Package: dualtrees (via r-universe)

November 3, 2024

Title Decimated and Undecimated 2D Complex Dual-Tree Wavelet Transform

Version 0.1.5

Maintainer Sebastian Buschow <s6sebusc@uni-bonn.de>

Description An implementation of the decimated two-dimensional complex dual-tree wavelet transform as described in Kingsbury (1999) <doi:10.1098/rsta.1999.0447> and Selesnick et al. (2005) <doi:10.1109/MSP.2005.1550194>. Also includes the undecimated version and spectral bias correction described in Nelson et al. (2018) <doi:10.1007/s11222-017-9784-0>. The code is partly based on the 'dtcwt' Python library.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Repository https://s6sebusc.r-universe.dev

RemoteUrl https://github.com/s6sebusc/dualtrees

RemoteRef HEAD

RemoteSha a3f830ba9b9c0a14c2518442f9ab1d71a234398f

Contents

A *Bias correction matrices*

Description

Matrices needed in order to eliminate the effect of "spectral leakage" for the local dtcwt-spectra.

Usage

```
A_b_bp

A_b
```

Format

A list with entries N1024, N512, N256, N128, N64, N32, each containing the bias correction matrix of appropriate size.

Details

As descibed in Nelson et al. (2018), the squared coefficients of the undecimated dtcwt (the local wavelet spectrum) can be used to obtain an estimate of local correlations in space. This estimator has a bias which mostly consists of an over-emphasis on the largest scales. It can be removed by multiplying each local spectrum with a matrix which depends only on the choice of wavelet and the dimensions of the field.

Source

Calculated by brute force.

References

Nelson, J. D. B., A. J. Gibberd, C. Nafornita, and N. Kingsbury (2018) <doi:10.1007/s11222-017-9784-0>.

Examples

```
image( A_b_bp$N512 )
```

blossom *Two meteorologists in front of cherry blossoms*

Description

A photograph of two meteorologists in front of the famous Japanese cherry trees in Bonn.

Usage

```
blossom
```

Format

A 256x256 matrix of gray-scale values

Source

Armin Blanke, used with permission.

Examples

```
image(blossom, col=gray.colors(128,0,1))
```

boundaries *Various boundary conditions for the 2D wavelet transform.*

Description

Extend a matrix to the desired size.

Usage

```
pad(x, N, Ny = N, value = min(x, na.rm = TRUE))

put_in_mirror(x, N, Ny = N)

period_bc(x, N, Ny = N)
```

Arguments

x	a real matrix
N	the number of rows of the desired output
Ny	the number of columns of the desired output, defaults to N
value	the value with which the picture is padded by pad

Details

pad pads the fields with a constant value on all sides, be careful what you pick here. put_in_mirror reflects the input at all edges (with repeated end samples), period_bc simply repeats the input periodically. In any case, you can retrieve the initial area via bc$res[bc$px, bc$py].

Value

a list containing the extended matrix ($res) and the positions of the original matrix within the extended one ($px and $py).

Note

N and Ny must be at least as big as the input.

Examples

```
bc <- put_in_mirror( blossom, N=300 )
plot( bc )
print( range( bc$res[ bc$px, bc$py ] - blossom ) )
```

---

cen2uv *centre to vector*

Description

transforms the angle and radius component of the spectral centre into vector components for visualization

Usage

```
cen2uv(cen)
```

Arguments

cen	an nx x ny x 3 array, the output of dt2cen

Details

The resulting vector field represents the degree of anisotropy and the dominant direction, averaged over all scales.

Value

an nx x ny x 2 array, the two matrices representing the u- and v-component of the vector field.

See Also

dt2cen, uvplot

Examples

```
dt <- fld2dt(blossom)
ce <- dt2cen(dt)
uv <- cen2uv(ce)
uvplot( uv, z=blossom )
```

---

cen_xy *xy <-> cen*

Description

Translate the centre of mass back and forth between polar and cartesian coordinates.

Usage

```
cen2xy(cen)

xy2cen(xy)
```

Arguments

cen	the centre of mass of a wavelet spectrum (rho, phi, z), output of dt2cen
xy	the centre of mass in cartesian coordinates (x, y, z), output of cen2xy

Details

These functions allow you to translate back and forth between the two coordinate systems. dt2cen represents the sepctrum's centre in cylinder coordinates because that is more intuitive than the x-y-z position within the hexagonal geometry. If you want to compare two spectra, it makes more sense to consider their distance in terms of x1-x2, y1-y2 since the difference in angle is only meaningful for reasonably large radii.

Note

cen2xy is not the same thing as cen2uv !

See Also

dt2cen, cen2uv

dt2cen *centre of the DT-spectrum*

Description

calculate the centre of mass of the local spectra in hexagonal geometry

Usage

```
dt2cen(dt, mask = NULL)
```

Arguments

dt	a J x nx x ny x 6 array of spectral energies, the output of fld2dt
mask	a nx x ny array of logical values

Details

Each of the J x 6 spectral values is assigned a coordinate in 3D space with x(d,j)=cos(60*(d-1)), y(d,j)=sin(60*(d-1)), z(d,j)=j, where j denotes the scale and d the direction. Then the centre of mass in this space is calculated, the spectral values being the masses at each vertex. The x- and y-cooridnate are then transformed into a radius rho=sqrt(x^2+y^2) and an angle phi=15+0.5*atan2(y,x). rho measures the degree of anisotropy at each pixel, phi the orientation of edges in the image, and the third coordinate, z, the central scale. If a mask is provided, values where mask==TRUE are set to NA.

Value

a nx x ny x 3 array where the third dimension denotes degree of anisotropy, angle and central scale, respectively.

Note

Since the centre of mass is not defined for negative mass, any values below zero are removed at this point.

Examples

```
dt <- fld2dt(blossom)
ce <- dt2cen(dt)
image( ce[,,3], col=gray.colors(32, 0, 1) )
```

dtmean *spatial mean spectrum*

Description

average the output of fld2dt or dtcwt over space

Usage

```
dtmean(x)
```

Arguments

x	either a J x nx x ny x 6 array of energies (output of dtcwt) or a list of complex wavelet coefficients (the output of `dtcwt(...,dec=FALSE)`)

Value

a J x 6 matrix of spatially averaged energies

Note

In the undecimated case, the coefficients are not averaged but summed up and then scaled by the area of the first level. This yields a comparable scale as the undecimated case.

dualtree-transform *The 2D forward and inverse dualtree complex wavelet transform*

Description

These functions perform the dualtree complex wavelet analysis and synthesis, either with or without decimation.

Usage

```
dtcwt(fld, fb1 = near_sym_b, fb2 = qshift_b, J = NULL, dec = TRUE,
  verbose = FALSE)

idtcwt(pyr, fb1 = near_sym_b, fb2 = qshift_b, verbose = TRUE)
```

Arguments

fld	real matrix representing the field to be transformed
fb1	A list of filter coefficients for the first level. Currently only near_sym_b and near_sym_b_bp are implemented
fb2	A list of filter coefficients for all following levels. Currently only qshift_b and qshift_b_bp are implemented
J	number of levels for the decomposition. Defaults to log2(min(Nx,Ny)) in the decimated case and log2(min(Nx,Ny)) - 3 otherwise
dec	whether or not the decimated transform is desired
verbose	if TRUE, the function tells you which level it is working on
pyr	a list containing arrays of complex coefficients for each level of the decomposition, produced by dtcwt(..., dec=TRUE)

Details

This is the 2D complex dualtree wavelet transform as described by Selesnick et al. (2005). It consists of four discerete wavelet transform trees, generated from two filter banks a and b by applying one set of filters to the rows and another (or the same) one to the columns. The 12 resulting coefficients are combined into six complex values representing six directions (15°, 45°, 75°, 105°, 135°, 165°). In the decimated case (dec=TRUE), each convolution is followed by a downsampling by two, meaining that the size of the six coefficient fields is cut in half at each level. The decimated transform can be reversed to recover the original image. For the near_sym_b and qshift_b filter banks, this reconstrcution should be basically perfect. In the case of the the b_bp filters, non-negligible artifacts appear near +-45° edges.

Value

if dec=TRUE a list of complex coefficient fields, otherwise a complex J x Nx x Ny x 6 array.

Note

At present, the inverse transform only works if the input image had dimensions 2^N x 2^N. You can use boundaries to achieve that.

Author(s)

Nick Kingsbury (canonical MATLAB implementation), Rich Wareham (open source Python implementation, https://github.com/rjw57/dtcwt), Sebastian Buschow (R port).

References

Kingsbury, Nick (1999) <doi:10.1098/rsta.1999.0447>. Selesnick, I.W., R.G. Baraniuk, and N.C. Kingsbury (2005) <doi:10.1109/MSP.2005.1550194>

See Also

filterbanks, fld2dt

Examples

```
oldpar <- par( no.readonly=TRUE )
# forward transform
dt <- dtcwt( blossom )
par( mfrow=c(2,3), mar=rep(2,4) )
for( j in 1:6 ){
    image( blossom, col=grey.colors(32,0,1) )
    contour( Mod( dt[[3]][ ,,j ] )**2, add=TRUE, col="green" )
}
par( oldpar )

# exmaple for the inverse transform
blossom_i <- idtcwt( dt )
image( blossom - blossom_i )

# example for a non-square case
boy <- blossom[50:120, 50:150]
bc  <- put_in_mirror(boy, 128)
dt  <- dtcwt(bc$res)
idt <- idtcwt(dt)[ bc$px, bc$py ]
```

`filterbanks` *filterbanks for the dtcwt*

Description

Some of the filters implemented in the python package dtcwt.

Usage

```
qshift_b

qshift_b_bp

near_sym_b

near_sym_b_bp
```

Format

A list of high- and low-pass filters for analysis and synthesis

Details

The near-sym filterbanks are biorthogonal wavelets used for the first level, they have 13 and 19 taps. The qshift filterbanks, each with 14 taps, are suitable for all higher levels of the dtcwt. The a- and b-filters form an approximate Hilbert-pair. The naming convention follows the python-package:

h:	analysis
g:	synthesis
0:	low-pass
1:	high-pass
a,b:	shifted filters

The b_bp-versions of the filterbanks contain a second high-pass for the diagonal directions, denoted by 2. They allow for better directional selectivity but prohibit perfect reconstruction.

Source

'dtcwt' python package (https://github.com/rjw57/dtcwt)

References

Selesnick, I.W., R.G. Baraniuk, and N.C. Kingsbury (2005) <doi:10.1109/MSP.2005.1550194>

Kingsbury, N. (2006) <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7071567&isnumber=7065146>

`fld2dt` *transform a field into an array of spectral energies*

Description

Handles the transformation itself, boundary conditions and bias correction and returns the unbiased local wavelet spectrum at each grid-point.

Usage

```
fld2dt(fld, Nx = NULL, Ny = NULL, J = NULL, correct = TRUE,
  rsm = 0, verbose = FALSE, boundaries = "pad",
  fb1 = near_sym_b_bp, fb2 = qshift_b_bp)
```

Arguments

`fld`	a real matrix
`Nx`	size to which the field is padded in x-direction
`Ny`	size to which the field is padded in y-direction
`J`	number of levels for the decomposition
`correct`	logical, whether or not to apply the bias correction
`rsm`	number of pixels to be linearly smoothed along each edge before applying the boundary conditions (see `smooth_borders`).
`verbose`	whether or not you want the transform to talk to you
`boundaries`	how to handle the boundary conditions, either "pad", "mirror" or "periodic"
`fb1`	filter bank for level 1
`fb2`	filter bank for all further levels

Details

The input is blown up to Nx x Ny and transformed by dtcwt(..., dec=FALSE). Then the original domain is cut out, the coefficients are squared and the bias is corrected (for details on the bias, see A).

Value

an array of size J x nx x ny x 6 where dim(fld)=c(nx,ny)

References

Nelson, J. D. B., A. J. Gibberd, C. Nafornita, and N. Kingsbury (2018) <doi:10.1007/s11222-017-9784-0>

See Also

A

Examples

```
oldpar <- par( no.readonly=TRUE )
dt <- fld2dt( blossom )
par( mfrow=c(2,2), mar=rep(2,4) )
for( j in 1:4 ){
    image( blossom, col=gray.colors(128, 0,1), xaxt="n", yaxt="n" )
    for(d in  1:6) contour( dt[j,,,d], levels=quantile(dt[,,,], .995),
                           col=d+1, add=TRUE, lwd=2, drawlabels=FALSE )
    title( main=paste0("j=",j) )
}
x0  <- seq( .1,.5,,6 )
y0  <- rep( 0.01,6 )
a   <- .075
phi <- seq( 15,,30,6 )*pi/180
x1  <- x0 + a*cos( phi )
y1  <- y0 + a*sin( phi )
rect( min(x0,x1)-.05, min(y0,y1)-.05,
      max(x0,x1)+.05, max(y0,y1), col="black", border=NA )
arrows( x0, y0, x1, y1, length=.05, col=2:7, lwd=2, code=3 )
par( oldpar )
```

---

get_en *get energy from the dualtree transform*

Description

square the wavelet coefficients and apply the bias correction

Usage

```
get_en(dt, correct = "none", N = ncol(dt))
```

Arguments

`dt`	array of `J x nx x ny x 6` complex wavelet coefficients, output of `dtcwt(..., dec=FALSE)`
`correct`	type of correction, either `"b"` or `"b_bp"`, any other value results in no correction at all.
`N`	the smallest whole power of two larger than or equal to the dimensions of the input image, usually just `ncol(dt)`

Details

The bias correction matrix should correspond to the filter bank used in the transform, for details on the matrices see `A`.

Value

an array of the same dimensions as `dt`

References

Nelson, J. D. B., A. J. Gibberd, C. Nafornita, and N. Kingsbury (2018) <doi:10.1007/s11222-017-9784-0>

See Also

`A`

`smooth_borders` *smoother borders*

Description

let a field decrease linearly towards its edges

Usage

```
smooth_borders(x, r)
```

Arguments

`x`	a real matrix
`r`	a positive integer

Details

Values within the field are linearly reduced from their original value to the field minimum, starting `r` pixels away from the edge. This enforces truely periodic boundaries and removes sharp edges.

Value

a matrix of the same dimensions as x

Note

r must not be larger than `min( dim(x) )/2`.

Examples

```
image( smooth_borders(blossom, r=64), col=gray.colors(128,0,1) )
```

`uvplot` *plot centre as vectors*

Description

display the radial and angular component of the spectrum's centre as arrows.

Usage

```
uvplot(uv, z = NULL, x = NULL, y = NULL, col = "green",
  zcol = grDevices::gray.colors(32, 0, 1), n = 42, f = 1,
  length = 0.05, ...)
```

Arguments

`uv`	an array of dimension `nx x ny x 2` , containing the u- and v-component, result of `cen2uv`
`z`	image to show in the background, defaults to `sqrt(x^2+y^2)`
`x, y`	optional x- and y-coordinates for the plot, must match the dimensions of `z`
`col`	color of the arrows
`zcol`	color scale for the image
`n`	number of arrows in one direction
`f`	factor by which to enlarge the arrows
`length`	length of the arrowhead in inches
`...`	further arguments passed to `image`

Details

The pivot of the arrows is at the location to which the u- and v-component belong. No arrowhead is displayed since the egdges detcted by the cdtwt have an orientation but no sign. The default size of the arrows is such that a 'velocity' of 1 corresponds to 5% of the shorter image side.

See Also

`cen2uv`

Examples

```
uv <- cen2uv( dt2cen( fld2dt( blossom ) ) )
uvplot( uv, z=blossom )
```

Index