7/1/2019 Optimal Aperture Curve

#Program finds the position of the max photon count per column, then applies a fit to the positions to create a curve for an optimal aperture.

# import packages to use

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.pyplot import \*

from astropy.utils.data import get\_pkg\_data\_filename

from astropy.table import Table

from astropy.io import fits

import glob

import batman

import lmfit

import corner

import matplotlib.pyplot as plt

from matplotlib import pylab

from numpy import arange,array,ones

from scipy import stats

# directory='/Users/annaburkholder/exp\_det\_scripts/visit23\_defringed/'

# directory='/home/ian/Desktop/WebbData/visit23\_defringed/'

directory='/visit23\_defringed/' #Change directory to proper location

number\_of\_rows=64

number\_of\_columns=1024

number\_of\_images=43

#Load images into a list

list=glob.glob(directory+"\*.fits")

#print, first image in list

print(list[0])

##Example load first fits image

hdul=fits.open(list[1])

#Get MJD mid time of exposure from Header, which has start and end MJD times

mjd\_start=hdul[0].header['EXPSTART']

mjd\_end=hdul[0].header['EXPEND']

mjd=(mjd\_end+mjd\_start)/2.

print(mjd)

#load fits file image into an array called 'data'

data = hdul[0].data

data.shape #size of image

data.dtype.name #type of image

print(np.sum(data)) #total counts in image

#close fits after loading in data needed

hdul.close()

index\_of\_images=np.arange(number\_of\_images)

index\_of\_rows=np.arange(number\_of\_rows)

index\_of\_columns=np.arange(number\_of\_columns)

total\_counts=np.zeros((number\_of\_images))

row\_sum=np.zeros(number\_of\_rows) #sums pixel count for each row of image

maxRow=np.zeros(number\_of\_columns) #number of row in which max count exists in cols

col\_vals=np.zeros(number\_of\_columns) #

col\_max=np.zeros(number\_of\_columns) #to contain max photon count per column in an image

col\_posn\_max=np.zeros(number\_of\_columns) #to contain position of max photon count per column in an image

col\_max\_posn=np.zeros((number\_of\_images, number\_of\_columns)) #contains all column pc among all images

col\_max\_value=np.zeros((number\_of\_images, number\_of\_columns)) #contains all position pc among all images

for i in index\_of\_images:

img=list[i]

#print(img)

hdul=fits.open(img)

data = hdul[0].data

#print(np.sum(data)) #total counts in image

for j in index\_of\_columns:

#total\_counts[i]=np.sum(data[0:64,j]) #total counts in column j of image

col\_posn\_max[j]=np.argmax(data[0:64,j])

col\_max[j]=max(data[0:64,j])

#print(total\_counts)

# print(row\_vals) #prints array for each image w/ max photon count per column

#col\_max\_posn[i]=total\_counts

col\_max\_value[i]=col\_max

print(col\_max\_value)

col\_max\_posn[i]=col\_posn\_max

print(col\_max\_posn)

all\_linear\_lines = np.zeros((number\_of\_images, number\_of\_columns)) #collect all fit lines for every image

for i in index\_of\_images:

y=col\_max\_posn[i]

x=index\_of\_columns

slope, intercept, r\_value, p\_value, std\_err = stats.linregress(x,y)

line = slope\*x+intercept

all\_linear\_lines[i]=line

plt.plot(x,y)

pylab.title('Linear Fit of Image 1')

print(all\_linear\_lines)

#Generate a 10th degree polynomial fit

z = np.polyfit(x, y, 10)

f = np.poly1d(z)

x\_new = np.linspace(x[0], x[-1], 50)

y\_new = f(x\_new)

plt.plot(x,y,'o', x\_new, y\_new)

plt.xlim([x[0]-1, x[-1] + 1 ])

plt.show()