wiimatch Documentation

Release 0.3.2

Mihai Cara

Nov 11, 2023

CONTENTS

`wiimatch` is a package that provides core computational algorithms for optimal "matching" of weighted N-dimensional image intensity data using (multivariate) polynomials.

CHAPTER

ONE

CONTENT

1.1 LSQ Image Intensity Matching

A module that provides main API for optimal (LSQ) "matching" of weighted N-dimensional image intensity data using (multivariate) polynomials.

Author

Mihai Cara (contact: help@stsci.edu)

License

LICENSE

`wiimatch.match.`**`match_lsq`**(*images*, *masks=None*, *sigmas=None*, *degree=0*, *center=None*, *image2world=None*, *center_cs='image'*, *ext_return=False*, *solver='RLU'*, *default_container=WMInMemoryData*)

Compute coefficients of (multivariate) polynomials that once subtracted from input images would provide image intensity matching in the least squares sense.

Parameters

images

[list of WMData and/or numpy.ndarray] A list of *WMData* to 1D, 2D, etc. `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) data arrays whose "intensities" must be "matched". All arrays must have identical shapes. When `images` is a list of `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), the container class specified by the `default_container` will be used to convert `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) to `WMData` objects. Input list may mix `WMData`, `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), and `None` (https://docs.python.org/3/library/constants.html#None) objects.

masks

[list of WMData and/or numpy.ndarray and/or None, None, optional] A list of `WMData` of the same length as `images`. Non-zero mask elements in data arrays indicate valid data in the corresponding `images` array. Mask arrays must have identical shape to that of the arrays in input `images`. Default value of `None` (https://docs.python.org/3/library/constants.html#None)

indicates that all pixels in (the corresponding) input images are valid. When `masks` is a list of numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), the container class specified by the `default_container` will be used to convert numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) to `WMData` objects. Input list may mix `WMData`, numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), and None (https://docs.python.org/3/library/constants.html#None) objects.

sigmas

[list of WMData and/or numpy.ndarray and/or numbers, None, optional] A list of `WMData` of the same length as `images` representing the uncertainties of the data in the corresponding array in `images`. Uncertainty arrays must have identical shape to that of the arrays in input `images`. A numeric value for a `sigmas` element will apply to all pixels in the corresponding `images` element. The default value of None (https://docs.python.org/3/library/constants.html#None) indicates that all pixels will be assigned equal weights. When `sigmas` is a list of numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), the container class specified by the `default_container` will be used to convert numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) to `WMData` objects. When `sigmas` is None (https://docs.python.org/3/library/constants.html#None), then all pixels in all images will be assigned weight 1.

degree

[iterable, int, optional] A list of polynomial degrees for each dimension of data arrays in `images`. The length of the input list must match the dimensionality of the input images. When a single integer number is provided, it is assumed that the polynomial degree in each dimension is equal to that integer.

center

[iterable, None, optional] An iterable of length equal to the number of dimensions in `image_shape` that indicates the center of the coordinate system in **image** coordinates when `center_cs` is `'image'` otherwise center is assumed to be in **world** coordinates (when `center_cs` is `'world'`). When `center` is None (https://docs.python.org/3/library/constants.html#None) then `center` is set to the middle of the "image" as `center[i]=image_shape[i]//2`. If `image2world` is not None (https://docs.python.org/3/library/constants.html#None) and `center_cs` is `'image'`, then supplied center will be converted to world coordinates.

image2world

[function, None, optional] Image-to-world coordinates transformation function. This function must be of the form `f(x,y, z,...)` and accept a number of arguments numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) arguments equal to the dimensionality of images.

center_cs

[{'image', 'world'}, optional] Indicates whether `center` is in image coordi-

nates or in world coordinates. This parameter is ignored when `center` is set to `None` (https://docs.python.org/3/library/constants.html#None): it is assumed to be `False` (https://docs.python.org/3/library/constants.html#False). `center_cs` *cannot be* `'world'` when `image2world` is `None` (https://docs.python.org/3/library/constants.html#None) unless `center` is `None` (https://docs.python.org/3/library/constants.html#None).

ext_return
[bool, optional] Indicates whether this function should return additional values besides optimal polynomial coefficients (see `bkg_poly_coeff` return value below) that match image intensities in the LSQ sense. See **Returns** section for more details.

solver
[{'RLU', 'PINV'}, optional] Specifies method for solving the system of equations.

default_container
[class] A class that is a subclass of `WMData` that will be used to wrap input and internal `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) arrays. Must be able to instantiate from a single argument - a data aray.

Returns

bkg_poly_coeff
[numpy.ndarray] When `nimages` is `None` (https://docs.python.org/3/library/constants.html#None), this function returns a 1D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) that holds the solution (polynomial coefficients) to the system.

When `nimages` is **not** `None` (https://docs.python.org/3/library/constants.html#None), this function returns a 2D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) that holds the solution (polynomial coefficients) to the system. The solution is grouped by image.

a
[numpy.ndarray] A 2D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarr that holds the coefficients of the linear system of equations. This value is returned only when `ext_return` is `True` (https://docs.python.org/3/library/constants.html#True).

b
[numpy.ndarray] A 1D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarr that holds the free terms of the linear system of equations. This value is returned only when `ext_return` is `True` (https://docs.python.org/3/library/constants.html#True).

coord_arrays
[list] A list of `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#n coordinate arrays each of `image_shape` shape. This value is returned only when `ext_return` is `True` (https://docs.python.org/3/library/constants.html#True).

eff_center
: [tuple] A tuple of coordinates of the effective center as used in generating coordinate arrays. This value is returned only when `ext_return` is `True` (https://docs.python.org/3/library/constants.html#True).

coord_system
: [{'image', 'world'}] Coordinate system of the coordinate arrays and returned `center` value. This value is returned only when `ext_return` is `True` (https://docs.python.org/3/library/constants.html#True).

Notes

match_lsq() builds a system of linear equations

$$a \cdot c = b$$

whose solution c is a set of coefficients of (multivariate) polynomials that represent the "background" in each input image (these are polynomials that are "corrections" to intensities of input images) such that the following sum is minimized:

$$L = \sum_{n,m=1,n\neq m}^{N} \sum_{k} \frac{\left[I_n(k) - I_m(k) - P_n(k) + P_m(k)\right]^2}{\sigma_n^2(k) + \sigma_m^2(k)}.$$

In the above equation, index $k = (k_1, k_2, ...)$ labels a position in input image's pixel grid [NOTE: all input images share a common pixel grid].

"Background" polynomials $P_n(k)$ are defined through the corresponding coefficients as:

$$P_n(k_1, k_2, ...) = \sum_{d_1=0,d_2=0,...}^{D_1,D_2,...} c^n_{d_1,d_2,...} \cdot k_1^{d_1} \cdot k_2^{d_2} \cdot ....$$

Coefficients $c^n_{d_1,d_2,...}$ are arranged in the vector c in the following order:

$$(c^1_{0,0,...}, c^1_{1,0,...}, \ldots, c^2_{0,0,...}, c^2_{1,0,...}, \ldots).$$

match_lsq() returns coefficients of the polynomials that minimize *L*.

Examples

```
>>> import wiimatch
>>> from wiimatch.containers import WMInMemoryData
>>> import numpy as np
>>> im1 = np.zeros((5, 5, 4), dtype=float)
>>> cbg = 1.32 * np.ones_like(im1)
>>> ind = np.indices(im1.shape, dtype=float)
>>> im3 = cbg + 0.15 * ind[0] + 0.62 * ind[1] + 0.74 * ind[2]
>>> mask = WMInMemoryData(np.ones_like(im1, dtype=np.int8))
```

(continues on next page)

(continued from previous page)

```
>>> sigma = WMInMemoryData(np.ones_like(im1, dtype=float))
>>> wiimatch.match.match_lsq([WMInMemoryData(im1), WMInMemoryData(im3)],
... [mask, mask], [sigma, sigma], degree=(1,1,1), center=(0,0,0))
array([[-6.60000000e-01, -7.50000000e-02, -3.10000000e-01,
        -6.96331881e-16, -3.70000000e-01, -1.02318154e-15,
        -5.96855898e-16,  2.98427949e-16],
       [ 6.60000000e-01,  7.50000000e-02,  3.10000000e-01,
         6.96331881e-16,  3.70000000e-01,  1.02318154e-15,
         5.96855898e-16, -2.98427949e-16]])
```

1.2 Data Containers

Data containers for accessing image data (i.e., numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.nda
uniformly whether they are kept in memory, as memory mapped files (load), or stored to/loaded from a file
as whole arrays.

Author
Mihai Cara (contact: help@stsci.edu)

License
LICENSE

class wiimatch.containers.**WMData**

Base class for all data containers. Provides a common interface to access data.

abstract property data

Sets/Gets linked data.

Parameters

data
[object] Data to be set.

kind = 'mapped'

Hints to how data are stored: `'mapped'`, `'file'`, or `'in-memory'`. May be used by code for performance optimization.

abstract property shape

Returns a tuple describing the shape of linked data.

class wiimatch.containers.**WMInMemoryData**(*data*)

Acessor for in-memory numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#nump
data.

property data

Sets/gets linked numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.

Parameters

data
[object] Data to be set.

`kind = 'in-memory'`

Hints to how data are stored: `'mapped'`, `'file'`, or `'in-memory'`. May be used by code for performance optimization.

`property shape`

Returns a tuple describing the shape of linked data.

`class wiimatch.containers.WMMappedData(data, tmpfile=None, prefix='tmp_wiimatch_', suffix='.npy', tmpdir='')`

Data container for arrays stored in temporary files. This is best suited when array data are needed in memory all at once and when array is not needed - it can be stored to a file.

To access small segments of data, use cls:*WMMemMappedData*.

`property data`

Sets/gets linked `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.

Parameters

data
[object] Data to be set.

`kind = 'file'`

Hints to how data are stored: `'mapped'`, `'file'`, or `'in-memory'`. May be used by code for performance optimization.

`property shape`

Returns a tuple describing the shape of linked data.

`class wiimatch.containers.WMMemMappedData(data, tmpfile=None, prefix='tmp_wiimatch_', suffix='.npy', tmpdir='')`

Data container for arrays stored in temporary files. This is best suited when array data are needed in memory all at once and when array is not needed - it can be stored to a file.

To access entire data arrays, use cls:*WMMappedData*.

`property data`

Sets/gets linked `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.

Parameters

data
[object] Data to be set.

`kind = 'mapped'`

Hints to how data are stored: `'mapped'`, `'file'`, or `'in-memory'`. May be used by code for performance optimization.

`property shape`

Returns a tuple describing the shape of linked data.

1.3 LSQ Equation Construction and Solving

A module that provides core algorithm for optimal matching of backgrounds of N-dimensional images using (multi-variate) polynomials.

Author

Mihai Cara (contact: help@stsci.edu)

License

LICENSE

wiimatch.lsq_optimizer.**build_lsq_eqs**(*images*, *masks*, *sigmas*, *degree*, *center=None*, *image2world=None*, *center_cs='image'*, *container_cls=<class 'wiimatch.containers.WMInMemoryData'>*)

build_lsq_eqs(images, masks, sigmas, degree, center=None, image2world=None, center_cs='image', container_cls=WMInMemoryData): Build system of linear equations whose solution would provide image intensity matching in the least squares sense.

Parameters

images

[list of WMData] A list of *WMData* to 1D, 2D, etc. numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) data arrays whose "intensities" must be "matched". All arrays must have identical shapes. When `images` is a list of numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), the container class specified by the `default_container` will be used to convert numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) to `WMData` objects. Input list may mix `WMData`, numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), and None (https://docs.python.org/3/library/constants.html#None) objects.

masks

[list of WMData and/or None] A list of `WMData` of the same length as `images`. Non-zero mask elements in data arrays indicate valid data in the corresponding `images` array. Mask arrays must have identical shape to that of the arrays in input `images`. Default value of None (https://docs.python.org/3/library/constants.html#None) indicates that all pixels in (the corresponding) input images are valid. When `masks` is a list of numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), the container class specified by the `default_container` will be used to convert numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) to `WMData` objects. Input list may mix `WMData`, numpy.ndarray (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), and None (https://docs.python.org/3/library/constants.html#None) objects.

sigmas

[list of WMData, list of None] A list of `WMData` of the same length

as `images` representing the uncertainties of the data in the corresponding array in `images`. Uncertainty arrays must have identical shape to that of the arrays in input `images`. The default value of `None` (https://docs.python.org/3/library/constants.html#None) indicates that all pixels in all images will be assigned equal weights of 1. When `sigmas` is a list of `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray), the container class specified by the `default_container` will be used to convert `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) to `WMData` objects.

degree

[iterable] A list of polynomial degrees for each dimension of data arrays in `images`. The length of the input list must match the dimensionality of the input images.

center

[iterable, None, optional] An iterable of length equal to the number of dimensions of images in `images` parameter that indicates the center of the coordinate system in **image** coordinates when `center_cs` is `'image'` otherwise center is assumed to be in **world** coordinates (when `center_cs` is `'world'`). When `center` is `None` (https://docs.python.org/3/library/constants.html#None) then `center` is set to the middle of the "image" as `center[i]=image.shape[i]//2`. If `image2world` is not `None` (https://docs.python.org/3/library/constants.html#None) and `center_cs` is `'image'`, then supplied center will be converted to world coordinates.

image2world

[function, None, optional] Image-to-world coordinates transformation function. This function must be of the form `f(x,y, z,...)` and accept a number of arguments `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) arguments equal to the dimensionality of images.

center_cs

[{'image', 'world'}, optional] Indicates whether `center` is in image coordinates or in world coordinates. This parameter is ignored when `center` is set to `None` (https://docs.python.org/3/library/constants.html#None): it is assumed to be `False` (https://docs.python.org/3/library/constants.html#False). `center_cs` *cannot be* `'world'` when `image2world` is `None` (https://docs.python.org/3/library/constants.html#None) unless `center` is `None` (https://docs.python.org/3/library/constants.html#None).

Returns

a

[numpy.ndarray] A 2D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarr that holds the coefficients of the linear system of equations.

b

[numpy.ndarray] A 1D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarr that holds the free terms of the linear system of equations.

coord_arrays
: [list] A list of `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#n coordinate arrays each of `images[0].shape` shape.

eff_center
: [tuple] A tuple of coordinates of the effective center as used in generating coordinate arrays.

coord_system
: [{'image', 'world'}] Coordinate system of the coordinate arrays and returned `center` value.

Notes

build_lsq_eqs() builds a system of linear equations

$$a \cdot c = b$$

whose solution c is a set of coefficients of (multivariate) polynomials that represent the "background" in each input image (these are polynomials that are "corrections" to intensities of input images) such that the following sum is minimized:

$$L = \sum_{n,m=1,n\neq m}^{N} \sum_{k} \frac{[I_n(k) - I_m(k) - P_n(k) + P_m(k)]^2}{\sigma_n^2(k) + \sigma_m^2(k)}.$$

In the above equation, index $k = (k_1, k_2, ...)$ labels a position in input image's pixel grid [NOTE: all input images share a common pixel grid].

"Background" polynomials $P_n(k)$ are defined through the corresponding coefficients as:

$$P_n(k_1, k_2, ...) = \sum_{d_1=0,d_2=0,...}^{D_1,D_2,...} c^n_{d_1,d_2,...} \cdot k_1^{d_1} \cdot k_2^{d_2} \cdot ....$$

Coefficients $c^n_{d_1,d_2,...}$ are arranged in the vector c in the following order:

$$(c^1_{0,0,...}, c^1_{1,0,...}, \ldots, c^2_{0,0,...}, c^2_{1,0,...}, \ldots).$$

Examples

```
>>> from wiimatch.lsq_optimizer import build_lsq_eqs
>>> from wiimatch.containers import WMInMemoryData
>>> import numpy as np
>>> im1 = np.zeros((5, 5, 4), dtype=float)
>>> cbg = 1.32 * np.ones_like(im1)
>>> ind = np.indices(im1.shape, dtype=float)
>>> im3 = cbg + 0.15 * ind[0] + 0.62 * ind[1] + 0.74 * ind[2]
>>> mask = np.ones_like(im1, dtype=np.int8)
```

(continues on next page)

(continued from previous page)

```
>>> sigma = np.ones_like(im1, dtype=float)
>>> a, b, ca, ef, cs = build_lsq_eqs(
...     [WMInMemoryData(im1), WMInMemoryData(im3)],
...     [WMInMemoryData(mask), WMInMemoryData(mask)],
...     [WMInMemoryData(sigma), WMInMemoryData(sigma)],
...     degree=(1, 1, 1), center=(0, 0, 0)
... )
>>> print(a)
[[   50.   100.   100.   200.    75.   150.   150.   300.   -50.  -100.
   -100.  -200.   -75.  -150.  -150.  -300.]
 [  100.   300.   200.   600.   150.   450.   300.   900.  -100.  -300.
   -200.  -600.  -150.  -450.  -300.  -900.]
 [  100.   200.   300.   600.   150.   300.   450.   900.  -100.  -200.
   -300.  -600.  -150.  -300.  -450.  -900.]
 [  200.   600.   600.  1800.   300.   900.   900.  2700.  -200.  -600.
   -600. -1800.  -300.  -900.  -900. -2700.]
 [   75.   150.   150.   300.   175.   350.   350.   700.   -75.  -150.
   -150.  -300.  -175.  -350.  -350.  -700.]
 [  150.   450.   300.   900.   350.  1050.   700.  2100.  -150.  -450.
   -300.  -900.  -350. -1050.  -700. -2100.]
 [  150.   300.   450.   900.   350.   700.  1050.  2100.  -150.  -300.
   -450.  -900.  -350.  -700. -1050. -2100.]
 [  300.   900.   900.  2700.   700.  2100.  2100.  6300.  -300.  -900.
   -900. -2700.  -700. -2100. -2100. -6300.]
 [  -50.  -100.  -100.  -200.   -75.  -150.  -150.  -300.    50.   100.
    100.   200.    75.   150.   150.   300.]
 [ -100.  -300.  -200.  -600.  -150.  -450.  -300.  -900.   100.   300.
    200.   600.   150.   450.   300.   900.]
 [ -100.  -200.  -300.  -600.  -150.  -300.  -450.  -900.   100.   200.
    300.   600.   150.   300.   450.   900.]
 [ -200.  -600.  -600. -1800.  -300.  -900.  -900. -2700.   200.   600.
    600.  1800.   300.   900.   900.  2700.]
 [  -75.  -150.  -150.  -300.  -175.  -350.  -350.  -700.    75.   150.
    150.   300.   175.   350.   350.   700.]
 [ -150.  -450.  -300.  -900.  -350. -1050.  -700. -2100.   150.   450.
    300.   900.   350.  1050.   700.  2100.]
 [ -150.  -300.  -450.  -900.  -350.  -700. -1050. -2100.   150.   300.
    450.   900.   350.   700.  1050.  2100.]
 [ -300.  -900.  -900. -2700.  -700. -2100. -2100. -6300.   300.   900.
    900.  2700.   700.  2100.  2100.  6300.]]
>>> print(b)
[ -198.5  -412.   -459.   -948.   -344.   -710.5  -781.  -1607.    198.5
   412.    459.    948.    344.    710.5   781.   1607. ]
```

wiimatch.lsq_optimizer.**pinv_solve**(*matrix*, *free_term*, *nimages*, *tol=None*)

Solves a system of linear equations

$$a \cdot c = b.$$

using Moore-Penrose pseudoinverse.

Parameters

matrix
[numpy.ndarray] A 2D array containing coefficients of the system.

free_term
[numpy.ndarray] A 1D array containing free terms of the system of the equations.

nimages
[int] Number of images for which the system is being solved.

tol
[float, None, optional] Cutoff for small singular values for Moore-Penrose pseudoinverse. When provided, singular values smaller (in modulus) than `tol * |largest_singular_value|` are set to zero. When `tol` is `None` (https://docs.python.org/3/library/constants.html#None) (default), cutoff value is determined based on the type of the input `matrix` argument.

Returns

bkg_poly_coeff
[numpy.ndarray] A 2D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarr
that holds the solution (polynomial coefficients) to the system. The solution is grouped by image.

Examples

```
>>> from wiimatch.lsq_optimizer import build_lsq_eqs, pinv_solve
>>> from wiimatch.containers import WMInMemoryData
>>> import numpy as np
>>> im1 = np.zeros((5, 5, 4), dtype=float)
>>> cbg = 1.32 * np.ones_like(im1)
>>> ind = np.indices(im1.shape, dtype=float)
>>> im3 = cbg + 0.15 * ind[0] + 0.62 * ind[1] + 0.74 * ind[2]
>>> mask = np.ones_like(im1, dtype=np.int8)
>>> sigma = np.ones_like(im1, dtype=float)
>>> a, b, _, _, _ = build_lsq_eqs(
...     [WMInMemoryData(im1), WMInMemoryData(im3)],
...     [WMInMemoryData(mask), WMInMemoryData(mask)],
...     [WMInMemoryData(sigma), WMInMemoryData(sigma)],
...     degree=(1, 1, 1), center=(0, 0, 0)
... )
>>> pinv_solve(a, b, 2)
array([[-6.60000000e-01, -7.50000000e-02, -3.10000000e-01,
```

(continues on next page)

(continued from previous page)

```
       -4.44089210e-15, -3.70000000e-01, -7.66053887e-15,
        3.69704267e-14,  8.37108161e-14],
      [ 6.60000000e-01,  7.50000000e-02,  3.10000000e-01,
        3.55271368e-15,  3.70000000e-01,  4.32986980e-15,
        4.88498131e-14,  7.87148124e-14]])
```

wiimatch.lsq_optimizer.**rlu_solve**(*matrix*, *free_term*, *nimages*)

Computes solution of a "reduced" system of linear equations

$$a' \cdot c' = b'.$$

using LU-decomposition. If the original system contained a set of linearly-dependent equations, then the "reduced" system is formed by dropping equations and unknowns related to the first image. The unknowns corresponding to the first image initially are assumed to be 0. Upon solving the reduced system, these unknowns are recomputed so that mean correction coefficients for all images are 0. This function uses `lu_solve` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu_solve.html#scipy.linalg.lu_solve) and `lu_factor` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lu_factor.html#scipy.linalg.lu_factor functions.

Parameters

matrix
[numpy.ndarray] A 2D array containing coefficients of the system.

free_term
[numpy.ndarray] A 1D array containing free terms of the system of the equations.

nimages
[int] Number of images for which the system is being solved.

Returns

bkg_poly_coeff
[numpy.ndarray] A 2D `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarr that holds the solution (polynomial coefficients) to the system. The solution is grouped by image.

Examples

```
>>> from wiimatch.lsq_optimizer import build_lsq_eqs, pinv_solve
>>> from wiimatch.containers import WMInMemoryData
>>> import numpy as np
>>> im1 = np.zeros((5, 5, 4), dtype=float)
>>> cbg = 1.32 * np.ones_like(im1)
>>> ind = np.indices(im1.shape, dtype=float)
>>> im3 = cbg + 0.15 * ind[0] + 0.62 * ind[1] + 0.74 * ind[2]
>>> mask = np.ones_like(im1, dtype=np.int8)
```

(continues on next page)

(continued from previous page)

```
>>> sigma = np.ones_like(im1, dtype=float)
>>> a, b, _, _, _ = build_lsq_eqs(
...     [WMInMemoryData(im1), WMInMemoryData(im3)],
...     [WMInMemoryData(mask), WMInMemoryData(mask)],
...     [WMInMemoryData(sigma), WMInMemoryData(sigma)],
...     degree=(1, 1, 1), center=(0, 0, 0)
... )
>>> rlu_solve(a, b, 2)
array([[-6.60000000e-01, -7.50000000e-02, -3.10000000e-01,
        -6.96331881e-16, -3.70000000e-01, -1.02318154e-15,
        -5.96855898e-16,  2.98427949e-16],
       [ 6.60000000e-01,  7.50000000e-02,  3.10000000e-01,
         6.96331881e-16,  3.70000000e-01,  1.02318154e-15,
         5.96855898e-16, -2.98427949e-16]])
```

1.4 Utilities used by `wiimatch`

This module provides utility functions for use by `wiimatch` module.

Author

Mihai Cara (contact: help@stsci.edu)

License

LICENSE

`wiimatch.utils.`**`create_coordinate_arrays`**(*image_shape*, *center=None*, *image2world=None*, *center_cs='image'*, *container_cls=<class 'wiimatch.containers.WMInMemoryData'>*)

Create a list of coordinate arrays/grids for each dimension in the image shape. This function is similar to `numpy.indices` (https://numpy.org/doc/stable/reference/generated/numpy.indices.html#numpy.indices) except it returns the list of arrays in reversed order. In addition, it can center image coordinates to a provided `center` and also convert image coordinates to world coordinates using provided `image2world` function.

Parameters

image_shape

[sequence of int] The shape of the image/grid.

center

[iterable, None, optional] An iterable of length equal to the number of dimensions in `image_shape` that indicates the center of the coordinate system in **image** coordinates when `center_cs` is `'image'` otherwise center is assumed to be in **world** coordinates (when `center_cs` is `'world'`). When `center` is `None` (https://docs.python.org/3/library/constants.html#None) then `center` is set to the middle of the "image" as `center[i]=image_shape[i]//2`. If `image2world`

is not `None` (https://docs.python.org/3/library/constants.html#None) and `center_cs` is `'image'`, then supplied center will be converted to world coordinates.

image2world
: [function, None, optional] Image-to-world coordinates transformation function. This function must be of the form `f(x,y, z,...)` and accept a number of arguments `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray) arguments equal to the dimensionality of images.

center_cs
: [{'image', 'world'}, optional] Indicates whether `center` is in image coordinates or in world coordinates. This parameter is ignored when `center` is set to `None` (https://docs.python.org/3/library/constants.html#None): it is assumed to be `False` (https://docs.python.org/3/library/constants.html#False). `center_cs` *cannot be* `'world'` when `image2world` is `None` (https://docs.python.org/3/library/constants.html#None) unless `center` is `None` (https://docs.python.org/3/library/constants.html#None).

Returns

coord_arrays
: [list] A list of `numpy.ndarray` (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#n coordinate arrays each of `image_shape` shape.

eff_center
: [tuple] A tuple of coordinates of the effective center as used in generating coordinate arrays.

coord_system
: [{'image', 'world'}] Coordinate system of the coordinate arrays and returned `center` value.

Examples

```
>>> import wiimatch
>>> wiimatch.utils.create_coordinate_arrays((3, 5, 4))
((array([[[-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.]],
       [[-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.]],
       [[-1.,  0.,  1.,  2.],
```

(continues on next page)

(continued from previous page)

```
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.],
        [-1.,  0.,  1.,  2.]]]),
array([[[-2., -2., -2., -2.],
        [-1., -1., -1., -1.],
        [ 0.,  0.,  0.,  0.],
        [ 1.,  1.,  1.,  1.],
        [ 2.,  2.,  2.,  2.]],
       [[-2., -2., -2., -2.],
        [-1., -1., -1., -1.],
        [ 0.,  0.,  0.,  0.],
        [ 1.,  1.,  1.,  1.],
        [ 2.,  2.,  2.,  2.]],
       [[-2., -2., -2., -2.],
        [-1., -1., -1., -1.],
        [ 0.,  0.,  0.,  0.],
        [ 1.,  1.,  1.,  1.],
        [ 2.,  2.,  2.,  2.]]]),
array([[[-2., -2., -2., -2.],
        [-2., -2., -2., -2.],
        [-2., -2., -2., -2.],
        [-2., -2., -2., -2.],
        [-2., -2., -2., -2.]],
       [[-1., -1., -1., -1.],
        [-1., -1., -1., -1.],
        [-1., -1., -1., -1.],
        [-1., -1., -1., -1.],
        [-1., -1., -1., -1.]],
       [[ 0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.]]])), (1.0, 2.0, 2.0), 'image')
```

1.5 LICENSE

the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

CHAPTER

TWO

DEVELOPMENT NOTES

2.1 Release Notes

2.1.1 0.3.2 (11-November-2023)

- Maintenance release.

2.1.2 0.3.1 (20-July-2022)

- Maintenance release.

2.1.3 0.3.0 (20-July-2022)

- Added data containers module and updated main code to support these containers with the purpose of minimizing memory usage by writing/loading data arrays to temporary files when needed. [#21]

2.1.4 0.2.1 (08-July-2022)

- Updated code to reduce warnings with latest `numpy` versions. [#16]
- Optimized code to improve performance and minimize memory usage when either `masks` and/or `sigmas` have default values. [#18, #19]

2.1.5 0.2.0 (07-August-2019)

Added

- Added a new, more stable, solver `rlu_solve()`. `match_lsq()` now takes a new parameter `solver` which, by default, is set to `'LU'` - the new solver. [#1]

Fixed

- Updated package structure, setup, docs. [#1]

2.1.6 0.1.2 (12-June-2017)

Added

- Several functions now return more values that can be used to analyse returned results:
  - *wiimatch.utils.create_coordinate_arrays()* now returns effective `center` values used in generating coordinate array and coordinate system type (`'image'` or `'world'`);
  - *wiimatch.lsq_optimizer.build_lsq_eqs()* now returns coordinate arrays, effective `center` values used in generating coordinate array, and the coordinate system type of coordinates in addition to coefficients of linear equations;
  - *wiimatch.match.match_lsq()* now optionally returns coefficients of linear equations, coordinate arrays, effective `center` values used in generating coordinate array, and the coordinate system type of coordinates in addition to optimal solution to the matching problem. New parameter `ext_return` indicates to return extended information.

2.1.7 0.1.1 (06-June-2017)

Added

- `center_cs` parameter to *wiimatch.utils.create_coordinate_arrays()* *wiimatch.match.match_lsq()* and *wiimatch.lsq_optimizer.build_lsq_eqs()* in order to allow specification of the coordinate system of the center (`'image'` or `'world'`) when it is explicitly set.

Fixed

- Broken logic in *wiimatch.utils.create_coordinate_arrays()* code for generating coordinate arrays.

2.1.8 0.1.0 (09-May-2017)

Initial release.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

W

INDEX