Package 'dualtrees'

October 28, 2019

Title Decimated and Undecimated 2D complex dual-tree wavelet transform

Version 0.0.1

Description What the package does (one paragraph).

Depends R (>= 3.5.0)

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

`A` *Bias correction matrices*

Description

Matrices supposedly needed in order to eliminate the effect of "spectral leakage" for the local dtcwt-spectra. Used by biascor(...).

Usage

```
A_b_bp

A_b
```

Format

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

Source

Calculated by hand via /user/s6sebusc/wavelets_verification/general_scripts/Amats_cdtwt.r

Examples

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

biascor *spectral bias correction, implemented in FORTRAN*

Description

spectral bias correction, implemented in FORTRAN

Usage

```
biascor(en, a)
```

blossom *Two meteorologists in front of cherry blossoms*

Description

A very beautiful image.

Usage

```
blossom
```

Format

A 512x512 matrix of gray-scale values

Source

real life

Examples

```
image(blossom, col=gray.colors(32,0,1))
```

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

Description

Various boundary conditions for the 2D wavelet transform.

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

an object of class `bc_field`

Examples

```
bc <- pad( boys, N=300 )
image( bc$res, col=grey.colors(32) )
print( range( bc$res[ bc$px, bc$py ] - boys ) )
```

boys *Two stromchasers in the sun*

Description

Another classic image.

Usage

```
boys
```

Format

A 256x256 matrix of gray-scale values

Source

real life

Examples

```
image(boys, col=gray.colors(32,0,1))
```

cen2uv *transform angle and anisotropy into vector components for plotting*

Description

transform angle and anisotropy into vector components for plotting

Usage

```
cen2uv(cen)
```

dt2cen *get the centre of the DT-spectrum*

Description

get the centre of the DT-spectrum

Usage

```
dt2cen(pyr)
```

dtcwt *The 2D forward dualtree complex wavelet transform*

Description

This function performs the dualtree complex wavelet analysis, either with or withour decimation

Usage

```
dtcwt(mat, fb1 = near_sym_b, fb2 = qshift_b, J = NULL, dec = TRUE,
  mode = NULL, verbose = TRUE, boundaries = "periodic")
```

Arguments

`mat` the real matrix we wish to transform

`fb1` A list of analysis filter coefficients for the first level. Currently only near_sym_b and near_sym_b_bp are implemented

`fb2` A list of analysis 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

`mode` how to perform the convolutions, either "direct" (default if dec=TRUE) or "FFT" (default if dec=FALSE)

`verbose` if TRUE, the function tells you which level it is working on

`boundaries` how to handle the internal boundary conditions of the convolutions, has no effect if mode="direct"

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 the same ot the other to the columns. In the decimated case (dec=TRUE), each convolution is followed by a downsampling, meaining that the size of the six coefficient fields is cut in half at each level. In this case, it is supposedly efficient to use direct convolutions (mode="direct"), the boundary conditions of which are steered by the boundaries-argument. If dec=FALSE, direct convolutions may be slow and you should use mode="FFT". In that case, you need to handle the boundary conditions externally (enter a nice 2^N x 2^M matrix) and the maximum level J is smaller than log2(N) due to the construction of the filters via an 'algorithme a trous'.

Value

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

Note

Periodic and reflective boundaries are both implemented for the decimated case, but only the periodic boundaries are actually invertible at this point.

References

Selesnick, I.W., R.G. Baraniuk, and N.C. Kingsbury. "The Dual-Tree Complex Wavelet Transform." IEEE Signal Processing Magazine 22, no. 6 (November 2005): 123–51. `https://doi.org/10.1109/MSP.2005.1550194`.

See Also

`idtcwt`

Examples

```
dt <- dtcwt( boys )
par( mfrow=c(2,3), mar=rep(2,4) )
for( j in 1:6 ){
    image( boys, col=grey.colors(32,0,1) )
    contour( Mod( dt[[3]][ ,,j ] )**2, add=TRUE, col="green" )
}
```

---

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, as the a- and b-filters for 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.

Source

dtcwt python package

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, mode = NULL,
  correct = NULL, verbose = FALSE, boundary = "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
mode	how to handle the convolutions
correct	how to correct the bias, either "fast", "b" or "b_bp" - any other value results in no correction
verbose	whether or not you want the transform to talk to you
boundary	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 the thrown into dtcwt. Then the original domain is cut out, the coefficients are squared and the bias is corrected.

Value

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

See Also

biascor

Examples

```
dt <- fld2dt( boys )
par( mfrow=c(2,2), mar=rep(2,4) )
for( j in 1:4 ){
    image( boys, 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 )
```

get_en *get energy from the dualtree transform*

Description

get energy from the dualtree transform

Usage

```
get_en(pyr, correct = "fast")
```

idtcwt *The 2D inverse dualtree complex wavelet transform*

Description

Reconstructs an image from the pyramid of complex directional wavelet coefficients.

Usage

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

Arguments

pyr	a list containing arrays of complex coefficients for each level of the decomposition, produced by dtcwt(..., dec=TRUE).
fb1	the filter bank for the first level
fb2	the filter bank for all following levels
verbose	if true, the function will say a few words while doing its thing.
boundaries	how to handle the boundary conditions, should be the same as for the decomposition.

Details

This function re-arranges the six complex daughter coefficients back into the four trees, convolves them with the synthesis wavelets and adds everything up to recover an 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

a real array of size `2N x 2M` where `dim( pyr[[1]] ) = (M,N,6)` .

Note

At present, only boundaries="periodic" actually works :(

References

Selesnick, I.W., R.G. Baraniuk, and N.C. Kingsbury. “The Dual-Tree Complex Wavelet Transform.” IEEE Signal Processing Magazine 22, no. 6 (November 2005): 123–51. https://doi.org/10.1109/MSP.2005.1550194.

See Also

`dtcwt`

Examples

```
py <- dtcwt( boys )
boys_i <- idtcwt( py )
image( boys - boys_i )
```

`my_conv` *Column-convolutions*

Description

This function convolves the columns of a matrix mat with a filter fil.

Usage

```
my_conv(mat, fil, dec = TRUE, mode = "direct", odd = FALSE,
  boundaries = "periodic")
```

Arguments

`mat`	a matrix
`fil`	the filter to convolve the columns with
`dec`	if `TRUE`, every second row is discarded after the convolution
`mode`	how to actually do the convolutions, must be either `"direct"` or `"FFT"`
`odd`	if `TRUE`, the first row is discarded, otherwise the second row is.
`how`	to handle the boundaries, does nothing if `mode="FFT"`

Details

This functions does all of the actual computations inside the wavelet transform. The direct mode uses `filter(...)` and can handle any field size you like. It is supposedly faster when the filters are short, i.e., in the decimated case. The FFT-version really only works when the input dimensions are whole powers of two and the filter is not longer than the columns of the matrix.

Value

a matrix with as many columns and either the same (`dec=FALSE`) or half (`dec=TRUE`) the number of rows as mat

Examples

```
dboysdy <- my_conv( boys, c(-1,1), dec=FALSE )
dboysdx <- t( my_conv( t(boys), c(-1,1), dec=FALSE ) )
par( mfrow=c(1,2) )
image( dboysdx, col=gray.colors(32) )
image( dboysdy, col=gray.colors(32) )
```

Index