

Package ‘zernike’

June 13, 2021

Title Zernike Polynomials

Version 3.7.0

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

Depends R (>= 3.6.3)

Suggests rgl, robustbase, clue, data.table, dplyr, pixmap, mvtnorm

Imports Rcpp (>= 0.12.0), RcppParallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Description Routines for Manipulation of Zernike polynomials and
Interferogram fringe analysis

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

License MIT and GPL

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

R topics documented:

| | |
|------------------------|----|
| addfit | 3 |
| aiapsi | 3 |
| astig.bath | 5 |
| brcutpuw | 6 |
| circle.hough | 7 |
| circle.pars | 8 |
| col3d | 9 |
| convolve2d | 10 |
| crop | 11 |
| ffit | 11 |
| FFTUtilities | 13 |
| fitzernikes | 14 |
| foucogram | 15 |
| gblur | 16 |
| gpcapsi | 17 |

| | |
|---------------------|----|
| gradzpm_cart | 18 |
| gray256 | 19 |
| hypot | 19 |
| idiffpuw | 20 |
| id_dxy_uw | 21 |
| id_uw | 22 |
| load.images | 23 |
| lpspi | 24 |
| makezlist.iso | 25 |
| mpinv | 25 |
| norm_zpm | 26 |
| pcapsi | 27 |
| pick.sidelobe | 28 |
| plot.cmat | 29 |
| plot.pupil | 30 |
| plotn | 31 |
| plotxs | 32 |
| psfit_options | 33 |
| psfit | 34 |
| pupil | 36 |
| pupil.pars | 37 |
| pupil.rhotheta | 38 |
| pupilrms | 39 |
| PVr | 40 |
| qpuw | 41 |
| q_uw | 42 |
| readjpeg | 43 |
| rescale | 43 |
| rmap | 44 |
| rygcb | 45 |
| sconic | 46 |
| separate.wf | 47 |
| startest | 48 |
| summary.wf_fitted | 49 |
| synth.interferogram | 50 |
| turbwf | 51 |
| vortexfit | 52 |
| wf3d.pupil | 54 |
| wf_net | 55 |
| zconic | 56 |
| Zernike | 57 |
| zlist | 58 |
| zmoments | 59 |
| zpm | 60 |
| zpm_cart | 61 |

| | |
|--------|--|
| addfit | <i>Add zernike coefficients to a matrix.</i> |
|--------|--|

Description

Add zernike coefficients to a matrix.

Usage

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

Arguments

| | |
|-------|--|
| ... | One or more fits as from psifit, etc. |
| th | Rotation angles, in degrees |
| zcm | The matrix to be added to (defaults to NULL) |
| theta | The vector of rotation angles to be added to |

Author(s)

M.L. Peck

| | |
|--------|---|
| aiapsi | <i>Iterative algorithms for PSI with unknown phase shifts</i> |
|--------|---|

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 <i>matrix</i> 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 | <i>Zernike coefficients for astigmatism due to Bath astigmatism.</i> |
|------------|--|

Description

Calculates Bath astigmatism coefficients with optional rotation of phi degrees.

Usage

```
astig.bath(D, rc, s, lambda = 1e-06, phi = 0)
```

Arguments

| | |
|--------|---|
| D | Diameter |
| rc | Radius of curvature |
| s | separation of reference and test beams |
| lambda | Wavelength – defaults to 1 nm. |
| phi | angle of image horizontal relative to interferometer axis, in degrees |

Details

D, rc, s, and lambda must have the same units.

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.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.pars()`, `pupil.pars()`

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.pars

Pupil parameters

Description

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

Usage

```
circle.pars(im, fw=2, qt=0.995, excl=5,
  plots=TRUE, details=FALSE)
```

Arguments

| | |
|------------|--|
| im | A matrix containing an image of a circular disk |
| fw | 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? |
| obstructed | Logical: is there a central obstruction? |

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 `fw` 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.pars`. For example [psifit](#), [fftfit](#), [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](#)

convolve2d

2D convolution

Description

General 2D convolution using FFTs

Usage

```
convolve2d(im, kern)
```

Arguments

| | |
|------|--------------------------------|
| im | A matrix representing an image |
| kern | the convolution kernel |

Value

The filtered matrix im.

Author(s)

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

See Also

[gblur](#). Called by [circle.pars](#).

| | |
|------|----------------------|
| crop | <i>Crop an array</i> |
|------|----------------------|

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)
```

Arguments

| | |
|------|---|
| img | Array to be cropped. |
| cp | A list describing the pupil boundary. |
| npad | Amount of padding to leave around the edge. |

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>

| | |
|--------|---|
| fftfit | <i>Fourier transform interferogram analysis</i> |
|--------|---|

Description

High level routines for FFT analysis of interferograms.

Usage

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

[wf_net](#), [pupil.pars](#), [pick.sidelobe](#).

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.

fftshift shuffles the quadrants of a matrix around to put the DC element (1,1) in the center of the transformed matrix, with spatial frequencies increasing to the right and up.

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](#).

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, phi = 0, maxorder = 14, uselm = FALSE)
```

Arguments

| | |
|----------|--|
| wf | A vector of wavefront values |
| rho | A vector of radial coordinates. |
| theta | A vector of angular coordinates, in radians. |
| phi | Orientation of the image, in degrees |
| maxorder | Maximum Zernike polynomial order |
| uselm | Boolean: use <code>lm()</code> for least squares fit |

Details

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

Value

The model fit as returned by [lm](#), or the coefficients of the least squares fit if uselm is FALSE.

Note

The model fit is of the form $wf \sim Z_0 + Z_1 + Z_2 + \dots$. With the standard ordering of Zernikes Z_0 is the piston term, Z_1 and Z_2 are x and y tilts, Z_3 is defocus, etc.

Author(s)

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

See Also

[zpm](#), [psifit](#), [fftfit](#).

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 <code>pupil</code> 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`*Gaussian blur*

Description

Blur an image by `fw` pixels

Usage

```
gblur(X, fw=0, details=FALSE)
```

Arguments

| | |
|----------------------|---|
| <code>X</code> | A matrix representing an image |
| <code>fw</code> | Width of the Gaussian convolution kernel, in pixels |
| <code>details</code> | Return convolution kernel? |

Details

`fw` is the standard deviation of the Gaussian.

Value

The filtered matrix `X`.

Note

the `details` option is mostly for debugging purposes and may go away.

Author(s)

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

See Also[convolve2d](#)

| | |
|---------|---|
| gpcapsi | <i>Generalized Principal components algorithm for phase shifting interferometry</i> |
|---------|---|

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_cart | <i>Calculate Zernike polynomial values and Cartesian gradients in ISO/ANSI sequence for a set of Cartesian coordinates.</i> |
|--------------|---|

Description

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

Usage

```
gradzpm_cart(x, y, maxorder = 12L, unit_variance = FALSE, 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

polynomial values and their directional derivatives in Cartesian coordinates. These are known to be both efficient and numerically stable.

$m = -n, -(n-2), \dots, (n-2), n$, 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_cart(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 | <i>Phase unwrapping by Integrating DIFFerences</i> |
|----------|--|

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 2π 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 2π 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\(\)](#)

| | |
|-------------|--------------------|
| load.images | <i>Read images</i> |
|-------------|--------------------|

Description

Loads image files in jpeg, tiff 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 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

| | |
|---------------------|--|
| <code>images</code> | An array containing the interferogram images |
| <code>phases</code> | A vector of phase shifts |
| <code>wt</code> | 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:

| | |
|------------------|------------------------------------|
| <code>phi</code> | Estimated wrapped wavefront phase. |
| <code>mod</code> | Estimated modulation |

Author(s)

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

See Also

[psifit](#)

| | |
|---------------|--|
| makezlist.iso | <i>Construct list of ZP indexes in ISO/ANSI sequence with sine terms first</i> |
|---------------|--|

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)
```

| | |
|-------|--|
| mpinv | <i>Moore-Penrose generalized inverse</i> |
|-------|--|

Description

Computes the Moore-Penrose generalized inverse of a matrix using singular value decomposition.

Usage

```
mpinv(X)
```

Arguments

| | |
|---|----------|
| X | A matrix |
|---|----------|

Value

Matrix containing the generalized inverse. If X is an n x m matrix the return will have dimension m x n.

Note

The threshold for determining if a matrix is rank deficient is `eps <- .Machine$double.eps * max(dim(X)) * S$d[1]`

Author(s)

M. L. Peck

Examples

```
X <- matrix(rnorm(18), 6, 3) ## this should be full rank almost always
mpinv(X) %*% X

X <- matrix(1:18, 6, 3) ## this is not
mpinv(X) %*% X
```

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 = 12L)
```

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_cart()` 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 <i>matrix</i> of interferogram values |
| bgsub | Boolean - subtract the pixelwise mean as background estimate? |
| group_diag | 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. |
| 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](#),

| | |
|----------------------------|---|
| <code>pick.sidelobe</code> | <i>Select an interferogram sidelobe in the Fourier domain</i> |
|----------------------------|---|

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

| | |
|------------------------|---|
| <code>imagedata</code> | A matrix containing an interferogram image |
| <code>logm</code> | Logical: pass <code>fn="logMod"</code> to plot.cmat ? |
| <code>gamma</code> | gamma value for display |

Details

Uses the basic graphics utility [locator](#).

Value

A list with the following components:

| | |
|---------------------|--|
| <code>sl</code> | The coordinates $c(x, y)$ of the selected sidelobe |
| <code>filter</code> | 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 | <i>Plot a complex matrix</i> |
|-----------|------------------------------|

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 $\log(1+\text{Mod}(X))$, "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

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

| | |
|--------------------------|--|
| <code>...</code> | List of wavefront estimates as returned by wf_net . |
| <code>labels</code> | Labels to identify the wavefronts. |
| <code>addContours</code> | Boolean to add contours to top row plots |
| <code>wftype</code> | If the inputs are from <code>wf_net</code> , one of "net", "smooth", "residual". |
| <code>col</code> | Color palette for top row of plot |
| <code>qt</code> | 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 $\pi \cdot (N-1)/N$. 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.

psfit_options

*Options for PSI and FFT based fitting routines***Description**

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

Usage

```
psfit_options(...)
```

Arguments

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

Details

Calling psfit_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.

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 ([zpmC](#)) and threaded ([zpmCP](#)) cases. Speed improvements, if any, may vary.

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>

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 = psifit_options())
```

Arguments

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

pc2 pca with `group_diag = "u"`

gpc my generalized PC algorithm in [gpcapsi](#)

gpcthentilt first [gpcapsi](#) the [tiltpsi](#)

tilt [tiltpsi](#)

Value

A list with the following components

| | |
|--------------------------|---|
| <code>phi</code> | wrapped phase estimate |
| <code>mod</code> | modulation estimate |
| <code>phases</code> | phase shifts |
| <code>cp</code> | the interferogram boundary |
| <code>wf.net</code> | net, unfiltered wavefront (see wf_net) |
| <code>wf.smooth</code> | Zernike fit wavefront |
| <code>wf.residual</code> | the difference |
| <code>fit</code> | Coefficients of Zernike fit to wavefront |
| <code>zcoef.net</code> | Net Zernike coefficients |
| <code>extras</code> | 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 <- psfit_options()
zopt$sa_target <- sa.t
zopt$ptol <- 0.01
tfit <- psfit(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

```

pupil(zcoef=NULL, zlist=makezlist(), phi=0, piston=0,
      nrow=512, ncol=nrow, cp=list(xc=256,yc=256,rx=255,ry=255,obstruct=0))
pupil.arb(zcoef=NULL, zlist=makezlist(), phi=0, piston=0,
          nrow=512, ncol=nrow, cp=list(xc=256,yc=256,rx=255,ry=255,obstruct=0))

```

Arguments

| | |
|--------|--|
| zcoef | A vector of Zernike coefficients |
| zlist | List of indexes the same length as zcoef |
| 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 Zernikes.

[pupil](#) requires a complete set of Zernikes as returned by [makezlist](#).

Value

A matrix of size `nrow` x `ncol`. The matrix is assigned to the class "pupil". NA's 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

[Zernike](#), [makezlist](#), [pupilrms](#), [pupilpv](#), [strehlratio](#), [pupil.pars](#), [circle.pars](#).

Examples

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

`pupil.pars`

Pupil parameters

Description

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

Usage

```
pupil.pars(im = NULL, obstructed = FALSE)
```

Arguments

| | |
|-------------------------|--|
| <code>im</code> | A matrix containing an interferogram image |
| <code>obstructed</code> | Logical: is there a central obstruction? |

Details

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

Value

A list with the following components:

| | |
|-----------------------|---|
| <code>xc</code> | X coordinate of the center of the pupil |
| <code>yc</code> | Y coordinate of the center of the pupil |
| <code>rx</code> | Horizontal radius of the pupil |
| <code>ry</code> | Vertical radius of the pupil |
| <code>obstruct</code> | Obstruction fraction |

Note

`pupil.pars` 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 pupil, or an annular one with small obstruction. We allow `rx != ry` for imaging sensors with non-square aspect ratios.

Author(s)

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

See Also

Many routines require the pupil parameters in the form returned by `pupil.pars`. For example [psifit](#), [fftfit](#), [pupil](#), etc.

| | |
|-----------------------------|--------------------------|
| <code>pupil.rhotheta</code> | <i>Polar coordinates</i> |
|-----------------------------|--------------------------|

Description

Calculate matrixes of polar coordinates for [pupil](#)'s.

Usage

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

Arguments

| | |
|-------------------|---|
| <code>nrow</code> | Number of rows in interferogram images |
| <code>ncol</code> | Number of columns in interferogram images |
| <code>cp</code> | 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. strehlratio 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 2π 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()`

| | |
|----------|---------------------------------|
| readjpeg | <i>Read a jpeg or tiff file</i> |
|----------|---------------------------------|

Description

Reads a jpeg or tiff file and combines the channels to produce a monochrome image in a matrix.

Usage

```
readjpeg(filename, channels)
readtiff(filename, channels)
```

Arguments

| | |
|----------|---|
| filename | File name |
| channels | A vector of length 3 with the channel weights |

Details

Values in channels should be non-negative, but need not add to one.

Value

A double precision matrix with the image data.

Note

The matrix must have rows reversed and transposed to display properly with `image()`.

Author(s)

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

| | |
|---------|--------------------------|
| rescale | <i>Rescale an image.</i> |
|---------|--------------------------|

Description

Rescale a matrix containing a bitmapped image using bilinear interpolation.

Usage

```
rescale(im, scale)
```

Arguments

| | |
|-------|---------------------------|
| im | 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 | <i>A better rainbow.</i> |
|-------|--------------------------|

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))
```

| | |
|--------|---------------|
| sconic | <i>Sconic</i> |
|--------|---------------|

Description

twice the radial height difference between a sphere and conic surface

Usage

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

Arguments

| | |
|--------|--------------------------------|
| D | Diameter (mm) |
| rc | Radius of curvature (mm) |
| b | conic constant |
| lambda | source or test wavelength (nm) |
| nmax | maximum polynomial order |

Value

Zernike polynomial coefficients

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)
```

| | |
|-------------|----------------------------|
| separate.wf | <i>Separate wavefronts</i> |
|-------------|----------------------------|

Description

Separate “polished in” from “instrumental” aberrations if possible

Usage

```
separate.wf(zcm, theta, maxorder = 14)
```

Arguments

| | |
|----------|---|
| zcm | Matrix of observed Zernike coefficients |
| theta | Vector of rotation angles (in radians) |
| maxorder | Maximum Zernike order to extract |

Value

zcb Table of extracted coefficients and standard errors

Author(s)

M.L. Peck

startest

Star test simulator

Description

Simulates a star test.

Usage

```
startest(wf=NULL, zcoef=NULL, zlist=makezlist(), phi=0,
lambda = 1, defocus=5,
nrow = 255, ncol = nrow,
cp = list(xc=128,yc=128,rx=127,ry=127,obstruct=0),
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 |
| zlist | Indexes of Zernike coefficients |
| phi | Angle to rotate wavefront |
| lambda | Wavelength, in same units as coefficients |
| defocus | Amount of defocus in waves |
| nrow | # rows in pupil matrix |
| ncol | # columns in pupil matrix |
| 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](#).

Examples

```
# a random, but probably almost diffraction limited, wavefront

temp <- startest(zcoef=rnorm(length(makezlist())$n), mean=0, sd=0.01), zlist=makezlist(), displaywf=TRUE)
```

| | |
|-------------------|--------------------------------------|
| summary.wf_fitted | <i>Methods for class "wf_fitted"</i> |
|-------------------|--------------------------------------|

Description

Summary, print, and plot methods for the returned list of values from [psifit\(\)](#), [fftfit\(\)](#), or [vortexfit\(\)](#)

Usage

```
## S3 method for class 'wf_fitted'
summary(wffit, digits = 3)
```

Arguments

| | |
|--------|--|
| wffit | the return values from one of the fringe analysis routines |
| digits | number of digits to display in print and summary methods |
| ... | values passed to plot.pupil() |

Value

print method returns data frame with Zernike coefficients

| | |
|---------------------|--------------------------------|
| synth.interferogram | <i>Synthetic interferogram</i> |
|---------------------|--------------------------------|

Description

Compute and display a synthetic interferogram.

Usage

```
synth.interferogram(wf = NULL, zcoef = NULL, zlist = 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 |
| zlist | A list of Zernike indexes |
| 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 should be non-null, but not both. If zcoef is specified zlist 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, zlist=zlist.fr)

X11()

# show again with some tilt

iwf <- synth.interferogram(zcoef=zcoef, zlist=zlist.fr, 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(
  imagedata,
  cp = NULL,
  filter = NULL,
  fw.o = 10,
  options = psfit_options()
)
```

Arguments

| | |
|-----------|---|
| imagedata | 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 <code>psfit_options()</code> . |

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.

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 <- psfit_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)
```

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)
```

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? |

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 described at <http://rgl.neoscientists.org/about.shtml>, and 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 smoothing

Description

Calculate net and smoothed wavefronts from a raw wavefront containing low order nuisance aberrations.

Usage

```
wf_net(wf.raw, cp, options)
```

Arguments

| | |
|---------|--|
| wf.raw | Raw wavefront to be processed |
| cp | a list describing the pupil boundary |
| options | a list of options. See psfit_options |

Details

Called by [psifit](#)

Value

A list with the following components:

| | |
|-------------|---|
| 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 |

Author(s)

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

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 1 nm. |
| nmax | Maximum radial polynomial order |

Details

D, rc, and lambda must have the same units.

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](#)

Examples

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

ZernikeZernike 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, $0 \leq \rho \leq 1$ |
| 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)
```

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 | <i>Matrixes of Zernike polynomials</i> |
|-----|--|

Description

Create a matrix of Zernike polynomial values, or their derivatives or gradient.

Usage

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

zpmC is the C++ routine that does the computations for zpm. No column names are returned.

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](#)

| | |
|----------|---|
| zpm_cart | <i>Calculate Zernike polynomial values in ISO/ANSI sequence for a set of Cartesian coordinates.</i> |
|----------|---|

Description

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

Usage

```
zpm_cart(x, y, maxorder = 12L, 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_cart\(\)](#) 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.

Index

- * **Graphics**
 - plotn, [31](#)
 - plotxs, [32](#)
 - pupil, [36](#)
 - rygcb, [45](#)
 - wf_net, [55](#)
- * **IO**
 - load.images, [23](#)
 - readjpeg, [43](#)
- * **Models**
 - psifit, [34](#)
- * **Utilities**
 - crop, [11](#)
- * **Utility**
 - addfit, [3](#)
 - hypot, [19](#)
 - separate.wf, [47](#)
 - zmoments, [59](#)
- * **arith**
 - rescale, [43](#)
 - sconic, [46](#)
- * **array**
 - mpinv, [25](#)
 - rescale, [43](#)
- * **file**
 - load.images, [23](#)
 - readjpeg, [43](#)
- * **graphics**
 - col3d, [9](#)
 - foucogram, [15](#)
 - gray256, [19](#)
 - pick.sidelobe, [28](#)
 - plot.cmat, [29](#)
 - startest, [48](#)
 - synth.interferogram, [50](#)
 - wf3d.pupil, [54](#)
- * **hplot**
 - col3d, [9](#)
 - foucogram, [15](#)
 - startest, [48](#)
 - synth.interferogram, [50](#)
 - wf3d.pupil, [54](#)
- * **list**
 - psfit_options, [33](#)
- * **mathematics**
 - aiapsi, [3](#)
 - astig.bath, [5](#)
 - convolve2d, [10](#)
 - fftfrit, [11](#)
 - gblur, [16](#)
 - idiffpuw, [20](#)
 - lspsi, [24](#)
 - pcapsi, [27](#)
 - qpuw, [41](#)
 - rmap, [44](#)
 - turbwf, [51](#)
 - zconic, [56](#)
 - Zernike, [57](#)
 - zlist, [58](#)
 - zpm, [60](#)
- * **math**
 - mpinv, [25](#)
- * **misc**
 - brcutpuw, [6](#)
- * **models**
 - gpcapsi, [17](#)
- * **optimization**
 - gpcapsi, [17](#)
- * **optimize**
 - brcutpuw, [6](#)
- * **smooth**
 - wf_net, [55](#)
- * **statistics**
 - fitzernikes, [14](#)
 - pupilrms, [39](#)
- * **utilities**
 - circle.pars, [8](#)
 - FFTUtilities, [13](#)

- `pick.sidelobe`, 28
 - `plot.pupil`, 30
 - `pupil.pars`, 37
 - `pupil.rhotheta`, 38
- `.up2 (FFTUtilities)`, 13
- `addfit`, 3
- `aiapsi`, 3, 35
- `aiapsiC (aiapsi)`, 3
- `astig.bath`, 5
-
- `brcutpuw`, 6, 21, 41, 45
- `brcutpuw()`, 21, 22
-
- `circle.hough`, 7
- `circle.pars`, 8, 10–12, 37
- `circle.pars()`, 8
- `col3d`, 9
- `complex`, 29
- `convolve2d`, 10, 16
- `crop`, 11
-
- `drzernike (Zernike)`, 57
-
- `fftfit`, 9, 11, 14, 28, 29, 38, 59
- `fftfit()`, 49, 53
- `fftshift (FFTUtilities)`, 13
- `FFTUtilities`, 13
- `fitzernikes`, 12, 14, 55, 61
- `foucogram`, 15
-
- `gblur`, 10, 16
- `gpcapsi`, 17, 35
- `gpcapsiC (gpcapsi)`, 17
- `gradzpm_cart`, 18
- `gradzpm_cart()`, 26, 61
- `gray256`, 19
- `grey256`, 46
- `grey256 (gray256)`, 19
-
- `hkpsi (aiapsi)`, 3
- `https://doi.org/10.1364/JOSAA.18.001862`, 53
- `https://doi.org/10.1364/OPEX.13.008097`, 53
- `hypot`, 19
-
- `id_dxy_uw`, 21
- `id_uw`, 22
- `idiffpuw`, 6, 20, 41
-
- `idiffpuw()`, 21, 22, 42
- `image.default`, 29, 30
-
- `lm`, 14
- `load.images`, 23, 27, 44
- `load.pgm (load.images)`, 23
- `locator`, 28, 38
- `lspsi`, 24, 35
- `lspsiC (lspsi)`, 24
-
- `makezlist`, 37, 51, 58, 60, 61
- `makezlist (zlist)`, 58
- `makezlist.iso`, 25
- `mpinv`, 25
-
- `norm_zpm`, 26
-
- `padmatrix (FFTUtilities)`, 13
- `pcapsi`, 18, 27, 35
- `pick.sidelobe`, 12, 28, 29
- `plot.cmat`, 28, 29
- `plot.pupil`, 10, 30, 31, 32, 37, 54
- `plot.pupil()`, 50
- `plotn`, 31
- `plotxs`, 32
- `psfit_options`, 12, 33, 55
- `psfit_options()`, 53
- `psifit`, 5, 9, 14, 17, 18, 24, 28, 33, 34, 38, 55, 59
- `psifit()`, 49
- `pupil`, 9, 12, 15, 16, 20, 30, 36, 38, 39, 41, 49–52, 54, 55, 58, 59
- `pupil.pars`, 12, 29, 30, 32, 34, 37, 37, 38, 50
- `pupil.pars()`, 8
- `pupil.rhotheta`, 38
- `pupilpv`, 30, 37, 58
- `pupilpv (pupilrms)`, 39
- `pupilrms`, 30, 37, 39, 58
- `PVr`, 40
-
- `q_uw`, 42
- `qpuw`, 6, 41
- `qpuw()`, 42
-
- `readjpeg`, 43
- `readtiff (readjpeg)`, 43
- `rescale`, 43
- `rmap`, 21, 44
- `rygcb`, 34, 45
- `rygcbm (rygcb)`, 45

`rzernike` (Zernike), 57

`sconic`, 46

`separate.wf`, 47

`startest`, 14, 48

`strehlratio`, 30, 37, 58

`strehlratio` (pupilrms), 39

`submatrix` (FFTUtilities), 13

`summary.pupil`, 37, 39, 40

`summary.pupil` (plot.pupil), 30

`summary.wf_fitted`, 49

`synth.interferogram`, 50

`tiltpsi`, 34, 35

`tiltpsi` (aiapsi), 3

`tiltpsiC` (aiapsi), 3

`turbwf`, 51

`vortexfit`, 52

`vortexfit()`, 49

`wf3d` (wf3d.pupil), 54

`wf3d.pupil`, 54

`wf_net`, 12, 31, 33, 35, 55

`wf_net()`, 40

`wftophase` (FFTUtilities), 13

`wrap` (rmap), 44

`zconic`, 47, 56

`Zernike`, 37, 39, 49, 52, 56, 57, 61

`zlist`, 58

`zlist.fr`, 58, 61

`zlist.fr` (zlist), 58

`zmoments`, 59

`zmult`, 57, 58

`zmult` (zlist), 58

`zpm`, 4, 14, 58, 60

`zpm()`, 19

`zpm_cart`, 61

`zpm_cart()`, 19, 26

`zpmC`, 34

`zpmC` (zpm), 60

`zpmCP`, 34