montage_{wrapper} Documentation
Release 0.9.9.dev237

Thomas Robitaille

August 16, 2015
## Contents

I Introduction  
II Download  
III Installation  
IV Using `montage_wrapper`  
  1 Montage commands  
  2 High-level functions  
  3 MPI  
V Reference/API  
  4 `montage_wrapper.wrappers Module`  
    4.1 Functions  
  5 `montage_wrapper.commands Module`  
    5.1 Functions  
  6 `montage_wrapper.status Module`  
    6.1 Functions  
    6.2 Classes  
    6.3 Class Inheritance Diagram  
Python Module Index
Part I

Introduction
This Astropy-affiliated package provides a python wrapper to the Montage Astronomical Image Mosaic Engine, including both functions to access individual Montage commands, and high-level functions to facilitate mosaicking and re-projecting.
Part II

Download
The latest stable release can be downloaded from here.
Part III

Installation
This package is a wrapper, not a replacement, for the IPAC Montage mosaicking software. Therefore, Montage will need to be downloaded from http://montage.ipac.caltech.edu. Once you have downloaded the latest release on Montage from that website, the installation should look like:

```
tar xvzf Montage_v3.3.tar.gz
    cd Montage_v3.3
    make
```

then move the Montage_v3.3 directory to wherever you would like to keep the installation, and add 
<installation_dir>/Montage_v3.3/bin to your $PATH, where <installation_dir> is the directory inside which you put Montage_v3.3 (Montage does not support make install). To check that Montage is correctly installed, you can type:

```
mAdd
```

If you see something like:

```
[struct stat="ERROR", msg="Usage: mAdd [-p imgdir] [-n(o-areas)] [-a mean|median|count] [-e(xact-size)] [-d level] [-s statusfile] images.tbl template.hdr out.fits"]
```

then the installation succeeded. Otherwise, if you see:

```
mAdd: command not found
```

then you will need to make sure that the Montage_v3.3/bin contains commands like mAdd, and that the directory is correctly in your $PATH.

Once the IPAC Montage package is installed, you can install the Python wrapper with:

```
tar xvzf montage-wrapper-x.x.x.tar.gz
    cd montage-wrapper-x.x.x
    python setup.py install
```

(replacing x.x.x with the actual version number). The only dependencies are numpy and astropy.
Part IV

Using montage_wrapper
Montage commands

The Montage wrapper is imported using:

```
>>> import montage_wrapper
```

or, for clarity:

```
>>> import montage_wrapper as montage
```

All Montage commands (except mJPEG, mMakeImg, and mTileImage) are accessible via Python functions. For example, to access mProject, use:

```
>>> montage.mProject(...)
```

and see `mProject()` for available options. Each Montage command returns a `Struct` object that contains information/diagnostics. The following example shows how to use the Montage command wrappers, and how to access the diagnostics:

```
>>> montage.mArchiveList('2MASS', 'K', 'm31', 0.5, 0.5, 'm31_list.tbl')
count : 18
stat : OK
>>> montage.mMakeHdr('m31_list.tbl', 'header.hdr', north_aligned=True)
count : 18
lat4 : 41.744136
stat : OK
lat1 : 40.912238
lat3 : 41.744136
latsize : 0.831951
clon : 10.717965
lonsize : 0.830562
posang : 0.0
lon4 : 11.274528
lat2 : 40.912238
lon1 : 11.267467
clat : 41.32951
lon2 : 10.168464
lon3 : 10.161403
>>> s = montage.mMakeHdr('m31_list.tbl', 'header.hdr', north_aligned=True)
>>> s.stat
'OK'
>>> s.lon1
11.267467
```

See `Reference/API` for a full list of available commands and documentation.
High-level functions

In addition to wrappers to the individual Montage commands, the following high-level functions are available:

- `reproject`: reproject a FITS file
- `reproject_hdu`: reproject an FITS HDU object
- `mosaic`: mosaic all FITS files in a directory

For example, to mosaic all FITS files in a directory called `raw` using background matching, use:

```python
>>> montage.mosaic('raw', 'mosaic', background_match=True)
```

In this specific example, a mosaic header will automatically be constructed from the input files.

For more details on how to use these, see the Reference/API section.
A few Montage commands can be run using MPI for parallelization (see here). For MPI-enabled commands (such as `mProjExec`), the use of MPI is controlled via the `mpi=` argument. For example, to call `mProjExec` using MPI, call `montage.mProjExec(..., mpi=True)` (rather than `montage.mProjExecMPI`, which does not exist). Note however that this requires the MPI versions of the Montage commands to be installed (which is not the case by default).

Different MPI installations require different commands (e.g. `mpirun` vs `mpiexec`) as well as different options, so it is possible to customize the MPI command:

```python
>>> import montage_wrapper as montage
>>> montage.set_mpi_command('mpiexec -n {n_proc} {executable}')
```

The command string should include `{n_proc}`, which will be replaced by the number of processes, and `{executable}`, which will be replaced by the appropriate Montage executable. The current MPI command can be accessed with:

```python
>>> from montage_wrapper.mpi import MPI_COMMAND
>>> MPI_COMMAND
'mpiexec -n {n_proc} {executable}''
Part V

Reference/API
montage_wrapper.wrappers Module

4.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mProject_auto(*args, **kwargs)</td>
<td>Run mProject, automatically selecting whether to run mProject or mProjectPP if possible (fast plane-to-plane projection). For details on required and optional arguments, see help(mProject).</td>
</tr>
<tr>
<td>mosaic(input_dir, output_dir[, header, ...])</td>
<td>Combine FITS files into a mosaic</td>
</tr>
<tr>
<td>reproject(in_images, out_images[, header, ...])</td>
<td>General-purpose reprojection routine.</td>
</tr>
<tr>
<td>reproject_cube(in_image, out_image[, ...])</td>
<td>Cube reprojection routine.</td>
</tr>
<tr>
<td>reproject_hdu(in_hdu, **kwargs)</td>
<td>Reproject an image (HDU version)</td>
</tr>
</tbody>
</table>

4.1.1 mProject_auto

montage_wrapper.wrappers.mProject_auto(*args, **kwargs)

Run mProject, automatically selecting whether to run mProject or mProjectPP if possible (fast plane-to-plane projection). For details on required and optional arguments, see help(mProject).

4.1.2 mosaic

montage_wrapper.wrappers.mosaic(input_dir, output_dir, header=None, image_table=None, mpi=False, n_proc=8, background_match=False, imlist=None, combine='mean', exact_size=False, cleanup=True, bitpix=-32, level_only=True, work_dir=None, background_n_iter=None, subset_fast=False, hdu=None)

Combine FITS files into a mosaic

**Parameters**

- **input_dir**: str
  The directory containing the input FITS files

- **output_dir**: str
  The output directory to create

- **header**: str, optional
  The header to project to. If this is not specified, then an optimal header is chosen.

- **image_table**: str, optional
  The table file containing the list of input images. This can be specified to avoid recomputing it every time a mosaic is made from the same set of input files.
**mpi** : bool, optional

Whether to use MPI whenever possible (requires the MPI-enabled Montage binaries to be installed).

**n_proc** : int, optional

The number of processes to use if *mpi* is set to True

**background_match** : bool, optional

Whether to include a background-matching step

**imglist** : str, optional

A list of images to use (useful if not all the files inside *input_dir* should be combined).

**combine** : str, optional

How to combine the images - this should be one of 'mean', 'median', or 'count'.

**exact_size** : bool, optional

Whether the output mosaic should match the input header exactly, or whether the mosaic should be trimmed if possible.

**cleanup** : bool, optional

Whether to remove any temporary directories used for mosaicking

**bitpix** : int, optional

BITPIX value for the output FITS file (default is -32). Possible values are: 8 (character or unsigned binary integer), 16 (16-bit integer), 32 (32-bit integer), -32 (single precision floating point), -64 (double precision floating point).

**level_only** : bool, optional

When doing background matching, whether to only allow changes in the level of frames, not the slope.

**work_dir** : str, optional

The temporary directory to use for the mosaicking. By default, this is a temporary directory in a system-defined location, but it can be useful to specify this manually for debugging purposes.

**background_n_iter** : int, optional

Number of iterations for the background matching (defaults to 5000). Can be between 1 and 32767.

**subset_fast** : bool, optional

Whether to use the fast mode for mSubset

**hdu** : int, optional

Which HDU to use when mosaicing
4.1.3 reproject

```
montage_wrapper.wrappers.reproject(in_images, out_images, header=None, bitpix=None, north_aligned=False, system=None, equinox=None, factor=None, common=False, exact_size=False, hdu=None, cleanup=True, silent_cleanup=False)
```

General-purpose reprojection routine.

If one input/output image is specified, and the header argument is set, the routine is equivalent to using `mProject` or `mProjectPP`. If `header=None` is not set, a new header is computed by taking into account the `north_aligned`, `system`, and `equinox` arguments (if set).

If tuples of input/output images are specified, the tuples need to have the same number of elements. If `header=None` is specified, all images are projected to this common projection. If `header=None` is not specified, then a new header is computed a new header is computed by taking into account the `north_aligned`, `system`, and `equinox` arguments (if set). If `common=False`, then a header is computed for each individual image, whereas if `common=True`, an optimal header is computed for all images.

**Parameters**

- **in_images**: str or list
  Path(s) of input FITS file(s) to be reprojected.

- **out_images**: str or list
  Path(s) of output FITS file(s) to be created.

- **header**: str, optional
  Path to the header file to use for re-projection.

- **bitpix**: int, optional
  BITPIX value for the output FITS file (default is -64). Possible values are: 8 (character or unsigned binary integer), 16 (16-bit integer), 32 (32-bit integer), -32 (single precision floating point), -64 (double precision floating point).

- **north_aligned**: bool, optional
  Align the pixel y-axis with North.

- **system**: str, optional
  Specifies the coordinate system. Possible values are: ‘EQUJ’, ‘EQUB’, ‘ECLJ’, ‘ECLB’, ‘GAL’, ‘SGAL’

- **equinox**: str, optional
  If a coordinate system is specified, the equinox can also be given in the form ‘YYYY’. Default is ‘J2000’.

- **factor**: float, optional
  Drizzle factor (see `mProject`)

- **exact_size**: bool, optional
  Whether to reproject the image(s) to the exact header specified (i.e. whether cropping is unacceptable).

- **hdu**: int, optional
  The HDU to use in the file(s).

- **silent_cleanup**: bool, optional
Hide messages related to tmp directory removal

**common** : str, optional

Compute a common optimal header for all images (only used if header=None and if multiple files are being reprojected)

### 4.1.4 reproject_cube

`montage_wrapper.wrappers.reproject_cube`(*in_image*, *out_image*, *header=None*, *bitpix=None*, *north_aligned=False*, *system=None*, *equinox=None*, *factor=None*, *common=False*, *cleanup=True*, *clobber=False*, *silent_cleanup=True*, *hdu=0*)

Cube reprojection routine.  

If one input/output image is specified, and the header argument is set, the routine is equivalent to using mProject or mProjectPP. If header= is not set, a new header is computed by taking into account the north_aligned, system, and equinox arguments (if set).

**Parameters**

- **in_image** : str
  
  Path of input FITS file to be reprojected.

- **out_image** : str
  
  Path of output FITS file to be created.

- **header** : str, optional
  
  Path to the header file to use for re-projection.

- **bitpix** : int, optional
  
  BITPIX value for the output FITS file (default is -64). Possible values are: 8 (character or unsigned binary integer), 16 (16-bit integer), 32 (32-bit integer), -32 (single precision floating point), -64 (double precision floating point).

- **north_aligned** : bool, optional
  
  Align the pixel y-axis with North

- **system** : str, optional
  
  Specifies the coordinate system Possible values are: ‘EQUJ’, ‘EQUB’, ‘ECLJ’, ‘ECLB’, ‘GAL’, ‘SGAL’

- **equinox** : str, optional
  
  If a coordinate system is specified, the equinox can also be given in the form ‘YYYY’. Default is ‘J2000’.

- **factor** : float, optional
  
  Drizzle factor (see mProject)

- **clobber** : bool, optional
  
  Overwrite the data cube if it already exists?

- **silent_cleanup** : bool, optional
  
  Hide messages related to tmp directory removal (there will be one for each plane of the cube if set to False)
hdu: int

Defaults to zero. Selects which HDU to use from the FITS file.

### 4.1.5 reproject_hdu

```python
montage_wrapper.wrappers.reproject_hdu(in_hdu, **kwargs)
```

Reproject an image (HDU version)

**Parameters**

- **in_hdu**: PrimaryHDU or ImageHDU
  
  Input FITS HDU to be reprojected.

**Notes**

Additional keyword arguments are passed to `reproject()`
## 5.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mAdd(images_table, template_header, out_image)</code></td>
<td>Coadd the reprojected images in an input list to form an output mosaic with FITS header keywords specified in a header file.</td>
</tr>
<tr>
<td><code>mAddExec(images_table, template_header, ...)</code></td>
<td>Builds a series of outputs (which together make up a tiled output) through multiple executions of <code>mAdd</code>.</td>
</tr>
<tr>
<td><code>mArchiveExec(region_table[, debug_level])</code></td>
<td>Given a table of archive images (generated by <code>mArchiveList</code>), calls <code>mArchiveGet</code> on each one in sequence to retrieve all the files into the current directory.</td>
</tr>
<tr>
<td><code>mArchiveGet(remote_ref, local_file[, debug, raw])</code></td>
<td>Retrieve a single FITS image from a remote archive, using a basic URL GET method.</td>
</tr>
<tr>
<td><code>mArchiveList(survey, band, ...)</code></td>
<td>Given a location on the sky, archive name, and size in degrees, contact the IRSA server to retrieve a list of archive images.</td>
</tr>
<tr>
<td><code>mBackground(in_image, out_image, A, B, C[, ...])</code></td>
<td>Remove a background plane from a FITS image.</td>
</tr>
<tr>
<td><code>mBackground_tab(in_image, out_image, ...[, ...])</code></td>
<td>Remove a background plane from a FITS image.</td>
</tr>
<tr>
<td><code>mBestImage(images_table, ra, dec[, debug])</code></td>
<td>Given a list of images and a position on the sky, determine which image covers the location &quot;best&quot; (i.e., the one where the position is farthest from the nearest edge).</td>
</tr>
<tr>
<td><code>mBgExec(images_table, corrections_table, ...)</code></td>
<td>Runs <code>mBackground</code> on all the images in a metadata table, using the corrections generated by <code>mFitExec</code>.</td>
</tr>
<tr>
<td><code>mBgModel(images_table, fits_table, ...[, ...])</code></td>
<td><code>mBgModel</code> is a modelling/fitting program. It uses the image-to-image method to fit a plane to the data.</td>
</tr>
<tr>
<td><code>mCatMap(in_table, out_image, template_header)</code></td>
<td><code>mCatMap</code> is a point-source imaging program. The user defines a general location and size, and this program creates a FITS header table that can be used to create images.</td>
</tr>
<tr>
<td><code>mConvert(in_image, out_image[, debug_level, ...])</code></td>
<td><code>mConvert</code> changes the datatype of an image. When converting to floating point, it uses a basic algorithm to handle the conversion.</td>
</tr>
<tr>
<td><code>mCoverageCheck(in_table, out_table, mode[, ...])</code></td>
<td><code>mCoverageCheck</code> can be used to subset an image metadata table (containing information about the tiles of an image) and extract a subset of images based on certain criteria.</td>
</tr>
<tr>
<td><code>mDiff(in_image_1, in_image_2, out_image, ...)</code></td>
<td><code>mDiff</code> calculates a simple difference between a single pair of overlapping images.</td>
</tr>
<tr>
<td><code>mDiffExec(diffs_table, template_header, diff_dir)</code></td>
<td>Runs <code>mDiff</code> on all the pairs identified by <code>mOverlaps</code>.</td>
</tr>
<tr>
<td><code>mDiffFitExec(diffs_table, fits_table, diff_dir)</code></td>
<td>Using the table of overlaps found by <code>mOverlaps</code>, <code>mDiffFitExec</code> runs both <code>mDiff</code> and <code>mFitplane</code> for each record.</td>
</tr>
<tr>
<td><code>mExec(survey, band[, raw_dir, n_tile_x, ...])</code></td>
<td>The <code>mExec</code> module is a mosaicking executive for 2MASS, SDSS, and DSS data. It reprojects images to a common scale and then mosaics them together.</td>
</tr>
<tr>
<td><code>mFitExec(diffs_table, fits_table, diff_dir)</code></td>
<td>Runs <code>mFitplane</code> on all the difference images identified by <code>mOverlaps</code> and generated by <code>mDiff</code> or <code>mDiffExec</code>.</td>
</tr>
<tr>
<td><code>mFitplane(in_image[, border, debug_level, ...])</code></td>
<td>Uses least squares to fit a plane (excluding outlier pixels) to an image.</td>
</tr>
<tr>
<td><code>mFixNaN(in_image, out_image[, debug_level, ...])</code></td>
<td>Converts NaNs found in the image to some other value (given by the user), or converts a range of supplied values into NaNs.</td>
</tr>
<tr>
<td><code>mFlattenExec(images_table, flat_dir[, ...])</code></td>
<td>Runs both <code>mFitPlane</code> and <code>mBackground</code> on a set of images.</td>
</tr>
<tr>
<td><code>mGetHdr(in_image, img_header[, debug, hdu, ...])</code></td>
<td>Reads the header from a FITS image and prints it out to a text file.</td>
</tr>
<tr>
<td><code>mHdr(object_or_location, width, out_file[, ...])</code></td>
<td>Connects to the IRSA service HdrTemplate to create a header template based on a location.</td>
</tr>
<tr>
<td><code>mHdrCheck(in_image, status_file)</code></td>
<td><code>mHdrCheck</code> reads the header from a FITS image (or an ASCII header file), and checks that it is valid.</td>
</tr>
<tr>
<td><code>mHdrtbl(directory, images_table[, ...])</code></td>
<td><code>mHdrtbl</code> operates in a fashion similar to <code>mImgtbl</code>, but it is used on a set of files instead of a single image.</td>
</tr>
<tr>
<td><code>mImgtbl(directory, images_table[, ...])</code></td>
<td><code>mImgtbl</code> extracts the FITS header geometry information from a set of files.</td>
</tr>
<tr>
<td><code>mMakeHdr(images_table, template_header[, ...])</code></td>
<td>From a list of images to be mosaicked together, <code>mMakeHdr</code> generates the FITS header that best describes the output image.</td>
</tr>
<tr>
<td><code>mOverlaps(images_table, diffs_table[, ...])</code></td>
<td>Analyze an image metadata table to determine a list of overlapping images.</td>
</tr>
<tr>
<td><code>mPix2Coord(template_header, ipix, jypix[, ...])</code></td>
<td>Takes an image FITS header template and a pixel (x,y) coordinate, and outputs the corresponding sky location.</td>
</tr>
<tr>
<td><code>mProjExec(images_table, template_header, ...)</code></td>
<td>An executive which runs <code>mProject</code> (or, if possible for the input/output projections, <code>mProjectPP</code>) for each image in an image metadata table.</td>
</tr>
<tr>
<td><code>mProject(in_image, out_image, template_header)</code></td>
<td><code>mProject</code> reprojects a single image to the scale defined in a FITS header.</td>
</tr>
<tr>
<td><code>mProjectPP(in_image, out_image, template_header)</code></td>
<td><code>mProjectPP</code> reprojects a single image to the scale defined in an alternate FITS header.</td>
</tr>
<tr>
<td><code>mPutHdr(in_image, out_image, template_header)</code></td>
<td>Replaces the header of the input file with one supplied by the user.</td>
</tr>
<tr>
<td><code>mRotate(in_image, out_image[, debug_level, ...])</code></td>
<td>Rotates a FITS image by an arbitrary angle.</td>
</tr>
</tbody>
</table>
Table 5.1 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mShrink(in_image, out_image, factor[, ...])</td>
<td>A utility for reducing the size of a FITS file, by averaging blocks of pixels.</td>
</tr>
<tr>
<td>mSubimage(in_image, out_image, ra, dec, xsize)</td>
<td>Creates a subimage (or “cutout”) of a FITS file.</td>
</tr>
<tr>
<td>mSubimage_pix(in_image, out_image, ...[, ...])</td>
<td>Creates a subimage (or “cutout”) of a FITS file (‘pixel’ mode)</td>
</tr>
<tr>
<td>mSubset(images_table, template_header, ...)</td>
<td>Generates a table of images that is a subset of the input table, containing only...</td>
</tr>
<tr>
<td>mTANHdr(orig_header, new_header[, debug, ...])</td>
<td>Analyzes a template file and determines if there would be an adequate equivalent...</td>
</tr>
<tr>
<td>mTbSort(in_table, column_name, out_table[, ...])</td>
<td>Sorts a table on numeric values.</td>
</tr>
<tr>
<td>mTileHdr(orig_header, new_header, n_x, n_y, ...)</td>
<td>Takes a header template file and creates another which represents one of a regular...</td>
</tr>
<tr>
<td>mTileImage(in_image, tiles_x, tiles_y[, ...])</td>
<td>mTileImage splits up an image into a given number of tiles.</td>
</tr>
</tbody>
</table>

### 5.1.1 mAdd

```python
montage_wrapper.commands.mAdd(images_table, template_header, out_image, img_dir=None, no_area=False, type=None, exact=False, debug_level=None, status_file=None, mpi=False, n_proc=8)
```

Coadd the reprojected images in an input list to form an output mosaic with FITS header keywords specified in a header file. Creates two output files, one containing the coadded pixel values, and the other containing coadded pixel area values. The pixel area values can be used as a weighting function if the output pixel values are themselves to be coadded with other projected images, and may also be used in validating the fidelity of the output pixel values.

**Parameters**

- **images_table**: str
  ASCII table (generated by mImgtbl) containing metadata for all images to be coadded.

- **template_header**: str
  FITS header template to be used in generation of output FITS

- **out_image**: str
  Name of output FITS image.

- **img_dir**: str, optional
  Specifies path to directory containing reprojected images. If the img_dir option is not included, mAdd will look for the input images in the current working directory.

- **no_area**: bool, optional
  Co-addition ignores weighting by pixel areas and performs coaddition based only on pixel positions. Will not output an area image for the output image.

- **type**: str, optional
  Select type of averaging to be done on accumulated pixel values (either mean or median). To generate a map showing counts of how many times each pixel was overlapped by the input images, use count.

- **exact**: bool, optional
  Enables exact size mode. The output image will match the header template exactly, instead of shrinking the output to fit the data.

- **debug_level**: int, optional
  Turns on debugging to the specified level (1-3).

- **status_file**: str, optional
  mAdd output and errors will be written to status_file instead of stdout.
mpi : bool, optional
    If set to True, will use the MPI-enabled versions of the Montage executable.

n_proc : int, optional
    If mpi is set to True, n_proc is the number of processes to run simultaneously (default is 8)

5.1.2 mAddExec

    montage_wrapper.commands.mAddExec(images_table, template_header, tile_dir, out_image, img_dir=None, no_area=False, type=None, exact=False, debug_level=None, status_file=None, mpi=False, n_proc=8)

Builds a series of outputs (which together make up a tiled output) through multiple executions of the mAdd modules.

Parameters

images_table : str
    ASCII table (generated by mImgtbl) containing metadata for all images to be coadded.

template_header : str
    FITS header template to be used in generation of output FITS

tile_dir : str
    Directory to contain output tile images and header templates

out_image : str
    Prefix for output tile images

img_dir : str, optional
    Specifies path to directory containing reprojected images. If the img_dir option is not included, mAdd will look for the input images in the current working directory.

no_area : bool, optional
    Co-addition ignores weighting by pixel areas and performs coaddition based only on pixel positions. Will not output an area image for the output image.

type : str, optional
    Select type of averaging to be done on accumulated pixel values (either mean or median).

exact : bool, optional
    Enables exact size mode. The output image will match the header template exactly, instead of shrinking the output to fit the data.

debug_level : int, optional
    Turns on debugging to the specified level (1-3).

status_file : str, optional
    mAdd output and errors will be written to status_file instead of stdout.

mpi : bool, optional
    If set to True, will use the MPI-enabled versions of the Montage executable.
n_proc : int, optional
        If mpi is set to True, n_proc is the number of processes to run simultaneously (default is 8)

5.1.3 mArchiveExec

montage_wrapper.commands.mArchiveExec(region_table, debug_level=None)
        Given a table of archive images (generated by mArchiveList), calls mArchiveGet on each one in sequence to retrieve all the files into the current directory.

Parameters
    region_table : str
        Table of archive images, generated by mArchiveList.
    debug_level : int, optional
        Prints out additional debugging information; in this version, the only supported level is 1.

5.1.4 mArchiveGet

montage_wrapper.commands.mArchiveGet(remote_ref, local_file, debug=False, raw=False)
        Retrieve a single FITS image from a remote archive, using a basic URL GET but with a structured output.

Parameters
    remote_ref : str
        URL of remote FITS file to retrieve (should be in quotes). See mArchiveList for more information.
    local_file : str
        Full path/filename of the retrieved file.
    debug : bool, optional
        Print additional debugging information.
    raw : bool, optional
        “Raw” mode - use a raw HTTP GET (no “HTTP/1.1” etc in the header); necessary for communication with some servers.

5.1.5 mArchiveList

montage_wrapper.commands.mArchiveList(survey, band, object_or_location, width, height, out_file)
        Given a location on the sky, archive name, and size in degrees, contact the IRSA server to retrieve a list of archive images. The list contains enough information to support mArchiveGet downloads.

Parameters
    survey : str
        Can be one of: 2MASS DSS SDSS DPOSS
    band : str
        Case insensitive - can be one of: (2MASS) j, h, k (SDSS) u, g, r, i, z (DPOSS) f, j, n (DSS) DSS1, DSS1R, DSS1B, DSS2, DSS2B, DSS2R, DSS2IR
object_or_location : str
    Object name or coordinate string to be resolved by NED (if string includes spaces, must be surrounded by double quotes)

width : float
    Width of area of interest, in degrees

height : float
    Height of area of interest, in degrees

out_file : str
    Path to output table

5.1.6 mBackground

montage_wrapper.commands.mBackground(in_image, out_image, A, B, C, debug_level=None, no_area=False, status_file=None)
Remove a background plane from a FITS image. The background correction applied to the image is specified as Ax+By+C, where (x,y) is the pixel coordinate using the image center as the origin, and (A,B,C) are the background plane parameters specified as linear coefficients. To run in ‘table’ mode, see mBackground_tab.

Parameters
in_image : str
    Input FITS file

out_image : str
    Output FITS file

A, B, C : str
    Corrections (as given by mFitplane or mFitExec)

ddebug_level : int, optional
    Turns on debugging to the specified level.

no_area : bool, optional
    Indicates that no area images are present (assumes equal weighting for each data pixel)

status_file : str, optional
    mBackground output and errors will be written to status_file instead of stdout.

5.1.7 mBackground_tab

montage_wrapper.commands.mBackground_tab(in_image, out_image, images_table, corrections_table, debug_level=None, no_area=False, status_file=None)
Remove a background plane from a FITS image. The background correction applied to the image is specified as Ax+By+C, where (x,y) is the pixel coordinate using the image center as the origin, and (A,B,C) are the background plane parameters specified as linear coefficients. This method runs mBackground_tab in ‘table’ mode.

Parameters
in_image : str
    Input FITS file
5.1.8 **mBestImage**

`montage_wrapper.commands.mBestImage(images_table, ra, dec, debug=False)`

Given a list of images and a position on the sky, determine which image covers the location “best” (i.e., the one where the position is farthest from the nearest edge).

**Parameters**

- **images_table**: str
  Input table of image metadata (as generated by mImgtbl).

- **ra**: float
  RA of location of interest (in degrees)

- **dec**: float
  Declination of location of interest (in degrees)

- **debug_level**: int, optional
  Turn on debugging to the specified level (1 or 2)

5.1.9 **mBgExec**

`montage_wrapper.commands.mBgExec(images_table, corrections_table, corr_dir, proj_dir=None, status_file=None, debug=False, no_area=False, mpi=False, n_proc=8)`

Runs mBackground on all the images in a metadata table, using the corrections generated by mFitExec.

**Parameters**

- **images_table**: str
  Image metadata table generated by mImgtbl.

- **corrections_table**: str
  Table of corrections generated by mFitExec

- **corr_dir**: str
Directory where output images should be written

proj_dir : str, optional
    Specifies the path to the directory containing the projected images.

status_file : str, optional
    Writes output message to status_file instead of to stdout

debug : bool, optional
    Turns on debugging

no_area : bool, optional
    Indicates that no area images are present (assumes equal weighting for each pixel)

mpi : bool, optional
    If set to True, will use the MPI-enabled versions of the Montage executable.

n_proc : int, optional
    If mpi is set to True, n_proc is the number of processes to run simultaneously (default is 8)

5.1.10 mBgModel

```
montage_wrapper.commands.mBgModel(images_table, fits_table, corrections_table, n_iter=None, level_only=False, debug_level=None, ref_img=None, status_file=None)
```

mBgModel is a modelling/fitting program. It uses the image-to-image difference parameter table created by mFitExec to interactively determine a set of corrections to apply to each image in order to achieve a “best” global fit.

Parameters

images_table : str
    Image metadata table generated by mImgtbl.

fits_table : str
    Plane fitting table generated by mFitExec.

corrections_table : str
    Output table of background corrections

n_iter : int, optional
    Number of iterations (without option, defaults to 5000). Can be between 1 and 32767.

level_only : bool, optional
    Calculate level adjustments only (ie, don’t attempt to match the slopes)

debug_level : int, optional
    Turns on debugging to the specified level (1-3).

ref_img : str, optional
    Turns on additional debugging for the nth image in images_table.

status_file : str, optional

5.1. Functions 35
mBgModel output and errors are written to status_file instead of to stdout.

5.1.11 mCatMap

```python
montage_wrapper.commands.mCatMap(in_table, out_image, template_header, column=None, ref_mag=None, debug_level=None, size=None)
```

mCatMap is a point-source imaging program. The user defines a general output FITS image, and its pixels are populated from a table of point sources. The source fluxes (or just source counts) from the table are added into the appropriate pixel to create an output image.

**Parameters**

- `in_table` : str
  
  Input table of source metadata.

- `out_image` : str
  
  Path of output FITS file.

- `template_header` : str
  
  ASCII header template defining output FITS file.

- `column` : str, optional
  
  Name of the table column that contains flux levels. If not specified, pixels will be populated with source counts rather than summed flux values.

- `ref_mag` : float, optional
  
  Set a reference magnitude to use when calculating fluxes.

- `debug_level` : int, optional
  
  Turn on debugging to the specified level (1-3).

- `size` : int, optional
  
  Set a spread size for point sources (default is to use no spread). Allowed values are 3 or 5.

5.1.12 mConvert

```python
montage_wrapper.commands.mConvert(in_image, out_image, debug_level=None, status_file=None, bitpix=None, min_val=None, max_val=None, blank_value=None)
```

mConvert changes the datatype of an image. When converting to floating point, no additional information is needed. However, when converting from higher precision (e.g. 64-bit floating point) to lower (e.g. 16-bit integer), scaling information is necessary. This can be given explicitly by the user or guessed by the program.

**Parameters**

- `in_image` : str
  
  Input image filename.

- `out_image` : str
  
  Output image filename.

- `debug_level` : int, optional
  
  Turns on debugging to the specified level (1-3).

- `status_file` : str, optional
mBgModel output and errors are written to status_file instead of to stdout.

**bitpix** : int, optional

BITPIX value for the output FITS file (default is -64). Possible values are: 8 (character or unsigned binary integer), 16 (16-bit integer), 32 (32-bit integer), -32 (single precision floating point), -64 (double precision floating point).

**min_val** : int, optional

Pixel data value in the input image which should be treated as a minimum (value of 0) in the output image when converting from floating point to integer (default for BITPIX 8: 0; BITPIX 16: -32767; BITPIX 32: -2147483647)

**max_val** : int, optional

Pixel data value in the input image which should be treated as a maximum (value of 255 or 32768) in the output image when converting from floating point to integer (Default for BITPIX 8: 255; BITPIX 16: 32768; BITPIX 32: 2147483648)

**blank_value** : int, optional

If converting down to an integer scale: value to be used in the output image to represent blank pixels (NaN) from the input image. Default value is min_val.

### 5.1.13 mCoverageCheck

```python
mCoverageCheck(in_table, out_table, mode, polygon=None, ra=None, dec=None, width=None, height=None, rotation=None, radius=None, header=None, status_file=None)
```

mCoverageCheck can be used to subset an image metadata table (containing FITS/WCS information or image corners) by determining which records in the table represent images that overlap with a region definition (box or circle in the sky) given on the command line.

**Parameters**

- **in_table** : str
  
  Input metadata table.

- **out_table** : str
  
  Output metadata table, to contain subset of in_table.

- **mode** : str
  
  How to check for coverage:
  
  - ‘points’: use a polygon with points specified by polygon=
  - ‘box’: use a rectangular box with center given by ra= and dec=, and the width given by width=. Optionally, the height and rotation can be given by height= and rotation=
  - ‘circle’: use a circle with center given by ra= and dec= and radius given by radius=
  - ‘point’: use a point given by ra= and dec=
  - ‘header’: use a header file given by header=

- **polygon** : list
  
  A polygon which should be given as [(ra1, dec1), (ra2, dec2), ..., (raN, decN)].

- **ra** : float, optional
  
  The right ascension of the box, circle, or point
dec : float, optional
The declination of the box, circle, or point

width : float, optional
The width of the box

height : float, optional
The height of the box

rotation : float, optional
The rotation of the box

radius : float, optional
The radius of the circle

header : str, optional
A header file

status_file : str, optional
Output and errors are sent to status_file instead of to stdout

5.1.14 mDiff

montage_wrapper.commands.mDiff(in_image_1, in_image_2, out_image, template_header, debug_level=None, no_area=False, status_file=None)
mDiff calculates a simple difference between a single pair of overlapping images. This is meant for use on reprojected images where the pixels already line up exactly. mDiff analyzes an image metadata table to determine a list of overlapping images. Each image is compared with every other image to determine all overlapping image pairs. A pair of images are deemed to overlap if any pixel around the perimeter of one image falls within the boundary of the other image.

Parameters

in_image_1 : str
First input FITS file (Also needs area image in1_area_image, or use the no_area option)

in_image_2 : str
Second input FITS file.(Also needs area image in2_area_image, or use the no_area option)

out_image : str
Difference FITS image to be generated.

template_header : str
FITS header template used to generate output image.

debug_level : int, optional
Turns on debugging to the specified level (1-4).

no_area : bool, optional
No-area-images option. Creates difference image without requiring pixel area FITS image

status_file : str, optional
Output and errors are sent to status_file instead of to stdout

### 5.1.15 mDiffExec

```python
montage_wrapper.commands.mDiffExec(diffs_table, template_header, diff_dir, proj_dir=None, debug=False, no_area=False, status_file=None, mpi=False, n_proc=8)
```

Runs mDiff on all the pairs identified by mOverlaps.

**Parameters**

- **diffs_table**: str
  - Table generated by mOverlaps for the images in proj_dir.
- **template_header**: str
  - FITS header template for output files.
- **diff_dir**: str
  - Path to output files.
- **proj_dir**: str, optional
  - Specifies path to the directory containing reprojected input images.
- **debug**: bool, optional
  - Turns on debugging.
- **no_area**: bool, optional
  - No-area-images option. Creates difference image without requiring _area FITS images
- **status_file**: str, optional
  - Output and errors are sent to status_file instead of to stdout
- **mpi**: bool, optional
  - If set to True, will use the MPI-enabled versions of the Montage executable.
- **n_proc**: int, optional
  - If mpi is set to True, n_proc is the number of processes to run simultaneously (default is 8)

### 5.1.16 mDiffFitExec

```python
montage_wrapper.commands.mDiffFitExec(diffs_table, fits_table, diff_dir, debug=False, status_file=None)
```

Using the table of overlaps found by mOverlaps, mDiffFitExec runs both mDiff and mFitplane for each record. The fitting parameters are written to a file to be used by mBgModel.

**Parameters**

- **diffs_table**: str
  - Overlap table generated by mOverlaps, the last column of which contains the filenames of the difference images generated by mDiffExec.
- **fits_table**: str
  - Output table of difference paramaters.
- **diff_dir**: str
  - Path to output directory for difference images.
Directory containing difference images.

**debug** : bool, optional

Turns on debugging

**status_file** : str, optional

Writes output message to status_file instead of to stdout

### 5.1.17 mExec

```python
def mExec(survey, band, raw_dir=None, n_tile_x=None, n_tile_y=None, level_only=False, keep=False, remove=False, output_image=None, debug_level=None, region_header=None, header=None, workspace_dir=None)
```

The `mExec` module is a mosaicking executive for 2MASS, SDSS, and DSS data. It includes remote data and metadata access. Alternatively, users can mosaic a set of data already on disk.

**Parameters**

- **survey**, **band** : str
  
  If not mosaicking user-provided data (raw_dir option), must select one of the following combinations of survey and band: 2MASS [j, h, k] SDSS [u, g, r, i, z] DSS [DSS1, DSS1R, DSS1B, DSS2, DSS2B, DSS2R, DSS2IR]

- **raw_dir** : str, optional
  
  Provide path to directory containing original (“raw”) data which will be reprojected and mosaicked. Not necessary if using mExec to retrieve remote data from the 2MASS, SDSS or DSS surveys.

- **n_tile_x** : int, optional
  
  Number of output tiles to create along the X-axis - default is 1 for a single mosaicked image.

- **n_tile_y** : int, optional
  
  Number of output tiles to create along the Y-axis - default is equal to n_tile_x.

- **level_only** : bool, optional
  
  “Level-only” option (see mBgModel)

- **keep** : bool, optional
  
  If retrieving data from a remote archive, the “keep” option will leave the original data products on disk after generating a mosaic. Without this option, raw data will be deleted (unless it was provided by the user with the “-r” option).

- **remove** : bool, optional
  
  Remove all temporary files and intermediate data products. Note: if not using the ‘-o’ option to specify an output file, this will also remove mosaic_image.

- **output_image** : str, optional
  
  Provide your own filename for the output mosaic. Default filename is “mosaic_image.”

- **debug_level** : int, optional
  
  Print out additional debugging information (levels 1-4)

- **region_header** : str, optional
Path to header template used to create mosaic.

**header** : str, optional

Provide header template as text input rather than point to a file; see sample shell script that makes use of this option.

**workspace_dir** : str, optional

Directory where intermediate files will be created. If no workspace is given, a unique local subdirectory will be created (e.g.: ./MOSAIC_AAAaa17v)

### 5.1.18 mFitExec

montage_wrapper.commands.mFitExec(diff_table, fits_table, diff_dir, debug=False, status_file=None, mpi=False, n_proc=8)

Runs mFitplane on all the difference images identified by mOverlaps and generated by mDiff or mDiffExec. mFitExec creates a table of image-to-image difference parameters.

**Parameters**

- **diffs_table** : str
  
  Overlap table generated by mOverlaps, the last column of which contains the filenames of the difference images generated by mDiffExec.

- **fits_table** : str
  
  Output table of difference parameters.

- **diff_dir** : str
  
  Directory containing difference images.

- **debug** : bool, optional
  
  Turns on debugging

- **status_file** : str, optional
  
  Writes output message to status_file instead of to stdout

- **mpi** : bool, optional
  
  If set to True, will use the MPI-enabled versions of the Montage executable.

- **n_proc** : int, optional
  
  If mpi is set to True, n_proc is the number of processes to run simultaneously (default is 8)

### 5.1.19 mFitplane

montage_wrapper.commands.mFitplane(in_image, border=None, debug_level=None, status_file=None)

Uses least squares to fit a plane (excluding outlier pixels) to an image. It is used on the difference images generated using mDiff or mDiffExec.

**Parameters**

- **in_image** : str
  
  Input FITS file is a difference file between two other FITS files, as can be generated using mDiff.

- **border** : int, optional
Number of border pixels to ignore at edges of image.

**debug_level** : int, optional

Turns on debugging to the specified level (1-3).

**status_file** : str, optional

Output and errors are written to status_file instead of stdout.

### 5.1.20 mFixNaN

montage_wrapper.commands.mFixNaN(in_image, out_image, debug_level=None, nan_value=None, min_blank=None, max_blank=None)

Converts NaNs found in the image to some other value (given by the user), or converts a range of supplied values into NaNs.

**Parameters**

- **in_image** : str
  
  Input FITS image file

- **out_image** : str
  
  Path of output FITS file. To run in “count” mode without creating an output file, use a dash (“-”) for this argument.

- **debug_level** : int, optional
  
  Turn on debugging to the specified level (1-3)

- **nan_value** : float, optional
  
  Value to use in place of any NaNs

- **min_blank, max_blank** : str, optional
  
  If the nan_value option is not used, mFixNaN will replace all pixel values between min_blank and max_blank with NaN.

### 5.1.21 mFlattenExec

montage_wrapper.commands.mFlattenExec(images_table, flat_dir, img_dir=None, debug=False, no_area=False, status_file=None)

Runs both mFitPlane and mBackground on a set of images.

**Parameters**

- **images_table** : str
  
  Metadata table (generated by mImgtbl) describing images to be flattened.

- **flat_dir** : str
  
  Path to directory where output files should be created.

- **img_dir** : str, optional
  
  Specifies path to directory containing images to be flattened.

- **debug** : bool, optional
  
  Turns on debugging.

- **no_area** : bool, optional
No-area-images option, indicating that mFlattenExec should not require area images for all the input FITS images.

**status_file** : str, optional
Output and errors are sent to status_file instead of to stdout

### 5.1.22 mGetHdr

```python
montage_wrapper.commands.mGetHdr(in_image, img_header, debug=False, hdu=None, status_file=None)
```
Reads in the header from a FITS image and prints it out to a text file.

**Parameters**

- **in_image** : str
  Path to FITS image from which to retrieve the header.

- **img_header** : str
  Path to text file where FITS header should be written.

- **debug** : bool, optional
  Turns on debugging.

- **hdu** : int, optional
  Retrieve the header from the Fits extention given by hdu. “0” indicates the primary FITS extension, and is the default used by mGetHdr.

- **status_file** : str, optional
  Output and errors are sent to status_file instead of to stdout

### 5.1.23 mHdr

```python
montage_wrapper.commands.mHdr(object_or_location, width, out_file, system=None, equinox=None, height=None, pix_size=None, rotation=None)
```
Connects to the IRSA service HdrTemplate to create a header template based on a location, size, resolution and rotation.

**Parameters**

- **object_or_location** : str
  Object string or coordinate location

- **width** : float
  Width (x-axis) of area

- **out_file** : str
  Path to output header template

- **system** : str, optional
  Specify a coordinate system. Can be one of: “equatorial” or “eq” (default), “ecliptic” or “ec” “galactic”, “ga”, “supergalactic” or “sgal”

- **equinox** : str, optional
  Specify an equinox. Default is 2000.0

- **height** : float, optional
Height (y-axis) of area in degrees. Default is equal to width

**pix_size** : float, optional
Size of a pixel (in arcsec); default is 1

**rotation** : float, optional
Rotation of image; default is 0

### 5.1.24 mHdrCheck

**montage_wrapper.commands.mHdrCheck**(in_image, status_file=None)

mHdrCheck reads in the header from a FITS image (or an ASCII header template file) and checks to see if any header lines are invalid. If it finds one, it will print out a message stating which keyword is invalid and exit before checking the rest of the header. It will not report on multiple invalid values. If all value are correct, mHdrCheck will print out a “Valid FITS/WCS” message.

**Parameters**

- **in_image** : str
  Path of FITS file to be validated.

- **status_file** : str, optional
  Output and errors are sent to status_file instead of to stdout

### 5.1.25 mHdrtbl

**montage_wrapper.commands.mHdrtbl**(directory, images_table, recursive=False, corners=False, debug=False, output_invalid=False, status_file=None, img_list=None)

mHdrtbl operates in a fashion similar to mImgtbl, but is used on a set of header template files instead of FITS images.

**Parameters**

- **directory** : str
  Path to directory containing set of input header templates.

- **images_table** : str
  Path of output metadata table.

- **recursive** : bool, optional
  mHdrtbl can also be used as a standalone program to gather image metadata for other purposes (to populate a database, as a basis for spatial coverage searches, etc.) In this case it is often desirable to collect information on all the files in a directory tree recursively. The recursive option instructs mHdrtbl to search the given directory and all its subdirectories recursively.

- **corners** : bool, optional
  The corners option in mHdrtbl will cause eight extra columns to be added to the output metadata table containing the RA, Dec coordinates (ra1, dec1, ... ra4, dec4) of the image corners. The output is always Equatorial J2000, even if the input is some other system. This has been done to make the metadata uniform so that it can easily be used for coverage searches, etc. The corners option is not needed for normal Montage processing.

- **debug** : bool, optional
Turn on debugging

**output_invalid** : bool, optional

When this option is set, mHdrtbl will explicitly output each header file it finds that does not appear to be valid, along with information on the error.

**status_file** : str, optional

Output and errors are written to status_file instead of being written to stdout.

**img_list** : str, optional

mHdrtbl will only process files with names specified in table img_list, ignoring any other files in the directory.

### 5.1.26 mlImgtbl

**montage_wrapper.commands.mlImgtbl(directory, images_table, recursive=False, corners=False, include_area=False, debug=False, output_invalid=False, status_file=None, fieldlist=None, img_list=None)**

mlImgtbl extracts the FITS header geometry information from a set of files and creates an ASCII image metadata table which is used by several of the other programs. It only collects data from headers that comply with the FITS standard, but reports a count of images that fail that check.

**Parameters**

**directory** : str

Path to directory containing set of input FITS files.

**images_table** : str

Path of output metadata table.

**recursive** : bool, optional

mlImgtbl can also be used as a standalone program to gather image metadata for other purposes (to populate a database, as a basis for spatial coverage searches, etc.) In this case it is often desirable to collect information on all the files in a directory tree recursively. The recursive option instructs mlImgtbl to search the given directory and all its subdirectories recursively.

**corners** : bool, optional

The corners option in mlImgtbl will cause eight extra columns to be added to the output metadata table containing the RA, Dec coordinates \((ra1, dec1, \ldots ra4, dec4)\) of the image corners. The output is always Equatorial J2000, even if the input is some other system. Though not required for the core processing modules, we recommend using this option, as some of the utilities may require corner locations to function properly.

**include_area** : bool, optional

By default, mlImgtbl ignores FITS files with names ending in _area (i.e. name_area_image), assuming them to be Montage-created area images. If you want to generate information on these images, or if you have images with _area in the title other than those generated by Montage, you should turn on this option to force mlImgtbl to look at all images in the directory.

**debug** : bool, optional

Turn on debugging

**output_invalid** : bool, optional
When this option is set, mImgtbl will explicitly output each FITS file it finds that does not appear to be valid, along with information on the error.

**status_file** : str, optional
Output and errors are written to status_file instead of being written to stdout.

**fieldlist** : str, optional
Used to specify a fieldlist, which will list additional keywords to be read from the FITS headers and included in the output table. Fieldlists should specify the keyword name, type (int,char,double), and size.

**img_list** : str, optional
mImgtbl will only process files with names specified in table img_list, ignoring any other files in the directory.

### 5.1.27 mMakeHdr

```
montage_wrapper.commands.mMakeHdr(images_table, template_header, debug_level=None, status_file=None, cdelt=None, north_aligned=False, system=None, equinox=None)
```

From a list of images to be mosaicked together, mMakeHdr generates the FITS header that best describes the output image.

**Parameters**

**images_table** : str
Metadata table (generated by mImgtbl) describing the images to be mosaicked.

**template_header** : str
Path to header template to be generated.

**debug_level** : int, optional
Turns on debugging to the specified level (1-3).

**status_file** : str, optional
Output and errors are written to status_file instead of to stdout.

**cdelt** : float, optional
Specify a pixel scale for the header, if different from the input images

**north_aligned** : bool, optional
“North-aligned” option. By default, the FITS header generated represents the best fit to the images, often resulting in a slight rotation. If you want north to be straight up in your final mosaic, you should use this option.

**system** : str, optional
Specifies the system for the header (default is Equatorial). Possible values are: EQUJ EQUB ECLJ ECLB GAL SGAL

**equinox** : str, optional
If a coordinate system is specified, the equinox can also be given in the form YYYY. Default is J2000.
5.1.28 mOverlaps

`montage_wrapper.commands.mOverlaps(images_table, diffs_table, exact=False, debug_level=None, status_file=None)`

Analyze an image metadata table to determine a list of overlapping images. Each image is compared with every other image to determine all overlapping image pairs. A pair of images are deemed to overlap if any pixel around the perimeter of one image falls within the boundary of the other image.

**Parameters**

- `images_table`: str
  Table of image metadata generated by mImgtbl.

- `diffs_table`: str
  Path of output table to be generated containing overlap information.

- `exact`: bool, optional
  Enables 'exact' overlaps mode, as opposed to the default approximate algorithm. The default mode uses great-circle connecting lines between image corners to determine which images overlap. Exact mode will instead check the edge pixels of every image to determine which pixels are inside the others. Although the default mode will occasionally report some incorrect overlaps, this is not a concern since mDiff will detect and ignore these false positive results when processing the table.

- `debug_level`: int, optional
  Turns on debugging to the specified level (1 or 2)

- `status_file`: str, optional
  Output and errors are sent to status_file instead of to stdout

5.1.29 mPix2Coord

`montage_wrapper.commands.mPix2Coord(template_header, ixpix, jypix, debug=False)`

Takes an image FITS header template and a pixel (x,y) coordinate, and outputs the corresponding sky location.

**Parameters**

- `template_header`: str
  ASCII header template describing the image (either a FITS image, or a JPEG file created from the FITS file)

- `ixpix`: int
  X coordinate (pixel location) on image

- `jypix`: int
  Y coordinate (pixel location) on image

- `debug`: bool, optional
  Print out additional debugging information
5.1.30 mProjExec

```
montage_wrapper.commands.mProjExec(images_table, template_header, proj_dir, stats_table,
raw_dir=None, debug=False, exact=False, whole=False,
border=None, restart_rec=None, status_file=None,
scale_column=None, mpi=False, n_proc=8)
```

An executive which runs mProject (or, if possible for the input/output projections, mProjectPP) for each image in an image metadata table.

**Parameters**

- **images_table**: str
  ASCII table (generated by mImgtbl) containing metadata for all images to be reprojected.

- **template_header**: str
  FITS header template to be used in generation of output FITS.

- **proj_dir**: str
  Directory in which to create reprojected images.

- **stats_table**: str
  Name of table for output statistics (time of each reprojection, or error messages).

- **raw_dir**: str, optional
  Specifies the path to the directory containing the images to be reprojected. If the -p option is not included, mProjExec looks for the images in the current working directory.

- **debug**: bool, optional
  Turns on debugging

- **exact**: bool, optional
  Flag indicating output image should exactly match the FITS header template, and not crop off blank pixels

- **whole**: bool, optional
  Force reprojection of whole images, even if they exceed the area of the FITS header template

- **border**: int, optional
  Ignore border width of pixels around edge of images

- **restart_rec**: str, optional
  Allows restart at record number restart_rec, if mProjExec exits upon an error

- **status_file**: str, optional
  Output and errors are written to status_file instead of being written to stdout.

- **scale_column**: str, optional
  Turn on flux rescaling (e.g. magnitude zero point correction): scale_column is the name of a column in images_table which contains scale information.

- **mpi**: bool, optional
  If set to True, will use the MPI-enabled versions of the Montage executable.

- **n_proc**: int, optional
If mpi is set to True, n_proc is the number of processes to run simultaneously (default is 8)

### 5.1.31 mProject

`montage_wrapper.commands.mProject(in_image, out_image, template_header, factor=None, debug_level=None, status_file=None, hdu=None, scale=None, weight_file=None, threshold=None, whole=False)`

mProject reprojects a single image to the scale defined in a FITS header template file (read more about header templates here). The program produces a pair of images: the reprojected image and an “area” image consisting of the fraction input pixel sky area that went into each output pixel. The “drizzle” algorithm is implemented. The algorithm proceeds by mapping pixel corners (as adjusted by drizzle, if called) from the input pixel space to the output pixel space, calculating overlap area with each output pixel, and accumulating an appropriate fraction of the input flux into the output image pixels. In addition, the appropriate fraction of the input pixel area is accumulated into the area image pixels. Projection of points from input pixel space to output pixel space is calculated in two steps: first map from input pixel space to sky coordinates; second map from sky coordinates to output pixel space.

**Parameters**

- **in_image**: str
  Input FITS file to be reprojected.

- **out_image**: str
  Path of output FITS file to be created.

- **template_header**: str
  FITS header template to be used in generation of output image

- **factor**: float, optional
  Processing is done utilizing the drizzle algorithm. factor is a floating point number; recommended drizzle factors are from 0.5 to 1.

- **debug_level**: int, optional
  Causes additional debugging information to be printed to stdout. Valid levels are 1-5 (for higher debugging levels, it is recommended to redirect the output to a file).

- **status_file**: str, optional
  Output and errors are written to status_file instead of being written to stdout.

- **hdu**: int, optional
  Use the specified FITS extension (default is to use the first HDU with image data)

- **scale**: float, optional
  Apply a correction factor of scale to each pixel

- **weight_file**: str, optional
  Path to a weight map to be used when reading values from the input image.

- **threshold**: float, optional
  Pixels with weights below threshold will be treated as blank.

- **whole**: bool, optional
Makes the output region (originally defined in the header template) big enough to include all of the input images

### 5.1.32 mProjectPP

```python
montage_wrapper.commands.mProjectPP(in_image, out_image, template_header, factor=None, debug_level=None, border=None, status_file=None, alternate_header=None, hdu=None, scale=None, weight_file=None, threshold=None, whole=False)
```

mProjectPP reprojects a single image to the scale defined in an alternate FITS header template generated (usually) by mTANhdr. The program produces a pair of images: the reprojected image and an “area” image consisting of the fraction input pixel sky area that went into each output pixel. This area image goes through all the subsequent processing that the reprojected image does, allowing it to be properly coadded at the end.

#### Parameters

- **in_image**: str
  Input FITS file to be reprojected.

- **out_image**: str
  Path to output FITS file to be created.

- **template_header**: str
  FITS header template to be used in generation of output FITS

- **factor**: float, optional
  Processing is done utilizing the drizzle algorithm. factor is a floating point number; recommended drizzle factors are from 0.5 to 1.

- **debug_level**: int, optional
  Causes additional debugging information to be printed to stdout. Valid levels are 1-5; for levels greater than 1, it’s recommended to redirect the output into a text file.

- **border**: int, optional
  Ignores border pixels around the image edge when performing calculations.

- **status_file**: str, optional
  Output and errors are written to status_file instead of being written to stdout.

- **alternate_header**: str, optional
  Specifies an alternate FITS header for use in mProjectPP calculations, allows substitution of psuedo-TAN headers created by mTANHdr.

- **hdu**: int, optional
  Specify the FITS extension to re-project if the FITS image is multi-extension.

- **scale**: float, optional
  Multiply the pixel values by scale when reprojecting. For instance, each 2MASS image has a different scale factor (very near 1.0) to correct for varying magnitude-zero points.

- **weight_file**: str, optional
  Path to a weight map to be used when reading values from the input image.

- **threshold**: float, optional
If using a weight image; only use those pixels where the weight value is above threshold.

whole : bool, optional
Reproject the whole image even if part of it is outside the region of interest (don’t crop while re-projecting).

5.1.33 mPutHdr

montage_wrapper.commands.mPutHdr(in_image, out_image, template_header, debug=False, status_file=None, hdu=None)
Replaces the header of the input file with one supplied by the user.

Parameters
in_image : str
Input FITS file.
out_image : str
Path to output FITS file (with new header)
template_header : str
ASCII header template to write into out_image.
debug : bool, optional
Turns on debugging to the specified level (this version only supports level “1”).
status_file : str, optional
Output and errors are sent to status_file instead of to stdout
hdu : int, optional
Write to the specified FITS extension (HDU).

5.1.34 mRotate

montage_wrapper.commands.mRotate(in_image, out_image, debug_level=None, status_file=None, rotation_angle=None, ra=None, dec=None, xsize=None, ysize=None)
Rotates a FITS image by an arbitrary angle. This module is meant for quick-look only; it is not flux conserving.

Parameters
in_image : str
Input FITS image.
out_image : str
Path to output (rotated) FITS image.
debug_level : int, optional
Print out additional debugging information (level can be 1-3)
status_file : str, optional
Output and errors are written to status_file instead of stdout.
rotation_angle : float, optional
Provide an angle (in degrees) to rotate the image.
ra, dec, xsize : str, optional
Center location and width (in degrees) of output image - optional. By default, entire
input image area will be included in output image.

ysize : float, optional
Height (in degrees) of output image, if a new center location and width are provided.
Only used if ra, dec, and xsize are specified. Defaults to xsize.

5.1.35 mShrink

montage_wrapper.commands.mShrink(in_image, out_image, factor, fixed_size=False, debug_level=None,
status_file=None)
A utility for reducing the size of a FITS file, by averaging blocks of pixels.

Parameters
in_image : str
Input FITS file
out_image : str
Path to output FITS file.
factor : float
Size of blocks, in pixels, to average. File size will be reduced by 1/factor squared. If the
fixed_size option is used, factor is the desired width of the output image.

fixed_size : bool, optional
Fixed-size option - specify output size of image, instead of the size of blocks of pixels
to be averaged
deploy_level : int, optional
Turns on debugging to the specified level (1-4).
status_file : str, optional
Output and errors are sent to status_file instead of to stdout

5.1.36 mSubimage

montage_wrapper.commands.mSubimage(in_image, out_image, ra, dec, xsize, debug=False, all_pixels=False, hdu=None, status_file=None, status_file=None)
Creates a subimage (or “cutout”) of a FITS file. To use mSubimage in ‘pixel’ mode, see mSubimage_pix

Parameters
in_image : str
Input FITS file.
out_image : str
Path to output FITS file.
ra : float
RA of center of output image.
dec : float
Declination of center of output image.

\texttt{xsize} : float
Width of output image in degrees.

\texttt{debug} : bool, optional
Turns on debugging.

\texttt{all\_pixels} : bool, optional
All pixels - Force retrieval of whole image (useful to extract an entire HDU)

\texttt{hdu} : int, optional
Operate on the specified FITS header extension (HDU)

\texttt{status\_file} : str, optional
Output and errors are sent to status_file instead of to stdout

\texttt{ysize} : float, optional
Height of output image in degrees (default is equal to xsize).

\subsection{5.1.37 \texttt{mSubimage\_pix}}

\texttt{montage\_wrapper.commands.mSubimage\_pix(in\_image, out\_image, xstartpix, ystartpix, xpixsize, debug=False, hdu=None, status\_file=None, ypixsize=None)}

Creates a subimage (or “cutout”) of a FITS file (‘pixel’ mode)

\textbf{Parameters}

\texttt{in\_image} : str
Input FITS file.

\texttt{out\_image} : str
Path to output FITS file.

\texttt{xstartpix} : int
Pixel along the x-axis where the cutout image will begin

\texttt{ystartpix} : int
Pixel along the y-axis where the cutout image will begin

\texttt{xpixsize} : int
Width of output image in pixels

\texttt{debug} : bool, optional
Turns on debugging.

\texttt{hdu} : int, optional
Operate on the specified FITS header extension (HDU)

\texttt{status\_file} : str, optional
Output and errors are sent to status_file instead of to stdout

\texttt{ypixsize} : int, optional
Height of output image in pixels (default is equal to xpixsize)
5.1.38 mSubset

`montage_wrapper.commands.mSubset(images_table, template_header, subset_table, debug_level=None, fast_mode=False, status_file=None)`

Generates a table of images that is a subset of the input table, containing only those images that cover the area defined by a given FITS header.

**Parameters**
- **images_table**: str
  ASCII table (generated by mImgtbl) containing metadata for image collection.
- **template_header**: str
  FITS header template defining the area of interest.
- **subset_table**: str
  Path to output table, which will contain only those FITS images covering the area defined by template_header.
- **debug_level**: int, optional
  Turns on debugging to the specified level (1-3).
- **fast_mode**: bool, optional
  Fast mode - input file must include corners (corners option in mImgtbl) to utilize this mode.
- **status_file**: str, optional
  Output and errors are sent to status_file instead of to stdout.

5.1.39 mTANHdr

`montage_wrapper.commands.mTANHdr(orig_header, new_header, debug=False, order=None, max_iter=None, tolerance=None, status_file=None)`

Analyzes a template file and determines if there would be an adequate equivalent distorted TAN projection, within a specified tolerance, and outputs the alternate header. This header can be used in conjunction with mProjectPP to produce a TAN plane image. This process is considerably faster than projecting with the general purpose tool mProject.

**Parameters**
- **orig_header**: str
  Input FITS header
- **new_header**: str
  Path to output header to be created
- **debug**: bool, optional
  Print additional debugging information to stdout.
- **order**: str, optional
  Order of output header polynomial focal plane distortions (default = 4)
- **max_iter**: int, optional
  Maximum number of iteration attempts to produce header (default = 50)
- **tolerance**: float, optional
  Output and errors are sent to status_file instead of to stdout.
Distortion tolerance value for acceptable output (default = 0.01)

status_file : str, optional
Output and errors are written to status_file instead of stdout.

5.1.40 mTblSort

montage_wrapper.commands.mTblSort(in_table, column_name, out_table, debug=False)
Sorts a table on numeric values.

Parameters
in_table : str
Path to input table
column_name : str
Name of column to sort on (column must contain numeric values)
out_table : str
Path to output table
debug : bool, optional
Turns on debugging

5.1.41 mTileHdr

montage_wrapper.commands.mTileHdr(orig_header, new_header, n_x, n_y, ix, iy, debug=False, status_file=None, xpad=None, ypad=None)
Takes a header template file and creates another which represents one of a regular set of tiles covering the original. The user specifies the tile gridding and which tile is desired.

Parameters
orig_header : str
ASCII header template from which to derive tiled headers
new_header : str
Path to output header
n_x : int
Number of tiles in the x-direction
n_y : int
Number of tiles in the y-direction
ix : int
Integer identifying the x location of the output tile on the grid (counting from 0)
iy : int
Integer identifying the y location of the output tile on the grid (counting from 0)
debug : bool, optional
Turns on debugging.
status_file : str, optional
Output and errors are sent to status_file instead of to stdout

**xpad** : int, optional

Number of pixels to overlap tiles in the x direction (default is 0)

**ypad** : int, optional

Number of pixels to overlap tiles in the y direction (default is 0). Only used if xpad is present.

### 5.1.42 mTileImage

**mTileImage**

#### montage_wrapper.commands.mTileImage(in_image, tiles_x, tiles_y, overlap_x=None, overlap_y=None)

mTileImage splits up an image into a given number of tiles. An overlap between each tile can optionally be specified.

**Parameters**

- **in_image** : str
  
  Input FITS file

- **tiles_x** : int
  
  Number of tiles along x axes.

- **tiles_y** : int
  
  Number of tiles along y axes.

- **overlap_x** : int, optional
  
  Pixel overlap in x direction.

- **overlap_y** : int, optional
  
  Pixel overlap in y direction.
6.1 Functions

6.1.1 parse_struct

```python
montage_wrapper.status.parse_struct(command, string)
```

6.1.2 simplify

```python
montage_wrapper.status.simplify(value)
```

6.2 Classes

6.2.1 MontageError

```python
exception montage_wrapper.status.MontageError
```

6.2.2 Struct

```python
class montage_wrapper.status.Struct(command, string)
    Bases: object
```
6.3 Class Inheritance Diagram

![Class Inheritance Diagram]

- Struct
- MontageError
m

montage_wrapper.commands, 29
montage_wrapper.status, 57
montage_wrapper.wrappers, 23
M
mAdd() (in module montage_wrapper.commands), 30
mAddExec() (in module montage_wrapper.commands), 31
mArchiveExec() (in module montage_wrapper.commands), 32
mArchiveGet() (in module montage_wrapper.commands), 32
mArchiveList() (in module montage_wrapper.commands), 32
mBackground() (in module montage_wrapper.commands), 33
mBackground_tab() (in module montage_wrapper.commands), 33
mBestImage() (in module montage_wrapper.commands), 34
mBgExec() (in module montage_wrapper.commands), 34
mBgModel() (in module montage_wrapper.commands), 35
mCatMap() (in module montage_wrapper.commands), 36
mConvert() (in module montage_wrapper.commands), 36
mCoverageCheck() (in module montage_wrapper.commands), 37
mDiff() (in module montage_wrapper.commands), 38
mDiffExec() (in module montage_wrapper.commands), 38
mDiffFitExec() (in module montage_wrapper.commands), 39
mExec() (in module montage_wrapper.commands), 39
mFitExec() (in module montage_wrapper.commands), 40
mFitplane() (in module montage_wrapper.commands), 41
mFixNaN() (in module montage_wrapper.commands), 42
mFlattenExec() (in module montage_wrapper.commands), 42
mGetHdr() (in module montage_wrapper.commands), 43
mHdr() (in module montage_wrapper.commands), 43
mHdrCheck() (in module montage_wrapper.commands), 44
mHdrtbl() (in module montage_wrapper.commands), 44
mImgtbl() (in module montage_wrapper.commands), 45
mMakeHdr() (in module montage_wrapper.commands), 46
montage_wrapper.commands (module), 29
montage_wrapper.commands (module), 29
montage_wrapper.status (module), 57
montage_wrapper.wrappers (module), 23
MontageError, 57
mosaic() (in module montage_wrapper.wrappers), 23
mOverlaps() (in module montage_wrapper.commands), 47
mPix2Coord() (in module montage_wrapper.commands), 47
mProject() (in module montage_wrapper.commands), 49
mProject_auto() (in module montage_wrapper.wrappers), 23
mProjectPP() (in module montage_wrapper.commands), 50
mProjExec() (in module montage_wrapper.commands), 48
mPutHdr() (in module montage_wrapper.commands), 51
mRotate() (in module montage_wrapper.commands), 51
mShrink() (in module montage_wrapper.commands), 52
mSubimage() (in module montage_wrapper.commands), 52
mSubimage_pix() (in module montage_wrapper.commands), 53
mSubset() (in module montage_wrapper.commands), 54
mTANHdr() (in module montage_wrapper.commands), 54
mTileHdr() (in module montage_wrapper.commands), 55
mTileImage() (in module montage_wrapper.commands), 56
mTTileSort() (in module montage_wrapper.commands), 55
parse_struct() (in module montage_wrapper.status), 57
reproject() (in module montage_wrapper.wrappers), 25
reproject_cube() (in module montage_wrapper.wrappers), 26
reproject_hdu() (in module montage_wrapper.wrappers), 27

S

simplify() (in module montage_wrapper.status), 57
Struct (class in montage_wrapper.status), 57