Empirical Wavelet Transform Toolbox
Version 5.0

Jérôme Gilles

Chapter 1

Introduction

The purpose of this document is to provide useful information on how is organized and how use the Empirical Wavelet Transform Toolbox (EWTT), not to explain what are the principles of the Empirical Wavelet Transform. In this document, we assume that the reader knows what is the EWT and how it works. If it is not the case, I advise you to read the papers [1, 2, 3].

Different releases:

- Version 5.0 (July 12th, 2022): fix a few minor bugs + add the code to perform the empirical watershed transform as well as the empirical Voronoi transform.
- Version 4.0 (December 13th, 2019): Code clean up + add the prefix "`EWT_`" in most function names. The option `upenv` for the spectrum regularization is now called `closing`. The mirroring used in some 2D filtering has been removed. The 1D transform can now handle complex signals by detecting boundaries in both the positive and negative frequencies. This implies that the empirical wavelet filters are now non-necessarily symmetric hence themselves complex. The construction of the 2D curvelet filters has been revised, simplified and now guarantee perfect symmetry which permits to obtain almost perfect reconstruction. All other 2D functions have been cleaned/simplified whenever it was possible. The plotting functions now add some titles on each subplot to help to recognize what we see.
- Version 3.4 (November 20th, 2019): fix a bug in the construction of the 1D wavelet filters (for the last filter in the high frequencies) by adding the function `EWT_Meyer_Wavelet_last.m` and removal of using the mirroring in the 1D code before performing the filtering. These fixes now permit to get almost perfect reconstruction. A function called `test_tight_frame.m` was added in the Tests/1D folder to check if the sum of the square of the constructed filters equals 1. Finally a test was added in the `EWT_TF_Plan.m` function such that it can manage either horizontal or vertical boundaries vectors.

- Version 3.2 (September 6th, 2016): new faster code to compute the scale-space boundary detection + remove all parfor so the toolbox can be use if you don't have any parallel capabilities.
- Version 3.0 (July 21th, 2015): adding the EWT-Curvelet option 3 (scales detected per angular sector) + fix of image sizes issues + wrong curvelet filters for the last scale.
- Version 2.0 (April 23th, 2014): adding new functions (like scale-space detection, TF plane generation,...) + documentation, the FTC method is removed and no longer available.
- Version 1.2 (June 18th, 2013): bugs fixes
- Version 1.0 (June 10th, 2013): original version

This toolbox is organized into several folders:

- 1D: contains functions to perform the 1D EWT,
- 2D: contains functions in several subfolders to perform the different 2D EWT,
- Boundaries: contains all different algorithms to perform the Fourier boundaries detection (also organized into several subfolders),
- Documentation: contains this document,
- Tests: contains 1D and 2D test scripts (these are the scripts used to generate the experiments of the different papers + new options),
- Utilities: contains useful functions like visualization functions.

We can split the available functions into two main categories:

- main functions: these are the functions you are supposed to regularly use to perform the EWT,
- supporting functions: these functions are basically used by the main function. A priori, you are not supposed to use them directly except if you want to modify the behavior or add new functionalities to this toolbox.

The following of the document review the available functions. Only the main functions will described in details, the auxiliary functions will only be listed, if the reader is interested to learn about them, he is invited to read the Matlab help provided with each function.

Chapter 2

Fourier boundaries detection

Main functions

```
[boundaries,presig]=EWT_Boundaries_Detect(f,params)
```

Inputs

- `f`: the spectrum to segment (only the interval $[0, \pi]$ for real signals or $[0, 2\pi]$ for complex ones).
- `params`: parameters (see description below).

Outputs

- `boundaries`: the set of detected boundaries, normalized in the interval $[0, \pi]$ or $[0, 2\pi]$ (✎ the boundaries 0 and $\pi/2\pi$ are not stored in this vector).
- `presig`: the preprocessed spectrum obtained by the given options.

Description

This function performs the Fourier boundaries detection to find the Fourier supports which will be use to build the wavelet filters. This function executes three steps:

1. A global trend removal: this step estimates a global trend in the spectrum (useful when most of the Fourier energy is concentrated in the low frequencies, i.e. the spectrum have a kind of $1/f$ profile) and then remove this global trend to the spectrum. The available methods to perform this step are:
    - 'none': does nothing,

- 'plaw': estimates the global trend by a power law,
- 'poly': estimates the global trend by a polynomial interpolation,
- 'morpho': estimates the global trend by taking the average of the lower and upper envelopes (computed by the opening and closing mathematical morphology operators),
- 'tophat': returns the Top-Hat mathematical morphology signal.

2. A regularization: Useful when the spectrum looks "noisy". The available regularization methods are:
   - 'none': no regularization,
   - 'gaussian': applies a Gaussian filter to the spectrum,
   - 'average': applies an average filter to the spectrum,
   - 'closing': returns the upper envelope of the spectrum (computed by the closing mathematical morphology operator).
3. The detection itself can be perform by
   - 'locmax': computes the local maxima and then the boundaries are set as the midpoints between consecutive maxima,
   - 'locmaxmin': computes the local maxima and then the boundaries are set as the smallest minima between consecutive maxima (everything is computed over the preprocessed spectrum),
   - 'locmaxminf': same as 'locmaxmin' except that the smallest minima are computed on the original spectrum,
   - 'adaptivereg': starts from an initial set of boundaries and then adapts them as the smallest minima within a window around the considered boundary (the minima are computed from the preprocessed spectrum),
   - 'adaptive': same as previous except that the smallest minima are computed on the original spectrum.
   - 'scalespace': use the detection of meaningful modes in a scale-space representation of the spectrum.

Table 2.1 provides the complete list of the different parameters that must be fixed in the `params` structure.

```
[worm_bound,Th]=EWT_GSS_2D(f,params)
```

Inputs

- `f`: the 2D spectrum on which to detect the meaningful maxima.
- `params`: parameters (see description below).

Outputs

- `worm_bound`: collection of the coordinates of the significant worms
- `Th`: Detected threshold

Description

This function builds a scale-space representation of f via successive separable convolutions with a discrete Gaussian Kernel. The function automatically estimates how many scales are needed. It then extract the meaningful maxima by selected the long-living worms in the scale-space.

Parameter name	Value	Comments and other mandatory parameters
General Parameters		
`params.N`	integer	Maximum number of bands
`params.completion`	0 or 1	Indicates if an equidistant partitioning must be used to complete the number of bands in case of a lower number of detected bands than expected
`params.log`	0 or 1	Indicates if the boundaries detection must be performed on the logarithm of the spectrum instead of the spectrum itself
Global Trend Removal		
`params.globtrend`	'none'	
	'plaw'	
	'poly'	`params.degree`: interpolation polynomial degree
	'morpho'	
	'tophat'	
Regularization		
`params.reg`	'none'	
	'gaussian'	`params.lengthFilter`: length of the Gaussian mask `params.sigmaFilter`: Gaussian standard deviation
	'average'	`params.lengthFilter`: length of the average mask
	'closing'	`params.lengthFilter`: length of the structural element
Detection method		
`params.detect`	'none'	
	'locmax'	
	'locmaxmin'	
	'locmaxminf'	
	'adaptivereg'	`params.InitBounds`: list of the initial boundaries that should be adapted (the endpoint must NOT appear in this list)
	'adaptive'	`params.InitBounds`: list of the initial boundaries that should be adapted (the endpoint must NOT appear in this list)
	'scalespace'	`params.typeDetect`: must be set to one of: 'otsu', 'halfnormal', 'empiricallaw', 'mean','kmeans'

Table 2.1: List of all parameters needed for the boundaries detection function and their corresponding possible values.

Auxiliary functions

- EWT_Adaptive_Bounds_Adapt
- EWT_Boundaries_Completion
- EWT_RemoveTrend
- EWT_SpectrumRegularize
- EWT_SpectrumRegularize_2D
- LocalMaxima/EWT_LocalMax
- LocalMaxima/EWT_LocalMaxMin
- LocalMaxima/EWT_LocalMaxMin2
- MorphoMath/EWT_FunctionClosing
- MorphoMath/EWT_FunctionDilation
- MorphoMath/EWT_FunctionErosion
- MorphoMath/EWT_FunctionGranulometry
- MorphoMath/EWT_FunctionOpening
- MorphoMath/EWT_FunctionTopHat
- PowerLaw/EWT_Powerlaw_Estimator
- ScaleSpace/EWT_EmpiricalLaw
- ScaleSpace/EWT_EmpiricalLaw2D
- ScaleSpace/EWT_GSS_BoundariesDetect
- ScaleSpace/EWT_HalfNormalLaw
- ScaleSpace/EWT_HalfNormalLaw2D
- ScaleSpace/EWT_kmeansDetect
- ScaleSpace/EWT_kmeansDetect2D
- ScaleSpace/EWT_LengthScaleCurve
- ScaleSpace/EWT_MeaningfulScaleSpace
- ScaleSpace/EWT_MeanTh
- ScaleSpace/EWT_OtsuMethod
- ScaleSpace/EWT_OtsuMethod_2D

- ScaleSpace/EWT_PlanGaussianScaleSpace
- ScaleSpace/EWT_RemoveMerge
- ScaleSpace/EWWT_Get_Boundaries
- ScaleSpace/EWWT_removeBounds

Chapter 3

1D transform

Main functions

```
[ewt,mfb,boundaries]=EWT1D(f,params)
```

Inputs

- `f`: signal to transform
- `params`: parameters (see description below)

Outputs

- `ewt`: `ewt{i}` contains the wavelet coefficients for subband `i`. `ewt{1}` corresponds to the lower frequencies while `ewt{end}` to the higher frequencies.
- `mfb`: the filter bank (in the Fourier domain) built by the function and used to get each wavelet subband coefficients. Same ordering as `ewt`.
- `boundaries`: the set of detected boundaries, normalized in the interval $[0, \pi]$ for real signal or $[0, 2\pi]$ for complex ones (✎ the boundaries 0 and $\pi/2\pi$ are not stored in this vector).

Description

This function performs the 1D Empirical Wavelet Transform of `f` according to the options provided in `params`. These options correspond to the ones needed by the boundaries detection function described in chapter 2. This function automatically calls the boundaries detection function to find the empirical Fourier supports, then builds the empirical wavelet filter bank and computes the projection on the empirical wavelets.

```
rec=iEWT1D(ewt,mfb)
```

Inputs

- `ewt`: set of empirical wavelet coefficients.
- `mfb`: empirical wavelet filter bank used to get the coefficients.

Output

- `rec`: reconstructed signal

Description

This function performs the inverse 1D Empirical Wavelet Transform by using the dual formulation.

```
rec=EWT_Modes_EWT1D(ewt,mfb)
```

Inputs

- `ewt`: set of empirical wavelet coefficients.
- `mfb`: empirical wavelet filter bank used to get the coefficients.

Output

- `rec`: reconstructed IMFs, they are provided in the same order as `ewt`.

Description

This function reconstructs the IMFs as defined in the Empirical Mode Decomposition algorithm.

```
Hilb=EWT_InstantaneousComponents(ewt,boundaries)
```

Inputs

- `ewt`: set of empirical wavelet coefficients.
- `boundaries`: the set of detected boundaries, normalized in the interval $[0, \pi]$ (✎ the boundaries 0 and π are not stored in this vector).

Output

- `Hilb`: contains the instantaneous amplitudes and frequencies of each `ewt` component. The first index correspond to the subband while the second must be equal to 1 or 2 corresponding to the amplitudes and frequencies, respectively. (e.g. `Hilb{i}{1}` is the instantaneous amplitude of subband `i` and `Hilb{i}{2}` is the instantaneous frequencies of subband `i`)

Description

This functions works only for real signals!
This function extracts the instantaneous amplitudes and frequencies of each `ewt` component by using the Hilbert transform.
✎ A postprocessing is applied on the frequencies to correct the issue that the Matlab Hilbert and phase unwrapping functions do not guaranty to have a monotonic increasing phase.

Auxiliary functions

- `EWT_beta`
- `EWT_LP_FilterBank`
- `EWT_LP_Scaling`
- `EWT_LP_Wavelet`
- `EWT_LP_Wavelet_last`
- `EWT_IFcleaning`

Chapter 4

2D transforms

The toolbox proposes several type of 2D EWT: Tensor, Littlewood-Paley, Ridgelet and Curvelet. If the corresponding functions are stores in distinct folders, they can share common auxiliary functions.
✎ To be able to run the Littlewood-Paley, ridgelet and curvelet functions, the Pseudo-Polar-Fourier-Transform toolbox must be properly installed on your computer. This toolbox is freely available on Michael Elad's webpage `http://www.cs.technion.ac.il/~elad/Various/PolarLab.zip`

Main functions

```
[ewtc,mfb,Bw,Bt]=EWT2D_Curvelet(f,params)
```

Inputs

- `f`: image to transform
- `params`: parameters (see description below)

Outputs

- `ewtc`: cell containing each filtered output subband (`ewtc{1}` is the lowpass subband and for EWTC-I and EWTC-II the next `ewtc{s}{t}` are the bandpass filtered images, `s` corresponds to the scales and `t` to the direction; while for EWTC-III the next output follow the ordering `ewtc{t}{s}`)
- `mfb`: cell containing the set of 2D empirical filters in the Fourier domain (the indexation is the same as `ewtc` above)
- `Bw`: list of detected scale (radial) boundaries
- `Bt`: list of detected angle boundaries

Description

This function performs the empirical curvelet transform of the input image `f`. You must provide the same parameters as for the boundaries detection function described in chapter 2 as well as the complimentary ones listed in table 4.1.

Parameter name	Value	Comments and other mandatory parameters
General Parameters		
`params.option`	integer	1 = radial and angle Fourier boundaries detected independently 2 = radial boundaries detected first and then the angular ones 3= angular sectors detected first and then the scales per angular sector
`params.curvN`	integer	Maximum number of angles
Global Trend Removal along the angles		
`params.curvpreproc`	'none'	
	'plaw'	
	'poly'	`params.curvedegree`: interpolation polynomial degree
	'morpho'	
	'tophat'	
Regularization along the angles		
`params.curvreg`	'none'	
	'gaussian'	`params.curvlengthFilter`: length of the Gaussian mask `params.curvsigmaFilter`: Gaussian standard deviation
	'average'	`params.curvlengthFilter`: length of the average mask
	'closing'	`params.curvlengthFilter`: length of the structural element
Detection method along the angles		
`params.curvmethod`	'none'	
	'locmax'	
	'locmaxmin'	
	'scalespace'	uses the same parameters as for the scale case

Table 4.1: List of complimentary parameters needed for the curvelet transform.

```
rec=iEWT2D_Curvelet(ewt,mfb)
```

Inputs

- `ewt`: cell containing the empirical curvelet coefficients
- `mfb`: filter bank built by the empirical curvelet transform

Output

- `rec`: reconstructed image

Description

This function performs the inverse 2D Empirical Curvelet Transform.

```
[ewtLP,mfb,boundaries]=EWT2D_LittlewoodPaley(f,params)
```

Inputs

- `f`: image to transform
- `params`: parameters (see description below)

Outputs

- `ewtLP`: cell containing each filtered output subband (`ewtc{1}` is the low-pass subband and the next `ewtc{s}` are the bandpass filtered images, `s` corresponds to the scales)
- `mfb`: cell containing the set of 2D empirical filters in the Fourier domain (the indexation is the same as `ewtLP` above)
- `boundaries`: list of detected scale (radial) boundaries

Description

This function performs the 2D empirical Littlewood-Paley transform of the input image `f`. You must provide the same parameters as for the boundaries detection function described in chapter 2.

```
rec=iEWT2D_LittlewoodPaley(ewt,mfb)
```

Inputs

- `ewt`: cell containing the empirical Littlewood-Paley coefficients
- `mfb`: filter bank built by the empirical Littlewood-Paley transform

Output

- `rec`: reconstructed image

Description

This function performs the inverse 2D Empirical Littlewood-Paley Transform.

```
[ewtLP,mfb,boundaries]=EWT2D_Ridgelet(f,params)
```

Inputs

- `f`: image to transform
- `params`: parameters (see description below)

Outputs

- `ewtLP`: cell containing each filtered output subband (`ewtc{1}` is the low-pass subband and the next `ewtc{s}` are the bandpass filtered images, `s` corresponds to the scales)
- `mfb`: cell containing the set of 2D empirical filters in the Fourier domain (the indexation is the same as `ewtLP` above)
- `boundaries`: list of detected boundaries

Description

This function performs the 2D empirical Ridgelet transform of the input image `f`. You must provide the same parameters as for the boundaries detection function described in chapter 2.

```
rec=iEWT2D_Ridgelet(ewt,mfb)
```

Inputs

- `ewt`: cell containing the empirical Ridgelet coefficients
- `mfb`: filter bank built by the empirical Ridgelet transform

Output

- `rec`: reconstructed image

Description

This function performs the inverse 2D Empirical Ridgelet Transform.

```
[ewtC,mfbR,mfbC,BR,BC]=EWT2D_Tensor(im,params)
```

Inputs

- `im`: image to transform
- `params`: parameters (see description below)

Outputs

- `ewtC`: cell containing each filtered output subband (`ewtc{1}` is the lowpass subband and the next `ewtc{r}{c}` are the bandpass filtered images, `r` corresponds to the scales along the rows and `c` to the scales along the columns)
- `mfbR`: cell containing the set of 2D empirical filters for the rows in the Fourier domain (the indexation is the same as for the 1D transform)
- `mfbR`: cell containing the set of 2D empirical filters for the columns in the Fourier domain (the indexation is the same as for the 1D transform)
- `BR`: list of detected boundaries along the rows
- `BC`: list of detected boundaries along the columns

Description

This function performs the 2D empirical tensor wavelet transform of the input image `f`. You must provide the same parameters as for the boundaries detection function described in chapter 2.

```
im=iEWT2D_Tensor(ewt2d,mfbR,mfbC)
```

Inputs

- `ewt2d`: cell containing the empirical tensor wavelet coefficients
- `mfbR`: cell containing the set of 2D empirical filters for the rows
- `mfbR`: cell containing the set of 2D empirical filters for the columns

Output

- `im`: reconstructed image

Description

This function performs the inverse 2D Empirical Tensor Wavelet Transform.

```
[ewtc,mfb,centers,supports] = EWT2D_Watershed(f,params)
```

Inputs

- `f`: image to transform
- `params`: parameters (see description below)

Outputs

- `ewtc`: cell containing each filtered output subband (`ewtc{1}` is the low-pass subband and the next `ewtc{s}` are the bandpass filtered images, `s` corresponds to the scales)
- `mfb`: cell containing the set of 2D empirical filters in the Fourier domain (the indexation is the same as `ewtc` above)
- `centers`: centers: detected local maxima representing modes
- `supports`: images containing the obtained partition

Description

This function performs the Empirical Watershed Wavelet Transform. The partitioning is achieved by a combination of scale-space representations for mode detection and the watershed transform for actual partitioning. The watershed transform is is performed by the built-in matlab function based on F.Meyer's watershed algorithm. After, it uses the arbitrary shape empirical wavelet transform. You must provide the same parameters as for the boundaries detection function described in chapter 2. A regularization can be applied to the spectrum. The different available regularization are:

- 'none': no regularization,
- 'gaussian': applies a Gaussian filter to the spectrum,
- 'average': applies an average filter to the spectrum.

```
rec = iEWT2D_Watershed(ewtc,mfb)
```

Inputs

- `ewtc`: cell containing the empirical wavelet coefficients
- `mfb`: cell containing the set of 2D empirical watershed filters

Output

- `rec`: reconstructed image

Description

This function performs the inverse 2D Empirical Watershed Wavelet Transform.

```
[ewtc,mfb,maxima,vorpartition] = EWT2D_Voronoi(f,params)
```

Inputs

- `f`: image to transform
- `params`: parameters (see description below)

Output

- `ewtc`: collection of outputs of each EW filter
- `mfb`: the built filter bank
- `maxima`: coordinates of each meaningful detected maxima
- `vorpartition`: detected Voronoi partition

Description

This function compute the 2D EWT based on a Voronoi partitioning of the Fourier domain. If the real transform is required (i.e params.complex $\neq 1$), symmetric cells are grouped together to guarantee symmetric filters. A regularization can be applied to the spectrum. The different available regularization are:

- 'none': no regularization,
- 'gaussian': applies a Gaussian filter to the spectrum,
- 'average': applies an average filter to the spectrum.

```
function rec = iEWT2D_Voronoi(ewtc,mfb)
```

Inputs

- `ewtc`: cell containing the empirical wavelet coefficients
- `mfb`: cell containing the set of 2D empirical watershed filters

Output

- `rec`: reconstructed image

Description

This function performs the inverse Empirical Voronoi Wavelet Transform, returning a reconstruction of the image.

Auxiliary functions

- `Curvelet/EWT_AnglesLocalMax`

- Curvelet/EWT_AnglesLocalMaxMin
- Curvelet/EWT_Angular_sector
- Curvelet/EWT_CreateAngleGrid
- Curvelet/EWT2D_Curvelet_FilterBank
- Curvelet/EWT2D_Curvelet_Scaling
- Curvelet/EWT_Angles_Detect
- Littlewood-Paley/EWT2D_LP_FilterBank
- Littlewood-Paley/EWT2D_LP_Scaling
- Littlewood-Paley/EWT2D_LP_Wavelet
- Littlewood-Paley/EWT2D_UP_LP_Wavelet
- Empirical Watershed Wavelets/EWT2D_Watershed_Filterbank
- Empirical Watershed Wavelets/EWT_beta_2D
- Voronoi/EWT2D_get_voronoi_cell_index
- Voronoi/EWT2D_merge_symmetric
- Voronoi/EWT2D_Voronoi_Filterbank
- Voronoi/EWT2D_Voronoi_LP_function
- Voronoi/EWT2D_Voronoi_Partition

Chapter 5

Utilities

Main functions

```
tf=EWT_TF_Plan(ewt,boundaries,Fe,sig,rf,rt,resf,color)
```

Inputs

- `ewt`: 1D EWT coefficients.
- `boundaries`: the set of detected boundaries, normalized in the interval $[0, \pi]$.
- `Fe`: Sampling frequency if known ([ ] otherwise).
- `sig`: original signal or [ ]. It will be plotted on top of the figure if it is known.
- `rf`: frequency axis ratio or [ ] (only frequencies in the range [0,Fe/(2rf)] are plotted).
- `rt`: time axis ratio or [ ] (only the interval of time [0,full range/rt] is plotted).
- `resf`: frequency resolution ratio or [ ] (the number of frequency sample used is (Maximum number of points)/resf).
- `color`: 1 or [ ] = plot in color; 0 = plot in grayscale.

Outputs

- `tf`: matrix containing the image of the time-frequency plane

Description

This functions works only for real signals!
This function creates and plots the time-frequency representation by computing the instantaneous amplitudes and frequencies of each EWT components. By default the full plane in full frequency resolution is generated (e.g. [ ] for the above inputs is equivalent to Fe=1, rf=1, rt=1, resf=1).
✎ In most of the case, the full resolution for the frequencies gives a very large picture e.g. it necessary to zoom it to see the details. It is generally sufficient to reduce the frequency resolution by a factor of 2 or 4.

```
Show_EWT(ewt,f,rec)
```

Inputs

- `ewt`: 1D EWT coefficients.
- `f`: original signal.
- `rec`: reconstructed signal.

✎ the last two inputs are optional.

Description

This function plots all EWT components as well as the original signal and its reconstructed version if available.

```
Show_EWT_Boundaries(magf,boundaries,R,SamplingRate,InitBounds,presig)
```

Inputs

- `magf`: magnitude of the signal's spectrum.
- `boundaries`: set of detected Fourier boundaries.
- `R`: frequency axis ratio (only frequencies in the range [0,SamplingRate/(2rf)] are plotted).
- `SamplingRate`: sampling frequency used during the signal acquisition.
- `InitBounds`: initial set of boundaries used in the case of adaptive detection methods.
- `presig`: preprocessed spectrum used for the detection.

Description

This function plots the detected boundaries super-imposed on the signal's spectrum.

```
EWT_LP_boundaries(f,boundaries)
```

Inputs

- `f`: original image.
- `boundaries`: set of radial boundaries detected by the Littlewood-Paley transform.

Description

This function plots the concentric boundaries corresponding to the detected scales in the 2D spectrum.

```
ShowCurveletFilters(mfb)
```

Inputs

- `mfb`: set of curvelet filters provided by the curvelet transform.

Description

This function displays the curvelet filter bank (one figure per scale or angular sector) + an image corresponding to $\sum |\psi_n(\omega)|^2$ (useful to check the tight frame property) and print the value of $\|1 - \sum |\psi_n(\omega)|^2\|_\infty$.

```
Show_Curvelets_boundaries(f,Bw,Bt,option)
```

Inputs

- `f`: original image.
- `Bw`: list of detected scale (radial) boundaries.
- `Bt`: list of detected angle boundaries.
- `option`: 1=EWTC-I;2=EWTC-II; 3=EWTC-III (must be the same as the one used in the curvelet transform).

Description

This function plots the angular sectors corresponding to the detected scales and angles in the 2D spectrum.

```
Show_EWT2D(ewt)
```

Inputs

- `ewt`: set of 2D Littlewood-Paley EWT components.

Description

This function displays the Littlewood-Paley EWT components.

```
Show_EWT2D_Curvelet(ewtc,option)
```

Inputs

- `ewtc`: set of empirical curvelet components.
- `option`: 1=EWTC-I;2=EWTC-II; 3=EWTC-III (must be the same as the one used in the curvelet transform).

Description

This function displays the empirical curvelet components (one figure per scale or angular sector).

```
Show_EWT2D_Filters(fil)
```

Inputs

- `fil`: set of 2D Littlewood-Paley filters built by the transform.

Description

This function displays the Littlewood-Paley filters built by the transform.

```
Show_EWT2D_Tensor(ewt2d)
```

Inputs

- `ewt2d`: set of 2D empirical tensor wavelet components .

Description

This function displays the empirical tensor wavelet components.

```
EWT_Tensor_Plot_Boundaries(f,BR,BC)
```

Inputs

- `f`: original image.
- `BR`: list of detected boundaries along the rows.
- `BC`: list of detected boundaries along the columns.

Description

This function plots vertical and horizontal boundaries detected by the tensor transform in the 2D spectrum.

```
Show_EWT2D_Voronoi_boundaries(f,vor,color,logspec)
```

Inputs

- `f`: input image.
- `vor`: image containong the plot of the Voronoi partition.
- `color`: vector corresponding to the wanted color for the partition edges. This must be a vector of the form [R G B] where the quantities R,G,B are provided within [0,1]. If this parameter is not provided, the plot will be made in red by default ([1 0 0]).
- `logspec`: flag to indicate if the function plots the logarithm of the image spectrum (logspec = 1). If not provided, the default behavior is to NOT plot the logarithm (i.e logspec = 0).

Description

This function plot the edges of the Voronoi partition onto the magnitude of the Fourier spectrum of the input image.

```
Show_Watershed_boundaries(f,supports,loga)
```

Inputs

- `f`: Image
- `supports`: Image containing the labelled partition
- `loga`: 1=logarithm of spectrum; 0 otherwise

Description

This function displays the detected watershed boundaries (in red) and detected maxima in green) overtop the image's magnitude spectrum.

```
params=EWTDefaultParams()
```

Output

- `params`: structure containing all default parameters

Description

This function initializes a `params` structure with default values (look in the file to see what are those default values).

Auxiliary functions

- `EWT_drawArcEllipse`
- `EWT_drawEllipse`

Bibliography

[1] Jérôme Gilles, *Empirical Wavelet Transform*, IEEE Transactions on Signal Processing, 61(16), 3999–4010, 2013.

[2] Jérôme Gilles and Giang Tran and Stanley Osher, *2D Empirical transforms. Wavelets, Ridgelets and Curvelets Revisited*, SIAM Journal on Imaging Sciences, 7(1), 157–186, 2014.

[3] Jérôme Gilles and Kathryn Heal, *A parameterless scale-space approach to find meaningful modes in histograms - Application to image and spectrum segmentation*, International Journal of Wavelets, Multiresolution and Information Processing, Vol.12, No.6, 1450044-1–1450044-17, December 2014.