

---

# **RRtoolbox Documentation**

***Release 1***

**David Toro**

October 24, 2016



<b>1</b>	<b>RRtoolbox package</b>	<b>3</b>
1.1	Subpackages	3
1.1.1	RRtoolbox.lib package	3
	Subpackages	3
	Submodules	26
	RRtoolbox.lib.cache module	26
	RRtoolbox.lib.config module	31
	RRtoolbox.lib.descriptors module	33
	RRtoolbox.lib.directory module	35
	RRtoolbox.lib.image module	41
	RRtoolbox.lib.inspector module	53
	RRtoolbox.lib.plotter module	54
	RRtoolbox.lib.root module	63
	RRtoolbox.lib.serverServices module	69
	RRtoolbox.lib.session module	71
	Module contents	72
1.1.2	RRtoolbox.tools package	72
	Submodules	72
	RRtoolbox.tools.lens module	72
	RRtoolbox.tools.segmentation module	73
	RRtoolbox.tools.selectors module	75
	RRtoolbox.tools.sticher module	76
	Module contents	76
1.2	Submodules	76
1.3	RRtoolbox.core module	76
1.4	RRtoolbox.run module	76
1.5	RRtoolbox.shell module	76
1.6	Module contents	77
<b>2</b>	<b>Indices and tables</b>	<b>79</b>
	<b>Python Module Index</b>	<b>81</b>
	<b>Index</b>	<b>83</b>



Contents:



---

## RRtoolbox package

---

### 1.1 Subpackages

#### 1.1.1 RRtoolbox.lib package

##### Subpackages

##### RRtoolbox.lib.arrayops package

##### Submodules

**RRtoolbox.lib.arrayops.basic module** This module contains simple array operation methods

`RRtoolbox.lib.arrayops.basic.angle(v1, v2, deg=False)`

Angle between two N dimensional vectors.

##### Parameters

- **v1** – vector 1.
- **v2** – vector 2.
- **deg** – if True angle is in Degrees, else radians.

**Returns** angle in radians.

Example:

```
>>> angle_between((1, 0, 0), (0, 1, 0))
1.5707963267948966
>>> angle_between((1, 0, 0), (1, 0, 0))
0.0
>>> angle_between((1, 0, 0), (-1, 0, 0))
3.141592653589793
```

---

**Note:** obtained from <http://stackoverflow.com/a/13849249/5288758> and tested in <http://onlinemschool.com/math/assistance/vector/angl/>

---

`RRtoolbox.lib.arrayops.basic.angle2D(v1, v2, deg=False, absolute=None)`

Angle between two 2 dimensional vectors.

##### Parameters

- **v1** – vector 1.
- **v2** – vector 2.
- **deg** – if True angle is in Degrees, else radians.
- **absolute** – if None returns the angle (0 to 180(pi)) between v1 and v2. if True returns the absolute angle (0 to 360(2pi)) from v1 as axis to v2. if False returns the angle (0 to 180 or 0 to -180) from v1 as axis to v2, where v2 angle relative to v1 is positive or negative if counter-clock or clock wise.

**Returns** angle in radians.

---

**Note:** implemented according to <http://math.stackexchange.com/a/747992> and tested in <http://onlinemschool.com/math/assistance/vector/angle/>

---

RRtoolbox.lib.arrayops.basic.**angleXY**(*coorX*, *coorY*, *angle*)

Rotate coordinate.

**Parameters**

- **coorX** – x coordinate.
- **coorY** – y coordinate.
- **angle** – radian angle.

**Returns** rotated x,y

RRtoolbox.lib.arrayops.basic.**anorm**(*a*)

norm in array.

**Parameters** **a** –

**Returns**

RRtoolbox.lib.arrayops.basic.**anorm2**(*a*)

Summation of squares (helper function for [anorm\(\)](#))

**Parameters** **a** –


**Returns**

RRtoolbox.lib.arrayops.basic.**axesIntercept**(*coorSM*, *maxS*, *maxM*)

Intercept static axis (S) and mobile axis (M) with a coordinate connecting both axes from minS to minM.

S1 S2

S0  maxS

**coorSM** 

M1 M2

M0  maxM

**Parameters**

- **coorSM** – coordinate of vector from S=0 to M=0.
- **maxS** – value representing end of estatic axis.
- **maxM** – value representing end of mobile axis.

**Returns** S1,S2,M1,M2.



`RRtoolbox.lib.arrayops.basic.boxPads (bx, points)`

Get box pads to fit all.

**Parameters**

- **bx** – box coordinates or previous boxPads [left\_top, right\_bottom]
- **points** – array of points

**Returns** [(left,top),(right,bottom)] where bx and points fit.

`RRtoolbox.lib.arrayops.basic.centerM (coor, maxM)`

Center vector coor in M axis.

**Parameters**

- **coor** – coordinate of vector from S=0 to M center
- **maxM** – value representing end of mobile axis

**Returns** M centered coordinate

`RRtoolbox.lib.arrayops.basic.centerS (coor, maxS)`

Center vector coor in S axis.

**Parameters**

- **coor** – coordinate of vector from S center to M=0
- **maxS** – value representing end of estatic axis

**Returns** S centered coordinate

`RRtoolbox.lib.arrayops.basic.centerSM (coorSM, maxS, maxM)`

Center vector coorSM in both S and M axes.

**Parameters**

- **coorSM** – coordinate of vector from S to M centers.
- **maxS** – value representing end of estatic axis.
- **maxM** – value representing end of mobile axis.

**Returns** SM centered coordinate.

`RRtoolbox.lib.arrayops.basic.contours2mask (contours, shape=None, astype=<type 'bool'>)`

Creates an array with filled polygons formed by contours.

**Parameters**

- **contours** – list of contour or points forming objects
- **shape** – (None) shape of array. If None it creates an array fitted to contours.
- **astype** – (“bool”) numpy type

**Returns**

`RRtoolbox.lib.arrayops.basic.contoursArea (contours)`

Accumulates areas from list of contours.

**Parameters** **contours** – list of contours or binary array.

**Returns** area.

RRtoolbox.lib.arrayops.basic.**convertXY**(*x*, *y*, *backshape*, *foreshape*, *flag*=0, *quartile*=0, *angle*=None)

Convert absolute XY 0,0 coordinates to new system WZ.

**Parameters**

- **x** – x coordinate.
- **y** – y coordinate.
- **backshape** – shape of background image.

:param foreshape: shape of foreground image. :param flag: flag for position (default=0).

- flag==0 : foreground to left up.
- flag==1 : foreground to left down.
- flag==2 : foreground to right up.
- flag==3 : foreground to right down.
- flag==4 : foreground at center of background.
- flag==5 : XY 0,0 is at center of background.
- flag==6 : XY 0,0 is at center of foreground.
- flag==7 : XY 0,0 is at right down of foreground.

**Parameters**

- **quartile** – place Mobile image at quartile 1,2,3,4. if left quartile=0 image won't be moved.
- **angle** – angle in radians (default=None). if None it does not apply.

**Returns** W,Z

RRtoolbox.lib.arrayops.basic.**convexityRatio**(*cnt*, *hull*=None)

Ratio to test if contours are irregular

**Parameters** **cnt** – contour

:param hull:(None) convex hull :return: ratio

RRtoolbox.lib.arrayops.basic.**entropyTest**(*arr*)

Entropy test of intensity arrays. (Helper function for `entropy()`)

**Parameters** **arr** – array MxN of dim 2.

**Returns** entropy.

RRtoolbox.lib.arrayops.basic.**find\_near**(*m*, *thresh*=None, *side*=None)

helper function for findminima and findmaxima :param m: minima or maxima points :param thresh: guess or seed point :param side: left or right :return: value

RRtoolbox.lib.arrayops.basic.**findmaxima**(*hist*, *thresh*=None, *side*=None)

Get nearest peak value to a thresh point from a histogram.

**Parameters**

- **hist** – histogram
- **thresh** – initial seed
- **side** – find valley from left or right of thresh

**Returns**

RRtoolbox.lib.arrayops.basic.**findminima** (*hist, thresh=None, side=None*)

Get nearest valley value to a thresh point from a histogram.

**Parameters**

- **hist** – histogram
- **thresh** – initial seed
- **side** – find valley from left or right of thresh

**Returns**

RRtoolbox.lib.arrayops.basic.**getOtsuThresh** (*hist*)

From histogram calculate Otsu threshold value.

**Parameters** **hist** – histogram

**Returns** otsu threshold value

RRtoolbox.lib.arrayops.basic.**getTransformedCorners** (*shape, H*)

from shape gets transformed corners of array.

**Parameters**

- **shape** – H,W array shape
- **H** – transformation matrix

**Returns** upper\_left, upper\_right, lower\_right, lower\_left transformed corners.

RRtoolbox.lib.arrayops.basic.**getTransparency** (*array*)

Convert foreground to background.

**Parameters** **array** – image array.

**Returns** alfa (int or array)

RRtoolbox.lib.arrayops.basic.**get\_x\_space** (*funcs, step=10, xleft=-300, xright=300*)

get X axis space by brute force. This can be used to find the x points where the points in the y axis of any number of functions become stable.

**Parameters**

- **funcs** – list of functions
- **step** –  
10. step to close guess to maximum
- **xleft** – maximum left limit
- **xright** – maximum right limit

**Returns** linspace

RRtoolbox.lib.arrayops.basic.**getdataVH** (*array, ypad=0, xpad=0, bgrcolor=None, alfa=None*)

Get data from array according to padding (Helper function for `padVH()`).

**Parameters**

- **array** – list of arrays to get data
- **ypad** – how much to pad in y axis
- **xpad** – how much to pad in x axis

**Returns** matrix\_shapes, grid\_div, row\_grid, row\_gridpad, globalgrid

`RRtoolbox.lib.arrayops.basic.histogram`

Get image histogram.

**Parameters** `img` – gray or image with any bands

**Returns** histogram of every band

`RRtoolbox.lib.arrayops.basic.im2imFormat(src, dst)`

Tries to convert source image to destine image format.

**Parameters**

- **src** – source image.
- **dst** – destine image.

**Returns** reshaped source image.

`RRtoolbox.lib.arrayops.basic.im2shapeFormat(im, shape)`

Tries to convert image to intuited format from shape.

**Parameters**

- **im** – image.
- **shape** – shape to get format.  
shapes: \* (None, None): converts to gray \* (None, None, 2): converts to GR555 \* (None, None, 3): converts to BGR \* (None, None, 4): converts to BGRA

**Returns** reshaped image.

`RRtoolbox.lib.arrayops.basic.instability_bf(funcs, step=10, maximum=300, guess=0, tolerance=0.01)`

Find the instability of function approaching value by brute force,

**Parameters**

- **funcs** – list of functions
- **step** –  
10. step to close guess to maximum
- **maximum** –  
300. maximum value, if guess surpass this value then calculations are stopped.
- **guess** –  
0. initial guess
- **tolerance** – (0.01) tolerance with last step to check instability.

**Returns** (state, updated guess). state is True if successful, else False.

`RRtoolbox.lib.arrayops.basic.invertM(coorSM, maxM)`

Invert M axis.

**Parameters**

- **coorSM** – coordinate of vector for M inverted axes.
- **maxS** – value representing end of estatic axis.
- **maxM** – value representing end of mobile axis.

**Returns** SM coordinate on S axis and inverted M axis.

`RRtoolbox.lib.arrayops.basic.invertSM(coorSM, maxS, maxM)`

Invert S and M axes.

**Parameters**

- **coorSM** – coordinate of vector for SM inverted axes.
- **maxS** – value representing end of estatic axis.
- **maxM** – value representing end of mobile axis.

**Returns** SM coordinate on inverted SM axes.

`RRtoolbox.lib.arrayops.basic.isnumpy(arr)`

Test whether an object is a numpy array.

**Parameters** **arr** –

**Returns** True if numpy array, else false.

`RRtoolbox.lib.arrayops.basic.makeVis(globalgrid, bgrcolor=None)`

Make visualization (Helper function for [padVH\(\)](#))

**Parameters**

- **globalgrid** – shape
- **bgrcolor** – color of visualization

**Returns** array of shape globalgrid

`RRtoolbox.lib.arrayops.basic.matrixIntercept(x, y, staticm, *mobilem)`

Intercepts planes x and y of a static matrix (staticm) with N mobile matrices (mobilem) translated from the origin to x,y coordinates.

**Parameters**

- **x** – x coordinate.
- **y** – y coordinate.
- **staticm** – static matrix.
- **mobilem** – mobile matrices.

**Returns** ROI of intercepted matrices [staticm,\*mobilem].

`RRtoolbox.lib.arrayops.basic.multiple_superpose(base, fore, H, foremask=None)`

Superpose multiple foreground images to a single base image.

**Parameters**

- **base** – background, base or dipest level image (level -1)
- **fore** – foreground image list (in order of level i = 0, ... , N)
- **H** – transformation matrix of fore in level i to overlay in base
- **foremask** – foreground alfa mask in level i

**Returns** generator of each overlay

`RRtoolbox.lib.arrayops.basic.noisy(arr, mode)`

Add noise to arrays

**Parameters**

- **arr** – Input ndarray data (it will be converted to float).

- **mode** – noise method:
  - ‘gauss’ - Gaussian-distributed additive noise.
  - ‘poisson’ - Poisson-distributed noise generated from the data.
  - ‘s&p’ - Replaces random pixels with 0 or 1.
  - ‘speckle’ - **Multiplicative noise using  $out = arr + n * arr$ , where  $n$  is uniform noise with specified mean & variance.**

:return noisy arr

RRtoolbox.lib.arrayops.basic.**normalize**(arr)  
Normalize array to ensure range [0,1]

RRtoolbox.lib.arrayops.basic.**normalize2**(arr)  
Normalize with factor of absolute maximum value.

RRtoolbox.lib.arrayops.basic.**normalizeCustom**(arr, by=<function amax>, axis=None)  
Normalize array with custom operations.

#### Parameters

- **arr** – array (it does not correct negative values, use preferable NxM).
- **by** – np.max, np.sum or any function that gets an array to obtain factor.
- **axis** – if None it normalizes in all axes else in the selected axis.

**Returns** normalized to with factor.

RRtoolbox.lib.arrayops.basic.**overlay**(back, fore, alpha=None, alfainverted=False, under=False, flag=0)

Try to Overlay any dimension array.

#### Parameters

- **back** – BGRA background image.
- **fore** – BGRA foreground image.
- **alpha** – transparency channel.
- **alfainverted** – if True inverts alpha transparency.
- **under** – if True, place back as fore and fore as back.
- **flag** – (experimental)
  0. **Normally replace inverted transparency of alpha in back (N)**; superpose alpha in back (V).
  1. **Bloat and replace inverted transparency of alpha in back**; superpose bgr in back (V).
  2. Superpose inverted transparent COLOR of alpha in back.
  3. Superpose inverted transparent COLOR of alpha in back.
  4. **Superpose transparent of alpha in back**; superpose transparent COLOR of alpha in back.
  5. **Superpose transparent of alpha in back**; superpose transparent COLOR of alpha in back.

**Returns** overlaid array

See also:

`overlay2()`

`RRtoolbox.lib.arrayops.basic.overlay2(back,fore)`

Overlay foreground to x,y coordinates in background image.

#### Parameters

- **back** – background image (numpy array dim 3).
- **fore** – foreground image (numpy array dim 4). the fourth dimension is used for transparency.

**Returns** back (with overlay).

#Example:

```
import cv2
import numpy as np
import time
a= time.time()
back = cv2.imread("t1.jpg")
temp = back.shape
bgr = np.zeros((temp[0],temp[1],4), np.uint8)
points = [(86, 162), (1219, 1112), (2219, 2112), (1277,3000), (86, 162)]
col_in = (0, 0, 0,255)
thickness = 10
for i in range(len(points)-1):
    pt1 = (points[i][0], points[i][1])
    pt2 = (points[i+1][0], points[i+1][1])
    cv2.line(bgr, pt1, pt2, col_in, thickness)

overlay(back,bgr)

win = "overlay"
cv2.namedWindow(win,cv2.WINDOW_NORMAL)
cv2.imshow(win, back)
print time.time()-a
cv2.waitKey()
cv2.destroyAllWindows()
```

See also:

`overlay()`

`RRtoolbox.lib.arrayops.basic.overlayXY(x, y, back, fore, alfa=None, alfainverted=False, under=False, flag=0)`

Overlay foreground image to x,y coordinates in background image. This function support images of different sizes with formats: BGR background and BGRA foreground of Opencv or numpy images.

#### Parameters

- **x** – x position in background.
- **y** – y position in background.
- **back** – background image (numpy array dim 3).
- **fore** – foreground image (numpy array dim 4). the fourth dimension is used for transparency.

**Returns** back (with overlay)

Example:

```
import cv2
back = cv2.imread("t1.jpg")
bgr = cv2.imread("mustache.png",-1)
x,y=convertXY(0,0,back.shape,bgr.shape,flag=1)
overlayXY(x,y,back,bgr)
win = "overlay"
cv2.namedWindow(win,cv2.WINDOW_NORMAL)
cv2.imshow(win, back)
cv2.waitKey()
cv2.destroyAllWindows()
```

RRtoolbox.lib.arrayops.basic.**overlaypng**(*back, fore, alpha=None, alfainverted=False, under=False, flag=0*)

Overlay only BGRA.

#### Parameters

- **back** – BGRA background image.
- **fore** – BGRA foreground image.
- **alpha** – transparency channel.
- **alfainverted** – if True inverts alpha transparency.
- **under** – if True, place back as fore and fore as back.
- **flag** – (experimental)
  0. **Normally replace inverted transparency of alpha in back (N)**; superpose alpha in back (V).
  1. **Bloat and replace inverted transparency of alpha in back**; superpose bgr in back (V).
  2. Superpose inverted transparent COLOR of alpha in back.
  3. Superpose inverted transparent COLOR of alpha in back.
  4. **Superpose transparent of alpha in back**; superpose transparent COLOR of alpha in back.
  5. **Superpose transparent of alpha in back**; superpose transparent COLOR of alpha in back.

**Returns** overlaid array

**See also:**

`overlay()`, `overlay2()`

RRtoolbox.lib.arrayops.basic.**padvH**(*imgs, ypad=0, xpad=0, bgrcolor=None, alfa=None*)  
Pad Vertically and Horizontally image or group of images into an array.

#### Parameters

- **imgs** – image to pad or list of horizontal images (i.e. piled up horizontally as [V1,...,VN] where each can be a list of vertical piling VN = [H1,...,HM]. It can be successive like horizontals, verticals, horizontals, etc.
- **ypad** – padding in axis y
- **xpad** – padding in axis x
- **bgrcolor** – color of spaces



- **alfa** – transparency of imgs over background of bgrcolor color.

**Returns** visualization of padded and piled images in imgs.

`RRtoolbox.lib.arrayops.basic.pad_to_fit_H(shape1, shape2, H)`  
get boxPads to fit transformed shape1 in shape2.

**Parameters**

- **shape1** – shape of array 1
- **shape2** – shape of array 2
- **H** – transformation matrix to use in shape1

**Returns** [(left,top),(right,bottom)]

`RRtoolbox.lib.arrayops.basic.points2mask(pts, shape=None, astype=<type 'bool'>)`  
Creates an array with the filled polygon formed by points.

**Parameters**

- **pts** – points.
- **shape** – (None) shape of array. If None it creates an array fitted to points.
- **astype** – (“bool”) numpy type

**Returns** array.

Example:

```
pts = random_points([(-100, 100), (-100, 100)])
img = points2mask(pts)
Plotim("filled", img).show()
```

`RRtoolbox.lib.arrayops.basic.points_generator(shape=(10, 10), noints=None, convex=False, erratic=False, complete=False)`

generate points.

**Parameters**

- **shape** – enclosed frame (width, height)
- **noints** – number of points
- **convex** – if True make points convex, else points follow a circular pattern.

**Returns**

`RRtoolbox.lib.arrayops.basic.polygonArea(pts)`  
Area of points calculating polygon Area.

**Parameters** **pts** – points.

**Returns** area value.

**..note::**

- If polygon is incomplete (last is not first point) it completes the array.
- If the polygon crosses over itself the algorithm will fail.
- Based on <http://www.mathopenref.com/coordpolygonarea.html>

`RRtoolbox.lib.arrayops.basic.polygonArea_calcule` (*pts*)

Area of points calculating polygon Area.

**Parameters** *pts* – points.

**Returns** area value.

**..note::**

- If polygon is incomplete (last is not first point) it completes the array.
- If the polygon crosses over itself the algorithm will fail.
- Based on <http://www.mathopenref.com/coordpolygonarea.html>

`RRtoolbox.lib.arrayops.basic.polygonArea_contour` (*pts*)

Area of points using contours.

**Parameters** *pts* – points.

**Returns** area value.

**..note::** if polygon is incomplete (last is not first point) it completes the array.

`RRtoolbox.lib.arrayops.basic.polygonArea_fill` (*pts*)

Area of points using filled polygon and pixel count.

**Parameters** *pts* – points.

**Returns** area value.

**..note::** if polygon is incomplete (last is not first point) it completes the array.

`RRtoolbox.lib.arrayops.basic.process_as_blocks` (*arr*, *func*, *block\_shape*=(3, 3),  
*mask*=None, *asWindows*=False)

process with function over an array using blocks (using re-striding).

**Parameters**

- **arr** – array to process
- **func** – function to feed blocks
- **block\_shape** – (3,3) shape of blocks
- **mask** – (None) mask to process arr
- **asWindows** – (False) if True all blocks overlap each other to give a result for each position of arr, if False the results are given in blocks equivalent for each processed blocks of arr (faster).

**Returns** processed array.

`RRtoolbox.lib.arrayops.basic.quadrant` (*coorX*, *coorY*, *maxX*, *maxY*, *quadrant*=0)

Moves a point to a quadrant

**Parameters**

- **coorX** – point in x coordinate
- **coorY** – point in y coordinate
- **maxX** – max value in x axis
- **maxY** – max value in y axis
- **quadrant** – Cartesian quadrant, if 0 or False it leaves coorX and coorY unprocessed.

**Returns**

RRtoolbox.lib.arrayops.basic.**random\_points** (*axes\_range*=((-50, 50), ), *nopoints*=4, *complete*=False)

Get random points.

**Parameters**

- **axes\_range** – [x\_points\_range, y\_points\_range] where points\_range is (min,max) range in axis.
- **nopoints** – number of points.
- **complete** – last point is the first point (adds an additional point i.e. nopoints+1).

**Returns** numpy array.

RRtoolbox.lib.arrayops.basic.**recursiveMap** (*function*, *sequence*)

Iterate recursively over a structure using a function.

**Parameters**

- **function** – function to apply
- **sequence** – iterator

**Returns**

RRtoolbox.lib.arrayops.basic.**relativeQuadrants** (*points*)

Get quadrants of relative vectors obtained from points.

**Parameters** **points** – array of points.

**Returns** quadrants.

RRtoolbox.lib.arrayops.basic.**relativeVectors** (*pts*, *all*=True)

Form vectors from points.

**Parameters**

- **pts** – array of points [p0, ... ,(x,y)].
- **all** – (True) if True adds last vector from last and first point.

**Returns** array of vectors [V0, ... , (V[n] = x[n+1]-x[n],y[n+1]-y[n])].

RRtoolbox.lib.arrayops.basic.**rescale** (*arr*, *max*=1, *min*=0)

Rescales array values to range [min,max].

**Parameters**

- **arr** – array.
- **max** – maximum value in range.
- **min** – minimum value in range.

**Returns** rescaled array.

RRtoolbox.lib.arrayops.basic.**separePointsByAxis** (*pts*, *ptaxis*=(1, 0), *origin*=(0, 0))

Separate scattered points with respect to axis (splitting line).

**Parameters**

- **pts** – points to separate.
- **ptaxis** – point to form axis from origin
- **origin** – origin

**Returns** left, right points from axis.

`RRtoolbox.lib.arrayops.basic.splitPoints (pts, aslist=None)`  
from points get x,y columns

**Parameters**

- **pts** – array of points
- **aslist** – True to return lists instead of arrays

**Returns** x, y columns

`RRtoolbox.lib.arrayops.basic.standardizePoints (pts, aslist=False)`  
converts points to a standard form :param pts: list or array of points :param aslist: True to return list instead of array :return: standard points

`RRtoolbox.lib.arrayops.basic.superpose (back, fore, H, foreMask=None, grow=True)`  
Superpose foreground image to background image.

**Parameters**

- **back** – background image
- **fore** – foreground image
- **H** – transformation matrix of fore to overlay in back
- **foreMask** – (None) foreground alpha mask, None or function. foreMask values are from 1 for solid to 0 for transparency. If a function is provided the new back,fore parameters are provided to produce the foreMask. If None is provided as foreMask then it is equivalent to a foreMask with all values to 1 where fore is True.
- **grow** – If True, im can be bigger than back and is calculated according to how fore is superposed in back; if False im is of the same shape as back.

**Returns** im, H\_back, H\_fore

`RRtoolbox.lib.arrayops.basic.transformPoint (p, H)`  
Transform individual x,y point with Transformation Matrix.

**Parameters**

- **p** – x,y point
- **H** – transformation matrix

**Returns** transformed x,y point

`RRtoolbox.lib.arrayops.basic.transformPoints (p, H)`  
Transform x,y points in array with Transformation Matrix.

**Parameters**

- **p** – array of points
- **H** – transformation matrix

**Returns** transformed array of x,y point

`RRtoolbox.lib.arrayops.basic.unit_vector (vector)`  
Returns the unit vector of the vector.

`RRtoolbox.lib.arrayops.basic.vectorsAngles (pts, ptaxis=(1, 0), origin=(0, 0), dtype=None, deg=False, absolute=None)`

Angle of formed vectors in Cartesian plane with respect to formed axis vector.

i.e. angle between vector “Vn” (formed by point “Pn” and the “origin”)

and vector “Vaxis” formed by “ptaxis” and the “origin”.

where  $\text{pts-origin} = (\text{P0-origin} \dots \text{Pn-origin}) = \text{V0} \dots \text{Vn}$

#### Parameters

- **pts** – points to form vectors from origin
- **ptaxis** – point to form axis from origin
- **origin** – origin
- **dtype** – return array of type supported by numpy.
- **deg** – if True angle is in Degrees, else radians.
- **absolute** – if None returns angles (0 to 180(pi)) between pts-origin (V0 .. Vn) and Vaxis. if True returns any Vn absolute angle (0 to 360(2pi)) from Vaxis as axis to Vn. if False returns any Vn angle (0 to 180 or 0 to -180) from Vaxis as axis to Vn, where any Vn angle is positive or negative if counter-clock or clock wise from Vaxis.

#### Returns

`RRtoolbox.lib.arrayops.basic.vectorsQuadrants (vecs)`

Get quadrants of vectors.

**Parameters** **vecs** – array of vectors.

**Returns** quadrants.

`RRtoolbox.lib.arrayops.basic.vertexesAngles (pts, dtype=None, deg=False)`

Relative angle of vectors formed by vertexes (where vectors cross).

**i.e. angle between vectors “v01” formed by points “p0-p1” and “v12” formed by points “p1-p2” where “p1” is seen as a vertex (where vectors cross).**

#### Parameters

- **pts** – points seen as vertexes (vectors are recreated from point to point).
- **dtype** – return array of type supported by numpy.
- **deg** – if True angle is in Degrees, else radians.

**Returns** angles.

`RRtoolbox.lib.arrayops.basic.view_as_blocks (arr_in, block_shape=(3, 3))`

Provide a 2D block\_shape view to 2D array. No error checking made. Therefore meaningful (as implemented) only for blocks strictly compatible with the shape of arr\_in.

#### Parameters

- **arr\_in** –
- **block\_shape** –

#### Returns

`RRtoolbox.lib.arrayops.basic.view_as_windows (arr_in, window_shape, step=1)`

Provide a 2D block\_shape rolling view to 2D array. No error checking made. Therefore meaningful (as implemented) only for blocks strictly compatible with the shape of arr\_in.

#### Parameters

- **arr\_in** –

- **window\_shape** –
- **step** –

**Returns**

**RRtoolbox.lib.arrayops.convert module** This module unlike common and basic array operations classifies just the from-to-conversions methods

**class** `RRtoolbox.lib.arrayops.convert.SimKeyPoint` (\*args)  
Simulates opencv keypoint (it allows manipulation, conversion and serialization of keypoints).

---

**Note:** Used for conversions and data persistence.

---

`RRtoolbox.lib.arrayops.convert.apply2kp_pairs` (kp\_pairs, kp1\_rel, kp2\_rel, func=None)  
Apply to kp\_pairs.

**Parameters**

- **kp\_pairs** – list of (kp1,kp2) pairs
- **kp1\_rel** – x,y relation or function to apply to kp1
- **kp2\_rel** – x,y relation or function to apply to kp2
- **func** – function to build new copy of keypoint

**Returns** transformed kp\_pairs

`RRtoolbox.lib.arrayops.convert.cnt2pts` (contours)  
Convert contours to points. (cnt2pts)

**Parameters** **contours** – array of contours (cnt) ([[x,y]] only for openCV)

**Returns**

Example:

```
contours = np.array([[0, 0]], [[1, 0]]) # contours
points = contour2points(contours)
print points # np.array([[0, 0], [1, 0]])
```

`RRtoolbox.lib.arrayops.convert.contour2points` (contours)  
Convert contours to points. (cnt2pts)

**Parameters** **contours** – array of contours (cnt) ([[x,y]] only for openCV)

**Returns**

Example:

```
contours = np.array([[0, 0]], [[1, 0]]) # contours
points = contour2points(contours)
print points # np.array([[0, 0], [1, 0]])
```

`RRtoolbox.lib.arrayops.convert.conv3H4H` (M)  
Convert a 3D transformation matrix (TM) to 4D TM.

**Parameters** **M** – Matrix

**Returns** 4D Matrix

---

```
RRtoolbox.lib.arrayops.convert.dict2keyPoint (d, func=<built-in function KeyPoint>)
    KeyPoint([x, y, _size[, _angle[, _response[, _octave[, _class_id]]]]) -> <KeyPoint object>
```

```
RRtoolbox.lib.arrayops.convert.getSOpintRelation (source_shape, destine_shape, as-
    Matrix=False)
```

Return parameters to change scaled point to original point.

# destine\_domain = relation\*source\_domain

#### Parameters

- **source\_shape** – image shape for source domain
- **destine\_shape** – image shape for destine domain
- **asMatrix** – if true returns a Transformation Matrix H

**Returns** x, y coordinate relations or H if asMatrix is True

---

**Note:** Used to get relations to convert scaled points to original points of an Image.

---

```
RRtoolbox.lib.arrayops.convert.invertH (H)
    Invert Transformation Matrix.
```

**Parameters** **H** –

**Returns**

```
RRtoolbox.lib.arrayops.convert.keyPoint2tuple (keypoint)
    obj.angle, obj.class_id, obj.octave, obj.pt, obj.response, obj.size
```

```
RRtoolbox.lib.arrayops.convert.points2contour (points)
    Convert points to contours. (pts2cnt)
```

**Parameters** **points** – array of points ([x,y] for openCV, [y,x] for numpy)

**Returns**

Example:

```
points = np.array([[0, 0], [1, 0]]) # points
contours = points2contour(points)
print contours # np.array([[0, 0], [[1, 0]]])
```

```
RRtoolbox.lib.arrayops.convert.points2vectors (pts, origin=None)
    Convert points to vectors with respect to origin.
```

**Parameters**

- **pts** – array of points.
- **origin** – point of origin.

**Returns** vectors.

```
RRtoolbox.lib.arrayops.convert.pts2cnt (points)
    Convert points to contours. (pts2cnt)
```

**Parameters** **points** – array of points ([x,y] for openCV, [y,x] for numpy)

**Returns**

Example:

```
points = np.array([[0, 0], [1, 0]]) # points
contours = points2contour(points)
print contours # np.array([[[0, 0]], [[1, 0]])
```

RRtoolbox.lib.arrayops.convert.**sh2oh**(*sH, osrc\_sh, sscr\_sh, odst\_sh, sdst\_sh*)  
Convert scaled transformation matrix (sH) to original (oH).

**Parameters**

- **sH** – scaled transformation matrix
- **osrc\_sh** – original source’s shape
- **sscr\_sh** – scaled source’s shape
- **odst\_sh** – original destine’s shape
- **sdst\_sh** – scaled destine’s shape

**Returns**

RRtoolbox.lib.arrayops.convert.**spairs2opairs**(*kp\_pairs, osrc\_sh, sscr\_sh, odst\_sh, sdst\_sh, func=None*)

Convert scaled kp\_pairs to original kp\_pairs.

**Parameters**

- **kp\_pairs** – list of kp\_pairs
- **osrc\_sh** – original source’s shape
- **sscr\_sh** – scaled source’s shape
- **odst\_sh** – original destine’s shape
- **sdst\_sh** – scaled destine’s shape
- **func** – function to build new copy of keypoint

**Returns**

RRtoolbox.lib.arrayops.convert.**spoint2opointfunc**(*source\_shape, destine\_shape*)  
Return function with parameters to change scaled point to original point.

**Parameters**

- **source\_shape** –
- **destine\_shape** – shape of

**Returns**

Example:

```
forefunc = scaled2realfunc(imgf.shape,bgr.shape)
backfunc = scaled2realfunc(imgb.shape,back.shape)
p1fore = np.array([forefunc(i) for i in p1])
p2back = np.array([backfunc(i) for i in p2])
```

RRtoolbox.lib.arrayops.convert.**toTuple**(*obj*)  
Converts recursively to tuple

**Parameters** **obj** – numpy array, list structure, iterators, etc.

**Returns** tuple representation obj.



```
RRtoolbox.lib.arrayops.convert.translateQuadrants (quadrants, quadrantmap={(0, 1):
    'up', (-1, 1): 'left-up', (0, 0): 'ori-
    gin', (-1, 0): 'left', (-1, -1): 'left-
    down', (0, -1): 'down', (1, 0):
    'right', (1, -1): 'right-down', (1, 1):
    'right-up'})
```

Convert quadrants into human readable data.

#### Parameters

- **quadrants** – array of quadrants.
- **quadrantmap** – dictionary map to translate quadrants. it is of the form:

```
{(0,0):"origin", (1,0):"right", (1,1):"top-right", (0,1):"top", (-1,1):"top-left",
  (-1,0):"left",(-1,-1):"bottom-left",(0,-1):"bottom", (1,-1):"bottom-right"}
```

**Returns** list of translated quadrants.

```
RRtoolbox.lib.arrayops.convert.tuple2keyPoint (points, func=<built-in function Key-
    Point>)
    KeyPoint([x, y, _size[, _angle[, _response[, _octave[, _class_id]]]]) -> <KeyPoint object>
```

```
RRtoolbox.lib.arrayops.convert.vectors2points (vecs, origin=None)
    Convert points to vectors with respect to origin.
```

#### Parameters

- **vecs** – array of vectors.
- **origin** – point of origin.

**Returns** points.

**RRtoolbox.lib.arrayops.filters module** This module contains custom 1D and 2D-array filters and pre-processing (as in filtering phase) methods

```
class RRtoolbox.lib.arrayops.filters.Bandpass (alpha, beta1, beta2)
```

Bases: `RRtoolbox.lib.arrayops.filters.FilterBase`

Bandpass filter (recommended to use float types)

```
class RRtoolbox.lib.arrayops.filters.Bandstop (alpha, beta1, beta2)
```

Bases: `RRtoolbox.lib.arrayops.filters.FilterBase`

Bandstop filter (recommended to use float types)

```
class RRtoolbox.lib.arrayops.filters.BilateraParameter (scale, shift=33, name=None,
    alpha=100, beta1=-400,
    beta2=200)
```

Bases: `RRtoolbox.lib.arrayops.filters.Bandstop`

bilateral parameter

```
class RRtoolbox.lib.arrayops.filters.BilateralParameters (d=None, sigmaColor=None,
    sigmaSpace=None)
```

Bases: `object`

create instance to calculate bilateral parameters from image shape.

**d -> inf** then:

- computation is slower
- filtering is better to eliminate noise
- images look more cartoon-like

#### Parameters

- **d** – distance
- **sigmaColor** – sigma in color
- **sigmaSpace** – sigma in space

**d** = <RRtoolbox.lib.arrayops.filters.BilateraParameter object>

**filters**  
list of filters

**sigmaColor** = <RRtoolbox.lib.arrayops.filters.BilateraParameter object>

**sigmaSpace** = <RRtoolbox.lib.arrayops.filters.BilateraParameter object>

**class** RRtoolbox.lib.arrayops.filters.**FilterBase** (*alpha=None, beta1=None, beta2=None*)

Bases: object

base filter to create custom filters

**alpha**

**beta1**

**beta2**

**class** RRtoolbox.lib.arrayops.filters.**Highpass** (*alpha, beta1*)

Bases: *RRtoolbox.lib.arrayops.filters.FilterBase*

Highpass filter (recommended to use float types)

**class** RRtoolbox.lib.arrayops.filters.**InvertedBandpass** (*alpha, beta1, beta2*)

Bases: *RRtoolbox.lib.arrayops.filters.Bandpass*

inverted Bandpass filter (recommended to use float types)

**class** RRtoolbox.lib.arrayops.filters.**InvertedBandstop** (*alpha, beta1, beta2*)

Bases: *RRtoolbox.lib.arrayops.filters.Bandstop*

inverted Bandstop filter (recommended to use float types)

**class** RRtoolbox.lib.arrayops.filters.**Lowpass** (*alpha, beta1*)

Bases: *RRtoolbox.lib.arrayops.filters.FilterBase*

Lowpass filter (recommended to use float types)

RRtoolbox.lib.arrayops.filters.**bilateralFilter** (*im, d, sigmaColor, sigmaSpace*)

Apply bilateral Filter.

#### Parameters

- **im** –
- **d** –
- **sigmaColor** –
- **sigmaSpace** –

**Returns** filtered image

RRtoolbox.lib.arrayops.filters.**filterFactory** (*alpha*, *beta1*, *beta2=None*)

Make filter.

#### Parameters

- **alpha** – steepness of filter
  - **beta1** – first shift from origin
  - **beta2** – second shift from origin:  
alpha must be != 0 if beta2 = None:  
if alpha > 0: high-pass filter, if alpha < 0: low-pass filter
- else:
- **if beta2 > beta1:** if alpha > 0: band-pass filter, if alpha < 0: band-stop filter
  - **else:** if alpha > 0: inverted-band-pass filter, if alpha < 0: inverted-band-stop filter

**Returns** filter funtion with intup levels

Example:

```
alpha,beta1,beta2 = 10,20,100
myfilter = filter(alpha,beta1,beta2)
print myfilter,type(myfilter)
print myfilter.alpha,myfilter.beta1,myfilter.beta2
```

RRtoolbox.lib.arrayops.filters.**getBilateralParameters** (*shape=None*, *mode=None*)

Calculate from shape bilateral parameters.

#### Parameters

- **shape** – image shape. if None it returns the instace to use with shapes.
- **mode** – “mild”, “heavy” or “normal” to process noise

**Returns** instance or parameters

RRtoolbox.lib.arrayops.filters.**normsigmoid** (*x*, *alpha*, *beta*)

Apply normalized sigmoid filter.

#### Parameters

- **x** – data to apply filter
- **alpha** – if alpha > 0: pass high filter, if alpha < 0: pass low filter, alpha must be != 0
- **beta** – shift from origin

**Returns** filtered values normalized to range [-1 if x<0, 1 if x>=0]

RRtoolbox.lib.arrayops.filters.**sigmoid** (*x*, *alpha*, *beta*, *max=255*, *min=0*)

Apply sigmoid filter.

#### Parameters

- **x** – data to apply filter
- **alpha** – if alpha > 0: pass high filter, if alpha < 0: pass low filter, alpha must be != 0
- **beta** – shift from origin
- **max** – maximum output value
- **min** – minimum output value

**Returns** filtered values ranging as [min,max]

---

**Note:** Based from [http://www.itk.org/Doxygen/html/classitk\\_1\\_1SigmoidImageFilter.html](http://www.itk.org/Doxygen/html/classitk_1_1SigmoidImageFilter.html)

---

RRtoolbox.lib.arrayops.filters.**smooth**(x, window\_len=11, window='hanning',  
rect=False)

Smooth the data using a window with requested size.

This method is based on the convolution of a scaled window with the signal. The signal is prepared by introducing reflected copies of the signal (with the window size) in both ends so that transient parts are minimized in the beginning and end part of the output signal.

**input:** x: the input signal window\_len: the dimension of the smoothing window; should be an odd integer  
window: the type of window from 'flat', 'hanning', 'hamming', 'bartlett', 'blackman'

flat window will produce a moving average smoothing.

**output:** the smoothed signal

**Example:**

```
t=linspace(-2,2,0.1)
x=sin(t)+randn(len(t))*0.1
y=smooth(x)
```

**See also:**

numpy.hanning, numpy.hamming, numpy.bartlett, numpy.blackman, numpy.convolve, scipy.signal.lfilter

---

**Note:** length(output) != length(input), to correct this: return y[(window\_len/2-1):-(window\_len/2)] instead of just y.

---

**RRtoolbox.lib.arrayops.mask module** This module contains all basic masking and pre-processing (as in segmenting phase) methods

RRtoolbox.lib.arrayops.mask.**background**(gray, mask=None, iterations=3)  
get the background mask of a gray image. (this is the inverted of *foreground()*)

**Parameters**

- **gray** – gray image
- **mask** – (None) input mask to process gray
- **iterations** – (3) number of iterations to detect background with otsu threshold.

**Returns** output mask

RRtoolbox.lib.arrayops.mask.**biggestCnt**(contours)  
Filters contours to get biggest contour.

**Parameters** contours –

**Returns** cnt

RRtoolbox.lib.arrayops.mask.**biggestCntData**(contours)  
Gets index and area of biggest contour.

**Parameters** contours –

**Returns** index, area

`RRtoolbox.lib.arrayops.mask.brightness (img)`  
get brightness from an image :param img: BGR or gray image :return:

`RRtoolbox.lib.arrayops.mask.cnt_hist (gray)`  
Mask of a ellipse enclosing retina using histogram threshold.

**Parameters**

- **gray** – gray image
- **invert** – invert mask

**Returns** mask

`RRtoolbox.lib.arrayops.mask.foreground (gray, mask=None, iterations=3)`  
get the foreground mask of a gray image. (this it the inverted of `background()`)

**Parameters**

- **gray** – gray image
- **mask** – (None) input mask to process gray
- **iterations** – (3) number of iterations to detect foreground with otsu threshold.

**Returns** output mask

`RRtoolbox.lib.arrayops.mask.gethull (contours)`  
Get convex hull.

**Parameters** **contours** – contours or mask array

**Returns** cnt

`RRtoolbox.lib.arrayops.mask.hist_cdf (img, window_len=0, window='hanning')`  
Get image histogram and the normalized cumulative distribution function.

**Parameters**

- **img** – imaeg
- **window\_len** –
- **window** –

**Returns** histogram (int), normalized cdf (float)

`RRtoolbox.lib.arrayops.mask.mask_watershed (BGR, GRAY=None)`  
Get retinal mask with watershed method.

**Parameters**

- **BGR** –
- **GRAY** –

**Returns** mask

`RRtoolbox.lib.arrayops.mask.multiple_otsu (gray, mask=None, flag=0L, iterations=1)`  
get the mask of a gray image applying Otsu threshold.

**Parameters**

- **gray** – gray image
- **mask** – (None) input mask to process gray
- **iterations** –

1. number of iterations to detect Otsu threshold.

**Returns** thresh, mask

`RRtoolbox.lib.arrayops.mask.thresh_biggestCnt(thresh)`

From threshold obtain biggest contour.

**Parameters** `thresh` – binary image

**Returns** cnt

`RRtoolbox.lib.arrayops.mask.thresh_hist(gray)`

Get best possible thresh to threshold object from the gray image.

**Parameters** `gray` – gray image.

**Returns** thresh value.

`RRtoolbox.lib.arrayops.mask.threshold_opening(src, thresh, maxval, type)`

Eliminate small objects from threshold.

**Parameters**

- `src` –
- `thresh` –
- `maxval` –
- `type` –

**Returns**

## Module contents

## Submodules

### RRtoolbox.lib.cache module

**platform** Unix, Windows

**synopsis** Serialize and Memoize.

Contains memoizing, caching, serializing and memory-mapping methods so as to let the package save its state (persistence) and to let a method “remember” what it processed in a session (with cache) or between sessions (memoization and serialization) of the same input contend once processed. It also wraps mmapping functions to let objects “live” in the disk (slower but almost unlimited) rather than in memory (faster but limited).

`@cache` is used as replacement of `@property` to compute a class method once. It is computed only one time after which an attribute of the same name is generated in its place.

`@cachedProperty` is used as replacement of `@property` to compute a class method depending on changes in its watched variables.

`@memoize` used as a general memoizer decorator for functions where metadata is generated to disk for persistence.

Made by Davtohi, powered by joblib. Dependent project: <https://github.com/joblib/joblib>

**class** `RRtoolbox.lib.cache.Cache(func)`

Bases: `object`

Descriptor (non-data) for building an attribute on-demand at first use. `@cache` decorator is used for class methods without inputs (only self reference to the object) and it caches on first compute. ex:

```
class x(object):
    @cache
    def method_x(self):
        return self.data
```

---

**Note:** Cached data can be deleted in the decorated object to recalculate its value.

---

```
class RRtoolbox.lib.cache.DynamicMemoizedFunc(func, cachedir=None, ignore=None,
                                              mmap_mode=None, compress=False, ver-
                                             bose=1, timestamp=None, banned=False)
```

Bases: object

**cachedir**

**call\_and\_shelve** (\*args, \*\*kwargs)

**clear** (warn=True)

**compress**

**enabled**

**func**

**ignore**

**mmap\_mode**

**verbose**

```
class RRtoolbox.lib.cache.LazyDict(getter, dictionary=None)
```

Bases: `_abcoll.MutableMapping`

Create objects on demand if needed. call the instance with keys to prevent it from using lazy evaluations (e.g. instead of `self[key]` use `self(key)` to prevent recursion). Containing operations are safe to prevent recursion (e.g. if `key` in `self` instead of `self[key]`). In addition use `self.isLazy` flag to enable or disable lazy operations to prevent possible recursions when `getter` is called.

```
class RRtoolbox.lib.cache.MemoizedDict(path, mode=None)
```

Bases: `_abcoll.MutableMapping`

memoized dictionary with keys and values persisted to files.

#### Parameters

- **path** – path to save memo file
- **mode** – loading mode from memo file {None, 'r+', 'r', 'w+', 'c'}

**Warning:** Some data structures cannot be memoize, so this structure is not save yet. Use at your own risk.

**clear** ()

Remove all items from D.

```
class RRtoolbox.lib.cache.Memoizer(ignore=(), ignoreAll=False)
```

Bases: object

**ignore**

**makememory** (cachedir=None, mmap\_mode=None, compress=False, verbose=0)

Make memory for `memoize()` decorator.

**Parameters**

- **cachedir** – path to save metadata, if left None function is not cached.
- **mmap\_mode** – {None, 'r+', 'r', 'w+', 'c'}, optional. The memmapping mode used when loading from cache numpy arrays. See `numpy.load` for the meaning of the arguments.
- **compress** – (boolean or integer) Whether to zip the stored data on disk. If an integer is given, it should be between 1 and 9, and sets the amount of compression. Note that compressed arrays cannot be read by memmapping.
- **verbose** – (int, optional) Verbosity flag, controls the debug messages that are issued as functions are evaluated.

**Returns**

**memoize** (*memory=None, ignore=None, verbose=0, mmap\_mode=False*)

Decorated functions are faster by trading memory for time, only hashable values can be memoized.

**Parameters**

- **memory** – (Memory or path to folder) if left None function is not cached.
- **ignore** – (list of strings) A list of arguments name to ignore in the hashing.
- **verbose** – (integer) Verbosity flag, controls the debug messages that are issued as functions are evaluated.
- **mmap\_mode** – {None, 'r+', 'r', 'w+', 'c'}, optional. The memmapping mode used when loading from cache numpy arrays. See `numpy.load` for the meaning of the arguments.

**Returns** decorator

**memoizers** = {140410707069776: <weakref at 0x7fb3ea29eaf8; to 'Memoizer' at 0x7fb3ea514b50>}

**class** RRtoolbox.lib.cache.**MemorizedFunc** (*func, cachedir, ignore=None, mmap\_mode=None, compress=False, verbose=1, timestamp=None*)

Bases: `joblib.memory.MemorizedFunc`

**class** RRtoolbox.lib.cache.**Memory** (*cachedir, mmap\_mode=None, compress=False, verbose=1*)

Bases: `joblib.memory.Memory`

A wrapper to `joblib.Memory` to have better control.

**class** RRtoolbox.lib.cache.**NotMemorizedFunc** (*func*)

Bases: `joblib.memory.NotMemorizedFunc`

**class** RRtoolbox.lib.cache.**ObjectGetter** (*callfunc=None, obj=None, callback=None, \*\*annotations*)

Bases: `object`

Creates or get instance object depending if it is alive.

**create** (*throw=False*)

Creates an object and keep reference.

**Parameters** **throw** – if there is not creation function throws error.

**Returns** created object.

**Warning:** previous object reference is lost even if it was alive.

---

**Note:** Recommended only to use when object from current reference is dead.

---



**getObj** (*throw=False*)

**isAlive** ()

test if object of reference is alive

**isCreatable** ()

test if can create object

**isGettable** ()

test if object can be gotten either by reference or creation.

**raw** ()

get object from reference. :return: None if object is dead, object itself if is alive.

**update** (*\*\*kwargs*)

**class** `RRtoolbox.lib.cache.ResourceManager` (*maxMemory=None, margin=0.8, unit='MB', all=True*)

Bases: `RRtoolbox.lib.cache.Retriever`

keep track of references, create objects on demand, manage their memory and optimize for better performance.

#### Parameters

- **maxMemory** – (None) max memory in specified unit to keep in check optimization (it does not mean that memory never surpasses maxMemory).
- **margin** – (0.8) margin from maxMemory to trigger optimization. It is in percentage of maxMemory ranging from 0 (0%) to maximum 1 (100%). So optimal memory is inside range:  $\text{maxMemory} * \text{margin} < \text{Memory} < \text{maxMemory}$
- **unit** – (MB) maxMemory unit, it can be GB (Gigabytes), MB (Megabytes), B (bytes)
- **all** – if True used memory is from all alive references, if False used memory is only from keptAlive references.

**all**

**Returns** all flag, if True: used memory is from all alive references, if False: used memory is only from keptAlive references.

**bytes2units** (*value*)

converts value from bytes to user units

**getSizeOf** (*item*)

**keepAlive** (*key, obj*)

**margin**

**Returns** margin used for triggering memory optimization from maxMemory.

**maxMemory**

**optimizeObject** (*key, getter, toWhiteList=False*)

**register** (*key, method=None, instance=None*)

Register object to retrieve.

#### Parameters

- **key** – hashable key to retrieve
- **method** – callable method to get object
- **instance** – object instance already created from method

---

**Note:** This method is used in `__setitem__` as `self.register(key, value)`. Overwrite this method to change key assignation behaviour.

---

Example:

```
def mymethod():
    class constructor: pass
    return constructor()

ret = retriever()
ret["obj"] = mymethod # register creating method in "obj"
im = ret["obj"] # get object (created obj +1, with reference)
assert im is ret["obj"] # check that it gets the same object
# it remembers that "obj" is last registered or fetched object too
assert ret() is ret()
# lets register with better control (created obj2 +1, no reference)
ret.register("obj2",mymethod(),mymethod)
# proves that obj2 is not the same as obj (created obj2 +1, no reference)
assert ret() is not ret["obj"]
print list(ret.iteritems()) # get items
```

**static resetGetter** (*getter*)

Helper function to reset getter parameters.

**Parameters** **getter** – any instance of objectGetter

**unit**

**Returns** user defined units

**units2bytes** (*value*)

converts value from user units two bytes

**usedMemory**

**Returns** used memory in user units

**class** `RRtoolbox.lib.cache.Retriever`

Bases: `_abcoll.MutableMapping`

keep track of references and create objects on demand if needed.

**register** (*key, method=None, instance=None*)

Register object to retrieve.

**Parameters**

- **key** – hashable key to retrieve
- **method** – callable method to get object
- **instance** – object instance already created from method

**Returns**

Example:

```
def mymethod():
    class constructor: pass
    return constructor()

ret = retriever()
```

```

ret["obj"] = mymethod # register creating method in "obj"
im = ret["obj"] # get object (created obj +1, with reference)
assert im is ret["obj"] # check that it gets the same object
# it remembers that "obj" is last registered or fetched object too
assert ret() is ret()
# lets register with better control (created obj2 +1, no reference)
ret.register("obj2", mymethod(), mymethod)
# proves that obj2 is not the same as obj (created obj2 +1, no reference)
assert ret() is not ret["obj"]
print list(ret.iteritems()) # get items

```

RRtoolbox.lib.cache.**cachedProperty** (watch=[], handle=[])

A memoize decorator of @property decorator specifying what to trigger caching.

#### Parameters

- **watch** – (list of strings) A list of arguments name to watch in the hashing.
- **handle** – (list of handles or empty list) Provided list is appended with the Memo handle were data is stored for the method and where a clear() function is provided.

#### Returns

RRtoolbox.lib.cache.**mapper** (path, obj=None, mode=None, onlynumpy=False)

Save and load or map live objects to disk to free RAM memory.

#### Parameters

- **path** – path to save mapped file.
- **obj** – the object to map, if None it tries to load obj from path if exist
- **mode** – {None, 'r+', 'r', 'w+', 'c'}.
- **onlynumpy** – if True, it saves a numpy mapper from obj.

**Returns** mmap image, names of mmap files

## RRtoolbox.lib.config module

**platform** Unix, Windows

**synopsis** Looking for a reference? look here!.

This module contains all config data to the package.

**class** RRtoolbox.lib.config.**ConfigTool**

Manage the configured Tools.

**static** **getTools** (package)

Obtains the tools of a directory for the RRtoolbox.

**Parameters** **package** – path to the directory or package object.

**Returns** a dictionary of imported modules.

**class** RRtoolbox.lib.config.**DirectoryManager** (path=None, raiseError=True, autosave=False)

Bases: object

Manage the configured variables, paths and files.

#### Parameters

- **path** – (None) path to configuration file. If None uses default path.

- **raiseError** – True to raise when not attribute in ConfigFile.
- **autosave** – (True) if True saves at each change.

---

**Note:** Any attribute that is not in ConfigFile returns None. Use raiseError to control this behaviour.

---

**default**

get directories from dictionary representing environment variables.

**Returns** dictionary of directories.

---

**Note:** Only directories in the scope of the module are detected.

---

**load()**

loads the configuration file and update.

**Returns** loaded configuration file dictionary.

**Warning:** Unsaved instance variables will be replaced by configuration file variables.

**reset()**

Returns the configuration file to default variables.

**Returns** False, if error. Dictionary of new data, if successful.

**Warning:** All custom data is lost in configuration file.

**Warning:** ConfigFile is purposely not updated. Call manually method load()

**save(mode=0)**

saves configuration file.

**Parameters** **mode** – 0- delete and save, 1- update without replace, 2- update replacing variables.

**Returns** False, if error. Dictionary of new data, if successful.

RRtoolbox.lib.config.**findModules**(package, exclude=None)

Find modules from a package.

**Parameters**

- **package** – imported packaged or path (str).
- **exclude** – list of modules to exclude.

**Returns** dictionary containing importer, ispkg

RRtoolbox.lib.config.**getModules**(package, exclude=None)

Import modules from a package.

**Parameters** **package** – imported packaged or path (str).

**Returns** dictionary containing imported modules.

RRtoolbox.lib.config.**getPackagePath**(package)

Get the path of a package object.

**Parameters** **package** – package object or path (str).

**Returns** path to the package.

## RRtoolbox.lib.descriptors module

```
RRtoolbox.lib.descriptors.ASIFT (feature_name, img, mask=None,
                                   pool=<multiprocessing.pool.ThreadPool object>)
asift(feature_name, img, mask=None, pool=None) -> keypoints, descrs
```

Apply a set of affine transformations to the image, detect keypoints and reproject them into initial image coordinates. See [http://www.ipol.im/pub/algo/my\\_affine\\_sift/](http://www.ipol.im/pub/algo/my_affine_sift/) for the details.

ThreadPool object may be passed to speedup the computation.

### Parameters

- **feature\_name** – feature name to create detector.
- **img** – image to find keypoints and its descriptors
- **mask** – mask to detect keypoints (it uses default, mask[:] = 255)
- **pool** – multiprocessing pool (dummy, it uses multithreading)

**Returns** keypoints, descriptors

```
RRtoolbox.lib.descriptors.ASIFT_iter (imgs, feature_name='sift-flann')
Affine-SIFT for N images.
```

### Parameters

- **imgs** – images to apply asift
- **feature\_name** – eg. SIFT SURF ORB

**Returns** [(kp1, desc1), ..., (kpN, descN)]

```
RRtoolbox.lib.descriptors.ASIFT_multiple (imgs, feature_name='sift-flann')
Affine-SIFT for N images.
```

### Parameters

- **imgs** – images to apply asift
- **feature\_name** – eg. SIFT SURF ORB

**Returns** [(kp1, desc1), ..., (kpN, descN)]

```
class RRtoolbox.lib.descriptors.Feature (pool=<multiprocessing.pool.ThreadPool object>, use-
                                         ASIFT=True, debug=True)
```

Bases: object

Class to manage detection and computation of features

### Parameters

- **pool** – multiprocessing pool (dummy, it uses multithreading)
- **useASIFT** – if True adds Affine perspectives to the detector.
- **debug** – if True prints to the stdout debug messages.

```
config (name, separator='-')
```

This function takes parameters from a command to initialize a detector and matcher.

### Parameters

- **name** – “[a-]<siftlsurflorb>[-flann]” (str) Ex: “a-sift-flann”

- **features** – it is a dictionary containing the mapping from name to the initialized detector, matcher pair. If None it is created. This feature is to reduce time by reusing created features.

**Returns** detector, matcher

**detectAndCompute** (*img*, *mask=None*)

detect keypoints and descriptors

**Parameters**

- **img** – image to find keypoints and its descriptors
- **mask** – mask to detect keypoints (it uses default, `mask[:] = 255`)

**Returns** keypoints, descriptors

`RRtoolbox.lib.descriptors.MATCH` (*feature\_name*, *kp1*, *desc1*, *kp2*, *desc2*)

Use matcher and asift output to obtain Transformation matrix (TM).

**Parameters**

- **feature\_name** – feature name to create detector. It is the same used in the detector which is used in `init_feature` function but the detector itself is ignored. e.g. if 'detector' uses BFMatcher, if 'detector-flann' uses FlannBasedMatcher.
- **kp1** – keypoints of source image
- **desc1** – descriptors of kp1
- **kp2** – keypoints of destine image
- **desc2** – descriptors of kp2

**Returns** TM

# [http://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_feature2d/py\\_feature\\_homography/py\\_feature\\_homography.html](http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html)

`RRtoolbox.lib.descriptors.MATCH_multiple` (*pairlist*, *feature\_name='sift-flann'*)

**Parameters**

- **pairlist** – list of keypoint and descriptors pair e.g. [(kp1, desc1), ..., (kpN, descN)]
- **feature\_name** – feature name to create detector

**Returns** [(H1, mask1, kp\_pairs1), ..., (HN, maskN, kp\_pairsN)]

`RRtoolbox.lib.descriptors.affine_skew` (*tilt*, *phi*, *img*, *mask=None*)

Increase robustness to descriptors by calculating other invariant perspectives to image.

**Parameters**

- **tilt** – tilting of image
- **phi** – rotation of image (in degrees)
- **img** – image to find Affine transforms
- **mask** – mask to detect keypoints (it uses default, `mask[:] = 255`)

**Returns** skew\_img, skew\_mask, Ai (invert Affine Transform)

Ai - is an affine transform matrix from skew\_img to img

`RRtoolbox.lib.descriptors.filter_matches` (*kp1*, *kp2*, *matches*, *ratio=0.75*)

This function applies a ratio test.

**Parameters**

- **kp1** – raw keypoints 1
- **kp2** – raw keypoints 2
- **matches** – raw matches
- **ratio** – filtering ratio of distance

**Returns** filtered keypoint 1, filtered keypoint 2, keypoint pairs

`RRtoolbox.lib.descriptors.init_feature(name, features=None)`

This function takes parameters from a command to initialize a detector and matcher.

#### Parameters

- **name** – “<siftlsurflorb>[-flann]” (str) Ex: “sift-flann”
- **features** – it is a dictionary containing the mapping from name to the initialized detector, matcher pair. If None it is created. This feature is to reduce time by reusing created features.

**Returns** detector, matcher

`RRtoolbox.lib.descriptors.inlineRatio(inlines, lines, thresh=30)`

Probability that a match was correct.

#### Parameters

- **inlines** – number of matched lines
- **lines** – number lines
- **thresh** – threshold for lines (i.e. very low probability  $\leq$  thresh < good probability)

**Returns**

## RRtoolbox.lib.directory module

This module holds all path manipulation methods and a string concept called directory (referenced paths and strings) designed to support `config` and be used with `session`.

### keywords:

*path*: it can be to a folder or file or url if specified *filename*: the file name without its path *filepath*: the path to a file *dirname*: the path to a folder *url*: Universal Resource Locator

**class** `RRtoolbox.lib.directory.Directory`

Bases: `str`

semi-mutable string representation of a immutable string with support for path representations.

#### Parameters

- **data** – list, directory instance, dictionary or string.
- **ispath** – True to add support for paths.
- **copy** – when data is a directory if copy is True then this instance data is independent of the passed directory otherwise both directories are a reference to the same dictionary data but they are not the same object.
- **kwargs** – additional data to add in directory.

**copy()**

Creates copy of itself.

**Returns** non-referenced directory copy.

**correctSTRBuiltin** ()

Decorate all the built-in functions of class directory.

**Returns** built-in decorated function.

**static filterdata** (*data*, *ispath=None*, *kwargs=None*)

Adequate data for dictionary creation.

**Parameters**

- **data** – any supported object.
- **ispath** – True to add support for paths.
- **kwargs** – additional data to add in directory.

**Returns** dictionary

**static repr2list** (*data*, *level=0*)

Converts the representation of a directory.repr to pickleable.

**Parameters** **data** – directory.repr of the form ["string",directory,...,directory.repr].

**Returns** pickleable list.

**static repr2str** (*data*, *ispath=True*)

Converts the representation of a directory.repr to string.

**Parameters** **data** – directory.repr of the form ["string",directory,...,directory.repr].

**Returns** converted string.

**update** (*data=None*)

Return an updated copy with provided data.

**Parameters** **data** – any supported object. If None return updated and referenced copy of itself.

**Returns** new directory referenced to itself.

**update\_left** (*other*)

Updates representation a the left.

**Parameters** **other** – any supported object.

**Returns** new directory referenced to itself.

---

**Note:** Equivalent to self - other e.g. directory([other, self])

---

**update\_right** (*other*)

Updates representation a the right.

**Parameters** **other** – any supported object.

**Returns** new directory referenced to itself.

---

**Note:** Equivalent to self + other e.g. directory([self, other])

---

**class** RRtoolbox.lib.directory.**FileDirectory**

Bases: *RRtoolbox.lib.directory.Directory*

Saves contents of a file as with directories.



**Parameters**

- **data** – list, directory instance, dictionary or string.
- **filename** – name of file.
- **path** – path to folder where file is (it must finish in /).
- **notes** – optional description string
- **kwargs** – additional data to add in directory.

**makeFile()**

Makes a file with its contents to path/filename.

**Returns** True if successful

`RRtoolbox.lib.directory.changedir(filepath, dirname, ext=True)`

Change path to file with dirname.

**Parameters**

- **filepath** – path to file.
- **dirname** – new path to replace in filepath.
- **ext** – True to keep extension of file if any.

**Returns** directory object of changed path.

`RRtoolbox.lib.directory.checkDir(dirname)`

checks if dirname exists.

**Parameters** **dirname** – path to folder

**Returns** True if exists, False if not

`RRtoolbox.lib.directory.checkFile(path)`

checks if filepath or filename exists.

**Parameters** **path** – filepath or filename

**Returns** True if exists, False if not

`RRtoolbox.lib.directory.checkPath(path)`

checks if path exists.

**Parameters** **path** – path to folder or file.

**Returns** True if exists, False if not

`RRtoolbox.lib.directory.checkURL(url)`

checks if url exists. :param url: path to url :return: True if exists, False if not

`RRtoolbox.lib.directory.correctPath(path, relative)`

Get path corrected from its relative path or level index.

**Parameters**

- **path** – path or file name.
- **relative** – pattern or level in directory.

**Returns** corrected path.

`RRtoolbox.lib.directory.correctSep(path='/mnt/4E443F99443F82AF/Dropbox/PYTHON/RRtools/RRtoolbox/lib/direct', separator='/')`

Replaces the path separators by custom or OS standard separator.

**Parameters**

- **path** – relative or absolute path (str). Default is `__file__` or module's path.
- **separator** – desired separators, By default uses system separator (`os.path.sep`).

**Returns** path with corrected separator.

`RRtoolbox.lib.directory.decoratePath (relative, sep='/')`

Decorated path is controlled to give absolute path from relative path.

**Parameters**

- **relative** – int or path.
- **sep** – path separator

**Returns** decorator

`RRtoolbox.lib.directory.getData (path='/mnt/4E443F99443F82AF/Dropbox/PYTHON/RRtools/RRtoolbox/lib/directory.p`

Get standard path from path.

**Parameters** **path** – it can be to a folder or file. Default is `__file__` or module's path.

**Returns** [drive,dirname,filename,ext]. 1. drive or UNC (Universal Naming Convention) 2. dirname is path to folder. 3. filename is name of file. 4. ext is extension of file.

`RRtoolbox.lib.directory.getFileHandle (path)`

Gets a file handle from url or disk file.

**Parameters** **path** – filepath or url

**Returns** file object

`RRtoolbox.lib.directory.getFileSize (path)`

Gets a size from url or disk file.

**Parameters** **path** – filepath or url

**Returns** size in bytes

`RRtoolbox.lib.directory.getPath (path='/mnt/4E443F99443F82AF/Dropbox/PYTHON/RRtools/RRtoolbox/lib/directory.p`

Get standard path from path.

**Parameters** **path** – it can be to a folder or file. Default is `__file__` or module's path. If file exists it selects its folder.

**Returns** dirname (path to a folder)

---

**Note:** It is the same as `os.path.dirname(os.path.abspath(path))`.

---

`RRtoolbox.lib.directory.getSep (path, pattern='\')`

Get path separator or indicator.

**Parameters**

- **path** – relative or absolute path (str).
- **pattern** – guess characters to compare path (str).

**Returns** sep (str).

---

**Note:** It is equivalent to `os.path.sep` but obtained from the given path and patterns.

---

`RRtoolbox.lib.directory.getShortenedPath(path, comp)`

Path is controlled to give absolute path from relative path or integer.

#### Parameters

- **path** – absolute path (str).
- **comp** – pattern or relative path (str) or integer representing level of folder determined by the separator Ex. “/level 1/level 2/.../level N or -1”.

**Returns** path before matched to comp Ex: “C://level 1//comp → C://level 1”

Example:

```
>>> path = 'LEVEL1/LEVEL2/LEVEL3/LEVEL4/LEVEL5'
>>> print getShortenedPath(path, -2) # minus two levels
LEVEL1/LEVEL2/LEVEL3
>>> print getShortenedPath(path, 2) # until three levels
LEVEL1/LEVEL2
>>> print getShortenedPath(path, 'LEVEL1/LEVEL2/LEVEL3/')
LEVEL1/LEVEL2/LEVEL3/
>>> print getShortenedPath(path, 'LEVEL4/REPLACE5/NEWLEVEL')
LEVEL1/LEVEL2/LEVEL3/LEVEL4/REPLACE5/NEWLEVEL
>>> print getShortenedPath(path, '../../SHOULD_BE_LEVEL4')
LEVEL1/LEVEL2/LEVEL3/SHOULD_BE_LEVEL4
```

`RRtoolbox.lib.directory.getSplitted(path='/mnt/4E443F99443F82AF/Dropbox/PYTHON/RRtools/RRtoolbox/lib/dire`

Splits a file path by its separators.

**Parameters** **path** – it can be to a folder or file. Default is `__file__` or module’s path.

**Returns** splitted path.

`RRtoolbox.lib.directory.increment_if_exists(path, add='_[num]', force=None)`

Generates new name if it exists.

#### Parameters

- **path** – absolute path or filename
- **add** – if fn exists add pattern
- **force** – (None) force existent files even if they don’t. if True treats fn as existent or if it is a list it treats names from the list as existent names.

**Returns** un-existent fn

`RRtoolbox.lib.directory.joinPath(absolute, relative)`

Joins an absolute path to a relative path.

#### Parameters

- **absolute** – directory or path.
- **relative** – directory or path.

**Returns** joined path.

---

**Note:** It is equivalent to `os.path.join` but works with directories.

---

`RRtoolbox.lib.directory.mkPath(path)`

Make path (i.e. creating folder) for filepath.

**Parameters** **path** – path to nonexistent folder or file.

**Returns** created path.

RRtoolbox.lib.directory.**quickOps** (*path, comp*)  
(IN DEVELOPMENT) make quick matching operations in path.

**Parameters**

- **path** – path to folder
- **comp** – pattern

**Returns**

Requirements:

```
path = 'LEVEL1/LEVEL2/LEVEL3/LEVEL4/LEVEL5'
print quickOps (path, '../ROOT/../LEVEL1/../LEVEL2/LEVEL3/../../LEVEL4')
'LEVEL4'
print quickOps (path, 'ROOT/../LEVEL1/LEVEL2/../../LEVEL4')
'LEVEL3/LEVEL4'
print quickOps (path, '../LEVEL2/../../')
'LEVEL1/LEVEL3/LEVEL4/LEVEL5'
print quickOps (path, '../LEVEL2/..')
'LEVEL1/LEVEL3/LEVEL4/LEVEL5/'
print quickOps (path, 'LEVEL2/../../LEVEL4/')
'LEVEL2/LEVEL3/LEVEL4/'
print quickOps (path, 'ROOT/../LEVEL2/../../LEVEL4')
'ROOT/LEVEL3/LEVEL4'
print quickOps (path, 'LEVEL-1/../NEW7/LEVEL8')
'LEVEL-1/LEVEL1/LEVEL2/LEVEL3/LEVEL4/LEVEL5/NEW7/LEVEL8'
print
```

RRtoolbox.lib.directory.**resource\_path** (*relative\_path=''*)  
Get absolute path to resource, works for dev and for PyInstaller

RRtoolbox.lib.directory.**rmFile** (*filepath*)  
Remove file.

**Parameters** **filepath** – path to file.

**Returns** None

RRtoolbox.lib.directory.**rmPath** (*path, ignore\_errors=False, onerror=None*)  
Remove path from path.

**Parameters** **path** – path to nonexistent folder or file.

**Returns** None

**See also:**

shutil.rmtree

RRtoolbox.lib.directory.**strdifference** (*s1, s2*)  
Get string differences.

**Parameters**

- **s1** – string 1
- **s2** – string 2

**Returns** (splitted string 1, splitted string 2, index). A splitted string is a list with the string parts.  
Index is a list containing the indexes of different parts of the two splitted strings.

## RRtoolbox.lib.image module

Bundle of methods for handling images. Rather than manipulating specialized operations in images methods in this module are used for loading, outputting and format-converting methods, as well as color manipulation.

### SUPPORTED FORMATS

see [http://docs.opencv.org/2.4/modules/highgui/doc/reading\\_and\\_writing\\_images\\_and\\_video.html#imread](http://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html#imread)

Windows bitmaps - \*.bmp, \*.dib (always supported) JPEG files - \*.jpeg, \*.jpg, \*.jpe (see the Notes section) JPEG 2000 files - \*.jp2 (see the Notes section) Portable Network Graphics - \*.png (see the Notes section) Portable image format - \*.pbm, \*.pgm, \*.ppm (always supported) Sun rasters - \*.sr, \*.ras (always supported) TIFF files - \*.tiff, \*.tif (see the Notes section)

**class** `RRtoolbox.lib.image.GetCoors` (*im*, *win*='get coordinates', *updatefunc*=<function drawcoor-  
points>, *unique*=True, *col\_out*=(0, 0, 0), *col\_in*=(0, 0, 255))

Bases: `RRtoolbox.lib.plotter.Plotim`

Create window to select points from image.

#### Parameters

- **im** – image to get points.
- **win** – window name.
- **updatefunc** – function to draw interaction with points. (e.g. `limitaxispoints`, `drawcoor-perspective`, etc.).
- **prox** – proximity to identify point.
- **radius** – radius of drawn points.
- **unique** – If True no point can be repeated, else selected points can be repeated.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.

#### coors

**drawstats** (*points*, *col\_out*=(0, 0, 0), *col\_in*=(0, 255, 0), *radius*=2)

#### Parameters

- **self** –
- **points** –
- **col\_out** –
- **col\_in** –
- **radius** –

#### Returns

**mousefunc** ()

Parameters **self** –

#### Returns

**updatecoors** ()

Parameters **self** –

#### Returns

**class** RRtoolbox.lib.image.**ImCoors** (*pts, dtype=<type 'numpy.float32'>, deg=False*)  
Bases: object

Image's coordinates class. Example:

```
a = ImCoors(np.array([(116, 161), (295, 96), (122, 336), (291, 286)]))
print a.__dict__
print "mean depend on min and max: ", a.mean
print a.__dict__
print "after mean max has been already been calculated: ", a.max
a.data = np.array([(116, 161), (295, 96)])
print a.__dict__
print "mean and all its dependencies are processed again: ", a.mean
```

**dtype**

**pts**

**class** RRtoolbox.lib.image.**ImFactory** (*\*\*kwargs*)

image factory for RRToolbox to create scripts to standardize loading images and provide lazy loading (it can load images from disk with the customized options and/or create mapping images to load when needed) to conserve memory.

**Warning:** In development.

**get\_Func** ()

gets the loading function

**get\_code** ()

get the script code

**get\_conversionFunc** (*code*)

**get\_errorFunc** (*path=None, throw=None*)

**get\_loadFunc** (*flag=None*)

**get\_mapFunc** (*flag=None, RGB=None, mpath=None, mode=None, func=None, dsize=None, dst=None, fx=None, fy=None, interpolation=None*)

**get\_np2qi** ()

**get\_resizeFunc** (*dsize=None, dst=None, fx=None, fy=None, interpolation=None*)

**get\_transposeFunc** ()

**update** (*\*\*kwargs*)

**class** RRtoolbox.lib.image.**ImLoader** (*path, flag=0, dsize=None, dst=None, fx=None, fy=None, interpolation=None, mmode=None, mpath=None, throw=True*)

Class to load image array from path, url, server, string or directly from numpy array (supports databases).

#### Parameters

- **flag** – (default: 0) 0 to read as gray, 1 to read as BGR, -1 to read as BGRA, 2 to read as RGB, -2 to read as RGBA.

**It supports openCV flags:**

- cv2.CV\_LOAD\_IMAGE\_COLOR
- cv2.CV\_LOAD\_IMAGE\_GRAYSCALE
- cv2.CV\_LOAD\_IMAGE\_UNCHANGED

value	openCV flag	output
2.	N/A	RGB
1.	cv2.CV_LOAD_IMAGE_COLOR	BGR
0.	cv2.CV_LOAD_IMAGE_GRAYSCALE	GRAY
(-1)	cv2.CV_LOAD_IMAGE_UNCHANGED	UNCHANGED
(-2)	N/A	RGBA

- **dsize** – (None) output image size; if it equals zero, it is computed as:  
`exttt{dsize = Size(round(fx*src.cols), round(fy*src.rows))}`
- **dst** – (None) output image; it has the size dsize (when it is non-zero) or the size computed from `src.size()`, `fx`, and `fy`; the type of `dst` is `uint8`.
- **fx** – scale factor along the horizontal axis; when it equals 0, it is computed as  
`exttt{(double)dsize.width/src.cols}`
- **fy** – scale factor along the vertical axis; when it equals 0, it is computed as  
`exttt{(double)dsize.height/src.rows}`
- **interpolation** – interpolation method compliant with `opencv`:

flag	Operation	Description
0.	INTER_NEAREST	nearest-neighbor interpolation
1.	INTER_LINEAR	bilinear interpolation (used by default)
2.	INTER_CUBIC	bicubic interpolation over 4x4 pixel neighborhood
3.	INTER_AREA	resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method.
4.	INTER_LANCZOS4	Lanczos interpolation over 8x8 pixel neighborhood

- **mmode** – (None) `mmode` to create mapped file. if `mpath` is specified loads image, converts to mapped file and then loads mapping file with mode `{None, 'r+', 'r', 'w+', 'c'}` (it is slow for big images). If `None`, loads mapping file to memory (useful to keep image copy for session even if original image is deleted or modified).
- **mpath** – (None) path to create mapped file. `None`, do not create mapping file `""`, uses path directory; `"*"`, uses working directory; else, uses specified directory.

---

**Note:** If `mmode` is `None` and `mpath` is given it creates `mmap` file but loads from it to memory. It is useful to create physical copy of data to keep loading from (data can be reloaded even if original file is moved or deleted).

---

**getConfiguration** (*\*\*kwargs*)

get Custom configuration from default configuration :param *kwargs*: keys to customize default configuration.

If no key is provided default configuration is returned.

**Returns** dictionary of configuration

**temp** (*\*\*kwargs*)

loads from temporal loader created with customized and default parameters.

**Parameters** *kwargs* – keys to customize default configuration.

**Returns** loaded image.

**class** `RRtoolbox.lib.image.Image` (*name=None, ext=None, path=None, shape=None, verbosity=False*)

Bases: `object`

Structure to load and save images

**BGR**

**BGRA**

**RGB**

**RGBA**

**ext**

**gray**

**load** (*name=None, path=None, shape=None*)

**path**

**save** (*name=None, image=None, overwrite=None*)

save restored image in path.

**Parameters**

- **name** – filename, string to format or path to save image. if path is not a string it would be replaced with the string “{path}restored\_{name}{ext}” to format with the formatting “{path}”, “{name}” and “{ext}” from the `baseImage` variable.
- **image** – (self.BGRA)
- **overwrite** – If True and the destine filename for saving already exists then it is replaced, else a new filename is generated with an index “{filename}\_{index}.{extension}”

**Returns** saved path, status (True for success and False for fail)

**shape**

**class** `RRtoolbox.lib.image.LoaderDict` (*loader=None, maxMemory=None, margin=0.8, unit='MB', all=True, config=None*)

Bases: `RRtoolbox.lib.cache.ResourceManager`

Class to standardize loading objects and manage memory efficiently.



**Parameters**

- **loader** – default loader for objects (e.g. load from file or create instance object)
- **maxMemory** – (None) max memory in specified unit to keep in check optimization (it does not mean that memory never surpasses maxMemory).
- **margin** – (0.8) margin from maxMemory to trigger optimization. It is in percentage of maxMemory ranging from 0 (0%) to maximum 1 (100%). So optimal memory is inside range:  $\text{maxMemory} * \text{margin} < \text{Memory} < \text{maxMemory}$
- **unit** – (MB) maxMemory unit, it can be GB (Gigabytes), MB (Megabytes), B (bytes)
- **all** – if True used memory is from all alive references, if False used memory is only from keptAlive references.
- **config** – (Not Implemented)

**register** (*key*, *path=None*, *method=None*)

**class** RRtoolbox.lib.image.**PathLoader** (*fns=None*, *loader=None*)

Bases: `_abcoll.MutableSequence`

Class to standardize loading images from list of paths and offer lazy evaluations.

**Parameters**

- **fns** – list of paths
- **loader** – path loader (loadcv, loadsfrom, or function from loadFunc)

Example:

```
fns = ["/path to/image 1.ext", "/path to/image 2.ext"]
imgs = pathLoader(fns)
print imgs[0] # loads image in path 0
print imgs[1] # loads image in path 1
```

**insert** (*index*, *value*)

RRtoolbox.lib.image.**bgra2bgr** (*im*, *bgrcolor=(255, 255, 255)*)

Convert BGR to BGRA image.

**Parameters**

- **im** – image
- **bgrcolor** – BGR color representing transparency. (information is lost when converting BGRA to BGR) e.g. [200,200,200].

**Returns**

RRtoolbox.lib.image.**checkLoaded** (*obj*, *fn=''*, *raiseError=False*)

Simple function to determine if variable is valid.

**Parameters**

- **obj** – loaded object
- **fn** – path of file
- **raiseError** – if True and obj is None, raise

**Returns** None

RRtoolbox.lib.image.**convertAs** (*fns*, *base=None*, *folder=None*, *name=None*, *ext=None*, *overwrite=False*, *loader=None*, *simulate=False*)

Reads a file and save as other file based in a pattern.

**Parameters**

- **fns** – file name or list of file names. It supports glob operations. By default glob operations ignore folders.
- **base** – path to place images.
- **folder** – (None) folder to place images in base’s path. If True it uses the folder in which image was loaded. If None, not folder is used.
- **name** – string for formatting new name of image with the {name} tag. Ex: if name is ‘new\_{name}’ and image is called ‘img001’ then the formatted new image’s name is ‘new\_img001’
- **ext** – (None) extension to save all images. If None uses the same extension as the loaded image.
- **overwrite** – (False) If True and the destine filename for saving already exists then it is replaced, else a new filename is generated with an index “{name}\_{index}.{extension}”
- **loader** – (None) loader for the image file to change image attributes. If None reads the original images untouched.
- **simulate** – (False) if True, no saving is performed but the status is returned to confirm what images where adequately processed.

**Returns** list of statuses (0 - no error, 1 - image not loaded, 2 - image not saved, 3 - error in processing image)

`RRtoolbox.lib.image.drawcoorarea(vis, points, col_out=(0, 0, 0), col_in=(0, 0, 255), radius=2)`

Function to draw interaction with points to obtain area.

**Parameters**

- **vis** – image array.
- **points** – list of points.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.
- **radius** – radius of drawn points.

**Returns**

`RRtoolbox.lib.image.drawcooraxes(vis, points, col_out=(0, 0, 0), col_in=(0, 255, 0), radius=2)`

Function to draw axes instead of points.

**Parameters**

- **vis** – image array.
- **points** – list of points.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.
- **radius** – radius of drawn points.

**Returns**

`RRtoolbox.lib.image.drawcoorperspective(vis, points, col_out=(0, 0, 0), col_in=(0, 0, 255), radius=2)`

Function to draw interaction with points to obtain perspective.

**Parameters**

- **vis** – image array.
- **points** – list of points.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.
- **radius** – radius of drawn points.

#### Returns

RRtoolbox.lib.image.**drawcoorpoints** (*vis, points, col\_out=(0, 0, 0), col\_in=(0, 0, 255), radius=2*)

Function to draw points.

#### Parameters

- **vis** – image array.
- **points** – list of points.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.
- **radius** – radius of drawn points.

#### Returns

RRtoolbox.lib.image.**drawcoorpolyArrow** (*vis, points, col\_out=(0, 0, 0), col\_in=(0, 0, 255), radius=2*)

Function to draw interaction with vectors to obtain polygonal.

#### Parameters

- **vis** – image array.
- **points** – list of points.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.
- **radius** – radius of drawn points.

#### Returns

RRtoolbox.lib.image.**drawcoorpolyline** (*vis, points, col\_out=(0, 0, 0), col\_in=(0, 0, 255), radius=2*)

Function to draw interaction with points to obtain polygonal.

#### Parameters

- **vis** – image array.
- **points** – list of points.
- **col\_out** – outer color of point.
- **col\_in** – inner color of point.
- **radius** – radius of drawn points.

#### Returns

RRtoolbox.lib.image.**fig2bgr** (*fig*)

Convert a Matplotlib figure to a RGB image.

**Parameters** **fig** – a matplotlib figure

**Returns** RGB image.

`RRtoolbox.lib.image.fig2bgra (fig)`

Convert a Matplotlib figure to a RGBA image.

**Parameters** `fig` – a matplotlib figure

**Returns** RGBA image.

`RRtoolbox.lib.image.getcoors (im, win='get coordinates', updatefunc=<function drawcoor-  
points>, coors=None, prox=8, radius=3, unique=True, col_out=(0,  
0, 0), col_in=(0, 0, 255))`

`RRtoolbox.lib.image.getgeometrycoors (*data)`

Get filled object coordinates. (function in progress)

`RRtoolbox.lib.image.getrectcoors (*data)`

Get ordered points.

**Parameters** `data` – list of points

**Returns** [Top\_left,Top\_right,Bottom\_left,Bottom\_right]

`RRtoolbox.lib.image.gray2qi (gray)`

Convert the 2D numpy array `gray` into a 8-bit QImage with a gray colormap. The first dimension represents the vertical image axis.

ATTENTION: This QImage carries an attribute `ndimage` with a reference to the underlying numpy array that holds the data. On Windows, the conversion into a QPixmap does not copy the data, so that you have to take care that the QImage does not get garbage collected (otherwise PyQt will throw away the wrapper, effectively freeing the underlying memory - boom!).

source from: <https://kogs-www.informatik.uni-hamburg.de/~meine/software/vigraqt/qimage2ndarray.py>

`RRtoolbox.lib.image.hist_match (source, template, alpha=None)`

Adjust the pixel values of an image to match those of a template image.

**Parameters**

- **source** – image to transform colors to template
- **template** – template image ()
- **alpha** –

**Returns** transformed source

`RRtoolbox.lib.image.interpretImage (toparse, flags)`

Interprets to get image.

**Parameters**

- **toparse** – string to parse or array. It can interpret:
  - \*connection to server (i.e. host:port) \*path to file (e.g. /path\_to\_image/image\_name.ext)
  - \*URL to image (e.g. [http://domain.com/path\\_to\\_image/image\\_name.ext](http://domain.com/path_to_image/image_name.ext)) \*image as string (i.g. numpy converted to string) \*image itself (i.e. numpy array)
- **flags** – openCV flags:

value	openCV flag	output
1.	cv2.CV_LOAD_IMAGE_COLOR	BGR
0.	cv2.CV_LOAD_IMAGE_GRAYSCALE	GRAY
(-1)	cv2.CV_LOAD_IMAGE_UNCHANGED	UNCHANGED

**Returns** image or None if not successful

RRtoolbox.lib.image.**limitaxispoints** (*c, maxc, minc=0*)

Limit a point in axis.

#### Parameters

- **c** – list of points..
- **maxc** – maximum value of point.
- **minc** – minimum value of point.

**Returns** return limited points.

RRtoolbox.lib.image.**loadFunc** (*flag=0, dsiz=None, dst=None, fx=None, fy=None, interpolation=None, mmode=None, mpath=None, throw=True, keepratio=True*)

Creates a function that loads image array from path, url, server, string or directly from numpy array (supports databases).

#### Parameters

- **flag** – (default: 0) 0 to read as gray, 1 to read as BGR, -1 to read as BGRA, 2 to read as RGB, -2 to read as RGBA.

**It supports openCV flags:**

- cv2.CV\_LOAD\_IMAGE\_COLOR
- cv2.CV\_LOAD\_IMAGE\_GRAYSCALE
- cv2.CV\_LOAD\_IMAGE\_UNCHANGED

value	openCV flag	output
2.	N/A	RGB
1.	cv2.CV_LOAD_IMAGE_COLOR	BGR
0.	cv2.CV_LOAD_IMAGE_GRAYSCALE	GRAY
(-1)	cv2.CV_LOAD_IMAGE_UNCHANGED	UNCHANGED
(-2)	N/A	RGBA

- **dsiz** – (None) output image size; if it equals zero, it is computed as:

```
exttt{dsiz = Size(round(fx*src.cols), round(fy*src.rows))}
```

If (integer,None) or (None,integer) it completes the values according to keepratio parameter.

- **dst** – (None) output image; it has the size dsize (when it is non-zero) or the size computed from `src.size()`, `fx`, and `fy`; the type of `dst` is `uint8`.
- **fx** – scale factor along the horizontal axis
- **fy** – scale factor along the vertical axis
- **interpolation** – interpolation method compliant with `opencv`:

flag	Operation	Description
0.	INTER_NEAREST	nearest-neighbor interpolation
1.	INTER_LINEAR	bilinear interpolation (used by default)
2.	INTER_CUBIC	bicubic interpolation over 4x4 pixel neighborhood
3.	INTER_AREA	resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method.
4.	INTER_LANCZOS4	Lanczos interpolation over 8x8 pixel neighborhood

- **mmode** – (None) `mmode` to create mapped file. if `mpath` is specified loads image, converts to mapped file and then loads mapping file with mode {`None`, `'r+'`, `'r'`, `'w+'`, `'c'`} (it is slow for big images). If `None`, loads mapping file to memory (useful to keep image copy for session even if original image is deleted or modified).
- **mpath** – (None) path to create mapped file. `None`, do not create mapping file `""`, uses path directory; `"*"`, uses working directory; else, uses specified directory.
- **keepratio** – True to keep image ratio when completing data from `dsize`, `fx` and `fy`, False to not keep ratio.

---

**Note:** If `mmode` is `None` and `mpath` is given it creates `mmap` file but loads from it to memory. It is useful to create physical copy of data to keep loading from (data can be reloaded even if original file is moved or deleted).

---

:return loader function

`RRtoolbox.lib.image.loadcv(path, flags=-1, shape=None)`  
Simple function to load using `opencv`.

#### Parameters

- **path** – path to image.
- **flag** – `openCV` flags:

value	openCV flag	output
1.	cv2.CV_LOAD_IMAGE_COLOR	BGR
0.	cv2.CV_LOAD_IMAGE_GRAYSCALE	YSCALE
(-1)	cv2.CV_LOAD_IMAGE_UNCHANGED	UNCHANGED

- **shape** – shape to resize image.

**Returns** loaded image

`RRtoolbox.lib.image.loadsfrom(path, flags=1L)`

Loads Image from URL or file.

**Parameters**

- **path** – filepath or url
- **flags** – openCV flags:

value	openCV flag	output
1.	cv2.CV_LOAD_IMAGE_COLOR	BGR
0.	cv2.CV_LOAD_IMAGE_GRAYSCALE	YSCALE
(-1)	cv2.CV_LOAD_IMAGE_UNCHANGED	UNCHANGED

**Returns**

`RRtoolbox.lib.image.myline(img, pt1, pt2, color, thickness=None)`

Funtion to draw points (experimental).

**Parameters**

- **img** –
- **pt1** –
- **pt2** –
- **color** –
- **thickness** –

**Returns**

`RRtoolbox.lib.image.np2qi(array)`

Convert numpy array to Qt Image.

source from: <https://kogs-www.informatik.uni-hamburg.de/~meine/software/vigraqt/qimage2ndarray.py>

**Parameters** **array** –

**Returns**

`RRtoolbox.lib.image.np2str(arr)`

`RRtoolbox.lib.image.plt2bgr(image)`

`RRtoolbox.lib.image.plt2bgra(image)`

`RRtoolbox.lib.image.qi2np (qimage, dtype='array')`

Convert QImage to numpy.ndarray. The dtype defaults to uint8 for QImage.Format\_Indexed8 or *bgra\_dtype* (i.e. a record array) for 32bit color images. You can pass a different dtype to use, or 'array' to get a 3D uint8 array for color images.

source from: <https://kogs-www.informatik.uni-hamburg.de/~meine/software/vigraqt/qimage2ndarray.py>

`RRtoolbox.lib.image.quadrants (points)`

Separate points respect to center of gravity point.

**Parameters** *points* – list of points

**Returns** [[Top\_left],[Top\_right],[Bottom\_left],[Bottom\_right]]

`RRtoolbox.lib.image.random_color (channels=1, min=0, max=256)`

Random color.

**Parameters**

- **channels** – number of channels
- **min** – min color in any channel
- **max** – max color in any channel

**Returns** random color

`RRtoolbox.lib.image.rgb2qi (rgb)`

Convert the 3D numpy array *rgb* into a 32-bit QImage. *rgb* must have three dimensions with the vertical, horizontal and RGB image axes.

ATTENTION: This QImage carries an attribute *ndimage* with a reference to the underlying numpy array that holds the data. On Windows, the conversion into a QPixmap does not copy the data, so that you have to take care that the QImage does not get garbage collected (otherwise PyQt will throw away the wrapper, effectively freeing the underlying memory - boom!).

source from: <https://kogs-www.informatik.uni-hamburg.de/~meine/software/vigraqt/qimage2ndarray.py>

`RRtoolbox.lib.image.separe (values, sep, axis=0)`

Separate values from separator or threshold.

**Parameters**

- **values** – list of values
- **sep** – separator value
- **axis** – axis in each value

:return:lists of greater values, list of lesser values

`RRtoolbox.lib.image.str2np (string, shape)`

`RRtoolbox.lib.image.transposeIm (im)`

`RRtoolbox.lib.image.try_loads (fns, func=<built-in function imread>, paths=None, debug=False, addpath=False)`

Try to load images from paths.

**Parameters**

- **fns** – list of file names
- **func** – loader function
- **paths** – paths to try. By default it loads working dir and test path
- **debug** – True to show debug messages



- **addpath** – add path as second argument

**Returns** image else None

## RRtoolbox.lib.inspector module

This module is an all purpose intended for debugging, tracking, auto-documenting and self-introspecting the package Made by DavtoH. Powered partially by pycallgraph. Dependent project: <https://github.com/gak/pycallgraph/#python-call-graph>

**class** RRtoolbox.lib.inspector.**Asynchronous** (*outputs, config*)

Bases: *RRtoolbox.lib.inspector.Synchronous*

**done** ()

**start** ()

**tracer** (*frame, event, arg*)

**class** RRtoolbox.lib.inspector.**GraphTrace** (*output=None, config=None*)

Bases: *pycallgraph.pycallgraph.PyCallGraph*

**get\_tracer\_class** ()

**saveSource** (*file*)

**source**

**class** RRtoolbox.lib.inspector.**GraphTraceOutput** (*source=None, saveflag=True, label='', \*\*kwargs*)

Bases: *pycallgraph.output.graphviz.GraphvizOutput*

**done** ()

**save** (*file=None, source=None*)

**saveSource** (*file, source=None*)

**class** RRtoolbox.lib.inspector.**Logger** (*\*\*kwargs*)

Bases: *object*

Logger for decorated functions. Holds important information of an instanced object and can be used with @trace decorator for traceback purposes.

### Parameters

- **func** – object reference.
- **funcname** – object name.
- **inputs** – inputs pass to the object.
- **outputs** – outputs given by the object execution.
- **time** – initial time of execution.
- **exectime** – time of execution in seconds.
- **writer** – writer function where messages are passed.
- **eventHandle** – event function where object is passed when Logger.broadcast() is called.
- **msg\_report** – message format to use in reports.
- **msg\_no\_executed** – message format to pass to writer when object has not been executed and Logger.report() is called.

- **msg\_executed** – message format to use when object is executed and `Logger.broadcast()` is called.

**Time\_**  
returns formatted time (str)

**Type\_**  
returns type name (str)

**broadcast** ()  
pass a notification message on object execution to the writer

**eventHandle = None**

**file = <open file ‘<stdout>’, mode ‘w’>**

**renew** ()  
renew Instance

**report** ()  
pass a report of the last executed object to the writer

**throwError** ()  
throw caught error :return:

**tracer**

**writer** (sender, \*arg)

**class** `RRtoolbox.lib.inspector.Synchronous` (outputs, config)

Bases: `pycallgraph.tracer.SynchronousTracer`

**start** ()

**stop** ()

`RRtoolbox.lib.inspector.funcData` (func)

`RRtoolbox.lib.inspector.load` (mod\_name, obj\_name)

Convert a string version of a class name to the object.

For example, `get_class('sympy.core.Basic')` will return class `Basic` located in module `sympy.core`

`RRtoolbox.lib.inspector.reloadFunc` (func)

`RRtoolbox.lib.inspector.tracer` (instance, broadcast=True, report=True)

Tracer for decorated functions.

#### Parameters

- **instance** – Logger instance
- **broadcast** –
- **report** –

#### Returns

## RRtoolbox.lib.plotter module

This module holds the plotting and data-visualization tools. Motto: don't know how it is interpreted? i'll show you!

```
#Plotim example filename = "t2.jpg" win = "test" img = cv2.resize(cv2.imread(filename), (400, 400)) # (height, width)
plot = Plotim(win,img) plot.show()
```

```

class RRtoolbox.lib.plotter.Edger(img, isSIZE=True, isEQUA=False, isCLAHE=False, isBFIL-
                                TER=False)
    Bases: RRtoolbox.lib.plotter.Plotim
    Test visualization for edges

    self.edge -> the edges in processed image self.img -> the processed image self.sample -> the rendered precessed
    image

    computeAll()
    computeEdge()
    getParameters(params=('d', 'sigmaColor', 'sigmaSpace', 'clipLimit', 'tileGridSize', 'isSIZE', 'isE-
                        QUA', 'isCLAHE', 'isBFILTER', 'th1', 'th2', 'size', 'apertureSize', 'L2gradient'))
    isActiveWindow()
    isBFILTER
    isCLAHE
    isEQUA
    isSIZE
    load(img, compute=True)
    maxth
    onTrackbar1(*args)
    onTrackbar2(*args)
    save(strname=None, ext='.png', name='img')
    showgray
    size
    th1
    th2
    windowfunc()

class RRtoolbox.lib.plotter.Imtester(img, win='Imtester plot')
    Bases: RRtoolbox.lib.plotter.Plotim
    Used to test some concepts as thresholds and filters

    static applythresh(img, type, adaptativetoggle, threshtoggle, th, blocksz, c, i='', ti='', info='', ti-
                        tle='')
    builtcmd()
    computefunc(image=None)
    detectType(type, i='', info='')
    static formatinfo(info, words=9)
    updatevisualization(image, channel, th=None, items=None, thresh1=None, thresh2=None)
    visualize()
    windowfunc()

```

**class** `RRtoolbox.lib.plotter.MatchExplorer` (*win, img1, img2, kp\_pairs=(), status=None, H=None, show=True, block=True, daemon=True*)

Bases: `RRtoolbox.lib.plotter.Plotim`

Draws a set of keypoint pairs obtained on a match method of a descriptor on two images `imgf` and `imgb`. (backend: Plotim).

#### Parameters

- **win** – window's name (str)
- **img1** – image1 (numpy array)
- **img2** – image2 (numpy array)
- **kp\_pairs** – zip(keypoint1, keypoint2)
- **status** – obtained from `cv2.findHomography`
- **H** – obtained from `cv2.findHomography` (default=None)
- **show** – if True shows Plotim using block and daemon, else do not show
- **block** – if True it wait for window close, else it detaches
- **daemon** – if True window closes if main thread ends, else windows must be closed to main thread to end

**Returns** Plotim object with visualization as `self.rimg` (image with matching result) (default=None)

---

**Note:** It supports BGR and gray images.

---

**drawline** ()

Draws background visualization without interaction

**drawrelation** ()

Draw keypoints where pointer is placed and pressed

**keyfunc** ()

**mousefunc** ()

**static randomColor** ()

**updaterenderer** (*img=None, zoom=True*)

update renderer when called.

#### Parameters

- **img** – image to update in renderer, if None use `self.img`
- **zoom** – True to enable zoom, else updates with original `img`.

**Returns** None

**class** `RRtoolbox.lib.plotter.Plotim` (*win, im=array([[1]]), bgrcolor=(250, 243, 238)*)

Bases: `object`

Show and image with events, animations, controls, internal commands and highly customizable by code.

#### Parameters

- **win** – window name
- **im** – image of numpy array
- **bgrcolor** – default color to use for transparent or background color.

**Warning:** Plotim is deprecated and will be replaced in the future (it was made to test concepts). Originally it was made for windows but some functions were removed to let it be multi-platform.

**builtincmd()**

Internal cmd control

**builtincontrol** (*control=False*)

Internal control. use self.usecontrol = True to activate.

**Parameters** **control** – if True, use control key.

**Returns**

**builtinplot** (*pixel=None, useritems=None, flag=1, xpad=0, ypad=0, bgrcolor=None, alpha=None*)

Internal plot.

**Parameters**

- **pixel** – pixel color where mouse is placed (placed for better control). Color can be from real image, showed image, original image or rendered image, or any color.
- **useritems** – items to show.
- **flag** – flag for position (default=0).
  - flag==0 : foreground to left up.
  - flag==1 : foreground to left down.
  - flag==2 : foreground to right up.
  - flag==3 : foreground to right down.
  - flag==4 : foreground at center of background.
  - flag==5 : XY 0,0 is at center of background.
  - flag==6 : XY 0,0 is at center of foreground.
  - flag==7 : XY 0,0 is at right down of foreground.
- **xpad** – padding in x
- **ypad** – padding in y
- **bgrcolor** – background color
- **alpha** – alpha mask or value for transparency

**Returns**

**builtinwindow()**

loads windowfunc, showfunc, starts window thread and mousecallback.

**clean()**

Attempt to clean the plotter dictionary for an error in garbage collection. :return:

**closefunc()**

Decoupled close function for Plotim (replace self.closefunc).

**Parameters** **self** – Plotim instance

**cmdfunc** (*execute=False*)

command function and decoupled cmd solver for Plotim. (repalce self.cmdfunc)

**Parameters**

- **self** –
- **execute** – True, enable execution of commands, False, disable execution.

**errorbackground**

**formatcmd** (*cmd*, *references*=(*'+'*, *'-'*, *'\*'*, *'='*), *lmissing*=*'self'*)

Decoupled cmd formatter for cmdfunc and Plotim.

**Parameters**

- **self** – Plotim instance
- **cmd** – command
- **references** –
- **lmissing** – assumed missing part in command

**Returns**

**help** (*showAll=False*)

function to print the quick help for the user with all the commands

**init** ()

Pseudo `__init__`. it is used to restart default values without destroying configurations.

**keyfunc** ()

Decoupled key function for Plotim (replace `self.keyfunc`).

**Parameters** **self** – Plotim instance

**makeoverlay** (*items*, *xpad*=0, *ypad*=0, *bgrcolor*=None, *alpha*=None)

overlay items over image.

**Parameters**

- **self** – instance
- **items** – list of object to overlay
- **xpad** – pad in x
- **ypad** – pad in y
- **bgrcolor** – background color
- **alpha** – transparency color

**Returns** overlaid

**mousefunc** ()

Decoupled mouse function for Plotim (replace `self.mousefunc`).

**Parameters** **self** – Plotim instance

**static onmouse** (*event*, *x*, *y*, *flags*, *self*)

Mouse event function for Plotim. (replace `self.mousefunc`)

**Parameters**

- **event** – mouse event
- **x** – x position
- **y** – y position
- **flags** – mouse flag to use in control (it represents clicks)

- **self** – Plotim object

#### Returns

**plotatpointer** (*items, img=None, x=0, y=0, flag=6, xpad=0, ypad=0, bgrcolor=None, alpha=None, pixel=None*)

Plot message where mouse pointer is.

#### Parameters

- **items** – list of items supported by `self.makeoverlay()`
- **img** – image to place in items. If None it uses `self.remg`
- **x** – x position
- **y** – y position
- **flag** – flag for position (default=0).
  - `flag==0` : foreground to left up.
  - `flag==1` : foreground to left down.
  - `flag==2` : foreground to right up.
  - `flag==3` : foreground to right down.
  - `flag==4` : foreground at center of background.
  - `flag==5` : XY 0,0 is at center of background.
  - `flag==6` : XY 0,0 is at center of foreground.
  - `flag==7` : XY 0,0 is at right down of foreground.
- **xpad** – padding in x
- **ypad** – padding in y
- **bgrcolor** – background color
- **alpha** – alpha mask or value for transparency
- **pixel** – color to add as item in items,

#### Returns

**plotatxy** (*items, img=None, x=0, y=0, flag=0, xpad=0, ypad=0, bgrcolor=None, alpha=None*)

Plot message in xy position.

#### Parameters

- **items** – list of items supported by `makeoverlay()`
- **img** – image to place in items. If None it uses `self.remg`
- **x** – x position
- **y** – y position
- **flag** – flag for position (default=0).
  - `flag==0` : foreground to left up.
  - `flag==1` : foreground to left down.
  - `flag==2` : foreground to right up.
  - `flag==3` : foreground to right down.

- `flag==4` : foreground at center of background.
- `flag==5` : XY 0,0 is at center of background.
- `flag==6` : XY 0,0 is at center of foreground.
- `flag==7` : XY 0,0 is at right down of foreground.

- **xpad** – padding in x
- **ypad** – padding in y
- **bgrcolor** – background color
- **alpha** – alpha mask or value for transparency

**Returns**

**plotintime** (*items=None, wait=2, img=None, bgrcolor=None*)  
plots messages and events.

**Parameters**

- **items** – list of items supported by `makeoverlay()`
- **wait** – time of message.
- **img** – image to place in items. If None it uses `self.remg`
- **bgrcolor** – color of message.

**Returns**

**real2render** (*x, y, astype=None*)  
from real coordinates get rendered coordinates.

**Parameters**

- **x** – real x
- **y** – real y
- **astype** – (np.int32) return as the specified type

**Returns** rendered x, rendered y

**render2real** (*rx, ry, astype=<type 'numpy.int32'>*)  
from rendered coordinates get real coordinates.

**Parameters**

- **rx** – rendered x
- **ry** – rendered y
- **astype** – (np.int32) return as the specified type

**Returns** real x, real y

**rx1**

**rx2**

**ry1**

**ry2**



**save** (*strname=None, ext='.png', name='img'*)

Save image (save image if not Qt backend is installed) :param strname: name to save, a label with {win} can be used to be replaced with the plot win name :param ext: (".png") extension. :param name: ("img") name of image object from self. default is "img" that is self.img

(it allows better control to get custom image)

**Returns** True if saved, False if not saved (possibly because folder does not exists)

**show** (*frames=None, block=True, daemon=False, clean=True*)

Show function. calls buildinwindow, handles key presses and close events.

#### Parameters

- **frames** – show number of frames and close.
- **block** – if True it wait for window close, else it detaches (Experimental)
- **daemon** – if True window closes if main thread ends, else windows must be closed to main thread to end (Experimental)

#### Returns

**showfunc** (*img=None*)

Decoupled show function for Plotim (replace self.showfunc).

#### Parameters

- **self** – Plotim instance
- **img** – image to show

**textbackground**

**updaterenderer** (*img=None, zoom=True*)

update renderer when called.

#### Parameters

- **img** – image to update in renderer, if None use self.img
- **zoom** – True to enable zoom, else updates with original img.

#### Returns

None

**windowfunc** ()

Decoupled window function for Plotim (replace self.windowfunc).

#### Parameters

**self** – Plotim instance

RRtoolbox.lib.plotter.**background** (*color, x=1, y=1, flag=0*)

Creates background rectangle.

#### Parameters

- **color** – main color.
- **x** – x pixels in axis x.
- **y** – y pixels in axis y.
- **flag** – Not implemented.

**Returns** image of shape y,x and ndim == color.ndim.

RRtoolbox.lib.plotter.**convert2bgr** (*src, bgrcolor=None*)

Tries to convert any image format to BGR.

**Parameters**

- **src** – source image.
- **bgrcolor** – background or transparent color.

**Returns** BGR array image.

`RRtoolbox.lib.plotter.convert2bgra(src, bgrcolor=None, transparency=None)`  
Tries to convert any image format to BGRA.

**Parameters**

- **src** – source image.
- **bgrcolor** – background or transparent color.
- **transparency** – mask or A channel. (typically source image has not A channel, so user can provide it)

**Returns** BGRA array image.

`RRtoolbox.lib.plotter.echo(obj)`  
Printer (used when user wants to print an object from Plotim) :param obj: object

`RRtoolbox.lib.plotter.fastplt(image, cmap=None, title='visualazor', win=None, block=False, daemon=False)`

Fast plot.

**Parameters**

- **image** – image to show
- **cmap** – “gray” or None
- **title** – title of subplot
- **win** – title of window
- **block** – if True it wait for window close, else it detaches (Experimental)
- **daemon** – if True window closes if main thread ends, else windows must be closed to main thread to end (Experimental)

**Returns** plt

`RRtoolbox.lib.plotter.graph_filter(filters, levels=None, titles=None, win=None, single=True, legend=True, annotate=True, cols=3, scale=0.07, show=True, lxp=None, lyp=None)`

Graph filter with standard data to watch response.

**Parameters**

- **filters** – list of filters
- **levels** – numpy array with values. if None tries to fit data or assumes from 0 to 255
- **titles** – list of titles for each filter in filters. if None creates the titles
- **win** – window name
- **single** – True to plot all filters in one plot. else separate each filter in a plot.
- **legend** – True to add legends.
- **annotate** – True to add annotations.
- **cols** – number of columns to create plots

- **scale** – factor from maximum to draw annotations
- **show** – to show the figure

**Returns** figure

`RRtoolbox.lib.plotter.limitaxis(c, maxc, minc=0)`

Limit value in axis.

**Parameters**

- **c** – value
- **maxc** – max c value.
- **minc** – min c value.

**Returns** limited c value c E [minc,maxc]

`RRtoolbox.lib.plotter.plotPointsContour(pts, ax=None, lcor='k', pcor=None, deg=None, annotate=True, width=0.004, label='pt{pt}({x}, {y}, {a})', arrowprops=None)`

Plots points and joining lines in axes.

**Parameters**

- **pts** – points. [(x0,y0)...(xN,yN)]
- **ax** – axes handle to draw points.
- **lcor** – color of joining lines.
- **pcor** – color of points. If specified uses lines, else vectors.
- **deg** – angle of vertex, if True in degrees, if False in radians, if None do not add.
- **annotate** – (True) True to annotate
- **width** – adjust width of lines
- **label** – string to format point labels. add the point with {pt}, x and y coordinates with {x} and {y}, and angle with {a}. By default label is 'pt{pt}({x}, {y}, {a})'.
- **arrowprops** – dictionary to modify array properties

**Returns** axes

## RRtoolbox.lib.root module

This module holds core-like methods for library modules but not for the hole package

`class RRtoolbox.lib.root.ControlStdout(disable=True, buffer=None)`

Bases: object

Context manager to control output to stdout

**Parameters**

- **disable** – if True suppress output.
- **buffer** – (None) if True creates a buffer to collect all data printed to the stdout which can be retrieved with self.buffered. A file can be given but if it is write-only it cannot retrieve data to self.buffered so “w+” is recommended to be used with self.buffered.

**Warning:** If a references to `sys.stdout` is kept before the `ControlStdout` instance then output can be printed trough it and cannot be controlled by the `ControlStdout` context.

**class** `RRtoolbox.lib.root.FactorConvert` (*factor=None, abbreviate=True*)

Bases: `object`

Keep track of factor and converts to any available factor.

**convert** (*factor, to=None*)

Convert from actual factor to another factor.

**Parameters**

- **factor** – number
- **to** – factor to convert

**Returns** converted value, units

**convert2sample** (*factor, to=None*)

Convert to resemble sample.

**Parameters**

- **factor** – number
- **to** – sample factor.

**Returns** converted value, units

**exactFactorIndex** (*key*)

Find the index of a factor that contains a key.

**Parameters** **key** – anything to look in factors (i.e. factor name, factor value, abbreviation).

**Returns** factor structure, else None.

**factor**

**factors**

**getFactor** (*key*)

Tries to find factor value in factors.

**Parameters** **key** – anything to look in factors (i.e. factor name, factor value, abbreviation). If **key** is a factor value it will look for the nearest factor value.

**Returns** factor structure, else raises error.

**nearFactorIndex** (*factor*)

Find the index of nearest factor value.

**Parameters** **factor** – factor value.

**Returns** factor structure near factor value.

**static parts** (*value, precision=4*)

Get number parts.

**Parameters**

- **value** – number
- **precision** – decimal precision

**Returns** ([... ,Hundreds, Tens, Ones],[Tenths, ...])

**static split** (*value*)  
Get number fraction.

**Parameters** *value* – number

**Returns** integer, fraction

**class** `RRtoolbox.lib.root.Magnitude` (*value=0, factor=None, unit=None, precision=None, abbreviate=False*)

Bases: object

**format\_value** (*value*)

**class** `RRtoolbox.lib.root.Namespace`

Bases: object

used to store variables

**exception** `RRtoolbox.lib.root.NoParserFound`

Bases: `exceptions.Exception`

**exception** `RRtoolbox.lib.root.NotCallable`

Bases: `exceptions.Exception`

Defines objectGetter error: given object is not callable.

**exception** `RRtoolbox.lib.root.NotCreatable`

Bases: `exceptions.Exception`

Defines objectGetter error: objectGetter cannot create new object.

**class** `RRtoolbox.lib.root.Profiler` (*msg=None, tag=None*)

Bases: object

profiler for code points

**param** *msg* custom comment for profiling point

**param** *tag* classification tag

**parameter** *space* ("" )

**parameter** *format\_line* (“{space}{tag}{msg}{time}”)

**parameter** *format\_structure* (“

{space}[[{tag}{msg}{time}]{child}]{side}”)

**parameter** *points* profile instaces which are divided in “side” or “children” points according if they are side by side or are inside of the profiler.

**close** ()

close profiler and all their points

**formatter** (*level, tag, msg, time*)

format profiling point arguments.

**Parameters**

- **level** –
- **tag** – classification tag
- **msg** – custom comment of profiling point
- **time** – time of profiling

**Returns** formatted (spacing, tag, msg, time)

**lines\_formatted** (*collapse=None*)  
generate string lines

**Parameters** **collapse** – list for collapsing repeated tags or messages.

**Returns** list of lines

**lines\_unformatted** (*collapse=None*)  
generate structure lines

**Parameters** **collapse** – list for collapsing repeated tags or messages.

**Returns** generator with outputs (level, tag, msg, time)

**open\_point** (*msg=None, tag=None*)  
Open a profiling point to track time.

**Parameters**

- **msg** – custom comment for profiling point
- **tag** – classification tag

**Returns**

**restructure** (*structure, collapse*)  
reprocess an already created structure.

**Parameters**

- **structure** – structure.
- **collapse** – list for collapsing repeated tags or messages.

**Returns** reprocessed structure

**string\_lines** ()  
string with plain structure of profiling

**string\_structured** (*collapse=None, structure=None*)  
string with plain structure of profiling

**Parameters**

- **collapse** – list for collapsing repeated tags or messages.
- **structure** – (None) uses and already created structure. If None it creates the structure.

**Returns** string

**structure** (*collapse=None*)  
profiling structure.

**Parameters** **collapse** – list for collapsing repeated tags or messages.

**Returns** structure with format [tag,msg,time,children]

**time**

**Returns** overall time of profiling

**class** `RRtoolbox.lib.root.StdoutLOG` (*path, mode='w+', chain=False*)  
simple logger to save stdout output so anything printed in the console is logged to a file.

**Parameters**

- **path** – path to logging file

- **mode** – mode for opening the file.
- **chain** – if True closes previous logs and continues with new log

```
close (**kwargs)
flush (**kwargs)
printline (text, **kwargs)
printlines (lines, **kwargs)
write (text, **kwargs)
```

**class** RRtoolbox.lib.root.**StdoutMULTI** (*filelist*)

Enclose several file-like objects.

:param filelist = list of file-like objects

```
close (**kwargs)
flush (**kwargs)
printline (text, **kwargs)
printlines (lines, **kwargs)
write (text, **kwargs)
```

**class** RRtoolbox.lib.root.**StdoutSIM** (*disable=False, stdout=None*)

simple logger to simulate stdout output

```
close ()
flush ()
printline (text, **kwargs)
printlines (lines, **kwargs)
write (text, **kwargs)
```

**class** RRtoolbox.lib.root.**TimeCode** (*msg=None, factor=None, precision=None, abv=None, endmsg='{time}\n', enableMsg=True, printfunc=None, profiler=None, profile\_point=None*)

Bases: object

Context to profile code by printing a prelude and prologue with time.

#### Parameters

- **msg** – prelude or description message
- **factor** – factor supported by FactorConvert class
- **precision** – number of digits after a float point
- **abv** – if True prints “s”, if False “seconds” for time
- **endmsg** – prologue message
- **enableMsg** – (True) A flag specifying if context should be printed or not.
- **printfunc** – function to print messages. By default it is sys.stdout.write

```
time
time_end
```

**exception** `RRtoolbox.lib.root.TimeoutException`

Bases: `exceptions.Exception`

**exception** `RRtoolbox.lib.root.TransferException`

Bases: `exceptions.Exception`

**exception** `RRtoolbox.lib.root.VariableNotDeletable`

Bases: `exceptions.Exception`

**exception** `RRtoolbox.lib.root.VariableNotSettable`

Bases: `exceptions.Exception`

`RRtoolbox.lib.root.addto` (*instance, funcname=None*)

Decorator: Add function as method to instance.

#### Parameters

- **instance** – class instance.
- **funcname** – name to register in instance.

#### Returns

`RRtoolbox.lib.root.decorateInstanceMethods` (*self, decorator, excludeMth='\_\_init\_\_', includeMth=None*)

Decorate methods in an instance. It should be used in the `__init__` method of a class.

#### Parameters

- **self** – class instance.
- **decorator** – decorator function to apply to self.
- **excludeMth** – list of methods to exclude.
- **includeMth** – list of methods to include if not in exclude. if `excludeMth` is `None` then `decorateInstanceMethods` checks for `includeMth` list. if `includeMth` and `excludeMth` is `None` then all methods of `self` are decorated.

#### Returns

`self`

---

**Note:** It must be used at instance initialization (i.e. inside `__init__` method)

---

`RRtoolbox.lib.root.ensureList` (*obj*)

ensures that object is list

`RRtoolbox.lib.root.formatConsume` (*format\_string, kwargs, formatter=None, handle=None*)

Format with dictionary and consume keys.

#### Parameters

- **format\_string** – string to format
- **kwargs** – dictionary containing the keys and values to format string. The keys must be supported by the string formatter
- **formatter** – (None) formatter function to format string

#### Returns

formatted string

`RRtoolbox.lib.root.formatOnly` (*format\_string, \*\*kwargs*)

Format string only with provided keys

#### Parameters



- **format\_string** – string to format
- **kwargs** – format keys

**Returns** formatted string

`RRtoolbox.lib.root.glob(path, contents='*', check=<function isfile>)`

Return a list of paths matching a pathname pattern with valid files.

**Parameters**

- **path** – path to process ing glob filter
- **contents** – If path is a folder then looks for contents using
- **check** – function to filter contents. it must receive the path and return True to let it pass and False to suppress it.

**Returns** return list of files

`class RRtoolbox.lib.root.globFilter(include=None, exclude=None, case=False)`

Bases: object

glob filter for patterns

`RRtoolbox.lib.root.lookinglob(pattern, path=None, ext=None, forward=None, filelist=None, aslist=False, raiseErr=False)`

Look for patterns in Path. It looks as {if path}{if pattern}{if forward}{if ext}.

**Parameters**

- **pattern** – string to look for pattern.
- **path** – (None) path to look for pattern
- **ext** – (None) extension of pattern in path
- **forward** – (None) look changes after pattern and before ext parameter.
- **filelist** – (None) simulates the files in path and look patterns in this list.
- **aslist** – (False) if False it returns the first match case string else the list of matching cases.
- **raiseErr** – If true raises Exception if patter is not found in path or there are more than one match

**Returns** matched case if returnAll is False else the list of matched cases or if no match is found None

## RRtoolbox.lib.serverServices module

`class RRtoolbox.lib.serverServices.Conection(conn)`

represent a connection to interchange objects between servers and clients.

`getLen(timeout=None)`

`rcv()`

`recvall()`

`send(obj)`

`sendLen(length, timeout=None)`

`RRtoolbox.lib.serverServices.generateServer` (*host='localhost', to=63342*)  
generates a simple Server in available address.

**Parameters** `to` – until port.

**Returns** socket, address

`RRtoolbox.lib.serverServices.initClient` (*addr, timeout=None*)  
Inits a simple client from address. :param `addr`: (host, port) :return: socket

`RRtoolbox.lib.serverServices.initServer` (*addr*)  
Inits a simple server from address.

**Parameters** `addr` – (host, port)

**Returns** socket

`RRtoolbox.lib.serverServices.parseString` (*string, timeout=3*)

**Parameters**

- `string` –
- `timeout` –

**Returns**

`RRtoolbox.lib.serverServices.ping` (*host, port*)  
Ping to.

**Parameters**

- `host` – IP address
- `port` – port address

**Returns**

`RRtoolbox.lib.serverServices.rcvPickle` (*addr=('localhost', 50007), timeout=None*)  
Receive potentially any data using sockets.

**Parameters**

- `addr` – socket or address.
- `timeout` – NotImplemented

**Returns** data, else throws error.

`RRtoolbox.lib.serverServices.recv_into` (*viewable, socket*)  
Receive from socket into viewable object.

**Parameters**

- `viewable` – viewable object
- `socket` – source socket

**Returns** None

`RRtoolbox.lib.serverServices.scan_ports` (*host*)  
Scan opened ports in address.

**Parameters** `host` – host IP to filter opened ports.

**Returns** generator

RRtoolbox.lib.serverServices.**sendPickle**(*obj*, *addr*=('localhost', 50007), *timeout*=None, *threaded*=False)

Send potentially any data using sockets.

**Parameters**

- **obj** – packable object.
- **addr** – socket or address.
- **timeout** – NotImplemented

**Returns** True if sent successfully, else Throw error.

RRtoolbox.lib.serverServices.**send\_from**(*viewable*, *socket*)

Send from viewable object.

**Parameters**

- **viewable** – viewable object
- **socket** – destine socket

**Returns** None

RRtoolbox.lib.serverServices.**string\_is\_socket\_address**(*string*)

## RRtoolbox.lib.session module

This module have serializing methods for data persistence so to let the package “save” custom objects

session module made by Davtohi and powered by dill Dependency project: <https://github.com/uqfoundation/dill>

RRtoolbox.lib.session.**checkFromSession**(*filepath*, *varlist*)

Check that variables exists in session file.

**Parameters**

- **filepath** – path to session file.
- **varlist** – list of variables to checkLoaded.

**Returns** list checkLoaded results

RRtoolbox.lib.session.**deleteFromSession**(*filepath*, *varlist*)

Delete variables from session file.

**Parameters**

- **filepath** – path to session file.
- **varlist** – list of variables to delete.

**Returns** None

RRtoolbox.lib.session.**flushSession**(*filepath*)

Empty session in file.

**Parameters** **filepath** – path to session file.

**Returns**

RRtoolbox.lib.session.**getEnvironmentSession**(*environment*=None)

Gets the filtered session from the global variables.

**Returns** dictionary containing filtered session.

`RRtoolbox.lib.session.readSession(filepath, helper=None)`

Loads a dictionary session from file.

**Parameters**

- **filepath** – path to load session file.
- **helper** – function to pos-process session file

**Returns** session

`RRtoolbox.lib.session.saveSession(filepath, session, helper=None)`

Saves dictionary session to file.

**Parameters**

- **filepath** – path to save session file.
- **session** – dictionary
- **helper** – function to pre-process session

**Returns** filename of saved session

`RRtoolbox.lib.session.updateSession(filepath, session, replace=True, rdhelper=None, svhelper=None)`

Updates a dictionary session in file.

**Parameters**

- **filepath** – path to session file.
- **session** – dictionary.
- **replace** – if True key values are replaced else old key values are kept.
- **rdhelper** – read helper.
- **svhelper** – save helper.

**Returns** None

## Module contents

This module contains core-like, too-much-used and too-much-referenced modules

### 1.1.2 RRtoolbox.tools package

#### Submodules

#### RRtoolbox.tools.lens module

`RRtoolbox.tools.lens.drawCircle(array, cnt, color=0)`

project circle over array.

**Parameters**

- **array** – array to draw circle
- **cnt** – contours of segmentation to fit circle
- **color** – color of lens

**Returns** array

`RRtoolbox.tools.lens.drawEllipse(array, cnt, color=0)`  
 project ellipse over array.

**Parameters**

- **array** – array to draw ellipse
- **cnt** – contours of segmentation to fit ellipse
- **color** – color of lens

**Returns** array

`RRtoolbox.tools.lens.fitLens(img, mask, color=0, asEllipse=False, addmask=False)`  
 Place lens-like object in image.

**Parameters**

- **img** – image to place lens
- **mask** – mask to fit lens
- **color** – color of the lens
- **asEllipse** – True to fit lens as a ellipse, False to fit circle.
- **addmask** – return additional mask parameter

**Returns** image with simulated lens

`RRtoolbox.tools.lens.simulateLens(img, threshfunc=None, pshape=(300, 300), color=0, asEllipse=True)`  
 Place lens-like object in image.

**Parameters**

- **img** – image to place lens.
- **threshfunc** – function to segment retinal area and get its mask.
- **pshape** – shape to resize processing image to increase performance.
- **color** – color of the lens.
- **asEllipse** – True to fit lens as a ellipse, False to fit circle.

**Returns** image with simulated lens.

## RRtoolbox.tools.segmentation module

`RRtoolbox.tools.segmentation.find_optic_disc_watershed(img, P)`  
 Find optic disk in image using a watershed method.

**Parameters**

- **img** – BGR image
- **P** – gray image

**Returns** optic\_disc, Crs, markers, watershed

`RRtoolbox.tools.segmentation.get_beta_params_Otsu(P)`  
 Automatically find parameters for alpha masks using Otsu threshold value.

**Parameters** **P** – gray image

**Returns** beta1 for minimum histogram value, beta2 for Otsu value

`RRtoolbox.tools.segmentation.get_beta_params_hist(P)`

Automatically find parameters for bright alpha masks using a histogram analysis method.

**Parameters** *P* – gray image

**Returns** beta1 for minimum valley left of body, beta2 for brightest valley right of body where the body starts at the tallest peak in the histogram.

`RRtoolbox.tools.segmentation.get_bright_alpha(backgray,foregray,window=None)`

Get alpha transparency for merging foreground to background gray image according to brightness.

**Parameters**

- **backgray** – background image. (as float)
- **foregray** – foreground image. (as float)
- **window** – window used to customizing alfa. It can be a binary or alpha mask, values go from 0 for transparency to any value where the maximum is visible i.e a window with all the same values does nothing. A binary mask can be used, where 0 is transparent and 1 is visible. If not window is given alfa is not altered and the intended alpha is returned.

**Returns** alfa mask

`RRtoolbox.tools.segmentation.get_layered_alpha(back,fore)`

Get bright alpha mask (using Otsu method)

**Parameters**

- **back** – BGR background image
- **fore** – BGR foreground image

**Returns** alpha mask

`RRtoolbox.tools.segmentation.layeredfloods(img, gray=None, backmask=None, step=1, connectivity=4, weight=False)`

Create an alpha mask from an image using a weighted layered flooding algorithm,

**Parameters**

- **img** – BGR image
- **gray** – Gray image
- **backmask** – background mask
- **step** – step to increase upDiff in the floodFill algorithm. If weight is True step also increases the weight of the layers.
- **connectivity** – pixel connectivity of 4 or 8 to use in the floodFill algorithm
- **weight** – Increase progressively the weight of the layers using the step parameter.

**Returns** alpha mask

`RRtoolbox.tools.segmentation.retina_markers_thresh(P)`

Retinal markers thresholds to find background, retinal area and optic disc with flares based in the histogram.

**Parameters** *P* – gray image

**Returns** min,b1,b2,max

`RRtoolbox.tools.segmentation.retinal_mask(img, biggest=False, addalpha=False)`

Obtain the mask of the retinal area in an image. For a simpler and lightweight algorithm see [`retinal\_mask\_watershed\(\)`](#).

**Parameters**

- **img** – BGR or gray image
- **biggest** – True to return only biggest object
- **addalpha** – True to add additional alpha mask parameter

**Returns**

**if addalpha:** binary mask, alpha mask

**else:** binary mask

```
RRtoolbox.tools.segmentation.retinal_mask_watershed(img, parameters=(10, 30, None),
                                                    addMarkers=False)
```

Quick and simple watershed method to obtain the mask of the retinal area in an image. For a more robust algorithm see [retinal\\_mask\(\)](#).

**Parameters**

- **img** – BGR or gray image
- **parameters** – tuple of parameters to pass to `filterFactory()`
- **addMarkers** – True to add additional Marker mask. It contains 0 for unknown areas, 1 for background and 2 for retinal area.

**Returns**

**if addMarkers:** binary mask, Markers mask

**else:** binary mask

**RRtoolbox.tools.selectors module**

```
class RRtoolbox.tools.selectors.EntropyPlot(images, win='Entropy tests', func=None)
```

Bases: [RRtoolbox.lib.plotter.Plotim](#)

Plot entropy test

**getData** (*im*)

**getImage** (*im*)

**keyfunc** ()

**nextim** ()

**previousim** ()

**selectlist** (*imlist*)

```
RRtoolbox.tools.selectors.entropy(imlist, loadfunc=None, invert=False)
```

Entropy function modified from:

Yan Liu, Feihong Yu, An automatic image fusion algorithm for unregistered multiply multi-focus images, Optics Communications, Volume 341, 15 April 2015, Pages 101-113, ISSN 0030-4018, <http://dx.doi.org/10.1016/j.optcom.2014.12.015>. (<http://www.sciencedirect.com/science/article/pii/S0030401814011559>)

**Parameters** **imlist** – list of path to images or arrays

**Returns** sortedD,sortedImlist,D,fns

where **sortedD** is the ranking of the Entropy test, **D = [D0,...,DN]** **D0>DN** **sortedImlist** is fns sorted to match **sortedD**, **D** is the list of the absolute difference between entropy and the root mean square, **D = |E-RMS|**

`RRtoolbox.tools.selectors.hist_comp(imlist, loadfunc=None, method='correlation')`  
Histogram comparison

**Parameters** **imlist** – list of path to images or arrays

**Returns** comparison

## RRtoolbox.tools.sticher module

`RRtoolbox.tools.sticher.stich(**opts)`

## Module contents

## 1.2 Submodules

## 1.3 RRtoolbox.core module

`RRtoolbox.core.f(*args, **kwargs)`

**class** `RRtoolbox.core.rrbox(*args)`

Bases: object

**asift** (*fn*)

`RRtoolbox.core.tools(instance, modules)`

`RRtoolbox.core.tools2(instance, modules)`

## 1.4 RRtoolbox.run module

## 1.5 RRtoolbox.shell module

**class** `RRtoolbox.shell.Shell`

**generateParser** (*func*)

**getParser** (*func*)

**parse** (*func, args=None, namespace=None*)

**parser\_fastplt** ()

**parser\_loadFunc** ()

`RRtoolbox.shell.getDocParamLines(doc)`

gets each parameter line from reStructured doc.

**Parameters** **doc** – documentation

**Returns** lines



`RRtoolbox.shell.getDocParameters` (*doc*)  
gets param and comment from reStructured doc.

**Parameters** *doc* – documentation

**Returns** list of (param,comment) items.

`RRtoolbox.shell.shell_processor` (*commands*)

`RRtoolbox.shell.shell_processor_parser` (*syslist, flags='', longopts=('feature=', 'nnn=')*)

`RRtoolbox.shell.string_interpreter` (*empty=None, commahandler=None, handle=None*)  
create a string interpreter :param empty: (None) variable to handle empty strings :param commahandler: (tuple\_creator) function to handle comma separated strings :return: interpreter function

`RRtoolbox.shell.tuple_creator` (*string*)

Process string to get tuple.

**Parameters** *string* – string parameters with “,” (colon) as separator Ex: param1, param2, ..., paramN

**Returns** tuple

## 1.6 Module contents



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## r

- RRtoolbox, [77](#)
- RRtoolbox.core, [76](#)
- RRtoolbox.lib, [72](#)
- RRtoolbox.lib.arrayops, [26](#)
- RRtoolbox.lib.arrayops.basic, [3](#)
- RRtoolbox.lib.arrayops.convert, [18](#)
- RRtoolbox.lib.arrayops.filters, [21](#)
- RRtoolbox.lib.arrayops.mask, [24](#)
- RRtoolbox.lib.cache, [26](#)
- RRtoolbox.lib.config, [31](#)
- RRtoolbox.lib.descriptors, [33](#)
- RRtoolbox.lib.directory, [35](#)
- RRtoolbox.lib.image, [41](#)
- RRtoolbox.lib.inspector, [53](#)
- RRtoolbox.lib.plotter, [54](#)
- RRtoolbox.lib.root, [63](#)
- RRtoolbox.lib.serverServices, [69](#)
- RRtoolbox.lib.session, [71](#)
- RRtoolbox.run, [76](#)
- RRtoolbox.shell, [76](#)
- RRtoolbox.tools, [76](#)
- RRtoolbox.tools.lens, [72](#)
- RRtoolbox.tools.segmentation, [73](#)
- RRtoolbox.tools.selectors, [75](#)
- RRtoolbox.tools.sticher, [76](#)



## A

addto() (in module RRtoolbox.lib.root), 68  
 affine\_skew() (in module RRtoolbox.lib.descriptors), 34  
 all (RRtoolbox.lib.cache.ResourceManager attribute), 29  
 alpha (RRtoolbox.lib.arrayops.filters.FilterBase attribute), 22  
 angle() (in module RRtoolbox.lib.arrayops.basic), 3  
 angle2D() (in module RRtoolbox.lib.arrayops.basic), 3  
 angleXY() (in module RRtoolbox.lib.arrayops.basic), 4  
 anorm() (in module RRtoolbox.lib.arrayops.basic), 4  
 anorm2() (in module RRtoolbox.lib.arrayops.basic), 4  
 apply2kp\_pairs() (in module RRtoolbox.lib.arrayops.convert), 18  
 applythresh() (RRtoolbox.lib.plotter.Imtester static method), 55  
 ASIFT() (in module RRtoolbox.lib.descriptors), 33  
 asift() (RRtoolbox.core.rrbox method), 76  
 ASIFT\_iter() (in module RRtoolbox.lib.descriptors), 33  
 ASIFT\_multiple() (in module RRtoolbox.lib.descriptors), 33  
 Asynchronous (class in RRtoolbox.lib.inspector), 53  
 axesIntercept() (in module RRtoolbox.lib.arrayops.basic), 4

## B

background() (in module RRtoolbox.lib.arrayops.mask), 24  
 background() (in module RRtoolbox.lib.plotter), 61  
 Bandpass (class in RRtoolbox.lib.arrayops.filters), 21  
 Bandstop (class in RRtoolbox.lib.arrayops.filters), 21  
 beta1 (RRtoolbox.lib.arrayops.filters.FilterBase attribute), 22  
 beta2 (RRtoolbox.lib.arrayops.filters.FilterBase attribute), 22  
 BGR (RRtoolbox.lib.image.Image attribute), 44  
 BGRA (RRtoolbox.lib.image.Image attribute), 44  
 bgra2bgr() (in module RRtoolbox.lib.image), 45  
 biggestCnt() (in module RRtoolbox.lib.arrayops.mask), 24

biggestCntData() (in module RRtoolbox.lib.arrayops.mask), 24  
 bilateralFilter() (in module RRtoolbox.lib.arrayops.filters), 22  
 BilateralParameters (class in RRtoolbox.lib.arrayops.filters), 21  
 BilateralParameter (class in RRtoolbox.lib.arrayops.filters), 21  
 boxPads() (in module RRtoolbox.lib.arrayops.basic), 5  
 brightness() (in module RRtoolbox.lib.arrayops.mask), 24  
 broadcast() (RRtoolbox.lib.inspector.Logger method), 54  
 builtincmd() (RRtoolbox.lib.plotter.Imtester method), 55  
 builtincmd() (RRtoolbox.lib.plotter.Plotim method), 57  
 builtincontrol() (RRtoolbox.lib.plotter.Plotim method), 57  
 builtinplot() (RRtoolbox.lib.plotter.Plotim method), 57  
 builtinwindow() (RRtoolbox.lib.plotter.Plotim method), 57  
 bytes2units() (RRtoolbox.lib.cache.ResourceManager method), 29

## C

Cache (class in RRtoolbox.lib.cache), 26  
 cachedir (RRtoolbox.lib.cache.DynamicMemoizedFunc attribute), 27  
 cachedProperty() (in module RRtoolbox.lib.cache), 31  
 call\_and\_shelve() (RRtoolbox.lib.cache.DynamicMemoizedFunc method), 27  
 centerM() (in module RRtoolbox.lib.arrayops.basic), 5  
 centerS() (in module RRtoolbox.lib.arrayops.basic), 5  
 centerSM() (in module RRtoolbox.lib.arrayops.basic), 5  
 changedir() (in module RRtoolbox.lib.directory), 37  
 checkDir() (in module RRtoolbox.lib.directory), 37  
 checkFile() (in module RRtoolbox.lib.directory), 37  
 checkFromSession() (in module RRtoolbox.lib.session), 71  
 checkLoaded() (in module RRtoolbox.lib.image), 45  
 checkPath() (in module RRtoolbox.lib.directory), 37  
 checkURL() (in module RRtoolbox.lib.directory), 37  
 clean() (RRtoolbox.lib.plotter.Plotim method), 57

- clear() (RRtoolbox.lib.cache.DynamicMemoizedFunc method), 27
- clear() (RRtoolbox.lib.cache.MemoizedDict method), 27
- close() (RRtoolbox.lib.root.Profiler method), 65
- close() (RRtoolbox.lib.root.StdoutLOG method), 67
- close() (RRtoolbox.lib.root.StdoutMULTI method), 67
- close() (RRtoolbox.lib.root.StdoutSIM method), 67
- closefunc() (RRtoolbox.lib.plotter.Plotim method), 57
- cmdfunc() (RRtoolbox.lib.plotter.Plotim method), 57
- cnt2pts() (in module RRtoolbox.lib.arrayops.convert), 18
- cnt\_hist() (in module RRtoolbox.lib.arrayops.mask), 25
- compress (RRtoolbox.lib.cache.DynamicMemoizedFunc attribute), 27
- computeAll() (RRtoolbox.lib.plotter.Edger method), 55
- computeEdge() (RRtoolbox.lib.plotter.Edger method), 55
- computefunc() (RRtoolbox.lib.plotter.Imtester method), 55
- Conection (class in RRtoolbox.lib.serverServices), 69
- config() (RRtoolbox.lib.descriptors.Feature method), 33
- ConfigTool (class in RRtoolbox.lib.config), 31
- contour2points() (in module RRtoolbox.lib.arrayops.convert), 18
- contours2mask() (in module RRtoolbox.lib.arrayops.basic), 5
- contoursArea() (in module RRtoolbox.lib.arrayops.basic), 5
- Controlstdout (class in RRtoolbox.lib.root), 63
- conv3H4H() (in module RRtoolbox.lib.arrayops.convert), 18
- convert() (RRtoolbox.lib.root.FactorConvert method), 64
- convert2bgr() (in module RRtoolbox.lib.plotter), 61
- convert2bgra() (in module RRtoolbox.lib.plotter), 62
- convert2sample() (RRtoolbox.lib.root.FactorConvert method), 64
- convertAs() (in module RRtoolbox.lib.image), 45
- convertXY() (in module RRtoolbox.lib.arrayops.basic), 5
- convexityRatio() (in module RRtoolbox.lib.arrayops.basic), 6
- coors (RRtoolbox.lib.image.GetCoors attribute), 41
- copy() (RRtoolbox.lib.directory.Directory method), 35
- correctPath() (in module RRtoolbox.lib.directory), 37
- correctSep() (in module RRtoolbox.lib.directory), 37
- correctSTRBuiltin() (RRtoolbox.lib.directory.Directory method), 36
- create() (RRtoolbox.lib.cache.ObjectGetter method), 28
- ## D
- d (RRtoolbox.lib.arrayops.filters.BilateralParameters attribute), 22
- decorateInstanceMethods() (in module RRtoolbox.lib.root), 68
- decoratePath() (in module RRtoolbox.lib.directory), 38
- default (RRtoolbox.lib.config.DirectoryManager attribute), 32
- deleteFromSession() (in module RRtoolbox.lib.session), 71
- detectAndCompute() (RRtoolbox.lib.descriptors.Feature method), 34
- detectType() (RRtoolbox.lib.plotter.Imtester method), 55
- dict2keyPoint() (in module RRtoolbox.lib.arrayops.convert), 18
- Directory (class in RRtoolbox.lib.directory), 35
- DirectoryManager (class in RRtoolbox.lib.config), 31
- done() (RRtoolbox.lib.inspector.Asynchronous method), 53
- done() (RRtoolbox.lib.inspector.GraphTraceOutput method), 53
- drawCircle() (in module RRtoolbox.tools.lens), 72
- drawcoorarea() (in module RRtoolbox.lib.image), 46
- drawcooraxes() (in module RRtoolbox.lib.image), 46
- drawcoorperspective() (in module RRtoolbox.lib.image), 46
- drawcoorpoints() (in module RRtoolbox.lib.image), 47
- drawcoorpolyArrow() (in module RRtoolbox.lib.image), 47
- drawcoorpolyline() (in module RRtoolbox.lib.image), 47
- drawEllipse() (in module RRtoolbox.tools.lens), 72
- drawline() (RRtoolbox.lib.plotter.MatchExplorer method), 56
- drawrelation() (RRtoolbox.lib.plotter.MatchExplorer method), 56
- drawstats() (RRtoolbox.lib.image.GetCoors method), 41
- dtype (RRtoolbox.lib.image.ImCoors attribute), 42
- DynamicMemoizedFunc (class in RRtoolbox.lib.cache), 27
- ## E
- echo() (in module RRtoolbox.lib.plotter), 62
- Edger (class in RRtoolbox.lib.plotter), 54
- enabled (RRtoolbox.lib.cache.DynamicMemoizedFunc attribute), 27
- ensureList() (in module RRtoolbox.lib.root), 68
- entropy() (in module RRtoolbox.tools.selectors), 75
- EntropyPlot (class in RRtoolbox.tools.selectors), 75
- entroyTest() (in module RRtoolbox.lib.arrayops.basic), 6
- errorbackground (RRtoolbox.lib.plotter.Plotim attribute), 58
- eventHandle (RRtoolbox.lib.inspector.Logger attribute), 54
- exactFactorIndex() (RRtoolbox.lib.root.FactorConvert method), 64
- ext (RRtoolbox.lib.image.Image attribute), 44
- ## F
- f() (in module RRtoolbox.core), 76
- factor (RRtoolbox.lib.root.FactorConvert attribute), 64
- FactorConvert (class in RRtoolbox.lib.root), 64
- factors (RRtoolbox.lib.root.FactorConvert attribute), 64



fastplt() (in module RRtoolbox.lib.plotter), 62  
 Feature (class in RRtoolbox.lib.descriptors), 33  
 fig2bgr() (in module RRtoolbox.lib.image), 47  
 fig2bgra() (in module RRtoolbox.lib.image), 48  
 file (RRtoolbox.lib.inspector.Logger attribute), 54  
 FileDirectory (class in RRtoolbox.lib.directory), 36  
 filter\_matches() (in module RRtoolbox.lib.descriptors), 34  
 FilterBase (class in RRtoolbox.lib.arrayops.filters), 22  
 filterdata() (RRtoolbox.lib.directory.Directory static method), 36  
 filterFactory() (in module RRtoolbox.lib.arrayops.filters), 22  
 filters (RRtoolbox.lib.arrayops.filters.BilateralParameters attribute), 22  
 find\_near() (in module RRtoolbox.lib.arrayops.basic), 6  
 find\_optic\_disc\_watershed() (in module RRtoolbox.tools.segmentation), 73  
 findmaxima() (in module RRtoolbox.lib.arrayops.basic), 6  
 findminima() (in module RRtoolbox.lib.arrayops.basic), 7  
 findModules() (in module RRtoolbox.lib.config), 32  
 fitLens() (in module RRtoolbox.tools.lens), 73  
 flush() (RRtoolbox.lib.root.StdoutLOG method), 67  
 flush() (RRtoolbox.lib.root.StdoutMULTI method), 67  
 flush() (RRtoolbox.lib.root.StdoutSIM method), 67  
 flushSession() (in module RRtoolbox.lib.session), 71  
 foreground() (in module RRtoolbox.lib.arrayops.mask), 25  
 format\_value() (RRtoolbox.lib.root.Magnitude method), 65  
 formatcmd() (RRtoolbox.lib.plotter.Plotim method), 58  
 formatConsume() (in module RRtoolbox.lib.root), 68  
 formatinfo() (RRtoolbox.lib.plotter.Imtester static method), 55  
 formatOnly() (in module RRtoolbox.lib.root), 68  
 formatter() (RRtoolbox.lib.root.Profiler method), 65  
 func (RRtoolbox.lib.cache.DynamicMemoizedFunc attribute), 27  
 funcData() (in module RRtoolbox.lib.inspector), 54

## G

generateParser() (RRtoolbox.shell.Shell method), 76  
 generateServer() (in module RRtoolbox.lib.serverServices), 69  
 get\_beta\_params\_hist() (in module RRtoolbox.tools.segmentation), 73  
 get\_beta\_params\_Otsu() (in module RRtoolbox.tools.segmentation), 73  
 get\_bright\_alpha() (in module RRtoolbox.tools.segmentation), 74  
 get\_code() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_conversionFunc() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_errorFunc() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_Func() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_layered\_alpha() (in module RRtoolbox.tools.segmentation), 74  
 get\_loadFunc() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_mapFunc() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_np2qi() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_resizeFunc() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_tracer\_class() (RRtoolbox.lib.inspector.GraphTrace method), 53  
 get\_transposeFunc() (RRtoolbox.lib.image.ImFactory method), 42  
 get\_x\_space() (in module RRtoolbox.lib.arrayops.basic), 7  
 getBilateralParameters() (in module RRtoolbox.lib.arrayops.filters), 23  
 getConfiguration() (RRtoolbox.lib.image.ImLoader method), 44  
 GetCoors (class in RRtoolbox.lib.image), 41  
 getcoors() (in module RRtoolbox.lib.image), 48  
 getData() (in module RRtoolbox.lib.directory), 38  
 getData() (RRtoolbox.tools.selectors.EntropyPlot method), 75  
 getdataVH() (in module RRtoolbox.lib.arrayops.basic), 7  
 getDocParameters() (in module RRtoolbox.shell), 76  
 getDocParamLines() (in module RRtoolbox.shell), 76  
 getEnviromentSession() (in module RRtoolbox.lib.session), 71  
 getFactor() (RRtoolbox.lib.root.FactorConvert method), 64  
 getFileHandle() (in module RRtoolbox.lib.directory), 38  
 getFileSize() (in module RRtoolbox.lib.directory), 38  
 getgeometrycoors() (in module RRtoolbox.lib.image), 48  
 gethull() (in module RRtoolbox.lib.arrayops.mask), 25  
 getImage() (RRtoolbox.tools.selectors.EntropyPlot method), 75  
 getLen() (RRtoolbox.lib.serverServices.Conection method), 69  
 getModules() (in module RRtoolbox.lib.config), 32  
 getObj() (RRtoolbox.lib.cache.ObjectGetter method), 28  
 getOtsuThresh() (in module RRtoolbox.lib.arrayops.basic), 7  
 getPackagePath() (in module RRtoolbox.lib.config), 32  
 getParameters() (RRtoolbox.lib.plotter.Edger method), 55  
 getParser() (RRtoolbox.shell.Shell method), 76  
 getPath() (in module RRtoolbox.lib.directory), 38  
 getrectcoors() (in module RRtoolbox.lib.image), 48  
 getSep() (in module RRtoolbox.lib.directory), 38  
 getShortenedPath() (in module RRtoolbox.lib.directory), 38

[getSOPointRelation\(\)](#) (in module `RRtoolbox.lib.arrayops.convert`), 19  
[getSplitted\(\)](#) (in module `RRtoolbox.lib.directory`), 39  
[getTools\(\)](#) (`RRtoolbox.lib.config.ConfigTool` static method), 31  
[getTransformedCorners\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 7  
[getTransparency\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 7  
[glob\(\)](#) (in module `RRtoolbox.lib.root`), 69  
[globFilter](#) (class in `RRtoolbox.lib.root`), 69  
[graph\\_filter\(\)](#) (in module `RRtoolbox.lib.plotter`), 62  
[GraphTrace](#) (class in `RRtoolbox.lib.inspector`), 53  
[GraphTraceOutput](#) (class in `RRtoolbox.lib.inspector`), 53  
[gray](#) (`RRtoolbox.lib.image.Image` attribute), 44  
[gray2qi\(\)](#) (in module `RRtoolbox.lib.image`), 48

## H

[help\(\)](#) (`RRtoolbox.lib.plotter.Plotim` method), 58  
[Highpass](#) (class in `RRtoolbox.lib.arrayops.filters`), 22  
[hist\\_cdf\(\)](#) (in module `RRtoolbox.lib.arrayops.mask`), 25  
[hist\\_comp\(\)](#) (in module `RRtoolbox.tools.selectors`), 76  
[hist\\_match\(\)](#) (in module `RRtoolbox.lib.image`), 48  
[histogram\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 8

## I

[ignore](#) (`RRtoolbox.lib.cache.DynamicMemoizedFunc` attribute), 27  
[ignore](#) (`RRtoolbox.lib.cache.Memoizer` attribute), 27  
[im2imFormat\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 8  
[im2shapeFormat\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 8  
[Image](#) (class in `RRtoolbox.lib.image`), 44  
[ImCoors](#) (class in `RRtoolbox.lib.image`), 41  
[ImFactory](#) (class in `RRtoolbox.lib.image`), 42  
[ImLoader](#) (class in `RRtoolbox.lib.image`), 42  
[Imtester](#) (class in `RRtoolbox.lib.plotter`), 55  
[increment\\_if\\_exists\(\)](#) (in module `RRtoolbox.lib.directory`), 39  
[init\(\)](#) (`RRtoolbox.lib.plotter.Plotim` method), 58  
[init\\_feature\(\)](#) (in module `RRtoolbox.lib.descriptors`), 35  
[initClient\(\)](#) (in module `RRtoolbox.lib.serverServices`), 70  
[initServer\(\)](#) (in module `RRtoolbox.lib.serverServices`), 70  
[inlineRatio\(\)](#) (in module `RRtoolbox.lib.descriptors`), 35  
[insert\(\)](#) (`RRtoolbox.lib.image.PathLoader` method), 45  
[instability\\_bf\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 8  
[interpretImage\(\)](#) (in module `RRtoolbox.lib.image`), 48  
[InvertedBandpass](#) (class in `RRtoolbox.lib.arrayops.filters`), 22

[InvertedBandstop](#) (class in `RRtoolbox.lib.arrayops.filters`), 22  
[invertH\(\)](#) (in module `RRtoolbox.lib.arrayops.convert`), 19  
[invertM\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 8  
[invertSM\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 8  
[isActiveWindow\(\)](#) (`RRtoolbox.lib.plotter.Edger` method), 55  
[isAlive\(\)](#) (`RRtoolbox.lib.cache.ObjectGetter` method), 29  
[isBFILTER](#) (`RRtoolbox.lib.plotter.Edger` attribute), 55  
[isCLAE](#) (`RRtoolbox.lib.plotter.Edger` attribute), 55  
[isCreatable\(\)](#) (`RRtoolbox.lib.cache.ObjectGetter` method), 29  
[isEQUA](#) (`RRtoolbox.lib.plotter.Edger` attribute), 55  
[isGettable\(\)](#) (`RRtoolbox.lib.cache.ObjectGetter` method), 29  
[isnumpy\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), 9  
[isSIZE](#) (`RRtoolbox.lib.plotter.Edger` attribute), 55

## J

[joinPath\(\)](#) (in module `RRtoolbox.lib.directory`), 39

## K

[keepAlive\(\)](#) (`RRtoolbox.lib.cache.ResourceManager` method), 29  
[keyfunc\(\)](#) (`RRtoolbox.lib.plotter.MatchExplorer` method), 56  
[keyfunc\(\)](#) (`RRtoolbox.lib.plotter.Plotim` method), 58  
[keyfunc\(\)](#) (`RRtoolbox.tools.selectors.EntropyPlot` method), 75  
[keyPoint2tuple\(\)](#) (in module `RRtoolbox.lib.arrayops.convert`), 19

## L

[layeredfloods\(\)](#) (in module `RRtoolbox.tools.segmentation`), 74  
[LazyDict](#) (class in `RRtoolbox.lib.cache`), 27  
[limitaxis\(\)](#) (in module `RRtoolbox.lib.plotter`), 63  
[limitaxispoints\(\)](#) (in module `RRtoolbox.lib.image`), 49  
[lines\\_formatted\(\)](#) (`RRtoolbox.lib.root.Profiler` method), 66  
[lines\\_unformatted\(\)](#) (`RRtoolbox.lib.root.Profiler` method), 66  
[load\(\)](#) (in module `RRtoolbox.lib.inspector`), 54  
[load\(\)](#) (`RRtoolbox.lib.config.DirectoryManager` method), 32  
[load\(\)](#) (`RRtoolbox.lib.image.Image` method), 44  
[load\(\)](#) (`RRtoolbox.lib.plotter.Edger` method), 55  
[loadcv\(\)](#) (in module `RRtoolbox.lib.image`), 50  
[LoaderDict](#) (class in `RRtoolbox.lib.image`), 44  
[loadFunc\(\)](#) (in module `RRtoolbox.lib.image`), 49  
[loadsfrom\(\)](#) (in module `RRtoolbox.lib.image`), 51  
[Logger](#) (class in `RRtoolbox.lib.inspector`), 53  
[lookinglob\(\)](#) (in module `RRtoolbox.lib.root`), 69  
[Lowpass](#) (class in `RRtoolbox.lib.arrayops.filters`), 22

## M

Magnitude (class in RRtoolbox.lib.root), 65  
 makeFile() (RRtoolbox.lib.directory.FileDirectory method), 37  
 makememory() (RRtoolbox.lib.cache.Memoizer method), 27  
 makeoverlay() (RRtoolbox.lib.plotter.Plotim method), 58  
 makeVis() (in module RRtoolbox.lib.arrayops.basic), 9  
 mapper() (in module RRtoolbox.lib.cache), 31  
 margin (RRtoolbox.lib.cache.ResourceManager attribute), 29  
 mask\_watershed() (in module RRtoolbox.lib.arrayops.mask), 25  
 MATCH() (in module RRtoolbox.lib.descriptors), 34  
 MATCH\_multiple() (in module RRtoolbox.lib.descriptors), 34  
 MatchExplorer (class in RRtoolbox.lib.plotter), 55  
 matrixIntercept() (in module RRtoolbox.lib.arrayops.basic), 9  
 maxMemory (RRtoolbox.lib.cache.ResourceManager attribute), 29  
 maxth (RRtoolbox.lib.plotter.Edger attribute), 55  
 memoize() (RRtoolbox.lib.cache.Memoizer method), 28  
 MemoizedDict (class in RRtoolbox.lib.cache), 27  
 Memoizer (class in RRtoolbox.lib.cache), 27  
 memoizers (RRtoolbox.lib.cache.Memoizer attribute), 28  
 MemorizedFunc (class in RRtoolbox.lib.cache), 28  
 Memory (class in RRtoolbox.lib.cache), 28  
 mkPath() (in module RRtoolbox.lib.directory), 39  
 mmap\_mode (RRtoolbox.lib.cache.DynamicMemoizedFunc attribute), 27  
 mousefunc() (RRtoolbox.lib.image.GetCoors method), 41  
 mousefunc() (RRtoolbox.lib.plotter.MatchExplorer method), 56  
 mousefunc() (RRtoolbox.lib.plotter.Plotim method), 58  
 multiple\_otsu() (in module RRtoolbox.lib.arrayops.mask), 25  
 multiple\_superpose() (in module RRtoolbox.lib.arrayops.basic), 9  
 myline() (in module RRtoolbox.lib.image), 51

## N

Namespace (class in RRtoolbox.lib.root), 65  
 nearFactorIndex() (RRtoolbox.lib.root.FactorConvert method), 64  
 nextim() (RRtoolbox.tools.selectors.EntropyPlot method), 75  
 noisy() (in module RRtoolbox.lib.arrayops.basic), 9  
 NoParserFound, 65  
 normalize() (in module RRtoolbox.lib.arrayops.basic), 10  
 normalize2() (in module RRtoolbox.lib.arrayops.basic), 10  
 normalizeCustom() (in module RRtoolbox.lib.arrayops.basic), 10

normsigmoid() (in module RRtoolbox.lib.arrayops.filters), 23  
 NotCallable, 65  
 NotCreatable, 65  
 NotMemorizedFunc (class in RRtoolbox.lib.cache), 28  
 np2qi() (in module RRtoolbox.lib.image), 51  
 np2str() (in module RRtoolbox.lib.image), 51

## O

ObjectGetter (class in RRtoolbox.lib.cache), 28  
 onmouse() (RRtoolbox.lib.plotter.Plotim static method), 58  
 onTrackbar1() (RRtoolbox.lib.plotter.Edger method), 55  
 onTrackbar2() (RRtoolbox.lib.plotter.Edger method), 55  
 open\_point() (RRtoolbox.lib.root.Profiler method), 66  
 optimizeObject() (RRtoolbox.lib.cache.ResourceManager method), 29  
 overlay() (in module RRtoolbox.lib.arrayops.basic), 10  
 overlay2() (in module RRtoolbox.lib.arrayops.basic), 11  
 overlaypng() (in module RRtoolbox.lib.arrayops.basic), 12  
 overlayXY() (in module RRtoolbox.lib.arrayops.basic), 11

## P

pad\_to\_fit\_H() (in module RRtoolbox.lib.arrayops.basic), 13  
 padVH() (in module RRtoolbox.lib.arrayops.basic), 12  
 parse() (RRtoolbox.shell.Shell method), 76  
 parser\_fastplt() (RRtoolbox.shell.Shell method), 76  
 parser\_loadFunc() (RRtoolbox.shell.Shell method), 76  
 parseString() (in module RRtoolbox.lib.serverServices), 70  
 parts() (RRtoolbox.lib.root.FactorConvert static method), 64  
 path (RRtoolbox.lib.image.Image attribute), 44  
 PathLoader (class in RRtoolbox.lib.image), 45  
 ping() (in module RRtoolbox.lib.serverServices), 70  
 plotatpointer() (RRtoolbox.lib.plotter.Plotim method), 59  
 plotatxy() (RRtoolbox.lib.plotter.Plotim method), 59  
 Plotim (class in RRtoolbox.lib.plotter), 56  
 plotintime() (RRtoolbox.lib.plotter.Plotim method), 60  
 plotPointsContour() (in module RRtoolbox.lib.plotter), 63  
 plt2bgr() (in module RRtoolbox.lib.image), 51  
 plt2bgra() (in module RRtoolbox.lib.image), 51  
 points2contour() (in module RRtoolbox.lib.arrayops.convert), 19  
 points2mask() (in module RRtoolbox.lib.arrayops.basic), 13  
 points2vectos() (in module RRtoolbox.lib.arrayops.convert), 19  
 points\_generator() (in module RRtoolbox.lib.arrayops.basic), 13

- [polygonArea\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [13](#)  
[polygonArea\\_calculc\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [13](#)  
[polygonArea\\_contour\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [14](#)  
[polygonArea\\_fill\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [14](#)  
[previousim\(\)](#) (`RRtoolbox.tools.selectors.EntropyPlot` method), [75](#)  
[println\(\)](#) (`RRtoolbox.lib.root.StdoutLOG` method), [67](#)  
[println\(\)](#) (`RRtoolbox.lib.root.StdoutMULTI` method), [67](#)  
[println\(\)](#) (`RRtoolbox.lib.root.StdoutSIM` method), [67](#)  
[printlns\(\)](#) (`RRtoolbox.lib.root.StdoutLOG` method), [67](#)  
[printlns\(\)](#) (`RRtoolbox.lib.root.StdoutMULTI` method), [67](#)  
[printlns\(\)](#) (`RRtoolbox.lib.root.StdoutSIM` method), [67](#)  
[process\\_as\\_blocks\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [14](#)  
[Profiler](#) (class in `RRtoolbox.lib.root`), [65](#)  
[pts](#) (`RRtoolbox.lib.image.ImCoors` attribute), [42](#)  
[pts2cnt\(\)](#) (in module `RRtoolbox.lib.arrayops.convert`), [19](#)
- ## Q
- [qi2np\(\)](#) (in module `RRtoolbox.lib.image`), [51](#)  
[quadrant\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [14](#)  
[quadrants\(\)](#) (in module `RRtoolbox.lib.image`), [52](#)  
[quickOps\(\)](#) (in module `RRtoolbox.lib.directory`), [40](#)
- ## R
- [random\\_color\(\)](#) (in module `RRtoolbox.lib.image`), [52](#)  
[random\\_points\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [15](#)  
[randomColor\(\)](#) (`RRtoolbox.lib.plotter.MatchExplorer` static method), [56](#)  
[raw\(\)](#) (`RRtoolbox.lib.cache.ObjectGetter` method), [29](#)  
[rcv\(\)](#) (`RRtoolbox.lib.serverServices.Conection` method), [69](#)  
[rcvPickle\(\)](#) (in module `RRtoolbox.lib.serverServices`), [70](#)  
[readSession\(\)](#) (in module `RRtoolbox.lib.session`), [71](#)  
[real2render\(\)](#) (`RRtoolbox.lib.plotter.Plotim` method), [60](#)  
[recursiveMap\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [15](#)  
[recv\\_into\(\)](#) (in module `RRtoolbox.lib.serverServices`), [70](#)  
[recvall\(\)](#) (`RRtoolbox.lib.serverServices.Conection` method), [69](#)  
[register\(\)](#) (`RRtoolbox.lib.cache.ResourceManager` method), [29](#)  
[register\(\)](#) (`RRtoolbox.lib.cache.Retriever` method), [30](#)  
[register\(\)](#) (`RRtoolbox.lib.image.LoaderDict` method), [45](#)  
[relativeQuadrants\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [15](#)  
[relativeVectors\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [15](#)  
[reloadFunc\(\)](#) (in module `RRtoolbox.lib.inspector`), [54](#)  
[render2real\(\)](#) (`RRtoolbox.lib.plotter.Plotim` method), [60](#)  
[renew\(\)](#) (`RRtoolbox.lib.inspector.Logger` method), [54](#)  
[report\(\)](#) (`RRtoolbox.lib.inspector.Logger` method), [54](#)  
[repr2list\(\)](#) (`RRtoolbox.lib.directory.Directory` static method), [36](#)  
[repr2str\(\)](#) (`RRtoolbox.lib.directory.Directory` static method), [36](#)  
[rescale\(\)](#) (in module `RRtoolbox.lib.arrayops.basic`), [15](#)  
[reset\(\)](#) (`RRtoolbox.lib.config.DirectoryManager` method), [32](#)  
[resetGetter\(\)](#) (`RRtoolbox.lib.cache.ResourceManager` static method), [30](#)  
[resource\\_path\(\)](#) (in module `RRtoolbox.lib.directory`), [40](#)  
[ResourceManager](#) (class in `RRtoolbox.lib.cache`), [29](#)  
[restructure\(\)](#) (`RRtoolbox.lib.root.Profiler` method), [66](#)  
[retina\\_markers\\_thresh\(\)](#) (in module `RRtoolbox.tools.segmentation`), [74](#)  
[retinal\\_mask\(\)](#) (in module `RRtoolbox.tools.segmentation`), [74](#)  
[retinal\\_mask\\_watershed\(\)](#) (in module `RRtoolbox.tools.segmentation`), [75](#)  
[Retriever](#) (class in `RRtoolbox.lib.cache`), [30](#)  
[RGB](#) (`RRtoolbox.lib.image.Image` attribute), [44](#)  
[rgb2qi\(\)](#) (in module `RRtoolbox.lib.image`), [52](#)  
[RGBA](#) (`RRtoolbox.lib.image.Image` attribute), [44](#)  
[rmFile\(\)](#) (in module `RRtoolbox.lib.directory`), [40](#)  
[rmPath\(\)](#) (in module `RRtoolbox.lib.directory`), [40](#)  
[rrbox](#) (class in `RRtoolbox.core`), [76](#)  
[RRtoolbox](#) (module), [77](#)  
[RRtoolbox.core](#) (module), [76](#)  
[RRtoolbox.lib](#) (module), [72](#)  
[RRtoolbox.lib.arrayops](#) (module), [26](#)  
[RRtoolbox.lib.arrayops.basic](#) (module), [3](#)  
[RRtoolbox.lib.arrayops.convert](#) (module), [18](#)  
[RRtoolbox.lib.arrayops.filters](#) (module), [21](#)  
[RRtoolbox.lib.arrayops.mask](#) (module), [24](#)  
[RRtoolbox.lib.cache](#) (module), [26](#)  
[RRtoolbox.lib.config](#) (module), [31](#)  
[RRtoolbox.lib.descriptors](#) (module), [33](#)  
[RRtoolbox.lib.directory](#) (module), [35](#)  
[RRtoolbox.lib.image](#) (module), [41](#)  
[RRtoolbox.lib.inspector](#) (module), [53](#)  
[RRtoolbox.lib.plotter](#) (module), [54](#)  
[RRtoolbox.lib.root](#) (module), [63](#)  
[RRtoolbox.lib.serverServices](#) (module), [69](#)  
[RRtoolbox.lib.session](#) (module), [71](#)  
[RRtoolbox.run](#) (module), [76](#)  
[RRtoolbox.shell](#) (module), [76](#)  
[RRtoolbox.tools](#) (module), [76](#)  
[RRtoolbox.tools.lens](#) (module), [72](#)  
[RRtoolbox.tools.segmentation](#) (module), [73](#)



RRtoolbox.tools.selectors (module), 75

RRtoolbox.tools.sticher (module), 76

rx1 (RRtoolbox.lib.plotter.Plotim attribute), 60

rx2 (RRtoolbox.lib.plotter.Plotim attribute), 60

ry1 (RRtoolbox.lib.plotter.Plotim attribute), 60

ry2 (RRtoolbox.lib.plotter.Plotim attribute), 60

## S

save() (RRtoolbox.lib.config.DirectoryManager method), 32

save() (RRtoolbox.lib.image.Image method), 44

save() (RRtoolbox.lib.inspector.GraphTraceOutput method), 53

save() (RRtoolbox.lib.plotter.Edger method), 55

save() (RRtoolbox.lib.plotter.Plotim method), 60

saveSession() (in module RRtoolbox.lib.session), 72

saveSource() (RRtoolbox.lib.inspector.GraphTrace method), 53

saveSource() (RRtoolbox.lib.inspector.GraphTraceOutput method), 53

scan\_ports() (in module RRtoolbox.lib.serverServices), 70

selectlist() (RRtoolbox.tools.selectors.EntropyPlot method), 75

send() (RRtoolbox.lib.serverServices.Connection method), 69

send\_from() (in module RRtoolbox.lib.serverServices), 71

sendLen() (RRtoolbox.lib.serverServices.Connection method), 69

sendPickle() (in module RRtoolbox.lib.serverServices), 70

separe() (in module RRtoolbox.lib.image), 52

separePointsByAxis() (in module RRtoolbox.lib.arrayops.basic), 15

sh2oh() (in module RRtoolbox.lib.arrayops.convert), 20

shape (RRtoolbox.lib.image.Image attribute), 44

Shell (class in RRtoolbox.shell), 76

shell\_processor() (in module RRtoolbox.shell), 77

shell\_processor\_parser() (in module RRtoolbox.shell), 77

show() (RRtoolbox.lib.plotter.Plotim method), 61

showfunc() (RRtoolbox.lib.plotter.Plotim method), 61

showgray (RRtoolbox.lib.plotter.Edger attribute), 55

sigmaColor (RRtoolbox.lib.arrayops.filters.BilateralParameters attribute), 22

sigmaSpace (RRtoolbox.lib.arrayops.filters.BilateralParameters attribute), 22

sigmoid() (in module RRtoolbox.lib.arrayops.filters), 23

SimKeyPoint (class in RRtoolbox.lib.arrayops.convert), 18

simulateLens() (in module RRtoolbox.tools.lens), 73

size (RRtoolbox.lib.plotter.Edger attribute), 55

smooth() (in module RRtoolbox.lib.arrayops.filters), 24

source (RRtoolbox.lib.inspector.GraphTrace attribute), 53

spairs2opairs() (in module RRtoolbox.lib.arrayops.convert), 20

split() (RRtoolbox.lib.root.FactorConvert static method), 64

splitPoints() (in module RRtoolbox.lib.arrayops.basic), 16

spoint2opointfunc() (in module RRtoolbox.lib.arrayops.convert), 20

standarizePoints() (in module RRtoolbox.lib.arrayops.basic), 16

start() (RRtoolbox.lib.inspector.Asynchronous method), 53

start() (RRtoolbox.lib.inspector.Synchronous method), 54

StdoutLOG (class in RRtoolbox.lib.root), 66

StdoutMULTI (class in RRtoolbox.lib.root), 67

StdoutSIM (class in RRtoolbox.lib.root), 67

stich() (in module RRtoolbox.tools.sticher), 76

stop() (RRtoolbox.lib.inspector.Synchronous method), 54

str2np() (in module RRtoolbox.lib.image), 52

strdifference() (in module RRtoolbox.lib.directory), 40

string\_interpreter() (in module RRtoolbox.shell), 77

string\_is\_socket\_address() (in module RRtoolbox.lib.serverServices), 71

string\_lines() (RRtoolbox.lib.root.Profiler method), 66

string\_structured() (RRtoolbox.lib.root.Profiler method), 66

structure() (RRtoolbox.lib.root.Profiler method), 66

superpose() (in module RRtoolbox.lib.arrayops.basic), 16

Synchronous (class in RRtoolbox.lib.inspector), 54

## T

temp() (RRtoolbox.lib.image.ImLoader method), 44

textbackground (RRtoolbox.lib.plotter.Plotim attribute), 61

th1 (RRtoolbox.lib.plotter.Edger attribute), 55

th2 (RRtoolbox.lib.plotter.Edger attribute), 55

thresh\_biggestCnt() (in module RRtoolbox.lib.arrayops.mask), 26

thresh\_hist() (in module RRtoolbox.lib.arrayops.mask), 26

threshold\_opening() (in module RRtoolbox.lib.arrayops.mask), 26

throwError() (RRtoolbox.lib.inspector.Logger method), 54

time (RRtoolbox.lib.root.Profiler attribute), 66

time (RRtoolbox.lib.root.TimeCode attribute), 67

Time\_ (RRtoolbox.lib.inspector.Logger attribute), 54

time\_end (RRtoolbox.lib.root.TimeCode attribute), 67

TimeCode (class in RRtoolbox.lib.root), 67

TimeoutException, 67

tools() (in module RRtoolbox.core), 76

tools2() (in module RRtoolbox.core), 76

toTuple() (in module RRtoolbox.lib.arrayops.convert), 20

tracer (RRtoolbox.lib.inspector.Logger attribute), 54

tracer() (in module RRtoolbox.lib.inspector), 54  
tracer() (RRtoolbox.lib.inspector.Asynchronous method), 53  
TransferExeption, 68  
transformPoint() (in module RRtoolbox.lib.arrayops.basic), 16  
transformPoints() (in module RRtoolbox.lib.arrayops.basic), 16  
translateQuadrants() (in module RRtoolbox.lib.arrayops.convert), 20  
transposeIm() (in module RRtoolbox.lib.image), 52  
try\_loads() (in module RRtoolbox.lib.image), 52  
tuple2keyPoint() (in module RRtoolbox.lib.arrayops.convert), 21  
tuple\_creator() (in module RRtoolbox.shell), 77  
Type\_ (RRtoolbox.lib.inspector.Logger attribute), 54

## U

unit (RRtoolbox.lib.cache.ResourceManager attribute), 30  
unit\_vector() (in module RRtoolbox.lib.arrayops.basic), 16  
units2bytes() (RRtoolbox.lib.cache.ResourceManager method), 30  
update() (RRtoolbox.lib.cache.ObjectGetter method), 29  
update() (RRtoolbox.lib.directory.Directory method), 36  
update() (RRtoolbox.lib.image.ImFactory method), 42  
update\_left() (RRtoolbox.lib.directory.Directory method), 36  
update\_right() (RRtoolbox.lib.directory.Directory method), 36  
updatecoors() (RRtoolbox.lib.image.GetCoors method), 41  
updaterenderer() (RRtoolbox.lib.plotter.MatchExplorer method), 56  
updaterenderer() (RRtoolbox.lib.plotter.Plotim method), 61  
updateSession() (in module RRtoolbox.lib.session), 72  
updatevisualization() (RRtoolbox.lib.plotter.Imtester method), 55  
usedMemory (RRtoolbox.lib.cache.ResourceManager attribute), 30

## V

VariableNotDeletable, 68  
VariableNotSettable, 68  
vectorsAngles() (in module RRtoolbox.lib.arrayops.basic), 16  
vectorsQuadrants() (in module RRtoolbox.lib.arrayops.basic), 17  
vectos2points() (in module RRtoolbox.lib.arrayops.convert), 21  
verbose (RRtoolbox.lib.cache.DynamicMemoizedFunc attribute), 27

vertexesAngles() (in module RRtoolbox.lib.arrayops.basic), 17  
view\_as\_blocks() (in module RRtoolbox.lib.arrayops.basic), 17  
view\_as\_windows() (in module RRtoolbox.lib.arrayops.basic), 17  
visualize() (RRtoolbox.lib.plotter.Imtester method), 55

## W

windowfunc() (RRtoolbox.lib.plotter.Edger method), 55  
windowfunc() (RRtoolbox.lib.plotter.Imtester method), 55  
windowfunc() (RRtoolbox.lib.plotter.Plotim method), 61  
write() (RRtoolbox.lib.root.StdoutLOG method), 67  
write() (RRtoolbox.lib.root.StdoutMULTI method), 67  
write() (RRtoolbox.lib.root.StdoutSIM method), 67  
writer() (RRtoolbox.lib.inspector.Logger method), 54