `opiPresent()` and a record of fixation (if `checkFixationOK` present in stim objects returned from `makeStim`: 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: correct stimulus index
- column 5: TRUE/FALSE was fixating for all presentations in this trial according to `checkFixationOK`
- column 6...: columns from params
- ...: columns from opiPresent return

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

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("
}
```

```
## End(Not run)
```

`opiClose` *Close using OPI.*

Description

Generic function for closing the chosen OPI implementation that is set with `chooseOpi()`.

Usage

```
opiClose(...)
```

Arguments

`...`	Implementation specific parameters. See details.

Value

Returns NULL if close succeeded, otherwise an implementation dependant error.

Compass: Returns a list of `err`, which is an error code, and `fixations`, which is a matrix with 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), and one row per fixation.

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

See Also

`chooseOpi`

Examples

```
chooseOpi("SimGaussian")
if (!is.null(opiInitialize(sd=2)))
    stop("opiInitialize failed")
if (!is.null(opiClose()))
    stop("opiClose failed, which is very surprising!")
```

opiInitialize *Initialize OPI.*

Description

Generic function for initialization of the chosen OPI implementation that is set with `chooseOpi()`.

Usage

```
opiInitialize(...)
opiInitialise(...)
```

Arguments

...	Implementation specific parameters. See details.

Details

SimHenson: `opiInitialize(type="C",A=NA,B=NA,cap=6,display=NULL,maxStim=10000/pi)`

If the chosen OPI implementation is `SimHenson`, then `type` can be one of: `"N"`, for normal patients; `"G"`, for POAG patients; and `"C"`, for a combination. See Table 1 in Henson et al (2000). If `type` is `"X"` then `A` and `B` should be specified and are used in place of one of the three A/B combinations as in Henson et al (2000). `cap` is the maximum standard deviation value that the simulator will use for the slope/spread of the psychometric function.

If `display` is a vector of four numbers `c(xlow,xhi,ylow,yhi)`, then a plot area is created of dimension `xlim=range(xlow,xhi)` and `ylim=range(ylow,yhi)` and each call to `opiPresent` will display a point on the area. The color of the plot area can be set with `opiSetBackground`, and the color of the displayed point is determined by the stimulus passed to `opiPresent`.

`maxStim` is the maximum stimuls value in cd/m^2. This is used in converting cd/m^2 to dB values, and vice versa.

SimHensonRT: `opiInitialize(type="C",A=NA,B=NA,cap=6,display=NULL,maxStim=10000/pi,rtData,rtFP=1:1`

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"`, reponse 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.

SimGaussian: `opiInitialize(sd,display=NULL,maxStim=10000/pi)`

If the chosen OPI implementation is `SimGaussian`, then `sd` is the standard deviation value that the simulator will use for the slope/spread of the psychometric function.

`display` and `maxStim` is as for SimHenson.

Octopus900: `opiInitialize(serverPort=50001,eyeSuiteSettingsLocation,eye,gazeFeed=NA,bigWheel=FALSE`

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 (resposne). 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.

Octopus600: `opiInitialize(ipAddress,eye,pupilTracking=FALSE,pulsar=FALSE,eyeControl=0)`

If the chosen OPI implementation is `Octopus600`, then you must specify the IP address of the Octopus 600 and the eye to test.

`ipAddress` is the IP address of the Octopus 600 as a string.

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

`pupilTracking` is TRUE to turn on IR illumination and set pupil black level (which happens at the first stimulus presentation).

`pulsar` is TRUE for pulsar stimulus, FALSE for size III white-on-white.

`eyeControl`

- 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

KowaAP7000: `opiInitialize(ip,port)`

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

- `ipAddress` is the IP address of the AP-7000 server as a string.
- `port` is the TCP/IP port of the AP-7000 server as a number.

imo: `opiInitialize(ip,port)`

If the chosen OPI implementation is `imo`, then you must specify the IP address and port of the imo server.

- `ip` is the IP address of the imo server as a string.
- `port` is the TCP/IP port of the imo server as a number.

Compass: `opiInitialize(ip,port)`

If the chosen OPI implementation is `Compass`, then you must specify the IP address and port of the Compass server.

- `ip` is the IP address of the Compass server as a string.
- `port` is the TCP/IP port of the Compass server as a number.

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

DayDream: `opiInitialize(ip="127.0.0.1",port=50008,lut= rep(1000,256),degrees_to_pixels= function(x,y) return(50*c(x,y)))`

If the chosen OPI implementation is `Daydream`, then you must specify the IP address of the Android phone that is in the Daydream, and the port on which the server running on the phone is listening.

- `lut` is a vector of 256 luminance values, with `lut[i]` being the cd/m^2 value for grey level i.
- `degrees_to_pixels` is a function that takes in `x` and `y` in degrees and returns a vector of the (x,y) coordinates in pixels for one eye. This assumes the viewing distance (z-coordinate) is 30cm.

Value

Returns NULL if initialization succeeded, otherwise an implementation dependant error.

Octopus900: Returns NULL if successful, 1 if Octopus900 is already initialised by a previous call to `opiInitialize`, and 2 if some error occured that prevented ininitialisation.

The default background and stimulus setup is to white-on-white perimetry. Use `opiSetBackground` to change the background and stimulus colors.

Octopus600: Returns NULL if successful, or an Octopus 600 error code

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

Kowa AP-7000: Always returns NULL.

imo: Always returns NULL. Will `stop` if there is an error.

Compass: Returns 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.

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://people.eng.unimelb.edu.au/aturpin/opi/index.html

David B. Henson, Shaila Chaudry, Paul H. Artes, E. Brian Faragher, and Alec Ansons. Response Variability in the Visual Field: Comparison of Optic Neuritis, Glaucoma, Ocular Hypertension, and Normal Eyes. Investigative Ophthalmology & Visual Science, February 2000, Vol. 41(2).

See Also

chooseOpi, opiSetBackground, opiClose, opiPresent

Examples

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

    # Set up a simple simulation for white-on-white perimetry
    # and display the stimuli in a plot region
chooseOpi("SimHenson")
if (!is.null(opiInitialize(type="C", cap=6, display=c(-30,30,-30,30))))
    stop("opiInitialize failed")

    # 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=c(-30,30,-30,30), rtData=RtSigmaUnits, rtFP=1:100)
if (!is.null(oi))
    stop("opiInitialize failed")

    # Set up a simulation using a psychometric function that is
    # a cumulative gaussian of standard deviation 2
chooseOpi("SimGaussian")
if (!is.null(opiInitialize(sd=2)))
    stop("opiInitialize failed")

## Not run:
    # Set up the Octopus 900
chooseOpi("Octopus900")
if (!is.null(opiInitialize(
        eyeSuiteSettingsLocation="C:/ProgramData/Haag-Streit/EyeSuite/",
        eye="left")))
    stop("opiInitialize failed")

## End(Not run)

## Not run:
```

```
    # Set up the Kowa AP-7000
chooseOpi("KowaAP7000")
opiInitialize(ip="192.168.1.7", port=44965)

## End(Not run)

## Not run:
    # Set up the imo
chooseOpi("imo")
opiInitialize(ip="192.168.1.7", port=44965)

## End(Not run)

## Not run:
    # Set up the imo
chooseOpi("Compass")
result <- opiInitialize(ip="192.168.1.7", port=44965)
if (is.null(result$err)) {
    print(result$prl)
}

## End(Not run)
```

`opiKineticStimulus` *Stimulus parameter list.*

Description

List containing stimulus parameters with an S3 class attribute of `opiKineticStimulus`.

Details

The list should contain the following elements.

- `path` list of (x,y) coordinates in degrees that is usable by `xy.coords()`
- `image` `image[i]` is the image to display (in a machine specific format) in the section of the path specified by `path[i]..path[i+1]`.
- `levels` if `is.na(image)` then `levels[i]` is the stimulus level in cd/m^2 in the section of the path specified by `path[i]..path[i+1]`
- `sizes` `sizes[i]` is the size of stimulus (diameter in degrees) to use for the section of path specified by `path[i]..path[i+1]`, or a scaling factor for `images[i]`.
- `colors` `colors[i]` is the color to use for the stimulus in the section of path specified by `path[i]..path[i+1]`. Ignored if !is.na(image).
- `speeds` `speeds[i]` is the speed (degrees per second) for the stimulus to traverse the path specified by `path[i]..path[i+1]`.
- `...` machine specific parameters

Octopus 900: x and y are in degrees, with precision to three decimal places recognised.
image is not possible on an Octopus 900.
levels are in cd/m^2, and are rounded to the nearest one tenth of a dB for display.
colors are ignored. Use opiSetBackground() to alter stimulus color.
sizes are in degrees, but are rounded to the nearest Goldmann Size I..V for display.

Kowa AP 7000: Only a simple path with a start and an end point is supported by the AP-7000.
x and y are in degrees and should only be length 2. (precision?)
image is not possible on an Kowa AP 7000.
levels are in cd/m^2 in the range 0.03 to 3183, and are rounded to the nearest one tenth of a dB for display. (precision?)
colors one of .KowaAP7000Env$COLOR_WHITE, .KowaAP7000Env$COLOR_GREEN, .KowaAP7000Env$COLOR_BLUE, and .KowaAP7000Env$COLOR_RED.
sizes are in degrees, but are rounded to the nearest Goldmann Size I..V for display.
speeds are in degrees per second in the range 3 to 5.

Compass: Not implemented.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

Examples

```
    # A Size III white kinetic stimuli on
    # a bilinear path {(27,27), (15,20), (0,0)}
stim <- list(path=list(x=c(27,15,0), y=c(27,20,0)),
                sizes=rep(0.43,2),
                colors=rep("white",2),
                levels=rep(318,2),
                speeds=c(4,3))
class(stim) <- "opiKineticStimulus"
```

---

opiPresent *Use OPI to present stimulus.*

Description

Generic function for presentation of stimulus stim. Depending on your choice of OPI implementation set using chooseOpi(), different parameters are available for opiPresent.

Usage

```
opiPresent(stim, nextStim=NULL, ...)
```

Arguments

stim	A list of class opiStaticStimulus, opiKineticStimulus, or opiTemporalStimulus.
nextStim	As for stim, but the next presentation to be made. This might be useful on some machines, particularly projector based systems, where preparations for the next presentation can be made while waiting for a response to the current.
...	Parameters specific to your chosen opi implementation.

Details

opiPresent is blocking in that it will not return until either a response is obtained, or at least the responseWindow milliseconds has expired. (Note that more time might have expired.) Specifying nextStim allows the implementing machine to use the time waiting for a response to stim to make preparations for the next stimuli. (For example retargeting the projector or moving aperture and/or filter wheels.) There is no guarantee that the next call to opiPresent will have nextStim as the first argument; this could be checked by the machine specific implementations (but currently is not, I think).

Also note that to allow for different parameters depending on the implementation chosen with chooseOpi, every parameter MUST be named in a call to opiPresent.

SimHenson:

```
opiPresent(stim,nextStim=NULL,fpr=0.03,fnr=0.01,tt=30)
```

If the chosen OPI implementation is SimHenson, then the response to a stimuli is determined by sampling from a Frequency-of-Seeing (FoS) curve (also known as the psychometric function) with formula

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

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

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

The ceiling simH.global.cap is set with the call to opiInitialize, and A and B are from Table 1 in Henson et al (2000). Which values are used is determined by simH.type which is also set in the call to opiInitialize.

Note that if the stimulus value is less than zero, then the Henson formula is not used. The probability of seeing is fpr.

```
opiPresent(stim,nextStim=NULL,fpr=0.03,fnr=0.01,tt=NULL,criteria=0.95,rt_shape=5.3,rt_rate=1.4,rt_
```

For determinng seen/not-seen for kinetic, the first location (to a fidelity of 0.01 degrees) on the path (it only works for single paths now) where the probability of seeing is equal to criteria is found. If no such location exists, then the stimuli is not seen. The probability of seeing at each location is determined using a frequency-of-seeing curve defined as a cumulative Gaussian with parameters controlled by tt and opiInitialize. At each location along the path, the mean of the FoS is taken from the tt function, which takes a distance-along-path (in degrees) as an argument, and returns a dB value which is the static threshold at that distance along the path. Function tt

can return NA for not thresholds that are always not seen. At each location along the path, the standard deviation of the FoS is sampled from a Gaussion with mean taken from the formula of Henson et al (2000), as parametrised by `opiInitialize`, and standard deviation 0.25.

The location of a false positive response (for the total kinetic path) is sampled uniformly from the start of the path to the 'seeing' location, or the entire path if the stimuli is not seen.

Note that the false positive rate `fpr` and the false negative rate `fnr` are specified for the whole path, and not for the individual static responses along the way.

The actual location returned for a seen response is the location where the probability of seeing equals `criteria`, plus a response time sampled from a Gamma distribution parameterised by `rt_shape` and `rt_rate` and multiplied by `rt_scale`. That is: `rgamma(1,shape=rt_shape,rate=rt_rate) / rt_scale`.

SimHensonRT:

`opiPresent(stim,nextStim=NULL,fpr=0.03,fnr=0.01,tt=30,dist=stim$level -tt)`

For static stimuli, 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`.

SimGaussian:

`opiPresent(stim,nextStim=NULL,fpr=0.03,fnr=0.01,tt=30)`

If the chosen OPI implementation is `SimGaussian`, then the response to a stimuli is determined by sampling from a Frequency-of-Seeing (FoS) curve (also known as the psychometric function) with formula `fpr+(1-fpr-fnr)*(1-pnorm(x,tt,simG.global.sd))`, where `x` is the stimulus value in Humphrey dB, and `simG.global.sd` is set with `opiInitialize`.

SimYes:

`opiPresent(stim,nextStim=NULL)`

If the chosen OPI implementation is `SimYes`, then the response to a stimuli is always yes, hence `opiPresent` always returns `err=NULL`, `seen=TRUE`, and `time=0`.

SimNo:

`opiPresent(stim,nextStim=NULL)`

If the chosen OPI implementation is `SimNo`, then the response to a stimuli is always no, hence `opiPresent` always returns `err=NULL`, `seen=FALSE`, and `time=0`.

Octopus900F310:

`opiPresent(stim,nextStim=NULL)`

This functions as for the Octopus900, but responses are taken from the F310 Controller. 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.

Octopus600:

`opiPresent(stim,nextStim=NULL)`

If the chosen OPI implementation is `Octopus600`, then nextStim is ignored. If `eyeControl` is non-zero, as set in `opiInitialize`, answer codes describing patient state may arise (see `answer` field in the Value section).

KowaAP7000:

`opiPresent(stim,nextStim=NULL)`

If the chosen OPI implementation is `KowaAP7000`, then `nextStim` is ignored.

Compass:

`opiPresent(stim,nextStim=NULL)`

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.

DayDream: If the chosen OPI implementation is `DayDream`, then `nextStim` is ignored. Note that the dB level is rounded to the nearest cd/m^2 that is in the `lut` specified in `opiInitialise`.

Currently uses the most simple algorithm for drawing a 'circle' (ie not Bresenham's).

Currently only implemented for `opiStaticStimulus`.

Value

A list containing

`err` `NULL` if no error occurred, otherwise a machine specific error message. This should include errors when the specified size cannot be achieved by the device (for example, in a projection system with an aperture wheel of predefined sizes.) If `stim` is `NULL`, then `err` contains the status of the machine.

`seen` `TRUE` if a response was detected in the allowed `responseWindow`, `FALSE` otherwise. (Note, see Octopus900F310 above).

`time` The time in milliseconds from the onset (or offset, machine specific) of the presentation until the response from the subject if `seen` is `TRUE`. If `seen` is `FALSE`, this value is undefined. For kinetic perimetry on the O900, this value is unknown...

`answer` Only returned for `Octopus600`. Can be the following values:

- 0 = stimulus not seen;
- 1 = stimulus seen;
- 132 = Response button was pressed before stimulus presentation (Patient needs a break - hold on examination);
- 36 = Eye is closed before stimulus presentation;
- 68 = Fixation lost before stimulus presentation (pupil center is out of green window in video image);
- 260 = Forehead rest lost before stimulus presentation;
- 516 = Fast Eye movements before stimulus presentation;
- 258 = Forehead rest lost during stimulus presentation;
- 66 = Fixation lost during stimulus presentation (pupil center is out of green window in video image);
- 34 = Eye was closed during stimulus presentation;
- 18 = Patient answer was too early (<=100ms after stimulus presentation) - lucky punch;

- 514 = Fast Eye movements during stimulus presentation

`pupilX` Only returned for KowaAP7000 and an opiStaticStimulus or O900 and staic/kinetic. x-coordinate of centre of pupil in pixels during presentation.

`pupilY` Only returned for KowaAP7000 and an opiStaticStimulus or O900 and staic/kinetic. y-coordinate of centre of pupil in pixels during presentation.

`purkinjeX` Only returned for KowaAP7000 and an opiStaticStimulus. x-coordinate of centre of Purkinje Image in pixels during presentation.

`purkinjeY` Only returned for KowaAP7000 and an opiStaticStimulus. y-coordinate of centre of Purkinje Image in pixels during presentation.

`x` Only returned for KowaAP7000 or Octopus900 and an opiKineticStimulus. x coordinate of stimuli when button is pressed.

`y` Only returned for KowaAP7000 or Octopus900 and an opiKineticStimulus. y coordinate of stimuli when button is pressed.

`time_rec` Only returned for Compass. Time since epoch that the opiPresent command was received by the Compass in ms.

`time_hw` Only returned for Compass. Hardware time of button press or response window expired (integer ms). To get the hardware time that a presentation began, subtract responseWindow from `th` (for aligning with fixation data returned by `opiClose()`.

`time_resp` Only returned for Compass. Time since epoch that the response was received or response window expired (in ms).

`num_track_events` Only returned for Compass. The number of tracking events associated with this presentation.

`num_motor_fails` Only returned for Compass. The number of time the motor could not keep pace with eye movements.

`pupil_diam` Only returned for Compass. The diameter of the pupil on milimetres on presentation.

`loc_x` Only returned for Compass. The x location in pixels of the presentation on the retinal image returned by `opiInitialize`.

`loc_y` Only returned for Compass. The y location in pixels of the presentation on the retinal image returned by `opiInitialize`.

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://people.eng.unimelb.edu.au/aturpin/opi/index.html

David B. Henson, Shaila Chaudry, Paul H. Artes, E. Brian Faragher, and Alec Ansons. Response Variability in the Visual Field: Comparison of Optic Neuritis, Glaucoma, Ocular Hypertension, and Normal Eyes. Investigative Ophthalmology & Visual Science, February 2000, Vol. 41(2).

See Also

opiStaticStimulus, opiKineticStimulus, opiTemporalStimulus, chooseOpi opiInitialize

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, 10000/pi), size=0.43, color="white",
              duration=200, responseWindow=1500)
    class(s) <- "opiStaticStimulus"

    return(s)
}
chooseOpi("SimHenson")
if (!is.null(opiInitialize(type="C", cap=6)))
    stop("opiInitialize failed")

result <- opiPresent(stim=makeStim(10,0), tt=30, fpr=0.15, fnr=0.01)

    # Will not work as 'stim' is not named
#result <- opiPresent(makeStim(10,0), tt=30, fpr=0.15, fnr=0.01)

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

    # Same but with simulated reaction times
chooseOpi("SimHensonRT")
data(RtSigmaUnits)
if (!is.null(opiInitialize(type="C", cap=6, rtData=RtSigmaUnits)))
    stop("opiInitialize failed")

dist <- (10 - 30)/min(exp(-0.098 * 30 + 3.62), 6)
result <- opiPresent(stim=makeStim(10,0), tt=30, fpr=0.15, fnr=0.01, dist=dist)

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

---

opiQueryDevice *Query device using OPI.*

Description

Generic function for getting details of the chosen OPI implementation that is set with `chooseOpi()`.

Usage

```
opiQueryDevice(...)
```

Arguments

...	Implementation specific parameters. See details.

Details

Octopus600: If the chosen OPI is `Octopus600`, then this function returns information about the patient. See the Value section for details.

KowaAP7000: If the chosen OPI is `KowaAP7000`, then this function returns the current location of the pupil. See the Value section for details.

Daydream: Returns all constants in `.DayDreamEnv` as a list.

Value

Returns a list that contains `isSim` and implementation dependant data.

1. `isSim` is `TRUE` if the device is a simulation, or `FALSE` if the device is a physical machine.

Octopus600: Returns a list of 10 items:

1. answerButton [0 = not pressed, 1 = pressed]
2. headSensor [0 = no forehead detected, 1 = forehead detected]
3. eyeLidClosureLeft [0 = eye is open, 1 = eye is closed]
4. eyeLidClosureRight [0 = eye is open, 1 = eye is closed]
5. fixationLostLeft [1 = eye pos lost, 0 = eye pos ok)
6. fixationLostRight [1 = eye pos lost, 0 = eye pos ok)
7. pupilPositionXLeft [in px]
8. pupilPositionYLeft [in px]
9. pupilPositionXRight [in px]
10. pupilPositionYRight [in px]

KowaAP7000: Returns a list of 4 items:

- `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.

It also prints a list of constants that OPI knows about for the AP-7000.

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

See Also

chooseOpi

Examples

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

---

opiSetBackground *Set background using OPI.*

Description

Generic function for setting background of the chosen OPI implementation that is set with `chooseOpi()`.

Usage

```
opiSetBackground(...)
```

Arguments

...	Implementation specific parameters. See details.

Details

Octopus900:

`opiSetBackground(lum=NA,color=NA,fixation=NA,fixIntensity=NA)`

Allowable `lum` and `color` are defined in the .Octopus900Env environment.

- `lum` is intensity of the background and can be one of
  - `.Octopus900Env$BG_OFF`, which turns background off.
  - `.Octopus900Env$BG_1`, background of 1.27 cd/m^2.
  - `.Octopus900Env$BG_10`, background of 10 cd/m^2.
  - `.Octopus900Env$BG_100`, background of 100 cd/m^2.
- `color` can be one of the following choices.
  - `.Octopus900Env$MET_COL_WW` for white-on-white
  - `.Octopus900Env$MET_COL_RW` for red-on-white
  - `.Octopus900Env$MET_COL_BW` for blue-on-white
  - `.Octopus900Env$MET_COL_WY` for white-on-yellow
  - `.Octopus900Env$MET_COL_RY` for red-on-yellow
  - `.Octopus900Env$MET_COL_BY` for blue-on-yellow
- `fixation` is one of
  - `.Octopus900Env$FIX_CENTRE` or `.Octopus900Env$FIX_CENTER`
  - `.Octopus900Env$FIX_CROSS`

  - `.Octopus900Env$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`.

SimHenson and SimGaussian:

`opiSetBackground(col,gridCol)`

`col` is the background color of the plot area used for displaying stimuli, and `gridCol` the color of the gridlines. Note the plot area will only be displayed if `opiInitialize` is called with a valid display argument.

Octopus600:

This function has no effect.

KowaAP7000:

`opiSetBackground(lum,color,fixation,)`

`lum` and `color` are dependant for the Kowa AP-7000. A white background must be 10 cd/m^2, and a yellow background must be. 100 cd/m^2. If `lum` is 10 and `color` is not set, then `.KowaAP7000Env$BACKGROUND_WHITE` is assumed. If `lum` is 100 and `color` is not set, then `.KowaAP7000Env$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`).

`fixation` is one of

- `.KowaAP7000Env$FIX_CENTER`, fixation marker in the centre.
- `.KowaAP7000Env$FIX_CENTRE`, fixation marker in the centre.
- `.KowaAP7000Env$FIX_AUX`, fixation marker is ???.
- `.KowaAP7000Env$FIX_MACULA`, fixation marker is a circle(?).
- `.KowaAP7000Env$FIX_AUX_LEFT`, fixation marker is as for AUX but only lower left.

Compass: `opiSetBackground(fixation=NA,tracking_on=NA)`

- `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` is either 0 (tracking off) or 1 (tracking on).

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.

Daydream: `opiSetBackground(lum=NA,color=NA,fixation="Cross",fixation_size=21,fixation_color=c(0,1`

- `lum` in cd/m^2 is set to nearest grey value in `lut` from `opiInitialize`.
- `color` is ignored.
- `fixation` can only be `'Cross'` at the moment.
- `fixation_size` is the number of pixels one cross-hair is in length.
- `fixation_color` RGB value of coor of fixation cross. Values in range [0,255].

Value

Returns NULL if succeeded, otherwise an implementation dependant error as follows.

Octopus900:

-1 indicates `opiInitialize` has not been called.

-2 indicates could not set the background color.

-3 indicates could not set the fixation marker.

-4 indicates that all input parameters were NA.

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://people.eng.unimelb.edu.au/aturpin/opi/index.html

See Also

`chooseOpi`

Examples

```
chooseOpi("SimGaussian")
if (!is.null(opiInitialize(sd=2, display=c(-30,30,-30,30))))
    stop("opiInitialize failed")
if (!is.null(opiSetBackground(col="white",gridCol="grey")))
    stop("opiSetBackground failed, which is very surprising!")

## Not run:
chooseOpi("Octopus900")
oi <- opiInitialize(eyeSuiteJarLocation="c:/EyeSuite/",
        eyeSuiteSettingsLocation="c:/Documents and Settings/All Users/Haag-Streit/",
        eye="left")
if (!is.null(oi))
    stop("opiInitialize failed")
if (!is.null(opiSetBackground(fixation=.Octopus900Env$FIX_CENTRE)))
    stop("opiSetBackground failed")
if (!is.null(opiSetBackground(fixation=.Octopus900Env$FIX_RING, fixIntensity=0)))
    stop("opiSetBackground failed")
if (!is.null(opiSetBackground(color=.Octopus900Env$MET_COL_BY)))
    stop("opiSetBackground failed")
if (!is.null(opiSetBackground(lum=.Octopus900Env$BG_100, color=.Octopus900Env$MET_COL_RW)))
    stop("opiSetBackground failed")
opiClose()

## End(Not run)
```

`opiStaticStimulus` *Stimulus parameter list.*

Description

List containing stimulus parameters with an S3 class attribute of `opiStaticStimulus`.

Details

The list should contain the following elements.

`x` coordinate of the center of stimulus in degrees relative to fixation

`y` coordinate of the center of stimulus in degrees relative to fixation

`image` an image to display in a machine specific format

`level` stimulus level in cd/m^2 (ignored if !is.na(image))

`size` diameter of target in degrees, or scaling factor for image if specified

`color` machine specific stimulus color settings (ignored if !is.na(image))

`duration` total stimulus duration in milliseconds maximum

`responseWindow` time (>= 0) in milliseconds to wait for a response from the onset of the stimulus presentation

`...` machine specific parameters

SimHenson and SimGaussian: Only `level` is used. Duration and location are ignored, `color` is assumed "white" and `size` is assumed to be 26/60 (Goldmann III).

Octopus 900: `x` and `y` are in degrees, with precision to one decimal place recognised.
`image` is not possible on an Octopus 900.
`level` is in cd/m^2, and is rounded to the nearest one tenth of a dB for display.
`color` is ignored. Use `opiSetBackground()` to alter stimulus color.
`checkFixationOK` is a function that takes the return value from `opiPresent` and returns either `TRUE`, indicating that fixation was food for the presentation; or `FALSE`, indicating that fixation was not good for the presentation.

Octopus 900 F310 Controller: As for the Octopus 900, but a `responseWindow` of -1 means that the Octopus 900 server will wait until either the L and R button is pressed in the controller until returning.

Kowa AP 7000: `x` and `y` are in degrees. (precision?)
`image` is not possible on an Kowa AP 7000.
`level` are in cd/m^2 in the range 0.03 to 3183, nearest one tenth of a dB for display.
`size` is in degrees, but is rounded to the nearest Goldmann Size I..V for display.
`color` one of `.KowaAP7000Env$COLOR_WHITE`, `.KowaAP7000Env$COLOR_GREEN`, `.KowaAP7000Env$COLOR_BLUE`, and `.KowaAP7000Env$COLOR_RED`.

imo:

`x`, `y`, `level`, `size`, and `color` are not used.

`image` is a list of two matrices: the first for the right eye, the second for the left. Each image is a 1080x1080 matrix with each element in the range 0 to 80, which maps onto 0dB to 40dB in steps of 0.5dB. Thus 0 is 0dB, 3283.048 cd/m^2; 1 is 0.5dB; and 80 is 40dB, 10 cd/m^2.

`tracking` is `TRUE` if auto image placement to keep pupil centred is used, or `FALSE` to turn off imo auto-image placement to keep centred on pupil.

Compass: `x` and `y` are in degrees (floating point) (range [-30,30]).

`level` is in cd/m^2, and is rounded to the nearest whole dB for display (range 0 to 50). 0dB is 10000aps.

`responseWindow` is in millliseconds (range 0 to 2680).

Parameter `duration` is assumed to be 200ms, `size` is assumed to be Goldmann III (0.43), and `color` is assumed to be white.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://people.eng.unimelb.edu.au/aturpin/opi/index.html

See Also

`opiSetBackground`

Examples

```
stim <- list(x=9, y=9, image=NA, 314, size=0.43, color="white",
             duration=200, responseWindow=1500)
class(stim) <- "opiStaticStimulus"
```

`opiTemporalStimulus` *Stimulus parameter list.*

Description

List containing stimulus parameters with an S3 class attribute of `opiTemporalStimulus`.

Details

- x coordinate of the center of stimulus in degrees relative to fixation
- y coordinate of the center of stimulus in degrees relative to fixation
- image an image to display in a machine specific format
- lut if is.na(image) then this is a lookup table (vector) for stimulus level at each step of rate Hz in cd/m^2. If image is specified, then this is a list of images, in the same format as image, that is stepped through at rate Hz.
- size diameter of target in degrees, or scaling factor for image if specified
- color machine specific stimulus color settings (ignored if !is.na(image))
- rate frequency with which lut is processed in Hz
- duration total length of stimulus flash in milliseconds. There is no guarantee that duration %% length(lut)/rate == 0. That is, the onus is on the user to ensure the duration is a multiple of the period of the stimuli.
- responseWindow maximum time (>= 0) in milliseconds to wait for a response from the onset of the stimulus presentation
- ... machine specific parameters

Octopus 900: x and y are in degrees, with precision to one decimal place recognised.
image is not possible on an Octopus 900.
lut is not possible on an Octopus 900. Stimulus is at 0 dB.
rate is in Hz, with precision to one decimal place recognised.
color is ignored. Use opiSetBackground() to alter stimulus color.

Kowa AP-7000: Not supported.

Kowa AP-7000: Not implemented.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References

Please cite: 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.

http://perimetry.org/OPI

Examples

```
    # A Size III flickering with a 10Hz square wave at
    # location (7,7) with luminance 10 dB (HFA)
stim <- list(x=7, y=7, size=0.43, color="white",
               rate=20,        # one lut step per 50 ms
               lut=c(0,318),   # so one full lut per 100 ms == 10Hz
               duration=400,   # and 4 cycles per stimulus
               responseWindow=1500)
class(stim) <- "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
```

RtDbUnits *Response times to white-on-white Goldmann Size III targets for 12 subjects.*

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

```
data(RtDbUnits)
```

Format

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.

Source

A.M. McKendrick, J. Denniss and A. Turpin. "Response times across the visual field: empirical observations and application to threshold determination" In preparation, Aug 2013.

References

A.M. McKendrick, J. Denniss and A. Turpin. "Response times across the visual field: empirical observations and application to threshold determination" In preperation, Aug 2013.

Examples

```
data(RtDbUnits)
```

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

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

```
data(RtSigmaUnits)
```

Format

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.

Source

A.M. McKendrick, J. Denniss and A. Turpin. "Response times across the visual field: empirical observations and application to threshold determination" In preparation, Aug 2013.

References

A.M. McKendrick, J. Denniss and A. Turpin. "Response times across the visual field: empirical observations and application to threshold determination" In preperation, Aug 2013.

Examples

```
data(RtSigmaUnits)
```

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-pnorm(domain,tt,1))}),
     stopType="S", stopValue=1.5,
     minStimulus=head(domain,1),
     maxStimulus=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-pnorm(domain,tt,1))}),
     stopType="S", stopValue=1.5,
     minStimulus=head(domain,1),
     maxStimulus=tail(domain,1),
     maxSeenLimit=2, minNotSeenLimit=2,
     maxPresentations=100,
     makeStim,
     stimChoice="mean",
     ...)
ZEST.step(state, nextStim=NULL)
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)`.

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

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

`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`
See `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.

`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`.

Details

This is an implementation of King-Smith et al.'s ZEST procedure and Watson and Pelli's QUEST procedure. All presentaions 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 occured).

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.
- opiRespA 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
- 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.
- 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).
- opiRespA list of responses received from each call to opiPresent within ZEST.step.

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.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

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.

Please cite: 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.

http://perimetry.org/OPI

See Also

dbTocd, opiPresent

Examples

```
chooseOpi("SimHenson")
if (!is.null(opiInitialize(type="C", cap=6)))
    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()))
    warning("opiClose() failed")
```

Index