Package ‘zernike’

December 13, 2025

Title Zernike Polynomials and Fringe Analysis tools

Version 3.8.2

Author M.L. Peck <mpeck1@ix.netcom.com>

Depends R (>= 4.1.0)

Suggests rgl, robustbase, clue, data.table, dplyr, pixmap, mvtnorm, tinytable, jpeg, tiff, png

Imports Rcpp (>= 1.0.0), RcppParallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel, BH

Description Routines for Manipulation of Zernike polynomials and Interferogram fringe analysis

Maintainer M.L. Peck <mlpeck54@gmail.com>

License MIT, GPL, and BSD

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Contents

addfit 3
aiapsi 5
astig.bath 6
brcutpuw 7
circle.auto 8
circle.hough 9
circle.man 11
col3d 12
conv2_sep_complex 13
conv2_sep_real 13
convolve2d 14
crop 15
d_of_g 15
fft 16

fftfit 17
fftshift 18
FFTUtilities 19
fft_cx 20
fft_cx_pad 21
fft_pad 21
fitzernikes 22
foucogram 23
gblur 24
gblur_complex 25
gblur_fft 25
gol_welsch 26
gpcapsi 26
gradzpm 27
gray256 29
hypot 29
idiffpuw 30
id_dxy_uw 31
id_uw 32
ifft 32
ifftshift 33
ifft_real 34
load.images 34
lspsi 35
makezlist.iso 36
norm_zpm 37
pcapsi 37
pick.sidelobe 38
plot.cmat 39
plot.pupil 40
plotn 41
plotxs 42
psifit 43
pupil 45
pupil.rhotheta 47
pupilrms 48
PVr 49
qpuw 50
q_uw 51
rescale 52
rmap 53
rygcb 54
rzernike_ann 55
rzernike_ann_direct 56
sconic 57
startest 58
summary.wf_zfit 59
synth.interferogram 60

turbwf 61
vortexfit 62
wf3d.pupil 64
wf_net 65
zapm 67
zapm_128 69
zapm_direct 71
zapm_iso 71
zapm_iso_128 73
zapm_iso_direct 75
zconic 76
Zernike 77
zernike_options 79
zlist 81
zmoments 82
zpm 83
zpm_cart 84

Index **85**

`addfit` *Separate wavefronts*

Description

Separate a set of wavefronts measured at different orientations into "polished in" and instrumental or "test stand" components.

Usage

```
addfit(..., th = 0, zcm = NULL, theta = numeric(0))

separate.wf(
  zcm,
  theta,
  maxorder = 14,
  nrow = nrow.default,
  ncol = ncol.default,
  cp = cp.default
)
```

Arguments

`...`	one or more wavefront fits as returned by psifit, vortexfit, fftfit, or wf_net.
`th`	the orientation angle(s) of object under test, in degrees.
`zcm`	Zernike coefficient matrix from addfit.
`theta`	vector of angles from addfit.

`maxorder`	the maximum Zernike polynomial order of the fits.
`nrow`	number of rows in the reconstructed wavefronts.
`ncol`	number of columns in the reconstructed wavefronts.
`cp`	list of values describing the location and size of the wavefront as returned by circle.auto or circle.man.

Details

The two functions `addfit()` and `separate.wf()` work together to provide a rudimentary project management capability when multiple optical tests have been run on an optical system rotated to one or more orientations. All `addfit()` does is extract the net Zernike coefficients in the list of values returned by psifit, vortexfit, fftfit, or wf_net. It also accepts one or more orientation angles in degrees.

The odd parameter list is to provide some flexibility in data entry. For example if several fits are available they could be entered as a group with the `thetas` entered as a vector the same length as the number of fits. Alternately fits could be entered one at a time. If the latter strategy is followed be sure to recycle the variable name that the return is assigned to. Also, the Zernike polynomial fits should be made to the same polynomial order. This isn't checked and will surely cause an error if different fit orders are used.

`separate.wf()` makes use of the properties of Zernike polynomials under rotations to disentangle the contributions from the "mirror" and the instrument or test stand to the extent possible. Least squares fits are performed for each non-axysmmetric aberration and some possibly useful summary statistics from the fits are returned in the data frames `zcb` and `sumstats`.

Value

Lists with the following elements

function `addfit`

zcm Zernike coefficient matrix from the `zcoef.net` entry in the wavefront fits, minus the first 3 elements.

function `separate.wf`

zcb a data frame with zernike coefficients and standard errors of estimates for intrinsic and instrumental aberrations.

sumstats a data frame with summary statistics describing the fits to each set of coefficients.

wf.mirror the estimated derotated wavefront, stored in a matrix of size nrow x ncol with class `pupil`.

wf.inst the estimated instrumental (test stand) contribution to the wavefront.

Functions

- `addfit()`: Add a wavefront fit to tracking list
- `separate.wf()`: Separate wavefronts

aiapsi	*Iterative algorithms for PSI with unknown phase shifts*

Description

Three iterative algorithms for PSI with unknown phase shifts.

Usage

```
aiapsi(im.mat, phases, ptol = 0.001, maxiter=20, trace=1)
aiapsiC(im.mat, phases_init, ptol, maxiter, trace)
hkpsi(im.mat, phases, maxiter = 20, ptol = 0.001,
  trace = 1, plotprogress = TRUE)
tiltpsi(im.mat, phases, coords, ptol = 0.01, maxiter = 20, trace = 1)
tiltpsiC(im.mat, phases, coords, ptol, maxiter, trace)
```

Arguments

`im.mat`	a *matrix* of interferogram values
`phases`	Starting guess for phase shifts
`ptol`	Convergence criterion for phase shifts
`maxiter`	Maximum number of iterations
`trace`	Boolean: Print some summary data at each iteration.
`plotprogress`	Plot some summary data for each iteration?

Also, for `tiltpsi` and `tiltpsiC`

`coords`	Low order Zernike polynomial matrix

Details

The “variable tilt“ algorithm now allows an indefinite number of low order Zernike terms to be variable between phase steps. `coords` can be created with `zpm` setting `maxorder` to a small value, say 4, discarding the first (dc) column and retaining as many as desired. There must be at least two columns for tilts. The third will be defocus, the next two astigmatism, the next two primary coma, ...

`aiapsi` and `tiltpsi` are wrappers for the calls to the C++ code in `aiapsiC` and `tiltpsiC` with sensible defaults for `ptol`, `maxiter`, and `trace`.

Value

A list containing the following elements:

`phi`	The wrapped phase estimate. This is a vector as long as the number of rows in `im.mat`.
`mod`	Modulation estimate.

`phases`	Phase shift estimates.
`iter`	Number of iterations.
`sse`	Sum squared error at each iteration.

Also, for `tiltpsi`

`zcs`	Matrix of Zernike coefficients, with one row for each column in `coords` and number of columns = number of columns of im.mat.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

References

Zhaoyang Wang and Bongtae Han, “Advanced iterative algorithm for phase extraction of randomly phase-shifted interferograms,” *Opt. Lett.* 29, 1671-1673 (2004).

Han, G-S and Kim, S-W,, “Numerical correction of reference phases in phase-shifting interferometry by iterative least squares fitting,” *Applied Optics* 33, 7321-7325 (1994),

Lin, B-J et al., “An iterative tilt-immune phase-shifting algorithm,” OSA conference Optical Fabrication and Testing 2010.

See Also

`psifit`

`astig.bath` *Zernike coefficients for astigmatism due to Bath geometry.*

Description

Calculates Bath astigmatism coefficients with optional rotation of phi degrees.

Usage

```
astig.bath(D, rc, s, lambda = 632.8, phi = 0)
```

Arguments

`D`	Diameter
`rc`	Radius of curvature
`s`	separation of reference and test beams
`lambda`	Wavelength
`phi`	angle of image horizontal relative to interferometer axis, in degrees

Details

D, rc, s, must have the same units. If those units are mm the source wavelength lambda should be in nm.

Value

The Zernike coefficients for primary astigmatism terms.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

`brcutpuw` *Branch cut algorithm for phase unwrapping*

Description

Solves a modification of the assignment problem to minimize the total length of branch cuts.

Usage

```
brcutpuw(phase, pen = 0, details = FALSE)
```

Arguments

`phase`	Matrix containing the wrapped phase map
`pen`	Optional penalty value for connecting a residue to an edge
`details`	boolean: if TRUE return some extra details for diagnostics

Value

A matrix containing the unwrapped wavefront. If details==TRUE a named list starting with the unwrapped wavefront in puw.

Author(s)

M.L. Peck

See Also

`qpuw` `idiffpuw`

Examples

```
set.seed(1234)
PW <- wrap(matrix((0:100)*pi/10,101,101))
## need a border of NA's
PW <- cbind(rep(NA,101), PW, rep(NA,101))
PW <- rbind(rep(NA,103), PW, rep(NA,103))
PW <- PW + rnorm(103^2)
mtext(rmap(PW, plot=TRUE))
PU <- brcutpuw(PW, details=TRUE)
image(1:103, 1:103, PU$bcuts, col="blue", add=TRUE)
X11()
image(PU$puw, col=grey256, asp=1, useRaster=TRUE)
```

`circle.auto` *Auto circular edge detection*

Description

Automatically determine the center and radius of a circular interferogram image.

Usage

```
circle.auto(im, sigma=3, qt=0.995, excl=5,
    plots=TRUE, details=FALSE)
```

Arguments

`im`	A matrix containing an image of a circular disk
`sigma`	Amount to smooth image
`qt`	Threshold to accept an edge point, expressed as a quantile
`excl`	number of pixels around border of frame to exclude
`plots`	Plot edge candidates and fit?
`details`	Logical: return extra details?

Details

This routine partially implements the Canny algorithm for edge detection. After optionally smoothing the input image the gradient is calculated using a Sobel filter, and edge pixels are identified by locating local maxima in the magnitude of the gradient.

The edge pixels with `qt` percentile largest gradients are passed to `nlsrob` in package `robustbase` `lqs` in package `MASS` to determine robustly the best fit circle.

Value

A list with the following components:

`xc`	X coordinate of the center of the pupil
`yc`	Y coordinate of the center of the pupil
`rx`	Horizontal radius of the pupil
`ry`	Vertical radius of the pupil = rx
`obstruct`	Obstruction fraction (always = 0)

Note

This routine is only effective on modulation estimates, and will almost certainly fail on interferogram images. Since data quality varies widely considerable experimentation may be needed on any given image. Increasing the smoothing parameter `sigma` helps to suppress artifacts. Depending on how strong the actual edge is compared to artifacts `qt` may need to be either increased or decreased from the default value.

if `details==TRUE` several more pieces of data are returned. This is mostly for debugging purposes and may be eliminated in the future.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

See Also

Many routines require the pupil parameters in the form returned by `circle.auto`. For example `psifit`, `fftfit`, `pupil`, etc.

`circle.hough` *Estimate parameters of a circle using Hough Circle Transform*

Description

Uses a portion of the canny algorithm to find candidate edge points and the direction of the gradient at those points, then uses Hough Circle Transform to estimate circle parameters.

Usage

```
circle.hough(
  im,
  fw = 2,
  qt = 0.995,
  excl = 5,
  rmin = min(dim(im))/4,
  rmax = min(dim(im))/2,
  rstep = 1,
  dtheta_max = 0.5,
  dtheta_step = 0.05,
  nn = 7,
  plots = TRUE,
  details = FALSE
)
```

Arguments

`im`	The image to find a circle in (a modulation estimate is best)
`fw`	Size of Gaussian blur to smooth image
`qt`	Threshold to accept strong edge candidate
`excl`	Number of pixels to exclude around edge of frame as candidates
`rmin`	Minimum circle radius
`rmax`	Maximum circle radius
`rstep`	step size in constructing lookup table
`dtheta_max`	maximum assumed error in gradient direction
`dtheta_step`	increment for dtheta
`plots`	plot?
`details`	Return extra details?
`number`	of nearest neighbors for alternate calculation

Details

The Hough transform section first creates a lookup table of candidate radii and center points, then for each candidate edge point calculates potential centers along a fan of rays near the gradient direction. An inner join then finds matches in the lookup table and increments an accumulator vector. Highest vote at the end wins.

Value

If details is FALSE a named list with the circle parameters

Note

This is experimental and can be very slow. A good guess for the radius is very helpful. Experimental feature: find the nn nearest neighbors of the selected trio of parameters and calculate a vote weighted mean. This is returned as `rxy_alt` if details is TRUE.

See Also

`circle.auto()`, `circle.man()`

Examples

```
example("psifit", package="zernike", ask=FALSE)
X11()
cp2 <- circle.hough(tfit$mod, rmin=round(tfit$cp$rx)-10, rmax=round(tfit$cp$rx)+10)
```

circle.man *Circle parameters*

Description

Interactively determine the center, radius, and obstruction fraction of a circular or annular interferogram image.

Usage

```
circle.man(im = NULL, obstructed = FALSE)
```

Arguments

`im`	A matrix containing an interferogram image
`obstructed`	Logical: is there a central obstruction?

Details

In `circle.man`, if the image has already been plotted `im` can be `NULL`, which is the default.

Value

A list with the following components:

`xc`	X coordinate of the center of the pupil
`yc`	Y coordinate of the center of the pupil
`rx`	Horizontal radius of the pupil
`ry`	Vertical radius of the pupil
`obstruct`	Obstruction fraction

Note

`circle.man` uses the basic graphics library routine `locator` to interactively mark the edge of the pupil, and optionally the edge of the obstruction. After right clicking to terminate `locator()` a least squares fit is performed to the marked points to determine the center and radius of the pupil.

Note that all routines that make use of Zernikes implicitly assume a circular or annular pupil.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

Many routines require the pupil parameters in the form returned by `circle.man`. For example `psifit`, `fftfit`, `vortexfit`, `pupil`, etc.

col3d *OpenGL plot*

Description

Returns a vector of colors similar to `image()` display.

Usage

```
col3d(surf, surf.col=topo.colors(256), zlim = NULL, eqa=FALSE)
```

Arguments

`surf`	A matrix of surface values
`surf.col`	Color palette for surface
`zlim`	Range of values to display
`eqa`	Equal area per color?

Value

A vector of color values the same length as `surf`.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

References

The **rgl** package is described at `http://rgl.neoscientists.org/about.shtml`, and available from CRAN.

See Also

`plot.pupil`

`conv2_sep_complex` *2D separable convolution - complex matrix*

Description

2D convolution with a separable kernel

Usage

```
conv2_sep_complex(X, kernel)
```

Arguments

`X`	a complex valued matrix.
`kernel`	

Details

Having a separable kernel implies that a direct 2D convolution can be executed as a series of 1D column- and row-wise convolutions. This *may* be faster a direct 2D convlution or using FFT's, and certainly benefits from multithreading. The input matrix is implicitly zero padded, which differs from an fft based convolution, where the input is implicitly extended to infinity in all directions, therefore the outputs will in general differ

Value

a complex valued matrix with the same dimension as X.

`conv2_sep_real` *2D separable convolution - real matrix*

Description

2D convolution with a separable kernel

Usage

```
conv2_sep_real(X, kernel)
```

Arguments

`X`	a real valued matrix.
`kernel`	

Details

Having a separable kernel implies that a direct 2D convolution can be executed as a series of 1D column- and row-wise convolutions. This *may* be faster a direct 2D convlution or using FFT's, and certainly benefits from multithreading. The input matrix is implicitly zero padded, which differs from an fft based convolution, where the input is implicitly extended to infinity in all directions, therefore the outputs will in general differ

Value

a real matrix with the same dimension as X.

`convolve2d` *2D convolution - real matrix*

Description

General 2D dircet convolution

Usage

```
convolve2d(X, kernel)
```

Arguments

`X`	a real valued matrix.
`kernel`	a real valued matrix

Details

General 2D direct convolution with an arbitrary matrix kernel. This is a wrapper for the Armadillo function `conv2`.

Value

a real matrix with the same dimension as X.

`crop` *Crop an array*

Description

Crop a matrix or 3D array. Main application is to trim excess pixels from an image array, wavefront, etc.

Usage

```
crop(img, cp, npad = 20, nxy=NULL)
```

Arguments

`img`	Array to be cropped.
`cp`	A list describing the pupil boundary.
`npad`	Amount of padding to leave around the edge.
`nxy`	Number pixels per side in cropped array.

Details

`cp` is the list provided by `circle.pars`.

Value

`im`	The cropped array
`cp`	Revised value of cp

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

`d_of_g` *Derivative of Gaussian filter*

Description

2D filter that combines smoothing and (central) differencing.

Usage

```
d_of_g(X, sigma)
```

Arguments

`X`	a matrix
`sigma`	standard deviation in pixels of the Gaussian kernel

Value

A named list with components Dx, Dy, Dxy, and the calculated kernel

Calls arma::conv2(). Called by `circle.auto()`.

`fft` *2D FFT*

Description

2D FFT wrapping Armadillo's `fft2`

Usage

```
fft(X)
```

Arguments

`X`	a real valued matrix

Value

A complex valued matrix with the discrete Fourier transformed input.

This is just a wrapper for Armadillo's `fft2` function. It works well for dimensions that are *highly composite*, not just powers of 2 as implied by the online documentation. `X` need not be a square matrix.

References

Conrad Sanderson and Ryan Curtin. Armadillo: An Efficient Framework for Numerical Linear Algebra. International Conference on Computer and Automation Engineering, 2025.

Conrad Sanderson and Ryan Curtin. Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation. Mathematical and Computational Applications, Vol.24, No.3, 2019.

fftfit *Fourier transform interferogram analysis*

Description

High level routines for FFT analysis of interferograms.

Usage

```
fftfit(imagedata, cp = NULL,
  sl = c(1, 1), filter = NULL, taper = 2,
  options = zernike_options())
```

Arguments

`imagedata`	A matrix containing the interferogram
`cp`	A list describing the pupil boundary, as returned by `pupil.pars`
`sl`	Position of sidelobe in the form c(x,y)
`filter`	Size of background filter around DC
`taper`	Size of taper applied to edge of half plane cut
`options`	a list of parameters passed to other functions. See `psfit_options`

.

Details

If `is.null(filter)` (the default), `pick.sidelobe` will be called to select a Fourier domain sidelobe and background filter size.

If `is.null(cp)` `circle.pars` is applied to the modulation to estimate the pupil parameters.

See `wf_net` for details of the process of creating net and smoothed wavefronts from raw unwrapped wavefront maps.

Value

A list with the following components:

`phase`	Wrapped phase map
`mod`	The estimated modulation
`cp`	A list describing the pupil boundary
`cp.orig`	The precropped value of cp
`wf.net`	Net unsmoothed wavefront; a matrix of class `"pupil"`
`wf.smooth`	Net smoothed wavefront
`wf.residual`	Difference between net wavefront and polynomial fit
`fit`	Return value from `fitzernikes`
`zcoef.net`	Net Zernike coefficients from fit

Note

These functions are based largely on the work of Roddier and Roddier (1987).

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

References

Roddier, C. and Roddier, F. 1987, **Interferogram analysis using Fourier transform techniques**, *Applied Optics*, vol. 26, pp. 1668-1673.

See Also

vortexfit, wf_net, pupil.pars, pick.sidelobe.

Examples

```
# just run the example for 'vortexfit()'

example(vortexfit, package="zernike", ask=FALSE)
```

---

fftshift *2D fftshift*

Description

Shift quadrants of a matrix

Usage

```
fftshift(X)
```

Arguments

X	a real or complex valued matrix.

Details

This swaps the quadrants of a matrix to put the (0,0) element in the center. The most common use is to center the DC component of a Fourier Transformed matrix for visualization or to simplify other matrix operations

Value

a complex matrix with the same dimension as X.

See Also

`ifftshift()` is the inverse operation.

Examples

```
X <- matrix(1:16, 4, 4)
XS <- fftshift(X)
XS
ifftshift(XS)
```

`FFTUtilities` *FFT Utilities*

Description

Miscellaneous utilities for working with 2D images in the Fourier domain.

Usage

```
wftophase(X, lambda=1)
padmatrix(X, npad, fill = mean(X, na.rm=TRUE))
submatrix(X, size = 255)
fftshift(X)
.up2(nr, nc=nr)
```

Arguments

`X`	A matrix
`lambda`	Value of the wavelength, in the same units as X
`npad`	Size of padded matrix
`fill`	Values to be assigned to padded matrix elements
`size`	Size of returned matrix
`nr`	A number
`nc`	A number

Details

`wftophase` computes the complex phase from wavefront values.

`padmatrix` pads a matrix to size `npad` x `npad`, placing the original matrix in the lower left hand corner of the padded matrix.

`submatrix` extracts a `size` x `size` matrix from the center of a larger matrix.

Value

A matrix transformation of the input matrix `X`.

`.up2` returns the next higher power of 2 than `max(nr, nc)`.

Note

These low level routines are used by several higher level functions that operate in the Fourier domain.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`startest`, `fftfit`.

`fft_cx` *2D FFT - complex valued input*

Description

2D FFT wrapping Armadillo's `fft2`

Usage

```
fft_cx(X)
```

Arguments

`X`	a complex valued matrix

Value

A complex valued matrix with the discrete Fourier transformed input.

See Also

`fft()`

fft_cx_pad *2D FFT with padding - complex valued input*

Description

2D FFT wrapping Armadillo's `fft2`

Usage

```
fft_cx_pad(X, npad)
```

Arguments

`X`	a complex valued matrix
`npad`	size to pad to

Value

A complex valued matrix with the discrete Fourier transformed input.

This is provided for cases where the input matrix has one or both dimensions that are *not* highly composite (that is, have prime factors that are larger than 5). `npad` itself should be highly composite. This isn't checked. The R function `nextn()` can be used to find a suitable size.

See Also

`fft()`

fft_pad *2D FFT with padding*

Description

2D FFT wrapping Armadillo's `fft2`

Usage

```
fft_pad(X, npad)
```

Arguments

`X`	a real valued matrix
`npad`	size to pad to

Value

A complex valued matrix with the discrete Fourier transformed input.

This is provided for cases where the input matrix has one or both dimensions that are *not* highly composite (that is, have prime factors that are larger than 5). `npad` itself should be highly composite. This isn't checked. The R function `nextn()` can be used to find a suitable size.

See Also

`fft()`

`fitzernikes` *Least Squares fit to Zernike polynomials*

Description

Performs a least squares fit of a specified set of Zernike polynomials to a vector of wavefront measurements.

Usage

```
fitzernikes(wf, rho, theta, eps=0, phi = 0, maxorder = 14,
            nthreads = -1, isoseq = FALSE,
            usecirc = FALSE, ext_prec = FALSE)
```

Arguments

`wf`	A vector of wavefront values
`rho`	A vector of radial coordinates.
`theta`	A vector of angular coordinates, in radians.
`eps`	Obstruction fraction of annular aperture.
`phi`	Orientation of the image, in degrees
`maxorder`	Maximum Zernike polynomial order
`nthreads`	# threads for threaded function calls
`isoseq`	Boolean: use ISO/ANSI sequencing
`usecirc`	Boolean: Use circle Zernikes even if eps > 0?
`ext_prec`	Boolean: Use extended precision arithmetic for annular Zernike computation?

Details

`wf`, `rho`, and `theta` must be the same length.

As of version 3.7.0 Zernike polynomials in ISO/ANSI sequence can be used through a call to `zpm_cart`.

Value

The coefficients of the least squares fit.

Note

With the standard ordering of Zernikes Z0 is the piston term, Z1 and Z2 are x and y tilts, Z3 is defocus, etc.

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

See Also

zpm, zpm_cart, zapm, zapm_iso, psifit, fftfit, vortexfit, wf_net.

`foucogram` *Simulate a Foucaultgram*

Description

Simulates the appearance of a wavefront under the Foucault test.

Usage

```
foucogram(wf, edgex = 0, phradius = 0, slit = FALSE,
  pad = 4, gamma = 1, map = FALSE, lev = 0.5)
```

Arguments

wf	An object of class pupil containing wavefront values
edgex	lateral position of knife edge
phradius	radius of light source
slit	Logical: Is source a slit or pinhole?
pad	pad factor for FFT
gamma	Gamma value for graphics display
map	Logical: Overlay contours from wavefront map?
lev	Increment for contour levels, if used

Details

The default value of 0 for `phradius` simulates a monochromatic point source. Try values in the range 10-30 to suppress diffraction effects.

Value

A matrix of intensity levels in the simulated image.

Note

The key approximations here are treating the light source as monochromatic and spatially coherent, which is usually not the case for an extended source. Also, Fraunhofer diffraction theory is used.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

References

See `http://home.netcom.com/~mpeck1/astro/foucault/ext_foucault.pdf` for an outline of the mathematical treatment of an extended source.

See Also

`pupil`

`gblur` *2D Gaussian blur*

Description

2D Gaussian blur of a real matrix

Usage

```
gblur(X, sigma)
```

Arguments

`X`	a matrix
`sigma`	standard deviation in pixels of the Gaussian kernel

Value

A matrix with the same dimensions as the input

Performs direct convolution using `conv2_sep_real()` with a Gaussian kernel sized to span 5 standard deviations.

`gblur_complex` *2D Gaussian blur of a complex valued matrix*

Description

2D Gaussian blur of a complex matrix

Usage

```
gblur_complex(X, sigma)
```

Arguments

`X`	a complex valued matrix
`sigma`	standard deviation in pixels of the Gaussian kernel

Value

A complex valued matrix with the same dimensions as the input

Performs direct convolution using `conv2_sep_complex()` with a Gaussian kernel sized to span 5 standard deviations. Preliminary benchmarking indicates this is no faster than performing separate convolutions of the real and imaginary components and combining the results.

`gblur_fft` *2D Gaussian blur*

Description

2D Gaussian blur – FFT based version

Usage

```
gblur_fft(X, sigma)
```

Arguments

`X`	a real valued matrix
`sigma`	standard deviation in pixels of the Gaussian kernel

Value

the smoothed input matrix

Numerical experiments suggest that it's almost always faster to use the direct convolution based Gaussian blur in `gblur()`.

`gol_welsch` *Golub-Welsch method to find quadrature points and weights for Gauss-Legendre quadrature*

Description

Calculates the nodes and weights for Legendre polynomials shifted to the interval (eps^2, 1).

Usage

```
gol_welsch(eps, qwts)
```

Arguments

eps	obstruction fraction 0 <= eps < 1
qwts	an input R vector with length equal to the number of quatrature points.

Details

If N is the maximum polynomial order to be evaluated qwts should be at least of length N/2 + 1. Quadratures will then be nominally exact.

Value

a vector of quadrature nodes the same length as `qwts` in the open interval (eps^2, 1). `qwts` will be overwritten with the quadrature weights.

See Also

Called by `zapm()` and `zapm_iso()`.

`gpcapsi` *Generalized Principal components algorithm for phase shifting interferometry*

Description

A generalized principal components algorithm for phase shifting interferometry developed by the author. This is the "low level" implementation.

Usage

```
gpcapsi(im.mat, ptol = 0.001, maxiter = 20, trace = 1)
gpcapsiC(im.mat, ptol, maxiter, trace)
```

Arguments

`im.mat`	Matrix containing the unmasked pixels from a set of interferograms.
`ptol`	Convergence tolerance for phase shifts
`maxiter`	Maximum number of iterations
`trace`	Print progress of nonlinear solver every trace iterations. Use trace=0 for silent operation.

Details

`gpcapsi` is a wrapper to the C++ call in `gpcapsiC`.

Value

A list with the following items:

`phi`	Estimated wrapped phase.
`mod`	Estimated modulation.
`phases`	Estimated phase shifts.
`snr`	An estimate of the S/N of the interferograms.
`eigen`	Eigenvalues of the crossproduct matrix

Note

This is the low level interface to the algorithm. The matrix `im.mat` should contain the unmasked pixel values from the input interferogram array. No checks are made for valid data. This should normally be called through the high level function `psifit`.

Author(s)

M. L. Peck

See Also

`pcapsi` `psifit`

`gradzpm` *Zernike polynomials and cartesian gradients*

Description

Calculate Zernike polynomial values and Cartesian gradients in ISO/ANSI sequence for a set of Cartesian coordinates.

Usage

```
gradzpm(x, y, maxorder = 12L, unit_variance = TRUE, return_zpm = TRUE)
```

Arguments

`x`	a vector of x coordinates for points on a unit disk.
`y`	a vector of y coordinates.
`maxorder`	the maximum radial polynomial order (defaults to 12).
`unit_variance`	logical: return with orthonormal scaling? (default `false`)
`return_zpm`	logical: return Zernike polynomial matrix? (default `true`)

Details

Uses the recurrence relations in the above publication to calculate Zernike polynomial values and their directional derivatives in Cartesian coordinates. These are known to be both efficient and numerically stable.

Columns are in ISO/ANSI sequence: for each radial order n >= 0 the azimuthal orders m are sequenced m = -n, -(n-2), ..., (n-2), n, with sine components for negative m and cosine for positive m. Note this is the opposite ordering from the extended Fringe set and the ordering of aberrations is quite different. For example the two components of trefoil are in the 7th and 10th column while coma is in columns 8 and 9 (or 7 and 8 with 0-indexing). Note also that except for tilt and coma-like aberrations (m=1) non-axisymmetric aberrations will be separated.

All three matrices will have the same dimensions on return. Columns 0 and 1 of `dzdx` will be all 0, while columns 0 and 3 of `dzdy` are 0.

Value

a named list with the matrices `zm` (optional but returned by default), `dzdx`, `dzdy`.

References

Anderson, T.B. (2018) Optics Express 26, #5, 18878 https://doi.org/10.1364/OE.26.018878 (open access)

See Also

`zpm()` uses the same recurrence relations for polar coordinates and extended Fringe set ordering, which is the more common indexing scheme for optical design/testing software.

`zpm_cart()` calculates and returns the Zernike polynomial values only.

Examples

```
rho <- seq(0.2, 1., length=5)
theta <- seq(0, 1.6*pi, length=5)
rt <- expand.grid(theta, rho)
x <- c(0, rt[,2]*cos(rt[,1]))
y <- c(0, rt[,2]*sin(rt[,1]))
gzpm <- gradzpm(x, y)
```

`gray256` *8 bit Grayscale*

Description

A vector of gray scale levels

Usage

```
gray256
grey256
```

Value

Defined as `gray256 <- grey(seq(0,1,length=256))`

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>.

`hypot` *Hypotenuse*

Description

The Euclidean length of a vector

Usage

```
hypot(x)
```

Arguments

`x`	a vector

Value

the length of the vector

Author(s)

M.L. Peck

Examples

```
hypot(c(1,2))
```

idiffpuw *Phase unwrapping by Integrating DIFFerences*

Description

Simple path following algorithm for two dimensional phase unwrapping.

Usage

```
idiffpuw(phase, mask = phase, ucall = TRUE, dx = NULL, dy = NULL)
```

Arguments

`phase`	A matrix of wrapped phase values
`mask`	Matrix the same size as `phase` indicating masked pixels
`ucall`	Boolean: User call?
`dx`	Matrix of x differences
`dy`	Matrix of y differences

Details

`mask` indicates pixels that shouldn't be unwrapped. In the simplest (default) case these are just pixels where phase is undefined.

Value

`if(ucall)`, a matrix of class "`pupil`" with unwrapped wavefront values, otherwise a list with items:

`puw`	Unwrapped phase
`uw`	Matrix indicating pixels that have been unwrapped.

Note

`brcutpuw` calls `rmap` first to check for the presence of residues. If there are none `idiffpuw` is guaranteed to work and is called to do the phase unwrapping.

If there *are* residues `brcutpuw` creates a mask then calls `idiffpuw` to unwrap unmasked portions of the phase map.

This function is user callable as well; use a call of the form `idiffpuw(phase)`.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>. Thanks to Steve Koehler for programming ideas to considerably speed up the algorithm.

References

Ghiglia, D.C., and Pritt, M.D., 1998, **Two-Dimensional Phase Unwrapping**, New York: Wiley & Sons, Inc., ISBN 0-471-24935-1.

See Also

rmap, brcutpuw

---

id_dxy_uw *Compiled code via Rcpp for Itoh's method of phase unwrapping*

Description

Called by brcutpuw() for fast phase unwrapping

Usage

```
id_dxy_uw(nr, nc, phase, mask, dx, dy, uw)
```

Arguments

nr	number of rows in phase matrix
nc	number of columns in phase matrix
phase	phase matrix converted to vector
mask	matrix of mask values converted to vector
dx	wrapped phase differences in x direction
dy	wrapped phase differences in y direction

Details

This is called by brcutpuw() through idiffpuw() but is also user callable. Wrapped phase values and differences are divided by 2*pi before input making the input values in the range [-1/2, 1/2). In brcutpuw() the mask indicates areas outside the interferogram area and lines of branch cuts

Value

a vector with the unwrapped phase

Author(s)

M.L. Peck (mlpeck54 -at- gmail.com)

See Also

brcutpuw(), idiffpuw()

`id_uw` *Compiled code via Rcpp for Itoh's method of phase unwrapping*

Description

Called by `idiffpuw()` for fast phase unwrapping

Usage

```
id_uw(nr, nc, phase)
```

Arguments

`nr`	number of rows in phase matrix
`nc`	number of columns in phase matrix
`phase`	phase matrix converted to vector

Details

This is called by `idiffpuw()` but is also user callable. Wrapped phase values are divided by `2*pi` before input making the input values in the range [-1/2, 1/2). In `brcutpuw()` the mask indicates areas outside the interferogram area and lines of branch cuts

Value

a vector with the unwrapped phase

Author(s)

M.L. Peck (mlpeck54 -at- gmail.com)

See Also

`brcutpuw()`, `idiffpuw()`

`ifft` *2D Inverse FFT - complex valued input*

Description

2D Inverse FFT wrapping Armadillo's `ifft2`

Usage

```
ifft(X)
```

Arguments

`X`	a complex valued matrix

Value

A complex valued matrix with the inverse discrete Fourier transformed input.

In contrast to some implementations this returns the *scaled* inverse transform.

See Also

`fft()`

`ifftshift` *2D ifftshift*

Description

Inverse fftShift

Usage

```
ifftshift(X)
```

Arguments

`X`	a real or complex valued matrix.

Details

Inverts the fftshift operation, swapping quadrants of the input matrix.

Value

a complex matrix with the same dimension as X.

See Also

`fftshfit()`.

Examples

```
X <- matrix(1:16, 4, 4)
XS <- fftshift(X)
XS
ifftshift(XS)
```

`ifft_real` *2D Inverse FFT - real valued output*

Description

2D Inverse FFT wrapping Armadillo's `ifft2`

Usage

```
ifft_real(X)
```

Arguments

`X`	a complex valued matrix

Value

A real valued matrix with the real component of the inverse discrete Fourier transformed input.

In many cases where a real valued matrix is transformed, some operations performed in the Fourier domain, and then inverse transformed the output will be real but with a small imaginary component. This function simply discards the imaginary component. No check is made that this is actually appropriate.

See Also

`fft()`

`load.images` *Read images*

Description

Loads image files in jpeg, tiff, png, or raw format. `load.pgm` provides legacy support for reading files in pgm format.

Usage

```
load.images(files, channels=c(1,0,0), scale=1, FLIP=FALSE)
load.pgm(files, imdiff=NULL)
```

Arguments

`files`	A vector of character strings with file names
`channels`	channel weights
`scale`	scale factor for image resize
`FLIP`	flip image left for right?

Details

set FLIP=TRUE to reverse mirror imaged interferograms.

Any file extension other than jpg, jpeg, tif, tiff, or png is assumed to be in RAW format. Supported raw formats are determined by libraw and may not be up to date

Value

An array containing the contents of the image files.

Note

`load.pgm` is the original `load.images` included for legacy support of greyscale portable anymap files.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

`lspsi` *Phase Shifting Interferometry*

Description

Least squares fitting of phase shifted interferograms.

Usage

```
lspsi(images, phases, wt = rep(1, length(phases)))
lspsiC(images, phases, wt)
```

Arguments

`images`	An array containing the interferogram images
`phases`	A vector of phase shifts
`wt`	A vector of weights

Details

`images` is a 3 dimensional array with dimensions `nrow` x `ncol` x `length(phases)`, where `nrow` and `ncol` are the number of rows and columns in the individual interferogram images.

`lspsi` reshapes the image array into a matrix and calls `lspsiC` which in turn calls the compiled C++ routine.

Value

A list containing the following components:

`phi`	Estimated wrapped wavefront phase.
`mod`	Estimated modulation

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`psifit`

`makezlist.iso` *Construct list of ZP indexes in ISO/ANSI sequence with sine terms first*

Description

Construct list of ZP indexes in ISO/ANSI sequence with sine terms first

Usage

```
makezlist.iso(maxorder = 12)
```

Arguments

`maxorder`	maximum radial and azimuthal order

Value

a named list with n=radial indexes, m=azimuthal, and t indicating trig function to apply

Examples

```
zlist.iso <- makezlist.iso(maxorder=6)
zlist <- makezlist(0, 6)
```

`norm_zpm` *Normalize matrix of Zernike polynomial values.*

Description

Convert a matrix of Zernike polynomial values from unit scaled to unit variance aka orthonormal form.

Usage

```
norm_zpm(uzpm, maxorder)
```

Arguments

`uzpm`	matrix of Zernike polynomial values
`maxorder`	the maximum radial order.

Details

This is intended only for ISO/ANSI ordered matrices. The only check performed is that the number of columns in the matrix matches the expected number given by the argument `maxorder`. This is called by `gradzpm()` and `zpm_cart()` if `unit_variance` is set to `true` in the respective function calls.

Value

matrix in orthonormal form.

`pcapsi` *Vargas et al.'s Principal Components method for PSI*

Description

Compute the phase using the Principal components algorithm.

Usage

```
pcapsi(im.mat, bgsub = TRUE, group_diag = "v")
```

Arguments

`im.mat`	A *matrix* of interferogram values
`bgsub`	Boolean - subtract the pixelwise mean as background estimate?
`pcalg`	controls treatment of singular values of the data matrix

Details

Images are input into an array by `load.images`. This must be reshaped into a matrix for this function. Also, a mask should be applied if available prior to the call.

Value

A list containing the following elements:

`phi`	The wrapped phase estimate. This is a vector as long as the number of rows in `im.mat`.
`mod`	Modulation estimate.
`phases`	Phase shift estimates.
`wt`	implied least squares weights.
`snr`	An estimate of the signal to noise ratio in the input data.
`eigen`	Singular values of the crossproduct matrix.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

References

J. Vargas, J. Antonio Quiroga, and T. Belenguer, “Phase-shifting interferometry based on principal component analysis,” *Opt. Lett.* **36**, 1326-1328 (2011) http://www.opticsinfobase.org/ol/abstract.cfm?URI=ol-36-8-1326

J. Vargas, J. Antonio Quiroga, and T. Belenguer, “Analysis of the principal component algorithm in phase-shifting interferometry,” *Opt. Lett.* **36**, 2215-2217 (2011) http://www.opticsinfobase.org/ol/abstract.cfm?URI=ol-36-12-2215

See Also

`psifit`,

`pick.sidelobe` *Select an interferogram sidelobe in the Fourier domain*

Description

Interactively locate the center of a first order sidelobe in the FFT of an interferogram, and mark the width of the background filter.

Usage

```
pick.sidelobe(imagedata, logm=FALSE, gamma=3)
```

Arguments

`imagedata`	A matrix containing an interferogram image
`logm`	Logical: pass `fn="logMod"` to `plot.cmat`?
`gamma`	gamma value for display

Details

Uses the basic graphics utility `locator`.

Value

A list with the following components:

`sl`	The coordinates `c(x,y)` of the selected sidelobe
`filter`	Estimated size of background filter

Note

The high level FFT interferogram analysis routine `fftfit` requires the approximate location of the intended first order interferogram sidelobe to be specified.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`fftfit`,

`plot.cmat` *Plot a complex matrix*

Description

Plot a real valued function of a complex matrix

Usage

```
plot.cmat(X, fn = "Mod", col = grey256,
cp=NULL, zoom=1, gamma=1, ...)
```

Arguments

`X`	A complex valued matrix
`fn`	A function returning a real value
`col`	Color palette for graph
`cp`	pupil parameters as returned by `pupil.pars`
`zoom`	zoom factor for display
`gamma`	gamma value for display
`...`	Other parameters to pass to `image.default`

Details

In addition to the functions described in `complex` `fn` can be assigned the values `"logMod"`, which will call an internally defined function returning the value `log(1+Mod(X))`, `"Mod2"` to plot the power spectrum, and `"logMod2"` to plot the logarithm of the power spectrum.

If the parameter cp is passed axes will display spatial frequencies in cycles per pupil radius.

Value

none

Note

This is used primarily for displaying FFT's of interferograms. In the case of an interferogram in which the background has not been removed use `fn="logMod"` to make the first order sidelobes visible.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

See Also

`pick.sidelobe`, `fftfit`.

`plot.pupil` *Pupils and wavefronts*

Description

Plot and summary methods for objects of class `"pupil"`.

Usage

```
plot.pupil(wf, cp=NULL, col = topo.colors(256), addContours = TRUE, cscale = FALSE,
   eqa=FALSE, zlim=NULL, ...)
summary.pupil(wf)
```

Arguments

`wf`	An object of class `"pupil"`
`cp`	Pupil parameters; a list as returned by `pupil.pars`
`col`	Color palette for plot
`addContours`	Logical: add contour lines?
`cscale`	Add a color scale legend?
`eqa`	Perform an "equal area" plot?
`zlim`	z limits to pass to image
`...`	Additional parameters to pass to `image.default`

Details

These give simple plot and summary methods for objects of class `pupil`.

If `eqa` is `TRUE`, each color in the palette will be used for an equal number of pixels (as opposed to representing an equal interval). Note: the color scale (when `cscale` == `TRUE`) may be inaccurate if a very small number of colors are used.

Value

none

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`pupil`, `pupilrms`, `pupilpv`, `strehlratio`, `pupil.pars`.

`plotn` *Wavefront comparison plots*

Description

Plot an arbitrary number of wavefronts and all differences.

Usage

```
plotn(..., labels = NULL, addContours=FALSE, wftype = "net",
col = rygcb(400), qt = c(0.01, 0.99))
```

Arguments

`...`	List of wavefront estimates as returned by `wf_net`.
`labels`	Labels to identify the wavefronts.
`addContours`	Boolean to add contours to top row plots
`wftype`	If the inputs are from `wf_net`, one of "net", "smooth", "residual".
`col`	Color palette for top row of plot
`qt`	Quantiles of differences to plot in comparisons.

Details

... can be any number of objects containing wavefront estimates as returned for example by `wf_net`.

Wavefronts are displayed on the top row, and differences of all pairs on subsequent rows. Grayscale is used to render the difference plots, and the color palette given in `col` is used for the wavefronts.

Value

none

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

See Also

`plot.pupil` `wf_net`

`plotxs` *Plot cross-sections (profiles) through a wavefront map.*

Description

Plots an arbitrary number of cross-sections through a wavefront map, with one highlighted.

Usage

```
plotxs(wf, cp, theta0 = 0, ylim = NULL, N = 4, n = 101,
col0 = "black", col = "gray", lty = 2)
```

Arguments

`wf`	A matrix of wavefront values.
`cp`	List of pupil parameters as returned by `pupil.pars`.
`theta0`	Angle of highlighted profile, in degrees.
`ylim`	range of heights to plot.
`N`	Number of cross sections.
`n`	Number of points for each cross section.
`col0`	Highlight color.
`col`	Cross section color.
`lty`	Line type for plots.

Details

The cross sections are equally spaced in angle from 0 to `pi*(N-1)/N`. Any angle can be specified for the highlighted profile at `theta0`.

Value

none

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`plot.pupil` is the main wavefront plotting routine.

`psifit` *Phase Shifting Interferometry*

Description

High level function for Least squares analysis of phase shifted interferograms.

Usage

```
psifit(images, phases, cp = NULL, satarget = NULL, psialg = "ls", options = psfit_options())
```

Arguments

`images`	An array containing the interferogram images
`phases`	A vector of phase shifts
`cp`	A list describing the pupil boundary, as returned by `pupil.pars`
`satarget`	Target 4th and 6th order SA coefficients in non-null tests of aspheres
`psialg`	String identifying the PSI algorithm to use
`options`	a list of options

Details

`images` is a 3 dimensional array with dimensions `nrow` x `ncol` x `length(phases)`, where `nrow` and `ncol` are the number of rows and columns in the individual interferogram images.

The current values recognized for `psialg` are

ls least squares with known phase shifts

aia the "advanced iterative algorithm" `aiapsi`

pc1 pca with pcalg="pc1"

pc2 pca with pcalg="pc2"

pc3 pca with pcalg="pc3"

gpc my generalized PC algorithm in `gpcapsi`

pc1thenaia first `pcapsi` then `aiapsi`

pc1thentilt first `pcapsi` then `tiltpsi`

gpcthentilt first `gpcapsi` then `tiltpsi`

tilt tiltpsi

Value

A list with the following components

`phi`	wrapped phase estimate
`mod`	modulation estimate
`phases`	phase shifts
`cp`	the interferogram boundary
`wf.net`	net, unfiltered wavefront (see `wf_net`)
`wf.smooth`	Zernike fit wavefront
`wf.residual`	the difference
`fit`	Coefficients of Zernike fit to wavefront
`zcoef.net`	Net Zernike coefficients
`extras`	any extra data returned by low level functions

Author(s)

M.L. Peck <mlpeck54@gmail.com>

See Also

`lspsi`, `aiapsi`, `tiltpsi`, `gpcapsi`, `pcapsi`

Examples

```
  ## reuse the files from the demo for an example of two stage fitting
  ## using gpca then tiltpsi
require(zernike)
fpath <- file.path(find.package(package="zernike"), "psidata")
files <- scan(file.path(fpath, "files.txt"), what="character")
for (i in 1:length(files)) files[i] <- file.path(fpath, files[i])

# load the images into an array

images <- load.images(files)

# parameters for this run

source(file.path(fpath, "parameters.txt"))

# phase shifts

phases <- wrap((0:(dim(images)[3]-1))/frames.per.cycle*2*pi)
phases <- switch(ps.dir, ccw = -phases, cw = phases, phases)

# target SA coefficients for numerical null.

sa.t <- sconic(diam,roc,lambda=wavelength)
zopt <- zernike_options()
zopt$satarget <- sa.t
zopt$ptol <- 0.01
tfit <- psifit(images, phases, psialg="gpcthentilt", options=zopt)
```

`pupil` *Pupils and wavefronts*

Description

Create a pupil object and optionally fill it with a wavefront. For our purposes a “pupil” is defined to be a matrix representation of a circular or annular aperture. Simple plot and summary methods are also provided.

Usage

```
nrow.default <- 640
ncol.default <- nrow.default
cp.default <- list(xc=320.5, yc=320.5, rx=319.5, ry=319.5, obstruct=0)

pupil(zcoef=NULL, maxorder=14L, isoseq=FALSE,
    phi=0, piston=NULL,
    nrow=nrow.default, ncol=ncol.default,
    cp=cp.default)
pupil.arb(zcoef=NULL, zlist=makezlist(),
        phi=0, piston=0,
        nrow=nrow.default, ncol=ncol.default,
        cp=cp.default)
```

Arguments

zcoef	A vector of Zernike coefficients
maxorder	Maximum Zernike polynomial order
zlist	List of indexes the same length as zcoef
isoseq	ZPs in ISO/ANSI sequence
phi	Amount to rotate image, in degrees
piston	Constant to add to wavefront values
nrow	Number of rows in output matrix
ncol	Number of columns in output matrix
cp	A list with items xc - x coordinate of central pixel, yc - y coordinate of central pixel, rx - x radius in pixels, ry - y radius in pixels, obstruct - central obstruction fraction.

Details

plot.pupil and summary.pupil provide simple plot and summary methods for objects of class "pupil".

pupil.arb will accept an arbitrary list of Zernike indexes.

pupil requires a complete set of Zernikes as returned by zpm or zpm_cart.

Default values nrow.default, ncol.default and cp.default are now (as of version 3.7.0) stored in the package environment. These can be used or overridden by setting values in the user‘s global environment.

Version 3.7.0 of this package makes some fairly significant changes in pupil from previous releases. First, zlist is no longer used to indicate the set of Zernikes used. Instead the maximum radial order is passed as maxorder, defaulting to 14. The complete set of Zernike polynomial values including piston is calculated with either zpm or zpm_cart if isoseq is TRUE. The vector zcoef, if non-null, must be either the same length as the number of columns in the zernike matrix with a null value for piston, or one shorter with a non-null value for piston. The vector zcoef.net returned by wf_net has the first, piston, element removed so a non-null piston value must be passed as an argument.

Value

A matrix of size nrow x ncol. The matrix is assigned to the class "pupil". NAs are used to fill the matrix outside the defined area of the pupil.

Note

The parameter cp is used to define the dimensions of the pupil. See pupil.pars for details.

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

See Also

zpm, zpm_cart, wf_net, pupilrms, pupilpv, strehlratio, pupil.pars, circle.pars.

Examples

```
wf <- pupil(zcoef=rnorm(length(makezlist(0, 14)$n), 0, 0.01))
plot(wf, addContours=FALSE)
summary(wf)
```

---

pupil.rhotheta *Polar coordinates*

Description

Calculate matrixes of polar coordinates for pupil's.

Usage

```
pupil.rhotheta(nrow, ncol, cp)
```

Arguments

nrow	Number of rows in interferogram images
ncol	Number of columns in interferogram images
cp	A list describing the pupil boundary, as returned by pupil.pars

Value

A list with the following components:

rho	A matrix of radial coordinates
theta	A matrix of angular coordinates

Note

My Zernike polynomial routines work in polar coordinates, which this function provides. Also, NA's are used to fill the matrix outside the pupil boundary, making the returned values convenient for selecting pixels inside interferograms.

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

See Also

Zernike, pupil.

pupilrms *Wavefront statistics*

Description

Compute basic statistics of wavefronts stored in "pupil" objects.

Usage

```
pupilrms(pupil)
pupilpv(pupil)
strehlratio(rms)
```

Arguments

pupil	A matrix of class "pupil"
rms	An rms wavefront error

Value

Estimates of the RMS and P-V wavefront errors. `strehratio` calculates Mahajan's approximation to the Strehl ratio.

Note

`pupilrms` simply calculates the standard deviation of finite values in the matrix `pupil`. This is a crude, but usually accurate enough estimate of the true RMS wavefront error.

`summary.pupil` calls these functions.

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

References

Schroeder, D.J. 2000, *Astronomical Optics, 2nd Edition*, Academic Press, chapter 10.

See Also

`summary.pupil`.

Examples

```
zcoef <- rnorm(length(makezlist()$n), 0, 0.01)
wf <- pupil(zcoef=zcoef)
plot(wf)
summary(wf)
sqrt(crossprod(zcoef)) # A more accurate estimate of RMS
```

PVr *Zygo's "robust" PV*

Description

A peak to valley error estimate that reduces the effect of noise and artifacts

Usage

```
PVr(wf.zfit, wf.residual)
```

Arguments

`wf.zfit`	matrix containing the smoothed Zernike fit wavefront
`wf.residual`	matrix of the difference between the raw wavefront and the Zernike fit. These values are returned by `wf_net()`

Details

no check is performed on the wavefronts, so it's the user's responsibility to make sure these come from the same source

Value

the estimated PVr

References

Evans, C. (2009) Optical Engineering 48(4), 43605. https://doi.org/10.1117/1.3119307

qpuw — *Quality guided algorithm for phase unwrapping*

Description

Quality guided algorithm for two dimensional phase unwrapping.

Usage

```
qpuw(phase, qual)
```

Arguments

`phase`	A matrix of wrapped phase values
`qual`	A matrix of quality values the same size as `phase`.

Value

`puw`	A matrix of class "`pupil`" with the unwrapped wavefront.

Note

This is a straightforward implementation of the quality guided algorithm of G&P.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

References

Ghiglia, D.C., and Pritt, M.D., 1998, **Two-Dimensional Phase Unwrapping**, New York: Wiley & Sons, Inc., ISBN 0-471-24935-1.

See Also

`idiffpuw`, `brcutpuw`

`q_uw` *Compiled code via Rcpp for quality guided phase unwrapping*

Description

Called by `qpuw()` for fast quality guided phase unwrapping

Usage

```
q_uw(nr, nc, phase, qual)
```

Arguments

`nr`	number of rows in phase matrix
`nc`	number of columns in phase matrix
`phase`	phase matrix converted to vector
`qual`	quality matrix converted to vector

Details

This is called by `qpuw()` but is also user callable. Wrapped phase values are divided by `2*pi` before input making the input values in the range [-1/2, 1/2).

Value

a vector with the unwrapped phase

Author(s)

M.L. Peck (mlpeck54 -at- gmail.com) with valuable programming advice from Steve Koehler

See Also

`qpuw()`, `idiffpuw()`

`rescale` *Rescale an image.*

Description

Rescale a matrix containing a bitmapped image using bilinear interpolation.

Usage

```
rescale(img, scale)
```

Arguments

`img`	A matrix with image data.
`scale`	Scale factor.

Details

A value <1 will shrink the image.

Value

A matrix containing the rescaled image data.

Note

`NA`'s are OK.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

Called by `load.images` if necessary.

`rmap` *Utilities for phase unwrapping*

Description

Utility functions for use in 2D phase unwrapping.

Usage

```
rmap(phase, dx = NULL, dy = NULL, plot = FALSE, ...)
wrap(phase)
```

Arguments

`phase`	Matrix of wrapped phase values
`dx`	Matrix of x differences
`dy`	Matrix of y differences
`plot`	Boolean: plot residue positions?
`...`	additional arguments for `image`

Details

`dx` and `dy` must have the same dimension as `phase`.

Value

In `rmap` if `plot` == `TRUE`

`nr`	the number of residues identified in the map

otherwise

`phase`	wrapped phase returned by `wrap`
`residues`	Matrix the same size as `phase` with residues marked as + or - 1.

Note

These are primarily intended for internal use but can be used interactively. Calling `rmap(phase, plot=TRUE)` will plot the positions of residues and return nothing. If `(plot==FALSE)` in the call to `rmap` a matrix the same size as `phase` is returned with residues identified with values of +1 or -1.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`. Steve Koehler is responsible for the efficient implementation of the `wrap` function.

References

Ghiglia, D.C., and Pritt, M.D., 1998, **Two-Dimensional Phase Unwrapping**, New York: Wiley & Sons, Inc., ISBN 0-471-24935-1.

See Also

Called by `brcutpuw`.

`rygcb` *A better rainbow.*

Description

Produces a rainbow color palette with colors ranging from `"red"` to `"blue"` or `"magenta"`. Perceptual uniformity should be superior to R's `rainbow`.

Usage

```
rygcb(n)
rygcbm(n)
```

Arguments

`n`	Number of color levels

Details

The palette is created using `colorRampPalette`.

Value

A vector of colors.

Note

The call to `colorRampPalette` sets `space="Lab"` and `interpolate="spline"` with the intent of creating a more perceptually uniform rainbow.

Author(s)

M.L. Peck

See Also

`grey256`

Examples

```
plotsp <- function(spectrum) {
sl <- length(spectrum)
rgbv <- col2rgb(spectrum)
plot((0:(sl-1))+0.5, rgbv[1,], type="l", col="red", xlim=c(0,sl),ylim=c(0,300),xlab="Index",ylab="Channel value"
points((0:(sl-1))+0.5, rgbv[2,], type="l", col="green")
points((0:(sl-1))+0.5, rgbv[3,], type="l", col="blue")
grid()
rect(0:(sl-1), 260, 1:sl, 300, col=spectrum, density=NA)
}
plotsp(rygcb(400))
X11()
plotsp(rygcbm(500))
```

`rzernike_ann` *Radial Zernike Annular polynomials*

Description

Create a matrix of Radial Zernike annular polynomial values for a given azimuthal order `m` and all radial orders up to `n`.

Usage

```
rzernike_ann(rho, eps, n, m, xq, qwts)
```

Arguments

`rho`	a vector of radial coordinates.
`eps`	the obstruction fraction 0 <= eps < 1.
`n`	the maximum radial order required
`m`	azimuthal order
`xq`	nodes for Gauss-Legendre quadrature
`qwts`	weights for Gauss-Legendre quadrature

Details

To the author's knowledge no recurrence relations for radial Zernike annular polynomials have been published, even though several are well known for the closely related Zernike circle polynomials. The m=0 polynomials representing axially symmetric aberrations however are just shifted Legendre polynomials with an easily derived recurrence relation. For the m>0 polynomials this routine calculates the recurrence coefficients iteratively using a method attributed to Stieltjes.

Value

A length(rho) x (n-m)/2+1 column matrix of radial Zernike Annular polynomial values evaluated at the input radial coordinates. The radial indexes are in increasing order from m, m+2, ..., n.

References

Gautschi, W. 1982, "On Generating Orthogonal Polynomials", SIAM J. Sci. Stat. Comput. vol. 3, no.3, 289-317.\

Mahajan, V.N. 1981, "Zarnike annular polynomials...", JOSA, vol. 71, no. 1, 75-85.\

Mahajan, V.N. 1994, "Zernike annular polynomials...", Suppl. Applied Optics, vol. 5, No. 11, 8125-8128.\

See Also

This function is called by `zapm()` and `zapm_iso()`.

`rzernike_ann_direct` *Radial Zernike annular polynomials from formulas*

Description

The first 15 radial Zernike annular polynomials copied from Table 1 of Mahajan (1994)

Usage

```
rzernike_ann_direct(rho, eps, j)
```

Arguments

`rho`	a vector of radial coordinates.
`eps`	the obstruction fraction 0 <= eps < 1.
`j`	index of the value to fetch

Details

This function is included for testing and reference only.

Value

A vector of radial Zernike annular polynomial values at the coordinates `rho` for index `j`

References

Mahajan, V. N. (1994). Zernike Annular Polynomials and Optical Aberrations of Systems with Annular Pupils. *Supplement to Applied Optics* **33**, 8125-8127.

See Also

This function is called by `zapm_direct()` and `zapm_iso_direct()`. Complete sequences to arbitrary order can be calculated with `rzernike_ann()` and `rzernike_ann_128()`.

sconic *Sconic*

Description

twice the radial height difference between a sphere and conic surface

Usage

```
sconic(D, rc, b = -1, eps = 0., lambda = 632.8, nmax = 6)
```

Arguments

`D`	Diameter (mm)
`rc`	Radius of curvature (mm)
`b`	conic constant (default -1)
`eps`	obstruction fraction (default 0)
`lambda`	source or test wavelength (nm)
`nmax`	maximum polynomial order

Details

For the typical use case `D` and `rc` should be in millimeters and `lambda` in nanometers.

If `eps` > `0` Zernike annular coefficients will be computed and returned.

Value

Zernike polynomial coefficients. The default returns the 4th and 6th order coefficients in a length two vector.

Note

This estimates twice the radial distance between a sphere and conic surface with same paraxial radius of curvature, and returns Zernike coefficients of polynomial expansion. Intended for "numerical nulling" when testing an asphere at center of curvature, and should be more accurate than the vertical height difference calculated by `zconic` for that purpose.

Author(s)

M.L. Peck

See Also

`zconic`

Examples

```
2.*zconic(1000,5000)
sconic(1000,5000)
```

`startest` *Star test simulator*

Description

Simulates a star test.

Usage

```
startest(wf=NULL, zcoef=NULL, maxorder=14L, phi=0,
lambda = 1, defocus=5, cp=NULL,
obstruct=NULL, npad = 4,
gamma=2, psfmag=2, displaymtf=TRUE, displaywf=FALSE)
```

Arguments

`wf`	A matrix of class `pupil` containing wavefront values
`zcoef`	Vector of Zernike coefficients
`maxorder`	maximum Zernike polynomial order
`phi`	Angle to rotate wavefront
`lambda`	Wavelength, in same units as coefficients
`defocus`	Amount of defocus in waves
`cp`	pupil parameters
`obstruct`	Obstruction fraction
`npad`	Pad factor for FFT
`gamma`	Gamma value for graphics display
`psfmag`	Magnification factor for in focus PSF display
`displaymtf`	Logical: Display MTF?
`displaywf`	Logical: Display calculated wavefront?

Details

If `wf` is `NULL` the wavefront is calculated from the the Zernike coefficients (which should be non-`NULL`).

Value

A list with the following components:

`psf`	The in focus point spread function.
`otf`	The complex optical transfer function, a complex matrix of size `pupilsize`.
`mtf`	The modulation transfer function, a real matrix of size `pupilsize`.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

References

Born, M. and Wolf, E. 1999, *Principles of Optics, 7th Edition*, Cambridge University Press.

Suiter, H. R., 1994, *Star Testing Astronomical Telescopes*, Willman-Bell, Inc.

See Also

`Zernike`, `pupil`.

`summary.wf_zfit` *Methods for class "wf_zfit"*

Description

Summary, print, plot, report, and invert methods for the returned list of values from `psifit()`, `fftfit()`, or `vortexfit()`

Usage

```
## S3 method for class 'wf_zfit'
summary(wffit, digits = 3, printnow = TRUE)
```

Arguments

`wffit`	the return values from one of the fringe analysis routines or `wf_net()`
`digits`	number of digits to display in print and summary methods
`printnow`	send output to console?
`...`	values passed to `plot.pupil()`

Details

The report method combines the plot, summary, and print methods into an html document and displays it in a browser. The invert method negates the values of wavefronts and Zernike coefficients and returns the adjusted input

Value

summary and print methods return data frame with wavefront summaries and Zernike coefficients

synth.interferogram *Synthetic interferogram*

Description

Compute and display a synthetic interferogram.

Usage

```
synth.interferogram(wf = NULL, zcoef = NULL, maxorder = NULL,
                    nr = nrow(wf), nc = ncol(wf), cp = NULL,
                    phi = 0, addzc = rep(0, 4), fringescale = 1,
                    plots = TRUE)
```

Arguments

`wf`	A matrix of wavefront values
`zcoef`	A vector of Zernike coefficients
`maxorder`	Maximum Zernike polynomial order
`nr`	Number of rows in the output matrix
`nc`	Number of columns in the output matrix
`cp`	A list describing the pupil boundaries, as created by `pupil.pars`
`phi`	Amount to rotate the wavefront, in degrees
`addzc`	A 4-vector with piston, tilt, and defocus terms to be added
`fringescale`	Fringe scale. Should be 1 for single pass, 0.5 for double, etc.
`plots`	Logical: Plot the interferogram?

Details

Either `wf` or `zcoef` must be non-null, but not both. If `zcoef` is specified `maxorder` must be as well.

Additional piston, tilt, and defocus terms can be added to the calculated wavefront using `addzc`.

Value

A matrix of intensity levels in the calculated interferogram, assigned class `"pupil"`.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`pupil`.

Examples

```
# create a list of zernikes
zcoef <- rnorm(length(zlist.fr$n), mean=0, sd=0.01)

iwf <- synth.interferogram(zcoef=zcoef, maxorder=12)

X11()

# show again with some tilt

iwf <- synth.interferogram(zcoef=zcoef, maxorder=12, addzc=c(0,5,5,0))
```

`turbwf` *Kolmogorov Turbulence*

Description

Simulates the optical effects of atmospheric turbulence using Noll's (1976) calculation of the covariance matrix of Zernike polynomials under Kolmogorov turbulence.

Usage

```
turbwf(friedratio = 1, zlist = makezlist(2, 40), reps = 1)
```

Arguments

`friedratio`	Ratio of pupil diameter to Fried parameter
`zlist`	A list of Zernikes, as returned for example by `makezlist`
`reps`	Number of draws to simulate

Details

The default value of zlist has 440 elements, which may be more than necessary for a reasonable representation of an "atmospheric" wavefront.

Value

A list with the following components:

`zcoef.turb`	A `reps` x `length(zlist$n` matrix of simulated draws of Zernike coefficients.
`V`	Covariance matrix of the indexed Zernikes.

Note

Typos in the original source material have been corrected in the code. Note that scintillation is not modelled.

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

References

Noll, R.J. 1976, **Zernike polynomials and atmospheric turbulence**, *J. Opt. Soc. Am.*, Vol. 66, No. 3, p. 207.

See Also

Zernike, pupil.

Examples

```
# Simulate a single draw from a turbulent atmosphere
zcoef.turb <- turbwf(friedratio=5, zlist=makezlist(2,30), reps=1)$zcoef.turb
# Warning: this can take a while
wf <- pupil(zcoef=zcoef.turb, zlist=makezlist(2,30))
plot(wf)
summary(wf)
```

vortexfit *Vortex transform.*

Description

Fringe analysis by Vortex aka Spiral Quadrature transform.

Usage

```
vortexfit(
  img,
  cp = NULL,
  filter = NULL,
  fw.o = 10,
  options = zernike_options()
)
```

Arguments

img	matrix containing the interferogram data
cp	list with circle parameters describing interferogram location. Defaults to NULL
filter	size of filter to remove background
fw.o	size of gaussian blur to smooth orientation estimate
options	A list with general fitting and display options. See zernike_options().

Details

Implements the Vortex or spiral phase quadrature transform method of Larkin et al. (2001) https://doi.org/10.1364/JOSAA.18
including the fringe orientation estimation approach in Larkin (2005) https://doi.org/10.1364/OPEX.13.008097.
Thanks to Steve Koehler for ideas on implementation details.

Value

a list with wavefront estimates, wrapped phase, modulation, etc. The return has S3 class 'wf_zfit' with plot, print, summary, and report methods.

Warning

This routine is offered as is with no license, as it may be in violation of one or more US and international patents.

See Also

This is one of two routines provided for analysis of single interferograms, along with `fftfit()`. This *may* be suitable for interferograms with closed fringes.

Examples

```
require(zernike)
fpath <- file.path(find.package(package="zernike"), "psidata")
fname <- "Image197.jpg"
img <- load.images(file.path(fpath, fname))

# parameters for this run

source(file.path(fpath, "parameters.txt"))

# target SA coefficients for numerical null.

sa.t <- sconic(diam,roc,lambda=wavelength)
zopt <- zernike_options()
zopt$satarget <- sa.t

# display an interferogram

if (tolower(.Platform$OS.type) == "windows") windows() else x11()
image(1:nrow(img), 1:ncol(img), img, col=grey256, asp=1,
 xlab="X", ylab="Y", useRaster=TRUE)
mtext("Sample Interferogram")

if (tolower(.Platform$OS.type) == "windows") windows() else x11()
vfit <- vortexfit(img, filter=15, fw.o=10, options=zopt)

# do "classical FFT" based fit and compare results

dev.set(dev.next())
ftfit <- fftfit(img, cp=vfit$cp, sl=c(32, 0), filter=25, options=zopt)
```

```
plotn(ftfit, vfit, labels=c("fft", "vortex"))
```

`wf3d.pupil` *OpenGL wavefront plot*

Description

Interactive plot of a wavefront using the OpenGL package **rgl**. This is a 3D plotting method for objects of class "pupil".

Usage

```
wf3d.pupil(wf, cp=NULL, zoom.wf = 1, surf.col = topo.colors(256), bg.col = "black",
    eqa=FALSE, new.window=TRUE)
```

Arguments

`wf`	A matrix of wavefront values
`cp`	A list describing the pupil boundary
`zoom.wf`	Zoom factor for heights
`surf.col`	Color palette for surface
`bg.col`	Background color
`eqa`	Equal area per color?
`new.window`	If TRUE (the default), open a new rgl deice. Otherwise clears and re-uses an existing window if available

Details

The default color palette will match the colors in the default version of `plot.pupil`.

Value

none

Author(s)

M.L. Peck <mpeck1@ix.netcom.com>

References

The **rgl** package is available from CRAN.

See Also

`plot.pupil`

Examples

```
# create a random wavefront

wf <- pupil(zcoef=rnorm(length(makezlist()$n), mean=0, sd=0.01))
# the default method

plot(wf)

#this is more fun

wf3d(wf)
```

---

wf_net *Wavefront post-processing*

Description

Perform post-processing of phase and modulation estimates performed by any of the fringe analysis routines

Usage

```
wf_net(phi, mod, cp, options)
```

Arguments

`phi`	wrapped phase map
`mod`	modulation estimate
`cp`	a list describing the pupil boundary
`options`	a list of options. See `psfit_options`

Details

Called by `psifit`, `vortexfit`, `fftfit`. This now performs all of the post-processing on a calculated wrapped phase and modulus estimate from any of the fringe analysis routines in this package. This includes calculating the actual interferogram location parameters if not specified at call time, cropping if desired, unwrapping the phase, calculating a net and Zernike fit wavefront.

Value

A list with the following components:

`phi`	the input phase, cropped if requested
`mod`	the input modulus
`cp`	interferogram location parameters as returned by `circle.pars`
`cp.orig`	input cp

`wf.net`	Net unsmoothed wavefront; a matrix of class "`pupil`"
`wf.smooth`	Net smoothed wavefront
`wf.residual`	Difference between net wavefront and polynomial fit
`fit`	Return value from `fitzernikes`
`zcoef.net`	Net Zernike coefficients from fit

Note

As of ver. 3.7.2 the values of `options$isoseq` and `cp$obstruct` determine what is called to fit Zernikes. If `isoseq` is `TRUE` it's either `zpm_cart` or `zapm_cart`, with the latter called automatically if `cp$obstruct` > `0`. Otherwise the calls are made to `zpm` or `zapmC`.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

Examples

```
## illustrates effect of incremental addition of an obstruction
## and standalone use of wf_net

require(zernike)
fpath <- file.path(find.package(package="zernike"), "psidata")
files <- scan(file.path(fpath, "files.txt"), what="character")
for (i in 1:length(files)) files[i] <- file.path(fpath, files[i])

# load the images into an array

images <- load.images(files)

# parameters for this run

source(file.path(fpath, "parameters.txt"))

# phase shifts

phases <- wrap((0:(dim(images)[3]-1))/frames.per.cycle*2*pi)
phases <- switch(ps.dir, ccw = -phases, cw = phases, phases)

# target SA coefficients for numerical null.

sa.t <- sconic(diam,roc,lambda=wavelength)
zopt <- psfit_options(satarget=sa.t)
psfit <- psifit(images, phases, psialg="ls", options=zopt)

# get back the raw wavefront

cp <- psfit$cp

eps <- seq(0.1, 0.5, by=0.1)
ne <- length(eps)
```

```
pad0 <- rep(0, ne)

## collect some basic data

df.annfits <- data.frame(eps=c(0, eps), sa.null = c(sa.t[1], pad0), sa.obs = c(psfit$fit[9], pad0), sa.net = c(psfi
                rms.net = c(sqrt(crossprod(psfit$zcoef.net)), pad0), defocus=c(psfit$fit[4], pad0))
for (i in seq_along(eps)) {
  cp$obstruct <- eps[i]
  sa.t <- sconic(diam, roc, eps=eps[i], lambda=wavelength)
  zopt$satarget <- sa.t
  fiti <- wf_net(psfit$phi, psfit$mod, cp=cp, options=zopt)
 df.annfits[i+1, 2:ncol(df.annfits)] <- c(sa.t[1], fiti$fit[9], fiti$zcoef.net[8], sqrt(crossprod(fiti$zcoef.net
}

rm(eps)
attach(df.annfits)

X11()
plot(eps, sa.null, type="b", ylim=range(df.annfits[,2:ncol(df.annfits)]), xlim=c(0, 0.8), main="Trends with obstr
points(eps, sa.obs, type="b", pch=2, col=2, lty=2)
points(eps, sa.net, type="b", pch=3, col=3, lty=3)
points(eps, rms.net, type="b", pch=4, col=4, lty=4)
points(eps, defocus, type="b", pch=5, col=5, lty=5)
grid()
legend(x=0.6, y=0, legend=names(df.annfits)[-1], col=1:5, lty=1:5, pch=1:5)

## plot the first and last smoothed wavefronts to show difference is completely symmetrical

plotn(psfit, fiti, wftype="smooth", qt=c(0,1))
detach(df.annfits)
print(df.annfits, digits=2)
```

zapm *Zernike Annular polynomials*

Description

Create a matrix of Zernike Annular polynomial values in extended Fringe sequence for a set of polar coordinates.

Usage

```
zapm(rho, theta, eps, maxorder = 14L, nqplus = 6L)
```

Arguments

`rho`	a vector of radial coordinates with eps <= rho <= 1.
`theta`	a vector of angular coordinates, in radians.

`eps`	the obstruction fraction 0 <= eps < 1.
`maxorder`	the maximum radial polynomial order (defaults to 14).
`nqplus`	the number of *extra* quadrature points for numerical integration

Details

The *radial* polynomials are calculated using recurrence relations generated numerically using a method credited to Stieltjes. See the documentation for `rzernike_ann()` for further details and literature references. A formal presentation will be published elsewhere.

Value

a matrix of Zernike Annular polynomial values evaluated at the input polar coordinates and all radial orders from 0 through `maxorder` in Fringe sequence, with orthonormal scaling.

Examples

```
sample_az <- function(maxorder=12, eps=0.33, col=rev(zernike::rygcb(400)), addContours=TRUE, cscale=TRUE) {

  ## get coordinates for unobstructed and obstructed apertures
  cpa <- cp.default
  cpa$obstruct <- eps
  prt <- pupil.rhotheta(nrow.default,ncol.default,cp.default)
  prta <- pupil.rhotheta(nrow.default,ncol.default,cp=cpa)
  rho0 <- prt$rho[!is.na(prt$rho)]
  theta0 <- prt$theta[!is.na(prt$theta)]
  rhoa <- prta$rho[!is.na(prta$rho)]
  thetaa <- prta$theta[!is.na(prta$theta)]

  ## fill up matrixes of Zernikes and Annular Zernikes

  zm <- zpm(rho0, theta0, maxorder=maxorder)
  zam <- zapm(rhoa, thetaa, eps=eps, maxorder=maxorder, nq=maxorder/2+5)

  ## pick a column at random and look up its index pair

  zlist <- makezlist(0, maxorder)
  i <- sample(2:ncol(zm), 1)
  n <- zlist$n[i]
  m <- zlist$m[i]

  ## fill up the wavefront representations and plot them

  wf0 <- prt$rho
  wf0[!is.na(wf0)] <- zm[,i]
  class(wf0) <- "pupil"

  wfa <- prta$rho
  wfa[!is.na(wfa)] <- zam[,i]
  class(wfa) <- "pupil"
```

```
    plot(wf0, cp=cp.default, col=col, addContours=addContours, cscale=cscale)
    mtext(paste("Zernike, n =", n, " m =", m))

    x11()
    plot(wfa, cp=cpa, col=col, addContours=addContours, cscale=cscale)
    mtext(paste("Annular Zernike, n =", n, " m =", m))

    ## return Zernike matrices and wavefronts invisibly
    ## just in case user wants to do something with them

    invisible(list(zm=zm, wf0=wf0, zam=zam, wfa=wfa))
  }

  sample_az()
```

---

zapm_128 *Zernike Annular polynomials, extended precision version*

Description

Create a matrix of Zernike Annular polynomial values in extended Fringe sequence for a set of polar coordinates.

Usage

```
zapm_128(rho, theta, eps, maxorder = 14L, nqplus = 6L)
```

Arguments

rho	a vector of radial coordinates with eps <= rho <= 1.
theta	a vector of angular coordinates, in radians.
eps	the obstruction fraction 0 <= eps < 1.
maxorder	the maximum radial polynomial order (defaults to 14).
nqplus	the number of *extra* quadrature nodes (defaults to 6).

Details

The *radial* polynomials are calculated using recurrence relations generated numerically using Stieltje's procedure with nominally exact numerical quadrature. See the documentation for `rzernike_ann()`. A formal presentation is included in the package documentation.

Value

a matrix of Zernike Annular polynomial values evaluated at the input polar coordinates and all radial orders from 0 through `maxorder` in Fringe sequence, with orthonormal scaling.

Examples

```
sample_az_128 <- function(maxorder=12, eps=0.33, col=rev(zernike::rygcb(400)), addContours=TRUE, cscale=TRUE) {

   ## get coordinates for unobstructed and obstructed apertures
   cpa <- cp.default
   cpa$obstruct <- eps
   prt <- pupil.rhotheta(nrow.default,ncol.default,cp.default)
   prta <- pupil.rhotheta(nrow.default,ncol.default,cp=cpa)
   rho0 <- prt$rho[!is.na(prt$rho)]
   theta0 <- prt$theta[!is.na(prt$theta)]
   rhoa <- prta$rho[!is.na(prta$rho)]
   thetaa <- prta$theta[!is.na(prta$theta)]

   ## fill up matrixes of Zernikes and Annular Zernikes

   zm <- zpm(rho0, theta0, maxorder=maxorder)
   zam <- zapm_128(rhoa, thetaa, eps=eps, maxorder=maxorder)

   ## pick a column at random and look up its index pair

   zlist <- makezlist(0, maxorder)
   i <- sample(2:ncol(zm), 1)
   n <- zlist$n[i]
   m <- zlist$m[i]

   ## fill up the wavefront representations and plot them

   wf0 <- prt$rho
   wf0[!is.na(wf0)] <- zm[,i]
   class(wf0) <- "pupil"

   wfa <- prta$rho
   wfa[!is.na(wfa)] <- zam[,i]
   class(wfa) <- "pupil"

   plot(wf0, cp=cp.default, col=col, addContours=addContours, cscale=cscale)
   mtext(paste("Zernike, n =", n, " m =", m))

   x11()
   plot(wfa, cp=cpa, col=col, addContours=addContours, cscale=cscale)
   mtext(paste("Annular Zernike, n =", n, " m =", m))

   ## return Zernike matrices and wavefronts invisibly
   ## just in case user wants to do something with them

   invisible(list(zm=zm, wf0=wf0, zam=zam, wfa=wfa))
 }

 sample_az_128()
```

`zapm_direct` *Zernike Annular polynomials from formulas*

Description

Create a matrix of Zernike Annular polynomial values for the counterparts of primary and secondary optical aberrations.

Usage

```
zapm_direct(rho, theta, eps)
```

Arguments

`rho`	a vector of radial coordinates with eps <= rho <= 1.
`theta`	a vector of angular coordinates, in radians.
`eps`	the obstruction fraction 0 <= eps < 1.

Details

The values are from published formulas. This function is included for testing and reference only.

Value

a `length(rho) x 16` matrix of Zernike annular polynomial values.

See Also

Calls `rzernike_ann_direct()` for radial Zernike annular values. Use `zapm()` or `zapm_128()` for complete sequences of values to arbitrary order.

`zapm_iso` *Zernike Annular polynomials, ISO ordering*

Description

Create a matrix of Zernike Annular polynomial values in ISO/ANSI sequence for a set of polar coordinates.

Usage

```
zapm_iso(rho, theta, eps, maxorder = 14L, nqplus = 6L)
```

Arguments

`rho`	a vector of radial coordinates with eps <= rho <= 1.
`theta`	a vector of angular coordinates, in radians.
`eps`	the obstruction fraction 0 <= eps < 1.
`maxorder`	the maximum radial and azimuthal polynomial order (defaults to 14).
`nqplus`	the number of *extra* quadrature points for numerical integration

Details

The *radial* polynomials are calculated using recurrence relations generated numerically using a method credited to Stieltjes. See the documentation for `rzernike_ann()` for further details and literature references. A formal presentation will be published elsewhere.

Value

a matrix of Zernike Annular polynomial values evaluated at the input polar coordinates and all radial orders from 0 through `maxorder` in ISO/ANSI sequence, with orthonormal scaling.

Examples

```
sample_az_iso <- function(maxorder=12, eps=0.33, col=rev(zernike::rygcb(400)), addContours=TRUE, cscale=TRUE) {

  ## get coordinates for unobstructed and obstructed apertures
  cpa <- cp.default
  cpa$obstruct <- eps
  prt <- pupil.rhotheta(nrow.default,ncol.default,cp.default)
  prta <- pupil.rhotheta(nrow.default,ncol.default,cp=cpa)
  rho0 <- prt$rho[!is.na(prt$rho)]
  theta0 <- prt$theta[!is.na(prt$theta)]
  rhoa <- prta$rho[!is.na(prta$rho)]
  thetaa <- prta$theta[!is.na(prta$theta)]

  ## fill up matrixes of Zernikes and Annular Zernikes

  zm <- zpm_cart(x=rho0*cos(theta0), y=rho0*sin(theta0), maxorder=maxorder)
  zam <- zapm_iso(rhoa, thetaa, eps=eps, maxorder=maxorder, nq=maxorder/2+5)

  ## pick a column at random and look up its index pair

  zlist <- makezlist.iso(maxorder)
  i <- sample(2:ncol(zm), 1)
  n <- zlist$n[i]
  m <- zlist$m[i]

  ## fill up the wavefront representations and plot them

  wf0 <- prt$rho
  wf0[!is.na(wf0)] <- zm[,i]
  class(wf0) <- "pupil"
```

```
    wfa <- prta$rho
    wfa[!is.na(wfa)] <- zam[,i]
    class(wfa) <- "pupil"

    plot(wf0, cp=cp.default, col=col, addContours=addContours, cscale=cscale)
    mtext(paste("Zernike, n =", n, " m =", m))

    x11()
    plot(wfa, cp=cpa, col=col, addContours=addContours, cscale=cscale)
    mtext(paste("Annular Zernike, n =", n, " m =", m))

    ## return Zernike matrices and wavefronts invisibly
    ## just in case user wants to do something with them

    invisible(list(zm=zm, wf0=wf0, zam=zam, wfa=wfa))
  }

  sample_az_iso()
```

`zapm_iso_128` *Zernike Annular polynomials, ISO ordering - extended precision version*

Description

Create a matrix of Zernike Annular polynomial values in ISO/ANSI sequence for a set of polar coordinates.

Usage

```
zapm_iso_128(rho, theta, eps, maxorder = 14L, nqplus = 6L)
```

Arguments

`rho`	a vector of radial coordinates with eps <= rho <= 1.
`theta`	a vector of angular coordinates, in radians.
`eps`	the obstruction fraction 0 <= eps < 1.
`maxorder`	the maximum radial and azimuthal polynomial order (defaults to 14).
`nqplus`	the number of *extra* quadrature nodes (defaults to 6).

Details

The *radial* polynomials are calculated using recurrence relations generated numerically using Stieltje's procedure with nominally exact numerical quadrature. See the documentation for `rzernike_ann()`. A formal presentation is included in the package documentation.

Value

a matrix of Zernike Annular polynomial values evaluated at the input polar coordinates and all radial orders from 0 through `maxorder` in ISO/ANSI sequence, with orthonormal scaling.

Examples

```
sample_az_iso_128 <- function(maxorder=12, eps=0.33, col=rev(zernike::rygcb(400)), addContours=TRUE, cscale=TRU

   ## get coordinates for unobstructed and obstructed apertures
   cpa <- cp.default
   cpa$obstruct <- eps
   prt <- pupil.rhotheta(nrow.default,ncol.default,cp.default)
   prta <- pupil.rhotheta(nrow.default,ncol.default,cp=cpa)
   rho0 <- prt$rho[!is.na(prt$rho)]
   theta0 <- prt$theta[!is.na(prt$theta)]
   rhoa <- prta$rho[!is.na(prta$rho)]
   thetaa <- prta$theta[!is.na(prta$theta)]

   ## fill up matrixes of Zernikes and Annular Zernikes

   zm <- zpm_cart(x=rho0*cos(theta0), y=rho0*sin(theta0), maxorder=maxorder)
   zam <- zapm_iso_128(rhoa, thetaa, eps=eps, maxorder=maxorder)

   ## pick a column at random and look up its index pair

   zlist <- makezlist.iso(maxorder)
   i <- sample(2:ncol(zm), 1)
   n <- zlist$n[i]
   m <- zlist$m[i]

   ## fill up the wavefront representations and plot them

   wf0 <- prt$rho
   wf0[!is.na(wf0)] <- zm[,i]
   class(wf0) <- "pupil"

   wfa <- prta$rho
   wfa[!is.na(wfa)] <- zam[,i]
   class(wfa) <- "pupil"

   plot(wf0, cp=cp.default, col=col, addContours=addContours, cscale=cscale)
   mtext(paste("Zernike, n =", n, " m =", m))

   x11()
   plot(wfa, cp=cpa, col=col, addContours=addContours, cscale=cscale)
   mtext(paste("Annular Zernike, n =", n, " m =", m))

   ## return Zernike matrices and wavefronts invisibly
   ## just in case user wants to do something with them

   invisible(list(zm=zm, wf0=wf0, zam=zam, wfa=wfa))
 }
```

```
  sample_az_iso_128()
```

`zapm_iso_direct` *Zernike Annular polynomials from formulas*

Description

Create a matrix of Zernike Annular polynomial values complete through radial and azimuthal orders 6 in ISO/ANSI sequence.

Usage

```
zapm_iso_direct(rho, theta, eps)
```

Arguments

`rho`	a vector of radial coordinates with eps <= rho <= 1.
`theta`	a vector of angular coordinates, in radians.
`eps`	the obstruction fraction 0 <= eps < 1.

Details

The values are from published formulas. This function is included for testing and reference only.

Value

a `length(rho) x 28` matrix of Zernike annular polynomial values.

See Also

Calls `rzernike_ann_direct()` for radial Zernike annular values. Use `zapm_iso()` or `zapm_iso_128()` for complete sequences of values to arbitrary order.

Examples

```
sample_az_iso_direct <- function(eps=0.33, col=rev(zernike::rygcb(400)), addContours=TRUE, cscale=TRUE) {

 ## get coordinates for obstructed aperture
 cp <- cp.default
 cp$obstruct <- eps
 prt <- pupil.rhotheta(nrow.default,ncol.default,cp=cp)
 rho <- prt$rho[!is.na(prt$rho)]
 theta <- prt$theta[!is.na(prt$theta)]

 ## fill up matrixes of Annular Zernikes using direct and recursive formulas

 zam_dir <- zapm_iso_direct(rho, theta, eps=eps)
```

```
  zam_iso <- zapm_iso(rho, theta, eps=eps, maxorder=6)

  ## pick a column at random and look up its index pair

  zlist <- makezlist.iso(6)
  i <- sample(2:ncol(zam_iso), 1)
  n <- zlist$n[i]
  m <- zlist$m[i]

  ## fill up the wavefront representations and plot them

  wf_dir <- prt$rho
  wf_dir[!is.na(wf_dir)] <- zam_dir[,i]
  class(wf_dir) <- "pupil"

  wf_zapm <- wf_dir
  wf_zapm[!is.na(wf_zapm)] <- zam_iso[,i]
  class(wf_zapm) <- "pupil"

  plot(wf_dir, cp=cp, col=col, addContours=addContours, cscale=cscale)
  mtext(paste("Annular Zernike from formula, n =", n, " m =", m))

  x11()
  plot(wf_zapm, cp=cp, col=col, addContours=addContours, cscale=cscale)
  mtext(paste("Annular Zernike from `zapm_iso`, n =", n, " m =", m))

 cat(paste("Sample wavefronts: SD of diffs", format(sd(wf_dir-wf_zapm, na.rm=TRUE), digits=6), "; range of diffs",
     format(min(wf_zapm-wf_dir, na.rm=TRUE), digits=6),":",
     format(max(wf_zapm-wf_dir, na.rm=TRUE), digits=6),"\n\n"))

 cat(paste("Zernike matrixes: SD of diffs", format(sd(zam_dir-zam_iso), digits=6), "; range of diffs",
  format(min(zam_dir-zam_iso), digits=6),":",
  format(max(zam_dir-zam_iso), digits=6),"\n"))

  ## return Zernike matrices and wavefronts invisibly
  ## just in case user wants to do something with them

  invisible(list(zam_dir=zam_dir, zam_iso=zam_iso, wf_dir=wf_dir, wf_zapm=wf_zapm))
 }

sample_az_iso_direct()
```

zconic *Zernike coefficients for a conic surface*

Description

Calculates the radially symmetric Zernike coefficient values up to order nmax for a conic surface relative to a sphere of the same paraxial radius of curvature.

Usage

```
zconic(D, rc, b = -1, lambda = 1e-06, nmax = 6)
```

Arguments

`D`	Diameter
`rc`	Radius of curvature
`b`	Conic constant
`lambda`	Wavelength – defaults to 632.8 (nm)
`nmax`	Maximum radial polynomial order. Defaults to 6.

Details

For the typical use case `D` and `rc` should be in millimeters and `lambda` in nanometers.

Value

A vector of length `nmax/2-1` of coefficient values, in increasing radial order, n=c(4,6, ...).

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`Zernike`, `sconic`

Examples

```
zconic(200,2000)
zconic(10, 20, b=-1.05, lambda=632.8E-9, nmax=12)
```

`Zernike` *Zernike Polynomials*

Description

Routines for creating and manipulating Zernike polynomials.

Usage

```
Zernike(rho, theta, n, m, t)
rzernike(rho, n, m)
drzernike(rho, n, m)
```

Arguments

`rho`	normalized radius, $0 <= rho <= 1$
`theta`	angular coordinate
`n`	radial polynomial order
`m`	azimuthal order
`t`	character for trig function: one of c("n", "c", "s")

Note

These functions return Zernikes scaled such that they form an orthonormal basis set for the space of functions defined on the unit circle. Note that this is not the most commonly used definition (as given e.g. in *Born and Wolf*). The definition I use is often associated with *Noll (1976)*.

The function `zmult` can be used to convert between normalized and conventionally defined vectors of Zernike coefficients.

The basic low level functions `rzernike` and `drzernike` use numerically stable recurrence relationships for the radial Zernikes.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

References

Born, M. and Wolf, E. 1999, *Principles of Optics, 7th Edition*, Cambridge University Press, chapter 9 and appendix VII.

Noll, R.J. 1976, **Zernike polynomials and atmospheric turbulence**, *J. Opt. Soc. Am.*, Vol. 66, No. 3, p. 207.

`http://wyant.opt-sci.arizona.edu/zernikes/zernikes.htm`

`http://mathworld.wolfram.com/ZernikePolynomial.html`

See Also

`makezlist`, `zlist.fr`, `zmult`, `zpm`, `pupil`, `pupilrms`, `pupilpv`, `strehlratio`.

Examples

```
Zernike(1, 0, 4, 0, "n") # == sqrt(5)

# A slightly more complex example

rho <- seq(0, 1, length = 101)
theta <- rep(0, 101)

plot(rho, Zernike(rho, theta, 6, 0, "n"), type="l",
  ylim=c(-3.5,3.5), main="Some 6th order Zernike Polynomials")
lines(rho, Zernike(rho, theta, 5, 1, "c"), lty=2)
lines(rho, Zernike(rho, theta, 4, 2, "c"), lty=3)
lines(rho, Zernike(rho, theta, 3, 3, "c"), lty=4)
```

`zernike_options` *Options for routines in package 'zernike'.*

Description

Get and optionally set parameters controlling various aspects of PSI algorithms, Zernike polynomial fitting, and data display

Usage

```
zernike_options(...)
```

Arguments

`colors`	topo.colors(256)
`plots`	TRUE
`refine`	TRUE
`puw_alg`	"qual"
`fringescale`	1
`wt`	NULL
`bgsub`	TRUE
`maxiter`	20
`ptol`	1e-04
`trace`	1
`nzcs`	2
`zc0`	6:7
`satarget`	c(0, 0)
`astig.bath`	c(0, 0)
`maxorder`	14
`isoseq`	FALSE
`usecirc`	FALSE
`nthreads`	parallel::detectCores()/2
`crop`	FALSE
`sigma_ed`	3.
`qt_ed`	0.995

Details

Calling zernike_options with an empty argument list returns the default values of the options used in psifit and wf_net as itemized above. The list can be modified directly or by passing argument value pairs to the function call.

Parameters you might want to change include:

satarget sets the target SA for “numerical nulling.“ This is a vector of length 2 setting the target values of primary and 5th order SA.

ptol sets convergence tolerances for iterative PSI algorithms. These have different definitions and different values may be suitable for different algorithms. A value around 0.01 is appropriate for tiltpsi.

The number of variable Zernike terms in the algorithm tiltpsi is controlled by nzcs. Set it to 3 to include defocus, 5 to include primary astigmatism, 7 to include coma.

maxorder sets the maximum Zernike polynomial order for wavefront fitting. It must be even and at least 6. The default generally produces a good wavefront representation but you may want to experiment with higher order fits.

Parameters isoseq and usecirc control which sets of Zernike (annular) polynomials are called for wavefront fitting. If Zernikes in ANSI/ISO sequence are desired set isoseq = TRUE. The default FALSE uses extended Fringe ordering. Set usecirc = TRUE if you want to use Zernike circle polynomials even for obstructed wavefronts.

puw_alg specifies the phase unwrapping algorithm. Current options are "qual", "brcut", "lpbrcut", and "lp".

A new and somewhat experimental feature from version 3.6.0 of the package is threaded computation of Zernike polynomial matrices. The number of threads used is set with the option nthreads. The default is to use half the number of cores detected because on CPUs that support multi- or hyperthreading the number of cores reported by detectCores() is double the number of physical cores. Setting nthreads = 1 will turn off threading. Different C++ routines are used for the matrix fill in the unthreaded (zpm) and threaded (zpmCP) cases. Speed improvements, if any, may vary.

sigma_ed and qt_ed control the smoothing and threshold parameters passed to the automated edge detection function circle.auto

If you don’t like the default color palette there are many other choices. If you like rainbows rygcb defined in this package produces a relatively perceptually uniform version that’s well suited for display on an RGB monitor.

Value

A named list with the current values of the arguments.

Author(s)

M.L. Peck <mlpeck54@gmail.com>

zlist *Lists of Zernike polynomial indexes*

Description

Ordered lists of Zernike polynomial indexes.

Usage

```
makezlist(minorder = 2, maxorder = 14)
zlist.fr
zmult(zlist = makezlist())
```

Arguments

`minorder`	minimum value of `n+m`
`maxorder`	maximum value of `n+m`
`zlist`	a list of the form returned by `makezlist`

Details

Zernike polynomials are indexed by a radial index `n`, an azimuthal index `m`, and include cosine, sine, and radial terms. These routines return lists of indexes using a popular ordering scheme for Zernike polynomials.

Value

`makezlist` and `zlist.fr` return lists with the following components:

`n`	radial order
`m`	azimuthal order
`t`	one of `c("c", "s", "n")`

`zmult` returns a vector the same length as the components of `zlist`.

Note

`zlist.fr` is an augmented “Fringe” set of Zernike polynomials equivalent to makezlist(2,12).

`makezlist` returns a complete list of indexes for all orders from `minorder` through `maxorder`, where “order” is the value of `n+m`.

Author(s)

M.L. Peck `<mpeck1@ix.netcom.com>`

See Also

Virtually all high level functions that work with Zernike polynomials use these lists. See for example `pupil`, `psifit`, `fftfit`.

Examples

```
zlist <- makezlist(2,12)
zcoef <- rnorm(length(zlist))
zcoef # a vector of normalized Zernike coefficients
zcoef*zmult(zlist)   # Coefficients in conventional representation
sqrt(crossprod(zcoef)) # This is the RMS error of the wavefront
# constructed from these Zernikes
```

`zmoments` *Zernike moments*

Description

Calculate Zernike moments from a vector of coefficients

Usage

```
zmoments(zcoef, maxorder = 14)
```

Arguments

`zcoef`	Zernike coefficients
`maxorder`	Maximum order to return

Value

A table of the moments along with radial and azimuthal orders

References

M.L. Peck

zpm *Matrixes of Zernike polynomials*

Description

Create a matrix of Zernike polynomial values.

Usage

```
zpm(rho, theta, phi= 0 , maxorder = 14, nthreads=parallel::detectCores()/2)
zpmCP(rho, theta, maxorder)
zpm.arb(rho, theta, phi = 0, zlist = makezlist())
```

Arguments

`rho`	A vector of radial coordinates.
`theta`	A vector of angular coordinates, in radians.
`phi`	Orientation of the image, in degrees
`zlist`	A list of indexes, as returned by `makezlist`
`maxorder`	The maximum Zernike polynomial order
`nthreads`	Number of threads for threaded function call

Details

`rho` and `theta` must be the same length.

Value

`zpm.arb` returns a matrix of size `length(rho)` x `length(zlist$n)` with values of Zernike polynomials evaluated at the polar coordinates `(rho, theta-pi*phi/180)`.

`zpm`, and `zpmCP` return a matrix of size `length(rho)` x `(maxorder/2+1)^2` of Zernike polynomial values including a piston term.

Note

These are used by various routines to make least squares fits of sets of Zernike polynomials to measured wavefront values.

Threaded computation of the matrix is now available using `zpmCP`.

Author(s)

M.L. Peck <`mpeck1@ix.netcom.com`>

See Also

`Zernike`, `makezlist`, `zlist.fr`, `fitzernikes`, `zapm` for Annular Zernikes.

`zpm_cart` *Zernike polynomials*

Description

Calculate Zernike polynomial values in ISO/ANSI sequence for a set of Cartesian coordinates.

Usage

```
zpm_cart(x, y, maxorder = 14L, unit_variance = TRUE)
```

Arguments

`x`	a vector of x coordinates for points on a unit disk.
`y`	a vector of y coordinates.
`maxorder`	the maximum radial polynomial order (defaults to 12).
`unit_variance`	logical: return with orthonormal scaling? (default `false`)

Details

This is the same algorithm and essentially the same code as `gradzpm()` except directional derivatives aren't calculated.

Value

a matrix of Zernike polynomial values evaluated at the input Cartesian coordinates and all radial and azimuthal orders from 0 through `maxorder`.

References

Anderson, T.B. (2018) Optics Express 26, #5, 18878 https://doi.org/10.1364/OE.26.018878 (open access)

Index

∗ **Graphics**
plotn, 41
plotxs, 42
pupil, 45
rygcb, 54
wf_net, 65
∗ **IO**
load.images, 34
∗ **Models**
psifit, 43
∗ **Utilities**
crop, 15
∗ **Utility**
hypot, 29
zmoments, 82
∗ **arith**
rescale, 52
sconic, 57
∗ **array**
rescale, 52
∗ **file**
load.images, 34
∗ **graphics**
col3d, 12
foucogram, 23
gray256, 29
pick.sidelobe, 38
plot.cmat, 39
startest, 58
synth.interferogram, 60
wf3d.pupil, 64
∗ **hplot**
col3d, 12
foucogram, 23
startest, 58
synth.interferogram, 60
wf3d.pupil, 64
∗ **list**
zernike_options, 79
∗ **mathematics**
aiapsi, 5
astig.bath, 6
fftfit, 17
idiffpuw, 30
lspsi, 35
pcapsi, 37
qpuw, 50
rmap, 53
turbwf, 61
zconic, 76
Zernike, 77
zlist, 81
zpm, 83
∗ **misc**
brcutpuw, 7
∗ **models**
gpcapsi, 26
∗ **optimization**
gpcapsi, 26
∗ **optimize**
brcutpuw, 7
∗ **smooth**
wf_net, 65
∗ **statistics**
fitzernikes, 22
pupilrms, 48
∗ **utilities**
circle.auto, 8
circle.man, 11
FFTUtilities, 19
pick.sidelobe, 38
plot.pupil, 40
pupil.rhotheta, 47

addfit, 3
aiapsi, 5, *44*
aiapsiC *(aiapsi)*, 5
astig.bath, 6

brcutpuw, 7, *30*, *31*, *50*, *54*
brcutpuw(), *31*, *32*

circle.auto, *4*, 8, *80*
circle.auto(), *10*, *16*
circle.hough, 9
circle.man, *4*, 11
circle.man(), *10*
circle.pars, *15*, *17*, *47*, *65*
col3d, 12
complex, *40*
conv2_sep_complex, 13
conv2_sep_complex(), *25*
conv2_sep_real, 13
conv2_sep_real(), *24*
convolve2d, 14
crop, 15

d_of_g, 15
drzernike *(*Zernike*)*, 77

fft, 16
fft(), *20–22*, *33*, *34*
fft_cx, 20
fft_cx_pad, 21
fft_pad, 21
fftfit, *3*, *4*, *9*, *11*, 17, *20*, *23*, *39*, *40*, *65*, *82*
fftfit(), *59*, *63*
fftshfit(), *33*
fftshift, 18
FFTUtilities, 19
fitzernikes, *17*, 22, *66*, *83*
foucogram, 23

gblur, 24
gblur(), *25*
gblur_complex, 25
gblur_fft, 25
gol_welsch, 26
gpcapsi, 26, *44*
gpcapsiC *(*gpcapsi*)*, 26
gradzpm, 27
gradzpm(), *37*, *84*
gray256, 29
grey256, *54*
grey256 *(*gray256*)*, 29

hkpsi *(*aiapsi*)*, 5
https://doi.org/10.1364/JOSAA.18.001862, *63*
https://doi.org/10.1364/OPEX.13.008097, *63*
hypot, 29

id_dxy_uw, 31
id_uw, 32
idiffpuw, *7*, 30, *50*
idiffpuw(), *31*, *32*, *51*
ifft, 32
ifft_real, 34
ifftshift, 33
ifftshift(), *19*
image.default, *40*, *41*

load.images, 34, *38*, *52*
load.pgm *(*load.images*)*, 34
locator, *11*, *39*
lspsi, 35, *44*
lspsiC *(*lspsi*)*, 35

makezlist, *61*, *78*, *83*
makezlist *(*zlist*)*, 81
makezlist.iso, 36

nextn(), *21*, *22*
norm_zpm, 37

padmatrix *(*FFTUtilities*)*, 19
pcapsi, *27*, 37, *44*
pick.sidelobe, *17*, *18*, 38, *40*
plot.cmat, *39*, 39
plot.pupil, *12*, 40, *42*, *43*, *46*, *64*
plot.pupil(), *59*
plotn, 41
plotxs, 42
psfit_options, *17*, *65*
psifit, *3*, *4*, *6*, *9*, *11*, *23*, *27*, *36*, *38*, 43, *65*, *80*, *82*
psifit(), *59*
pupil, *9*, *11*, *17*, *23*, *24*, *30*, *41*, 45, *47*, *48*, *50*, *59*, *60*, *62*, *64*, *66*, *78*, *82*
pupil.pars, *17*, *18*, *40*, *41*, *43*, *46*, *47*, *60*
pupil.rhotheta, 47
pupilpv, *41*, *47*, *78*
pupilpv *(*pupilrms*)*, 48
pupilrms, *41*, *47*, 48, *78*
PVr, 49

q_uw, 51
qpuw, *7*, 50

qpuw(), *51*

rescale, 52
rmap, *30*, *31*, 53
rygcb, 54, *80*
rygcbm *(*rygcb*)*, 54
rzernike *(*Zernike*)*, 77
rzernike_ann, 55
rzernike_ann(), *56*, *68*, *69*, *72*, *73*
rzernike_ann_128(), *56*
rzernike_ann_direct, 56
rzernike_ann_direct(), *71*, *75*

sconic, 57, *77*
separate.wf *(*addfit*)*, 3
startest, *20*, 58
strehlratio, *41*, *47*, *78*
strehlratio *(*pupilrms*)*, 48
submatrix *(*FFTUtilities*)*, 19
summary.pupil, *46*, *48*
summary.pupil *(*plot.pupil*)*, 40
summary.wf_zfit, 59
synth.interferogram, 60

tiltpsi, *44*, *80*
tiltpsi *(*aiapsi*)*, 5
tiltpsiC *(*aiapsi*)*, 5
turbwf, 61

vortexfit, *3*, *4*, *11*, *18*, *23*, 62, *65*
vortexfit(), *59*

wf3d *(*wf3d.pupil*)*, 64
wf3d.pupil, 64
wf_net, *3*, *4*, *17*, *18*, *23*, *42*, *44*, *46*, *47*, 65, *80*
wf_net(), *49*, *59*
wftophase *(*FFTUtilities*)*, 19
wrap *(*rmap*)*, 53

zapm, *23*, 67, *83*
zapm(), *26*, *56*, *71*
zapm_128, 69
zapm_128(), *71*
zapm_cart, *66*
zapm_direct, 71
zapm_direct(), *56*
zapm_iso, *23*, 71
zapm_iso(), *26*, *56*, *75*
zapm_iso_128, 73
zapm_iso_128(), *75*
zapm_iso_direct, 75
zapm_iso_direct(), *56*
zapmC, *66*
zconic, *57*, 76
Zernike, *48*, *59*, *62*, *77*, 77, *83*
zernike_options, 79
zernike_options(), *62*
zlist, 81
zlist.fr, *78*, *83*
zlist.fr *(*zlist*)*, 81
zmoments, 82
zmult, *78*
zmult *(*zlist*)*, 81
zpm, *5*, *23*, *46*, *47*, *66*, *78*, *80*, 83
zpm(), *28*
zpm_cart, *22*, *23*, *46*, *47*, *66*, 84
zpm_cart(), *28*, *37*
zpmC *(*zpm*)*, 83
zpmCP, *80*