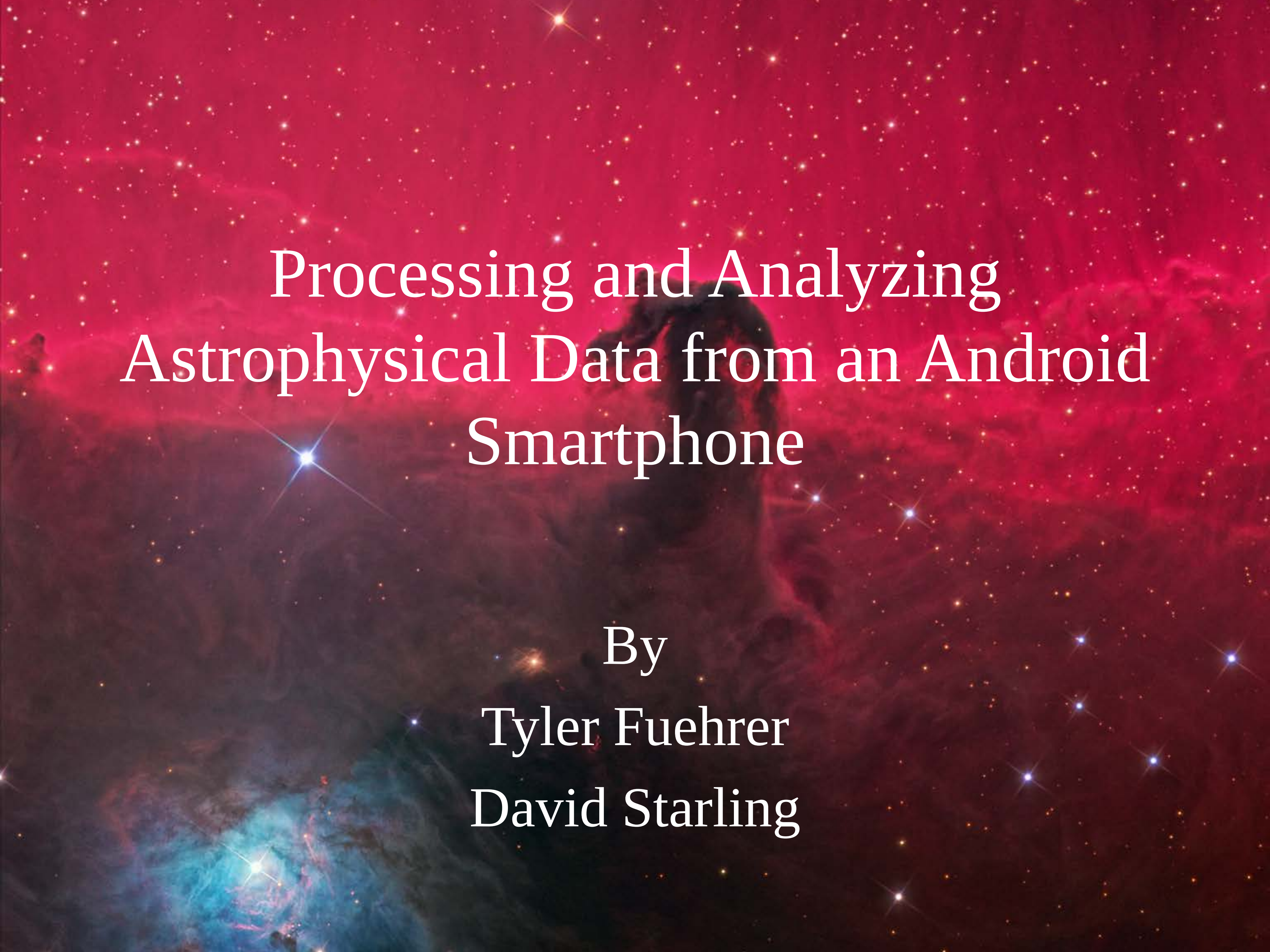


Processing and Analyzing Astrophysical Data from an Android Smartphone

By

Tyler Fuehrer

David Starling

Abstract

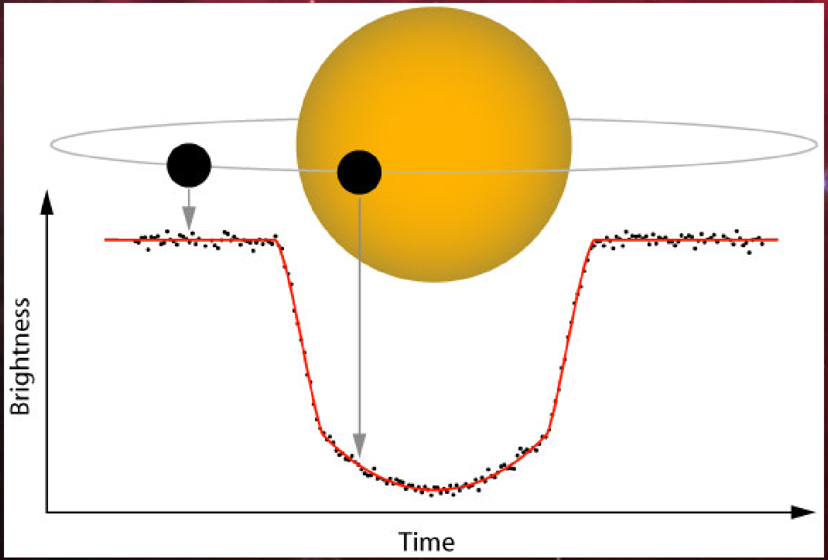
Computer programs, much like the one developed here, are constantly being created by astrophysicist in order to process data. The specialized nature of their research requires such custom programs. In particular, astrophysical data requires many hidden layers of processing before it is possible to analyze. Here, we developed code in the Python programing language that will process images from a custom Android smartphone app (developed by Dr. Joseph Ranalli) that was used to take photographs of celestial bodies. The android app takes a quick burst of photographs, waits a set amount of time, and then repeats this process. The Python program takes this raw data and compiles it into a single file as discussed below.

Our program first enumerates the number of files it will need to process using the file type of our photos. Using a specific naming standard for the images the program is able to search through the file names to find the amount of images in a single burst. We use this information to stack each burst of images into a single composite image. This process repeats on each set of bursts. Due to the time between each burst of images, the rotation of the Earth has caused a rotation between the images. Therefore, we rotate the composite images into the same orientation. We take these rotated images and store them in a data cube or stack them together, as needed. The program has been tested successfully on simulated images and is ready for testing with field data. We will present portions of the code along with the resulting data.

Motivation

- The purpose of this work is to process data obtained through astrophotography in a manner consistent with modern astrophysics research.

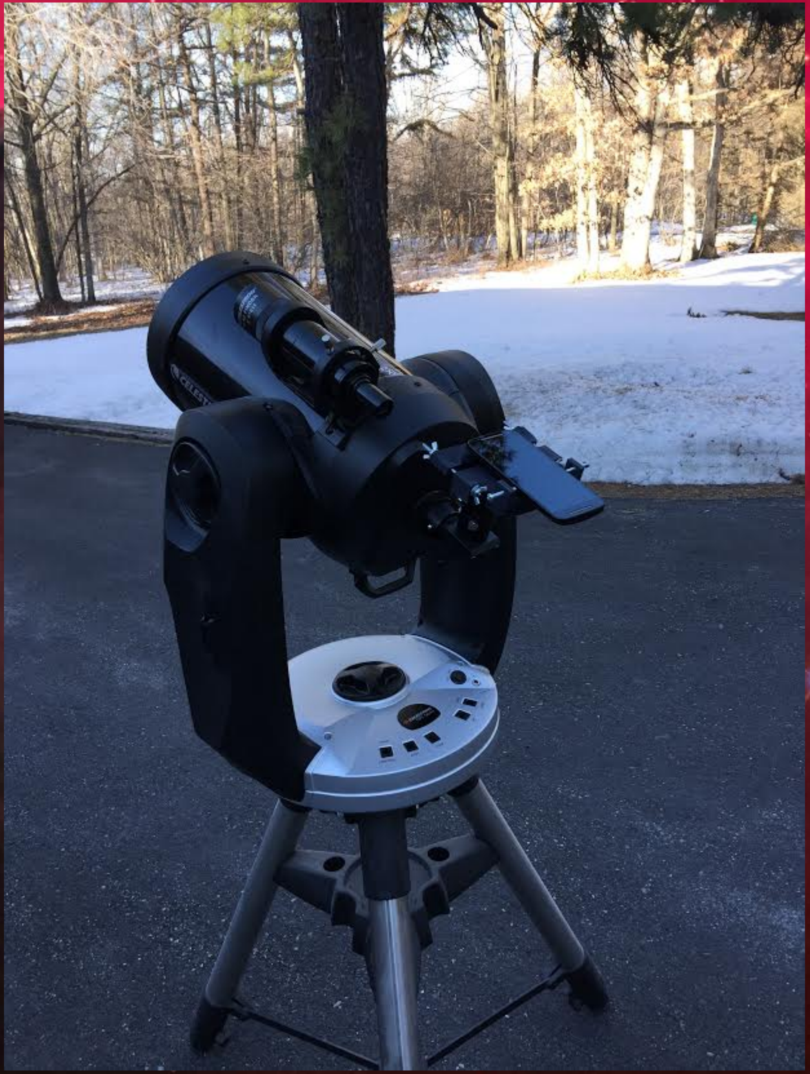
- The processed data can be used to:
  - track the movement of nearby celestial bodies (asteroids, space junk, etc.)
  - monitor the brightness of distant stars to detect exoplanets



(source: github.com)

Data Acquisition

- To acquire data we attach an android smartphone to a telescope using a custom built mount.
- The android smartphone uses an application developed by Dr. Joseph Ranalli.

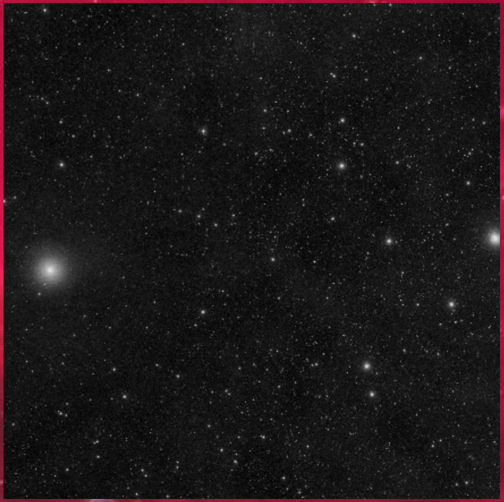


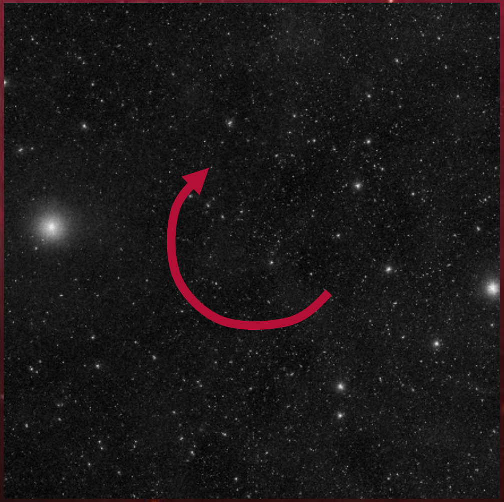
Rotation

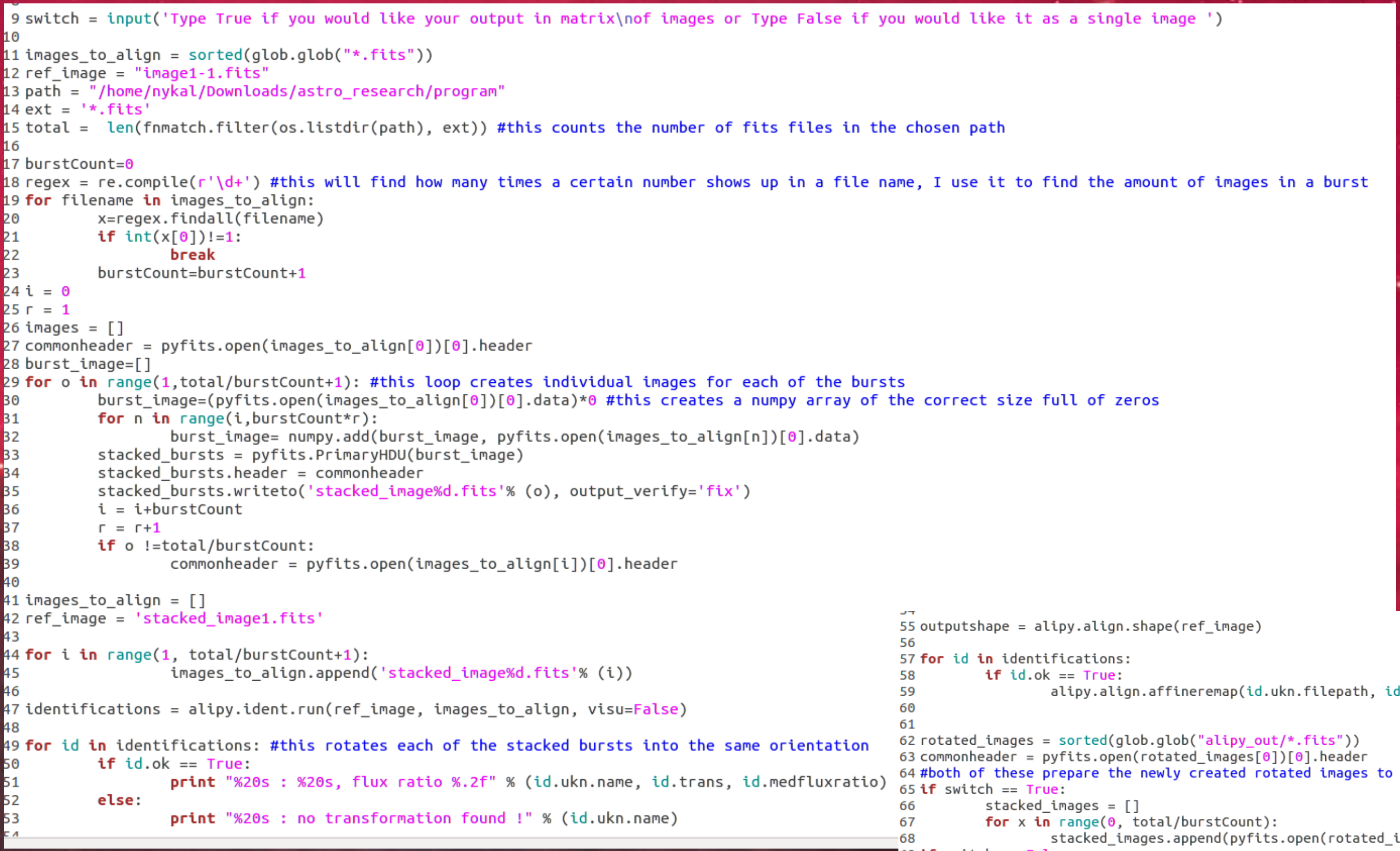
- When processing the data, we must take into consideration the circular movement of the sky throughout the night caused by the rotation of the Earth.

- The instrument we use to take data is a motorized alt-azimuth telescope. This will track a single object in the sky.

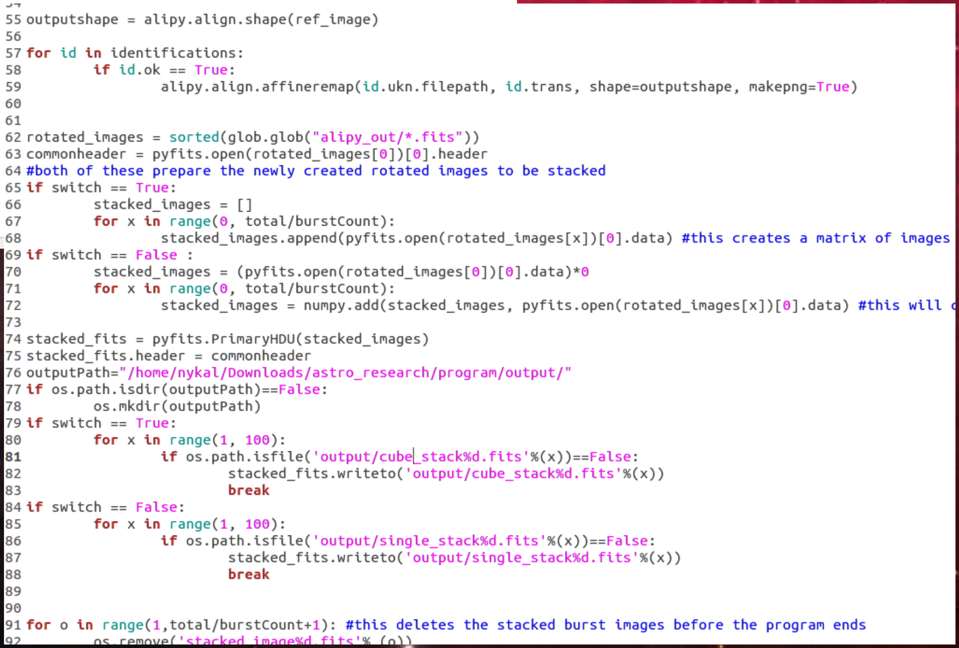
- As the telescope tracks this single object, the view will *still* appear to rotate (example: track the north star Polaris).







```python
switch = input('Type True if you would like your output in matrix\nof images or Type False if you would like it as a single image ')

images_to_align = sorted(glob.glob("*.fits"))
ref_image = "image1-1.fits"
path = "/home/nykal/Downloads/astro_research/program"
ext = '*.fits'
total =  len(fnmatch.filter(os.listdir(path), ext)) #this counts the number of fits files in the chosen path

burstCount=0
regex = re.compile(r'\d+') #this will find how many times a certain number shows up in a file name, I use it to find the amount of images in a burst
for filename in images_to_align:
        x=regex.findall(filename)
        if int(x[0])!=1:
                break
        burstCount=burstCount+1
i = 0
r = 1
images = []
commonheader = pyfits.open(images_to_align[0])[0].header
burst_image=[]
for o in range(1,total/burstCount+1): #this loop creates individual images for each of the bursts
        burst_image=(pyfits.open(images_to_align[0])[0].data)*0 #this creates a numpy array of the correct size full of zeros
        for n in range(i,burstCount*r):
                burst_image= numpy.add(burst_image, pyfits.open(images_to_align[n])[0].data)
        stacked_bursts = pyfits.PrimaryHDU(burst_image)
        stacked_bursts.header = commonheader
        stacked_bursts.writeto('stacked_image%d.fits'% (o), output_verify='fix')
        i = i+burstCount
        r = r+1
        if o !=total/burstCount:
                commonheader = pyfits.open(images_to_align[i])[0].header

images_to_align = []
ref_image = 'stacked_image1.fits'

for i in range(1, total/burstCount+1):
                images_to_align.append('stacked_image%d.fits'% (i))

identifications = alipy.ident.run(ref_image, images_to_align, visu=False)

for id in identifications: #this rotates each of the stacked bursts into the same orientation
        if id.ok == True:
                print "%20s : %20s, flux ratio %.2f" % (id.ukn.name, id.trans, id.medfluxratio)
        else:
                print "%20s : no transformation found !" % (id.ukn.name)
```

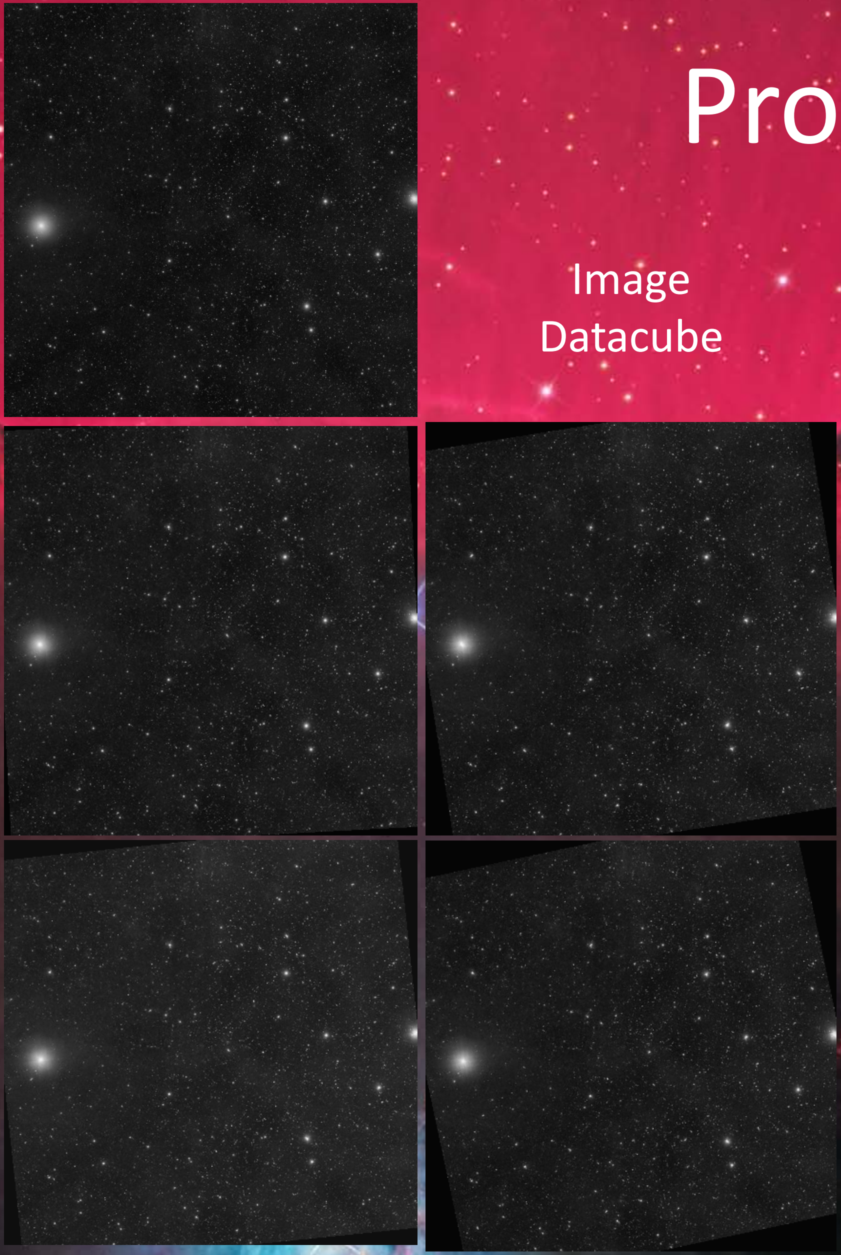


```python
outputshape = alipy.align.shape(ref_image)

for id in identifications:
        if id.ok == True:
                alipy.align.affineremap(id.ukn.filepath, id.trans, shape=outputshape, makepng=True)


rotated_images = sorted(glob.glob("alipy_out/*.fits"))
commonheader = pyfits.open(rotated_images[0])[0].header
#both of these prepare the newly created rotated images to be stacked
if switch == True:
        stacked_images = []
        for x in range(0, total/burstCount):
                stacked_images.append(pyfits.open(rotated_images[x])[0].data) #this creates a matrix of images
if switch == False :
        stacked_images = (pyfits.open(rotated_images[0])[0].data)*0
        for x in range(0, total/burstCount):
                stacked_images = numpy.add(stacked_images, pyfits.open(rotated_images[x])[0].data) #this will

stacked_fits = pyfits.PrimaryHDU(stacked_images)
stacked_fits.header = commonheader
outputPath="/home/nykal/Downloads/astro_research/program/output/"
if os.path.isdir(outputPath)==False:
        os.mkdir(outputPath)
if switch == True:
        for x in range(1, 100):
                if os.path.isfile('output/cube_stack%d.fits'%(x))==False:
                        stacked_fits.writeto('output/cube_stack%d.fits'%(x))
                        break
if switch == False:
        for x in range(1, 100):
                if os.path.isfile('output/single_stack%d.fits'%(x))==False:
                        stacked_fits.writeto('output/single_stack%d.fits'%(x))
                        break


for o in range(1,total/burstCount+1): #this deletes the stacked burst images before the program ends
        os.remove('stacked_image%d.fits'% (o))
```

The Code

Processed Data

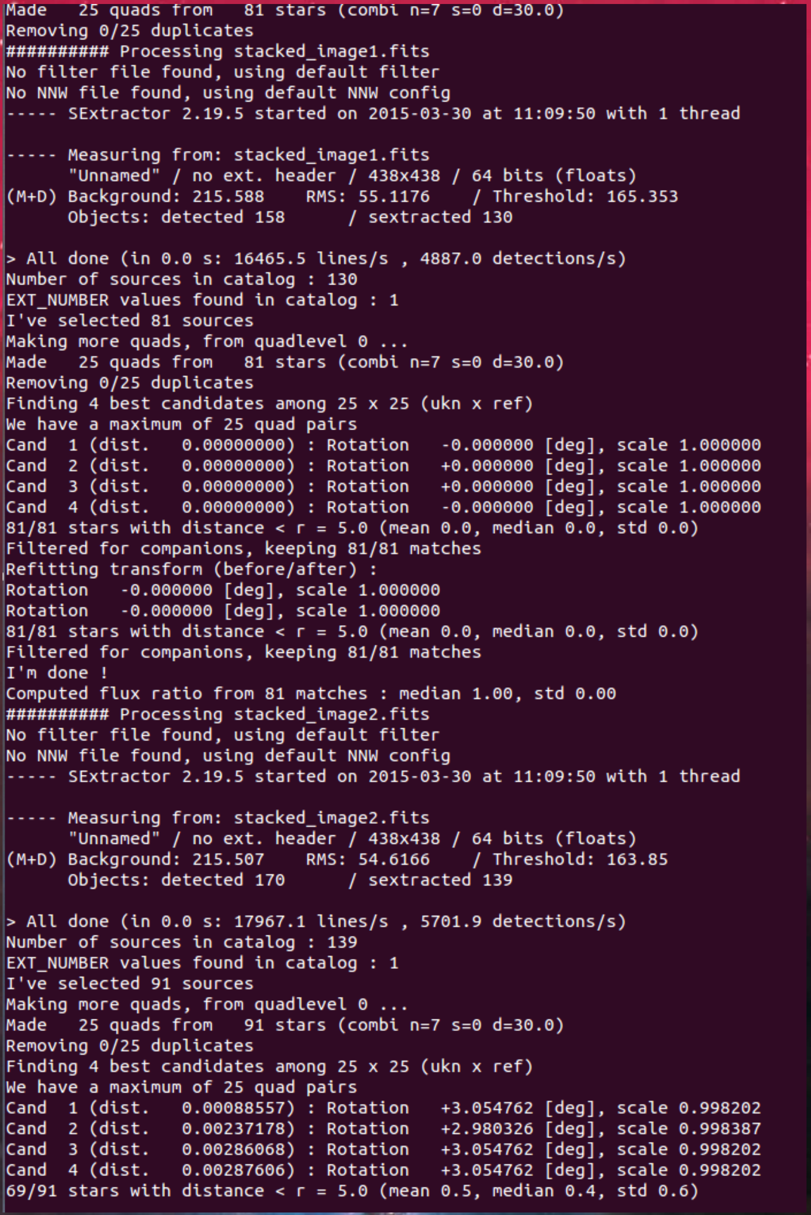
Image Datacube



Single stacked image

(files in standard astronomy .fits format)

The Future



```
Made    25 quads from    81 stars (combi n=7 s=0 d=30.0)
Removing 0/25 duplicates
########## Processing stacked_image1.fits
No filter file found, using default filter
No NNW file found, using default NNW config
----- SExtractor 2.19.5 started on 2015-03-30 at 11:09:50 with 1 thread

----- Measuring from: stacked_image1.fits
      "Unnamed" / no ext. header / 438x438 / 64 bits (floats)
(M+D) Background: 215.588     RMS: 55.1176     / Threshold: 165.353
      Objects: detected 158       / sextracted 130

> All done (in 0.0 s: 16465.5 lines/s , 4887.0 detections/s)
Number of sources in catalog : 130
EXT_NUMBER values found in catalog : 1
I've selected 81 sources
Making more quads, from quadlevel 0 ...
Made    25 quads from    81 stars (combi n=7 s=0 d=30.0)
Removing 0/25 duplicates
Finding 4 best candidates among 25 x 25 (ukn x ref)
We have a maximum of 25 quad pairs
Cand   1 (dist.    0.00000000) : Rotation    -0.000000 [deg], scale 1.000000
Cand   2 (dist.    0.00000000) : Rotation    +0.000000 [deg], scale 1.000000
Cand   3 (dist.    0.00000000) : Rotation    +0.000000 [deg], scale 1.000000
Cand   4 (dist.    0.00000000) : Rotation    -0.000000 [deg], scale 1.000000
81/81 stars with distance < r = 5.0 (mean 0.0, median 0.0, std 0.0)
Filtered for companions, keeping 81/81 matches
Refitting transform (before/after) :
Rotation    -0.000000 [deg], scale 1.000000
Rotation    -0.000000 [deg], scale 1.000000
81/81 stars with distance < r = 5.0 (mean 0.0, median 0.0, std 0.0)
Filtered for companions, keeping 81/81 matches
I'm done !
Computed flux ratio from 81 matches : median 1.00, std 0.00
########## Processing stacked_image2.fits
No filter file found, using default filter
No NNW file found, using default NNW config
----- SExtractor 2.19.5 started on 2015-03-30 at 11:09:50 with 1 thread

----- Measuring from: stacked_image2.fits
      "Unnamed" / no ext. header / 438x438 / 64 bits (floats)
(M+D) Background: 215.507     RMS: 54.6166     / Threshold: 163.85
      Objects: detected 170       / sextracted 139

> All done (in 0.0 s: 17967.1 lines/s , 5701.9 detections/s)
Number of sources in catalog : 139
EXT_NUMBER values found in catalog : 1
I've selected 91 sources
Making more quads, from quadlevel 0 ...
Made    25 quads from    91 stars (combi n=7 s=0 d=30.0)
Removing 0/25 duplicates
Finding 4 best candidates among 25 x 25 (ukn x ref)
We have a maximum of 25 quad pairs
Cand   1 (dist.    0.00088557) : Rotation    +3.054762 [deg], scale 0.998202
Cand   2 (dist.    0.00237178) : Rotation    +2.980326 [deg], scale 0.998387
Cand   3 (dist.    0.00286068) : Rotation    +3.054762 [deg], scale 0.998202
Cand   4 (dist.    0.00287606) : Rotation    +3.054762 [deg], scale 0.998202
69/91 stars with distance < r = 5.0 (mean 0.5, median 0.4, std 0.6)
```

- The newly processed data that the current program outputs is now ready to be analyzed
- The next step will be to write a program that
  - identifies movement of celestial bodies such as asteroids and comets.
  - calculates the change in brightness of objects using differential photometry
- The ultimate goal will be to track the motion of near-Earth objects and to identify exoplanets and their orbital periods.

References

[1] Zelle, John. (2010), Python Programming: An Introduction to Computer Science. Franklin, Beedle & Associates, Inc.; 2nd edition

[2] Alipy. (2013), Retrieved September 2014, http://obswww.unige.ch/~tewes/alipy/

[3] Kay, L., Smith, B. and Blumenthal. (2013), 21st Century Astronomy. W. W. Norton and Company, Inc.; 4th edition.