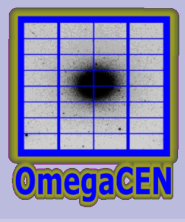
Image Stacking in Astro-WISE

- Why this talk?
  - Infrared data: lots of short exposures to stack
- Stacking with Eclipse
- Stacking with SWarp
- Future: stacking with DARMA

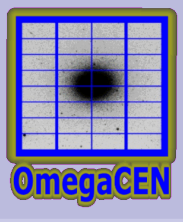
Eclipse

- Interface to C-code from Python
- Fast
- Not scalable for taking median of cube
- Images must be of equal dimensions
- No longer developed
- Weighting not incorporated in C-code



What do we use Eclipse for?

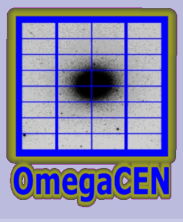
- Bias (average with sigma-rejection)
  - median() is used as first approx. of average
- Flat-fields
  - Master dome and twilight (average with sigma-rejection)
  - Nightsky flats (median)
- Fringe maps (median)



Eclipse: average of cube

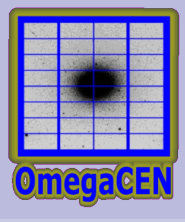
```
awe> filenames = ['image1.fits', 'image2.fits', 'image3.fits']
awe> cube = eclipse.cube.cube([eclipse.image.image(f) for f in
filenames])
awe> average = cube.average()

# do-it-yourself, scalable
awe> im1 = eclipse.image.image('image1.fits')
awe> for f in filenames[1:]:
            im1 += eclipse.image.image(f)
awe> im1 /= len(filenames)
awe> im1.save('average.fits')
```



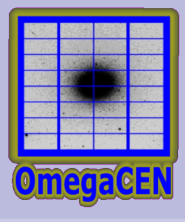
Eclipse: median of cube

```
# Calculate the median of a cube of images
awe> cube = eclipse.cube.cube([eclipse.image.image(filename) for
filename in filenames])
awe> med = cube.median()
awe> med.save('median.fits')
```



Eclipse: image difference

```
awe> im1 = eclipse.image.image('image1.fits')
awe> im2 = eclipse.image.image('image2.fits')
awe> im3 = im1-im2
awe> im3.save('diff.fits')
```



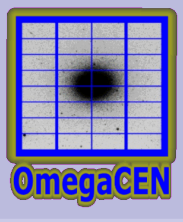
Abstraction in object model

```
awe> bias = (BiasFrame.filename != '').max('creation_date')
awe> bias.load_image()
awe> bias.image
<eclipse.image.image object at 0xb7bf1fac>


awe> sci.image = sci.image - bias.image
```

SWarp

- Fast (slower than Eclipse (bagage))
  - Meant to use world coordinates
- Coadd with or without resampling
  - Integer pixel shifts by changing CRPIX1, CRPIX2 etc.
- Pads output image with zeros



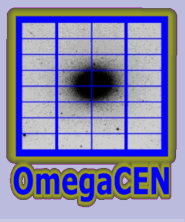
What do we use SWarp for?

- Resampling images to the same grid
- Coadd images
- Can be (ab)used to stack images to create fringe map

SWarp example

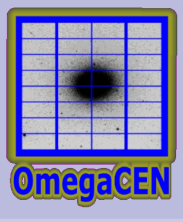
```
awe> from astro.external import  Swarp
awe> from astro.main.Config import SwarpConfig
awe> config = SwarpConfig()
awe> config.WEIGHT_TYPE = 'NONE'
awe> config.BACK_SIZE = 64

awe> Swarp.swarp(['image1.fits', 'image2.fits'], config=config)
```



SWarp abstraction in object model

- RegriddedFrame
  - result of swarp with COMBINE=”N” and RESAMPLE = “Y”
- CoaddedRegriddedFrame
  - result of swarp with COMBINE=”Y” and RESAMPLE = “N”



SWarp: in pipeline

using the DPU

awe> dpu.run('Regrid', d='2000-01-01', i='WFI', f='#843', o='Science1_?-*')

local processing

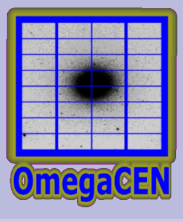
awe> task = RegridTask(date='2000-01-01', chip='ccd50', filter='#843', object='Science1_?-*')

awe> task.execute()

SWarp to make FringeFrame

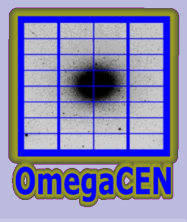
COMBINE = Y
COMBINE_TYPE = MEDIAN
RESAMPLE = N

> swarp -c swarp.conf <flat-fielded de-biased images>



DARMA

- PyFITS interface, much like Eclipse interface
  - Uses NumPy
- Not fast enough for some number crunching operations:
  - Convolution, Hough-transform (satellite detection)
  - NumPy extension (C-code) to speed up



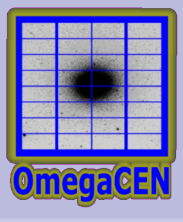
DARMA examples

```
awe> filenames = ['image1.fits', 'image2.fits', 'image3.fits']

awe> cube = darma.cube.cube([darma.image.image(f) for f in filenames])

# Calculate the average
awe> average = cube.average()

# Calculate the median
awe> median = cube.median()
```



Performance overview

- hardware: P4 2.8 Ghz, 1GB RAM
- data: ISAAC Hawaii CCD images (4.1Mb)

STACKING	ECLIPSE	SWARP	DARMA
MEDIAN #frames Computing time Comments	 199 48 sec limited by RAM	 199 58 sec	 199 93 sec
AVERAGE #frames Computing time	 500 60 sec	 500 186 sec	 500 65 sec