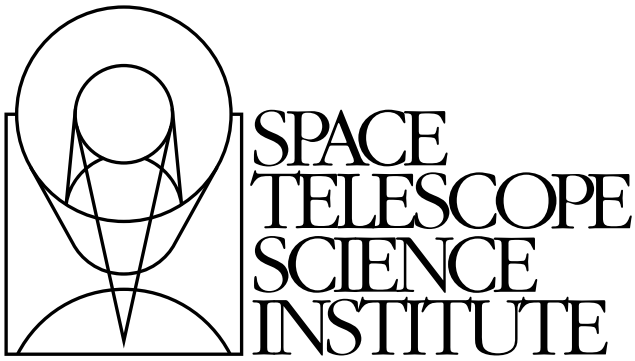

Image Documentation

Release 2.0

STScI

May 03, 2013

CONTENTS

This package contains various functions for performing processing of images.

Modules:

CHAPTER

ONE

IMAGE

This module provides access to various image manipulation functions.

stsci.image._image.**translate**(*a*, *sdx*, *sdy*, *output=None*, *mode='nearest'*, *cval=0.0*)

translate performs a translation of 'a' by (sdx, sdy) storing the result in 'output'.

sdx, sdy are float values.

supported 'mode's include:

'nearest' elements beyond boundary come from nearest edge pixel. 'wrap' elements beyond boundary come from the opposite array edge. 'reflect' elements beyond boundary come from reflection on same array

edge.

'constant' elements beyond boundary are set to 'cval'

stsci.image.combine.**average**(*arrays*, *output=None*, *outtype=None*, *nlow=0*, *nhigh=0*, *badmasks=None*)

average() nominally computes the average pixel value for a stack of identically shaped images.

arrays specifies a sequence of inputs arrays, which are nominally a
stack of identically shaped images.

output may be used to specify the output array. If none is specified,
either arrays[0] is copied or a new array of type 'outtype' is created.

outtype specifies the type of the output array when no 'output' is
specified.

nlow specifies the number of pixels to be excluded from average
on the low end of the pixel stack.

nhigh specifies the number of pixels to be excluded from average
on the high end of the pixel stack.

badmasks specifies boolean arrays corresponding to 'arrays', where true
indicates that a particular pixel is not to be included in the average calculation.

```
>>> a = num.arange(4)
>>> a = a.reshape((2,2))
>>> arrays = [a*16, a*4, a*2, a*8]
>>> average(arrays)
array([[ 0,  7],
       [15, 22]])
>>> average(arrays, nhigh=1)
array([[ 0,  4],
       [ 9, 14]])
>>> average(arrays, nlow=1)
```

```
array([[ 0,  9],
       [18, 28]])
>>> average(arrays, outtype=num.float32)
array([[  0. ,   7.5],
       [ 15. ,  22.5]], dtype=float32)
>>> bm = num.zeros((4,2,2), dtype=num.bool8)
>>> bm[2,...] = 1
>>> average(arrays, badmasks=bm)
array([[ 0,  9],
       [18, 28]])
>>> average(arrays, badmasks=threshhold(arrays, high=25))
array([[ 0,  7],
       [ 9, 14]])
```

stsci.image.combine.**iaverage**(*arrays*, *output=None*, *outtype=None*, *nlow=0*, *nhigh=0*, *badmasks=None*)

average() nominally computes the average pixel value for a stack of identically shaped images, filling pixels with no weight with the value from the first input array.

arrays specifies a sequence of inputs arrays, which are nominally a
stack of identically shaped images.

output may be used to specify the output array. If none is specified,
either arrays[0] is copied or a new array of type 'outtype' is created.

outtype specifies the type of the output array when no 'output' is
specified.

nlow specifies the number of pixels to be excluded from average
on the low end of the pixel stack.

nhigh specifies the number of pixels to be excluded from average
on the high end of the pixel stack.

badmasks specifies boolean arrays corresponding to 'arrays', where true
indicates that a particular pixel is not to be included in the average calculation.

```
>>> a = num.arange(4)
>>> a = a.reshape((2,2))
>>> arrays = [a*16, a*4, a*2, a*8]
>>> average(arrays)
array([[ 0,  7],
       [15, 22]])
>>> average(arrays, nhigh=1)
array([[ 0,  4],
       [ 9, 14]])
>>> average(arrays, nlow=1)
array([[ 0,  9],
       [18, 28]])
>>> average(arrays, outtype=num.float32)
array([[  0. ,   7.5],
       [ 15. ,  22.5]], dtype=float32)
>>> bm = num.zeros((4,2,2), dtype=num.bool8)
>>> bm[2,...] = 1
>>> average(arrays, badmasks=bm)
array([[ 0,  9],
       [18, 28]])
>>> average(arrays, badmasks=threshhold(arrays, high=25))
array([[ 0,  7],
       [ 9, 14]])
```

stsci.image.combine.**imedian**(*arrays*, *output=None*, *outtype=None*, *nlow=0*, *nhigh=0*, *badmasks=None*)

median() nominally computes the median pixels for a stack of identically shaped images, filling pixels with no weight with the value from the first input array.

arrays specifies a sequence of inputs arrays, which are nominally a
stack of identically shaped images.

output may be used to specify the output array. If none is specified,
either arrays[0] is copied or a new array of type 'outtype' is created.

outtype specifies the type of the output array when no 'output' is
specified.

nlow specifies the number of pixels to be excluded from median
on the low end of the pixel stack.

nhigh specifies the number of pixels to be excluded from median
on the high end of the pixel stack.

badmasks specifies boolean arrays corresponding to 'arrays', where true
indicates that a particular pixel is not to be included in the median calculation.

```
>>> a = num.arange(4)
>>> a = a.reshape((2,2))
>>> arrays = [a*16, a*4, a*2, a*8]
>>> median(arrays)
array([[ 0,  6],
       [12, 18]])
>>> median(arrays, nhigh=1)
array([[ 0,  4],
       [ 8, 12]])
>>> median(arrays, nlow=1)
array([[ 0,  8],
       [16, 24]])
>>> median(arrays, outtype=num.float32)
array([[  0.,   6.],
       [ 12.,  18.]], dtype=float32)
>>> bm = num.zeros((4,2,2), dtype=num.bool8)
>>> bm[2,...] = 1
>>> median(arrays, badmasks=bm)
array([[ 0,  8],
       [16, 24]])
>>> median(arrays, badmasks=threshhold(arrays, high=25))
array([[ 0,  6],
       [ 8, 12]])
```

stsci.image.combine.**median**(*arrays*, *output=None*, *outtype=None*, *nlow=0*, *nhigh=0*, *badmasks=None*)

median() nominally computes the median pixels for a stack of identically shaped images.

arrays specifies a sequence of inputs arrays, which are nominally a
stack of identically shaped images.

output may be used to specify the output array. If none is specified,
either arrays[0] is copied or a new array of type 'outtype' is created.

outtype specifies the type of the output array when no 'output' is
specified.

nlow specifies the number of pixels to be excluded from median
on the low end of the pixel stack.

nhigh specifies the number of pixels to be excluded from median
on the high end of the pixel stack.

badmasks specifies boolean arrays corresponding to 'arrays', where true
indicates that a particular pixel is not to be included in the median calculation.

```
>>> a = num.arange(4)
>>> a = a.reshape((2,2))
>>> arrays = [a*16, a*4, a*2, a*8]
>>> median(arrays)
array([[ 0,  6],
       [12, 18]])
>>> median(arrays, nhigh=1)
array([[ 0,  4],
       [ 8, 12]])
>>> median(arrays, nlow=1)
array([[ 0,  8],
       [16, 24]])
>>> median(arrays, outtype=num.float32)
array([[  0.,   6.],
       [ 12.,  18.]], dtype=float32)
>>> bm = num.zeros((4,2,2), dtype=num.bool8)
>>> bm[2,...] = 1
>>> median(arrays, badmasks=bm)
array([[ 0,  8],
       [16, 24]])
>>> median(arrays, badmasks=threshhold(arrays, high=25))
array([[ 0,  6],
       [ 8, 12]])
```

stsci.image.combine.**minimum**(*arrays*, *output=None*, *outtype=None*, *nlow=0*, *nhigh=0*, *badmasks=None*)

minimum() nominally computes the minimum pixel value for a stack of identically shaped images.

arrays specifies a sequence of inputs arrays, which are nominally a
stack of identically shaped images.

output may be used to specify the output array. If none is specified,
either arrays[0] is copied or a new array of type 'outtype' is created.

outtype specifies the type of the output array when no 'output' is
specified.

nlow specifies the number of pixels to be excluded from minimum
on the low end of the pixel stack.

nhigh specifies the number of pixels to be excluded from minimum
on the high end of the pixel stack.

badmasks specifies boolean arrays corresponding to 'arrays', where true
indicates that a particular pixel is not to be included in the minimum calculation.

```
>>> a = num.arange(4)
>>> a = a.reshape((2,2))
>>> arrays = [a*16, a*4, a*2, a*8]
>>> minimum(arrays)
array([[0, 2],
       [4, 6]])
>>> minimum(arrays, nhigh=1)
array([[0, 2],
       [4, 6]])
```

```
>>> minimum(arrays, nlow=1)
array([[ 0,  4],
       [ 8, 12]])
>>> minimum(arrays, outtype=num.float32)
array([[ 0.,  2.],
       [ 4.,  6.]], dtype=float32)
>>> bm = num.zeros((4,2,2), dtype=num.bool8)
>>> bm[2,...] = 1
>>> minimum(arrays, badmasks=bm)
array([[ 0,  4],
       [ 8, 12]])
>>> minimum(arrays, badmasks=threshhold(arrays, low=10))
array([[ 0, 16],
       [16, 12]])
```

stsci.image.combine.**threshhold**(*arrays*, *low=None*, *high=None*, *outputs=None*)

threshhold() computes a boolean array 'outputs' with corresponding elements for each element of arrays. The boolean value is true where each of the arrays values is < the low or >= the high threshholds.

```
>>> a=num.arange(100)
>>> a=a.reshape((10,10))
>>> (threshhold(a, 1, 50)).astype(num.int8)
array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], dtype=int8)
>>> (threshhold([ range(10)]*10, 3, 7)).astype(num.int8)
array([[1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 0, 0, 0, 0, 1, 1, 1]], dtype=int8)
>>> (threshhold(a, high=50)).astype(num.int8)
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], dtype=int8)
>>> (threshhold(a, low=50)).astype(num.int8)
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
```

```
[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],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=int8)
```

CHAPTER

TWO

NUMCOMBINE

This module allows the user to combine a stack of images into a single image using various methods.

class stsci.image.numcombine.**numCombine**(*arrObjectList*, *numarrayMaskList=None*, *combinationType='median'*, *nlow=0*, *nhigh=0*, *nkeep=1*, *upper=None*, *lower=None*)

A lite version of the imcombine IRAF task

Parameters

arrObjectList : list of ndarray

A sequence of inputs arrays, which are nominally a stack of identically shaped images.

numarrayMaskList : list of ndarray

A sequence of mask arrays to use for masking out 'bad' pixels from the combination The ndarray should be a numpy array, despite the variable name.

combinationType : {'median','imedian','iaverage','mean','sum','minimum'}

Type of operation should be used to combine the images The 'imedian' and 'iaverage' types ignore pixels which have been flagged as bad in all input arrays and returns the value from the last image in the stack for that pixel.

nlow : int [Default: 0]

Number of low pixels to throw out of the median calculation

nhigh : int [Default: 0]

Number of high pixels to throw out of the median calculation

nkeep : int [Default: 1]

Minimun number of pixels to keep for a valid computation

upper : float, optional

Throw out values >= to upper in a median calculation

lower: float, optional :

Throw out values < lower in a median calculation

Returns

combArrObj : ndarray

The attribute '.combArrObj' holds the combined output array.

Examples

This class can be used to create a median image from a stack of images with the following commands:

```
>>> import numpy as np
>>> from stsci.image import numcombine as nc
>>> a = np.ones([5,5],np.float32)
>>> b = a - 0.05
>>> c = a + 0.1
>>> result = nc.numCombine([a,b,c],combinationType='mean')
>>> result.combArrObj
array([[ 1.01666665,  1.01666665,  1.01666665,  1.01666665,  1.01666665],
       [ 1.01666665,  1.01666665,  1.01666665,  1.01666665,  1.01666665],
       [ 1.01666665,  1.01666665,  1.01666665,  1.01666665,  1.01666665],
       [ 1.01666665,  1.01666665,  1.01666665,  1.01666665,  1.01666665],
       [ 1.01666665,  1.01666665,  1.01666665,  1.01666665,  1.01666665]], dtype=float32)
```

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

S

S

A

I

M

N

S

T