M2CAD

1.0

Generated by Doxygen 1.8.10

Tue Dec 1 2015 11:58:28

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Namespace Documentation

2.1 M2CAD.colour_subtraction Namespace Reference

Functions

- def make_colour_sub (Sfile, Afile, Xfile, suffixe, prefix='./', cuts=['0')

Variables

- list infr = cuts[0]

  Pour simu Red levels infr = str(-0.1) maxr = str(0.6) Blue levels infb = str(-0.02) maxb = str(0.1) Green levels.
- list **maxr** = cuts[1]
- list **infg** = cuts[2]
- list **maxg** = cuts[3]
- list **infb** = cuts[4]
- list **maxb** = cuts[5]
- tuple infrn = str(-5∗sigmar)

  Pour Refsdal

  #### Red levels
- tuple **maxrn** = str(5∗sigmar)
- tuple **infbn** = str(-5∗sigmab)
- tuple **maxbn** = str(5∗sigmab)
- tuple **infgn** = str(-5∗sigmag)
- tuple **maxgn** = str(5∗sigmag)
- string **name_red** = prefix+'Red_'
- string **name_blue** = prefix+'Blue_'
- string **name_colour** = prefix+'All_'
- string **name_all** = prefix+'Res_'

2.1.1 Detailed Description

`@package M2CAD`

2.1.2 Function Documentation

2.1.2.1 def M2CAD.colour_subtraction.make_colour_sub (*Sfile, Afile, Xfile, suffixe, prefix* = './', *cuts* = ['0'])

```
 Creates colour images and visualisation of the residuals of the separation of estimated sources from a colour
 INPUTS:
    Sfile: name and path to a fits file with sources as estimated from M2CAD.
    Afile: name and path to a fits file with mixing coefficients as estimated from M2CAD.
    Xfile: name and path to a fits cube with original multi-band images used to feed M2CAD.
    suffixe: string that will be added at the end of the names of the png files showing the residuals.
OUTPUTS:
    none. The code writes fits files and png files with the resulting residuals:
        prefix+'Colour_images.fits'
        prefix+'Red_residuals.fits'
        prefix+'Blue_residuals.fits'
        prefix+'Colour_residuals.fits'
        prefix+'S1_'+suffixe+'.png'
        prefix+'S2_'+suffixe+'.png'
        prefix+'Red_'+suffixe+'.png'
        prefix+'Blue_'+suffixe+'.png'
        prefix+'All_'+suffixe+'.png'
        prefix+'Res_'+suffixe+'.png'
OPTIONS:
    prefix: string, location where to save fits and png files.
    cuts: colour cuts to apply to ds9 visualisation tool. cuts is an array
    with values [minR, maxR, minG, maxG, minB, maxB] where minR is the lower red cut and maxR
    is the maximum red cut (idem for Green and Blue)
```

2.1.3 Variable Documentation

2.1.3.1 list infr = cuts[0]

Pour simu Red levels infr = str(-0.1) maxr = str(0.6) Blue levels infb = str(-0.02) maxb = str(0.1) Green levels.

infg = str(-0.05) maxg = str(0.3) # Red levels infr = str(0) maxr = str(0.015) Blue levels infb = str(-0.01) maxb = str(0.02) Green levels

infg = str(-0.005) maxg = str(0.015) Pour Refsdal

Red levels

2.2 M2CAD.MCA Namespace Reference

Functions

- def mMCA
- def MOM
- def MAD (x)
- def mr_filter
- def linorm (A, nit)
- def PCA_initialise

Variables

- tuple reweight = (thmap[j,:,:,:])

  test
- tuple **sub** = (thmap[j-1,:,:,:])
- int **weight2** = 1
- tuple **kthr** = np.max([kmax, k])

- **X** = X+S
- **k** = k-step
- tuple **S** = np.zeros((ns,n1,n2))
- tuple **sig** = np.zeros((nb))
- int **serr** = sig+0
- tuple **err** = np.sum((img[i,:,:]-A[i,0]∗S[0,:,:]-A[i,1]∗S[1,:,:]))
- tuple **Chi** = np.sum(serr/sigma_y∗∗2)
- tuple **th** = np.multiply(th.T,levels[:sh[0]])
- int **imnew** = 0
- int **i** = 0
- **R** = img
- tuple **alpha** = mw.wave_transform(R,lvl, newwave = 0)
- tuple **Rnew** = mw.iuwt(M∗alpha)
- tuple **wmap** = mw.wave_transform(imnew,lvl)

2.2.1 Detailed Description

```
@package M2CAD
```

2.2.2 Function Documentation

2.2.2.1 def M2CAD.MCA.linorm (*A, nit*)

```
  Estimates the maximal eigen value of a matrix A

  INPUTS:
      A: matrix
      nit: number of iterations

  OUTPUTS:
      xn: maximal eigen value

   EXAMPLES
```

2.2.2.2 def M2CAD.MCA.MAD (*x*)

```
  Estimates noise level in an image from Median Absolute Deviation

  INPUTS:
      x: image

  OUTPUTS:
      sigma: noise standard deviation

  EXAMPLES
```

2.2.2.3 def M2CAD.MCA.mMCA (*img, A, kmax, niter, mode* = 'PCA', *PCA* = [2, *harder* = 0, *pos* = False, *threshmode* = 'mom', *lvl* = 6, *soft* = False, *reweighting* = 'none')

```
  mMCA runs the M2CAD algorithm over a cube of multi-band images.

  INPUTS:
      img: multiband cube with size nbxn1xn2 where nb is the number of bands and n1xn2,
the size of the images
      A: the mixing matrix. if mode is set to 'PCA', A will be ignored and can be set to 0
      kmax: detection threshold in units of noise standard deviation usually chosen between 3 and 5
      niter: number of iterations of the M2CAD algorithm
```

```
  OUTPUTS:
      S: extracted sources
      A: mixing matrix, either given by the user or estimate by PCA with option mode ='PCA'

  OPTIONS:
      mode: if set to 'PCA', the mixing matrix A will be estimated from PCA decomposition of the SEDs
      PCA: parameters for PCA sensitivity. if mode is set to 'PCA', the PCA estimator will take PCA[0]
as the number of sources to be extracted and PCA[1] as a sensitivity parameter to discriminate between
source. Values betwee 5 and 30 are usually recommended
      harder: if set to 1,
      pos: if set to True, the output of the hard thresholding procedure is constrined to be positive
      threshmode: if set to 'mom', adaptive method of moments is used at every iteration to decrease the  thre
      lvl: number of wavelet levels to use in the decompositions, default is 6.
      soft: if set to True, soft thresholding is used

  EXAMPLE:
```

2.2.2.4 def M2CAD.MCA.MOM (*R, sigma, lvl =* 6)

```
Estimates the best for a threshold from method of moments

  INPUTS:
      R: multi-sources cube with size nsxn1xn2 where ns is the number of sources
      and n1xn2, the size of an image
      sigma: noise standard deviation

  OUTPUTS:
      k: threshold level

  OPTIONS:
      lvl: number of wavelet levels used in the decomposition, default is 6.

  EXAMPLES
```

2.2.2.5 def M2CAD.MCA.mr_filter (*img, niter, k, sigma, lvl =* 6, *pos =* False, *harder =* 0, *mulweight =* 1, *subweight =* 0, *addweight =* 0, *soft =* False)

```
  Computes wavelet iterative filtering on an image.

  INPUTS:
      img: image to be filtered
      niter: number of iterations (10 is usually recommended)
      k: threshold level in units of sigma
      sigma: noise standard deviation

  OUTPUTS:
      imnew: filtered image
      wmap: weight map

  OPTIONS:
      lvl: number of wavelet levels used in the decomposition, default is 6.
      pos: if set to True, positivity constrain is applied to the output image
      harder: if set to one, threshold levels are risen. This is used to compensate for correlated noise
      for instance
      mulweight: multiplicative weight (default is 1)
      subweight: weight map derived from other sources applied to diminish the impact of a given set of coeffi
      addweight: weight map used to enhance previously detected features in an iterative process (default is 0
      soft: if set to True, soft thresholding is used

  EXAMPLES
```

2.2.2.6 def M2CAD.MCA.PCA_initialise (*cube, ns, angle =* 15, *npca =* 64)

```
  Estimates the mixing matrix of of two sources in a multi band set of images
```

```
  INPUTS:
      cube: multi-band cube from which to extract mixing coefficients
      ns: number of mixed sources

  OUTPUTS:
      A0: mixing matrix

  OPTIONS:
      angle: sensitivity parameter. The angular resolution at which the algorithm has to look for PCA coeffici
      npca: square root of the number of pixels to be used. Since too big images result in too big computation
      we propose to downsample the image in order to get reasonable calculation time

  EXAMPLES
```

2.3 M2CAD.mk_pca Namespace Reference

Functions

- def mk_pca
- def rec_pca

2.3.1 Detailed Description

```
@package M2CAD
```

2.3.2 Function Documentation

2.3.2.1 def M2CAD.mk_pca.mk_pca (*vectors, dec* = 0)

```
Perfoms Principal Component Analysis of a set of vectors
INPUTS:
    vectors: Set of vectors to be decomposed through PCA.
OUTPUTS:
    alpha: PCA coefficients resulting of the decomposition of the vectors.
    EN_2: PCA basis set.
OPTIONS:
    dec: if non zero, dec is used as a PCA basis to decompose the vectors. In this case,
    a simple projection is thus conducted instead of the PCA.
```

2.3.2.2 def M2CAD.mk_pca.rec_pca (*alpha, base, lim* = 0)

```
Reconstructs a signal in direct space from its PCA coefficients and the basis over which it has been decompose
INPUTS:
    alpha: sets of PCA coefficients.
    basis: the basis over which the signal has been decomposed.
OUTPUTS:
    rec: reconstructed signal
OPTIONS:
    lim: if lim is set to non-zero value, lim is maximal number of coefficients used in the reconstruction.
```

2.4 M2CAD.pca_ring_spectrum Namespace Reference

Functions

- def pca_ring_spectrum (images)
- def actg (X, Y)
- def pca_lines (alphas, sig, dt, ns)

Variables

- tuple noise = np.multiply(np.random.randn(100,s),std.T)

  Noise propagation in PCA space.
- tuple **alphanoise** = np.dot(base.T,noise.T)
- tuple **sig** = np.zeros(2)
- int **count** = 0
- tuple locky = np.zeros(np.size(theta))

  Second correction of attractors.
- tuple **distance** = np.abs(theta-attractors[i])
- tuple **bigloc** = np.where(distance >= np.pi)
- tuple **locator** = np.zeros(np.size(angle))
- tuple **images** = np.zeros([n2**0.5,n2**0.5])
- int **clus** = angle+0
- int **n_clus** = ns+1
- list **colors** = [[0.6,0,0],np.array([135, 233, 144])/255.,[0,0,0]]
- list **col** = [0,0,0.7]
- list xy = alphas[0:2,np.where(locator == k)[0]]

class_member_mask = (clus== k)

2.4.1 Detailed Description

```
@package M2CAD
```

2.4.2 Function Documentation

2.4.2.1 def M2CAD.pca_ring_spectrum.actg (*X, Y*)

```
Computes the arctan(x/y) of two vectors.
INPUTS:
    X: 1-d vector
    Y: 1-d vector
OUTPUTS:
    angle: 1-d vector with the result of arctan(X/Y)

EXAMPLE:
```

2.4.2.2 def M2CAD.pca_ring_spectrum.pca_lines (*alphas, sig, dt, ns*)

```
Finds alignments in PCA coefficients and identifies corresponding structures in direct space. It is actually a
INPUTS:
    alphas: PCA coefficients.
    sig: noise levels in the two first PCA components
    dt: angular resolution at which the algorithm has to discriminate between coefficients of a same group
    ns: number of alignments to identify.
OUTPUTS:
    images: 2-d map of strucutres with same colours. Each structure has all its pixels set to the same value.
    Pixels identified as non-significant are set to 0.
EXAMPLE:
```

2.4.2.3 def M2CAD.pca_ring_spectrum.pca_ring_spectrum (*images*)

```
Decomposes a set of SEDs from multiband images into PCA and filters the less significant coefficients
INPUTS:
    images: cube of muti-bandimages with size n1xn2xs where s is the number of bands and n1xn2, the size of ea
OUTPUTS:
    alphas: PCA coefficients for each SED at each pixel location.
    basis: corresponding PCA basis.
    sig: noise as propagated into PCA space.


EXAMPLE:
```

2.5 M2CAD.wave_transform Namespace Reference

Functions

- def wave_transform
- def iuwt

2.5.1 Detailed Description

```
@package M2CAD
```

2.5.2 Function Documentation

2.5.2.1 def M2CAD.wave_transform.iuwt (*wave*, *convol2d* = 0)

```
Inverse Starlet transform.
INPUTS:
    wave: wavelet decomposition of an image.
OUTPUTS:
    out: image reconstructed from wavelet coefficients
OPTIONS:
    convol2d:  if set, a 2D version of the filter is used (slower, default is 0)
```

2.5.2.2 def M2CAD.wave_transform.wave_transform (*img*, *lvl*, *Filter* = 'Bspline', *newwave* = 1, *convol2d* = 0)

```
Performs starlet decomposition of an image
INPUTS:
    img: image with size n1xn2 to be decomposed.
    lvl: number of wavelet levels used in the decomposition.
OUTPUTS:
    wave: starlet decomposition returned as lvlxn1xn2 cube.
OPTIONS:
    Filter: if set to 'Bspline', a bicubic spline filter is used (default is True).
    newave: if set to True, the new generation starlet decomposition is used (default is True).
    convol2d: if set, a 2D version of the filter is used (slower, default is 0).
```