Package ‘OPI’

July 14, 2016

Type Package
Title Open Perimetry Interface
Version 2.5
Date 2016-6-29
Author Andrew Turpin, David Lawson, Matthias Muller
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 February 2016 it is fully implemented on the Octopus 600 and Octopus 900 and partially on the Heidelberg Edge Perimeter, the Kowa AP 7000 and the CrewTimo. It also has a cousin: the R package visualFields, which has tools for analysing and manipulating visual field data.
License GPL-3
URL http://perimetry.org/OPI
Depends methods
NeedsCompilation no
Repository CRAN
Date/Publication 2016-07-14 10:14:50

R topics documented:

OPI-package ................................................. 2
cdTodb .................................................. 3
chooseOpi .............................................. 4
dbTocd .................................................. 5
fourTwo .................................................. 6
FT .......................................................... 9
opiClose ................................................. 12
opiInitialize .......................................... 13
opiKineticStimulus ................................. 17
Description

This package contains the current version of the Open Perimetry Interface (OPI), for 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 July 2014 it is fully implemented on the Octopus 900 and Octopus 600, and partially on the Heidelberg Edge Perimeter. For the HEP, you will need the Rheap package and permission from Heidelberg Engineering. For the Octopus 900, you will need the O900Server.jar program and for both Octopus machines you will need permission from Haag-Streit. See the OPI www site for details. There is also a cousin: the R package visualFields, which has tools for analysing and manipulating visual field data.

Details

Package: OPI
Type: Package
Version: 1.4
Date: 2012-10-26
License: GPL-3

Author(s)

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

References


http://perimetry.org/OPI
See Also

visualFields

---

\textit{cdTodb} \hspace{1cm} \textit{Convert cd/m}^2 \textit{to dB}

\section{Description}

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

\section{Usage}

\texttt{cdTodb(cd, maxStim=10000/pi)}

\section{Arguments}

- \texttt{cd} \hspace{1cm} Value to convert in cd/m$^2$
- \texttt{maxStim} \hspace{1cm} Stimulus value for 0dB in cd/m$^2$

\section{Value}

Returns dB value.

\section{Author(s)}

Andrew Turpin <aturpin@unimelb.edu.au>

\section{References}


http://perimetry.org/OPI

See Also

\texttt{dbTocd}

\section{Examples}

\begin{verbatim}
  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
\end{verbatim}
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 cumulative 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 cumulative 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


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.


http://perimetry.org/OPI

Examples

```r
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


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

---

**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

**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) 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 (mean 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


http://perimetry.org/OPI

## See Also

dbTocd, opiPresent

## Examples

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

chooseopi <- function(dbt, n) {
  if (!is.null(opiInitialize(type= DT, cap=6)))
    stop("opiInitialize failed")

  # This section is for multiple fourTows
  makeStimHelper <- function(dbt, n, x, y) {
    # returns a function of (db,n)
    ff <- function(dbt, n) db+n

    body(ff) <- substitute(
      {s <- list(x=x, y=y, level=dbTocd(dbt), 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(dbt, 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(lst) # choose a random,
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

**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 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) received 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 (including both staircases if run).

Author(s)
Andrew Turpin <aturpin@unimelb.edu.au>

References
http://perimetry.org/OPI

See Also
dbTocd, opiPresent

Examples
```r
# 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")
```
### opiClose

**Description**

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

**Function**

```r
# 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(0,-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")
```
opiInitialize

Usage

opiClose(...)

Arguments

... Implementation specific parameters. See details.

Value

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

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References

http://perimetry.org/OPI

See Also

choose0pi

Examples

choose0pi("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 choose0pi().
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 stimulus value in cd/m². This is used in converting cd/m² to dB values, and vice versa.

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

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.

**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(eyeSuiteSettingsLocation, eye, gazeFeed=0, bigWheel=FALSE, pres_buzzer=0, resp_buzzer=0)`

If the chosen OPI implementation is Octopus900, then you must specify a directory and the eye to be tested. 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 0 for no gaze tracking information. If gazeFeed is 1, then a single frame is returned as part of the value from opiPresent which is the most recent frame captured. If gazeFeed is 2, then all frames containing the asterix are returned as part of the value from opiPresent, which is the frames captured while a static stimulus is displayed. If gazeFeed is greater than zero, a Java driven window appears containing the live feed from the Octopus 900 gaze camera. 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.

**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.

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

**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.

**Author(s)**

Andrew Tuprin <aturpin@unimelb.edu.au>
References


http://perimetry.org/OPI


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(  eyeSuiteJarLocation="C:/Program Files (x86)/Haag-Streit/EyeSuite/",
                          eyeSuiteSettingsLocation="C:/ProgramData/Haag-Streit/EyeSuite/",
                          eye="left")))
  stop("opiInitialize failed")

## 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/m² 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/m², 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.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References


http://perimetry.org/OPI

Examples

```r
# 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=rep(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

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

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

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

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

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

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 proba-
bility of seeing is \(fpr\).

SimHensonRT:

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

This presentation 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.

**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.

- `frames` Only returned for Octopus 900. An array of integer 0..255 for gaze image. In version 1.5, only a single frame is returned. See the parameter gazeFeed for opiInitialize to control what frames are returned.

- `numframes` Only returned for Octopus 900. The number of frames in frames.

- `width` Only returned for Octopus 900. Width of frame in frames.
height

Only returned for Octopus 900. Height of frame in frames.

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. x-coordinate of centre of pupil in pixels during presentation.

pupilY

Only returned for KowaAP7000 and an opiStaticStimulus. 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.

Author(s)

Andrew Tuprin <aturpin@unimelb.edu.au>

References


http://perimetry.org/OPI

opiQueryDevice

Query device using OPI.

Description

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

Usage

opiQueryDevice(...)

See Also

opiStaticStimulus, opiKineticStimulus, opiTemporalStimulus, chooseOpi

Examples

# Stimulus is Size III white-on-white as in the HFA
makeStim <- function(db, n) {
  s <- list(x=9, y=9, level=dbToDb(db, 10000/pi), size=0.43, color="white",
            duration=200, responseWindow=1500)
  class(s) <- "opiStaticStimulus"
  return(s)
}
chooseOpi("SimHenson")
if (!is.null(chooseOpi(type="C", cap=6)))
  stop("chooseOpi 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(chooseOpi(type="C", cap=6)))
  warning("chooseOpi () failed")

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

Value

Returns implementation dependant data.

**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.
- publiy, 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


http://perimetry.org/OPI

See Also

chooseOpi
Examples

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

**Descrip**

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

**Usage**

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

**Arguments**

... Implementation specific parameters. See details.

**Details**

**Octopus900:**

```r
opiSetBackground(lum, color="white", background=NA, fixation=NA, fixIntensity=50)
```

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

- `lum` is intensity of the background and must 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.
**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/m², and a yellow background must be 100 cd/m². 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.

**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.

**Author(s)**

Andrew Tuprin <aturpin@unimelb.edu.au>

**References**


http://perimetry.org/OPI

**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")
oipIClose()

## End(Not run)

opiStaticStimulus    Stimulus parameter list.

Description

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

Details

The list must 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
**opiStaticStimulus**

**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/m², and is rounded to the nearest one tenth of a dB for display. color is ignored. Use opiSetBackground() to alter stimulus color.

**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/m² 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/m²; 1 is 0.5dB; and 80 is 40dB, 10 cd/m². 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.

**Author(s)**

Andrew Turpin <aturpin@unimelb.edu.au>

**References**


http://perimetry.org/OPI

**See Also**

opiSetBackground

**Examples**

```r
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/m². 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)`) color is ignored. Use `opiSetBackground()` to alter stimulus color.
- rate frequency with which lut is processed in Hz rate is in Hz, with precision to one decimal place recognised.
- 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.

Author(s)

Andrew Turpin <aturpin@unimelb.edu.au>

References


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"

---

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


References


Examples

data(RtDbUnits)
**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 cumulative 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**


**References**


**Examples**

data(RtSigmaUnits)
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,
    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 regardless of stopType.
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.

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).

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. 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, row 2 is 1/0 for seen/not-seen.
- 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.
**Multiple 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.

ZEST.step returns a list containing

- **state:** The new state after presenting a stimulii 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**


http://perimetry.org/OPI

**See Also**

dbToC, opiPresent

**Examples**

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

################################################################################
# This section is for single location ZESTs

```r
ZEST
```

# Stimulus is Size III white-on-white as in the HFA

```r
def makeStim <- function(db, n) {
  s <- list(x=9, y=9, size=dbToCcd(db), level=0.43, color="white",
            duration=200, responseWindow=1500)
  class(s) <- "opisStaticStimulus"
  return(s)
}
```

```r
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)
```

```r
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",])))
```

```r
# This section is for multiple ZESTs

```r
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, size=dbToCcd(db), level=0.43, color="white",
               duration=200, responseWindow=1500)
     class(s) <- "opisStaticStimulus"
     return(s)
    }, list(x=x,y=y))
  return(ff)
}
```

```r
# 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))
```

```r
# 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, unstopped state
  r <- ZEST.step(st)
  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 %d %d, locations[[i]][1], locations[[i]][2])
  #cat(sprintf("has threshold %4.2f\n", finals[[i]]))
}

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

*Topic datasets
RtDbUnits, 29
RtSigmaUnits, 30

*Topic misc
cdTodb, 3
chooseOpi, 4
dbTocd, 5
fourTwo, 6
FT, 9
opiClose, 12
opiInitialize, 13
opiKineticStimulus, 17
opiPresent, 18
 opiQueryDevice, 22
 opiSetBackground, 24
 opiStaticStimulus, 26
 opiTemporalStimulus, 28
 ZEST, 31

*Topic package
OPI-package, 2

cdTodb, 3, 6
chooseOpi, 4, 13, 16, 22, 23, 25

dbTocd, 3, 5, 8, 11, 33

fourTwo, 6
FT, 9

OPI (OPI-package), 2
OPI-package, 2
opiClose, 12, 16
opiInitialise (opiInitialize), 13
opiInitialize, 13, 20
opiKineticStimulus, 17, 19, 22
opiPresent, 8, 11, 14, 16, 18, 19, 20, 33
opiQueryDevice, 22
 opiSetBackground, 16, 24, 27
 opiStaticStimulus, 19, 22, 26
 opiTemporalStimulus, 19, 22, 28

QUEST (ZEST), 31
RtDbUnits, 29
RtSigmaUnits, 30
ZEST, 31