JWST - NIRCam - Level 3 TSO Pipeline

This pipeline was commissioned to take input from the Level 2 pipeline -- having been processed through the NIRCam `ncdhas` pipeline -- and now being further processed through from a stack of images into a time series

NEW METHOD

1. Develop a single routine that inputs
   A. String (fits file name) or array (loaded fits file)
   B. The expected location of the star (center of frame is default)
   C. Subframe size (for better center fitting)
   D. List of aperture radii (or a float for a single aperture radii)
2. This routine will load a single fits file or list of fits files (one at a time; recursive?)
3. For each single, or recursively for a list of fits files,
   A. load the data.
   B. Computer the time element
   C. subtract the background (store background level)
   D. isolate the star into a subframe
   E. Cross-correlate a Gaussian (or JWST psf) with the image to find predicted center (store CC center)
   F. Gaussian fit to subframe, starting at CC center (store GS center, width, amplitude)
   G. Perform apeture photometry with each radius given at the beginning (store aperture radii as a function of radius)

This routine ensures that the user can manipulate the inputs as needed. Users can either send a single fits array, a set of fits array, a single string with the location of a fits file, or a list of strings with the location of several fits files.

The result will be a 'DataFrame' of the same depth as the input structure, containing (labeled as keys) the

- 'sky background'
- 'cross correlation center'
- 'gaussian center'
- 'gaussian width'
- 'gaussian ampitude'
- 'aperture photometry dictionary' or 'aperture photometry dataframe'
  - the keys to the aperture photometry dictionary or data frame will be the float values of the aperture radii
- 'time' (in days?)

OLD METHOD - for Posterity and External Comparison

1. Input data from file directory from user
2. Access that file directory and grab all file names -- possible include a data file
3. Sequentially open all fits file in that directory (or from the data file)
4. During the opening process, store the data frame(s) necessary for production of time series
5. Remove the original data from RAM (too much space)
6. Subtract median background
7. Cross-Correlated Gaussian with center of image
8. Fit a Gaussian to center of image, starting from Cross-Correlation solution
9. Integrate (using 'exact') the aperture photometry
10. Store aperture photometry, gaussian centers, cross-correlation centers, gaussian widths, gaussian heights

Load All Necessary Libraries and Functions

```
`pylab`       : combination of array manipulation and plotting functions
`matplotlib`  : specialized plotting functions
`numpy`       : array more manipulation functions
`pandas`      : dataframe -- more advanced array / table -- functions
`photutils`   : astropy associated package for aperture photometry
`astroML`     : better histogram function for plotting
`astropy`     : `modeling` : access linear and gaussian functions with astropy formatting
                `fitting`  : access to astropy fitting routines
`jd`          : julian date from header info calculations
`julian_date`: julian data from header info calculations written by Ian Crossfield (IJC)
`datatime`    : assists `jd` with calculating header time
`os`          : Operating System level control for python
`glob`        : grab list of files in directory
`sklearn`     : `externals`: imports operating system (storage) level function (i.e. joblib)
`statsmodels`: `robust`   : robust statistical modeling packages; `scale.mad` == median average
 distance
`sys`         : python-os level functions (i.e. path)
`time`        : compute and convert current timestamps from python / os
```

In [1]:

```
# Matplotlib
%matplotlib inline
from pylab              import gcf, sort, linspace, indices, std, empty, concatenate, pi, sqrt, ones, dia
from pylab              import rcParams, array, get_current_fig_manager, twinx, figure, subplots_adjust

from matplotlib.ticker  import MaxNLocator
from matplotlib         import style
from matplotlib         import pyplot as plt

# Numpy & Pandas
from numpy              import min, max, median, mean, zeros, empty
from numpy              import ones, where, arange, indices
from pandas             import DataFrame, read_csv, scatter_matrix

# Astropy
from photutils          import CircularAperture, aperture_photometry
from astroML.plotting   import hist
from astropy.modeling   import models, fitting
from astropy.io         import fits

# Time Stamps
import jd

# Built in Libraries
from datetime           import datetime
from os                 import listdir
from glob               import glob

from julian_date        import gd2jd
# Adam Ginsburg
from image_registration import cross_correlation_shifts

# Data Storage from Sci-kits
from sklearn.externals import joblib

from seaborn            import *

# from socket             import gethostname
from statsmodels.robust import scale
from sys                import exit, stdout
from time               import time

style.use('fivethirtyeight')
```

This is an example input for the requests below. The directory contains JWST-NIRCam fits files within it

```
- only works on Jonathan Fraine's Laptop
- soon to 'upgrade' to working on surtr
```

'/Users/jonathan/Research/NIRCam/CV3/StabilityTest/fitsfilesonly/reduced_orig_flags/redfits/NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T18h00m43.red/'

There is also a test file in the current working directory named `'fits_input_file.txt'`. It was creating using the bash 'script'

```
cd /Users/jonathan/Research/NIRCam/CV3/StabilityTest/fitsfilesonly/reduced_orig_flags/redfits/NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T18h00m43.red/

ls > fits_input_file.txt
```

Responding to the inquiry with (including appostraphes) either

`'fits_input_file.txt'`

or

`'/Users/jonathan/Research/NIRCam/CV3/StabilityTest/fitsfilesonly/reduced_orig_flags/redfits/NRCN821CLRSUB`
`6012172256_1_481_SE_2016-01-12T18h00m43.red/'`

is successful

Request Directory with a Set of Fits Files OR a Text File with the Same List

In [2]:

```
list_of_data_file_types = ['.txt', '.dat', '.csv']
nircam_data = DataFrame()
found       = False
DataDir     = input()


for filetype in list_of_data_file_types:
    if filetype in DataDir:
        nircam_data['fitsfilenames'] = read_csv(DataDir)
        found = True

if not found:
    nircam_data['fitsfilenames'] = glob(DataDir+'/*')
```

```
'fits_input_file.txt'
```

Compute Julian Data from Header

This function is a wrapper for `julian_date` in the `jd.py` package (soon to be converted to `julian_date.py` package. It's utility is in taking in the time stamps from the headers and converting them to the julian date; to be saved in the 'master' data frame below.

In [3]:

```
def get_julian_date_from_header(header):
    from jd import julian_date
    fitsDate    = header['DATE-OBS']
    startTimeStr= header['TIME-OBS']
    endTimeStr  = header['TIME-END']

    yy,mm,dd    = fitsDate.split('-')

    hh1,mn1,ss1 = array(startTimeStr.split(':')).astype(float)
    hh2,mn2,ss2 = array(endTimeStr.split(':')).astype(float)

    startDate   = julian_date(yy,mm,dd,hh1,mn1,ss1)
    endDate     = julian_date(yy,mm,dd,hh2,mn2,ss2)

    return startDate, endDate
```

Load Data / Gaussian Fit / AperturePhot Image

This function is the **crux** of the entire algorithm. The operation takes in one fits file name and outputs its time stamp, aperture photometry, gaussian centering / widths / amplitude, cross-correlation centering, and background subtracted values. The routine does the following:

1. Input:
    A. String (fits file name) or array (loaded fits file)
    B. The expected location of the star (center of frame is default)
    C. Subframe size (for better center fitting)
    D. List of aperture radii (or a float for a single aperture radii)
2. Operation:
    A. load the data.
    B. Computer the time element
    C. subtract the background (store background level)
    D. isolate the star into a subframe
    E. Cross-correlate a Gaussian (or JWST psf) with the image to find predicted center (store CC center)
    F. Gaussian fit to subframe, starting at CC center (store GS center, width, amplitude)
    G. Perform apeture photometry with each radius given at the beginning (store aperture radii as a function of radius)
3. Output
    A. time stamp
    B. aperture photometry
    C. gaussian amplitude
    D. gaussian centering
    E. gaussian widths
    F. cross-correlation centering
    G. background subtracted values.

This routine ensures that the user can manipulate the inputs as needed. Users can either send a single fits array, a set of fits array, a single string with the location of a fits file, or a list of strings with the location of several fits files.

```
In [4]:
```

```
def load_fit_phot_time(fitsfile, guesscenter = None, subframesize = [10,10], aperrad = [5],
                           nGroupsBig = 100, stddev0 = 2.0):
    y,x     = 0,1
    zero    = 0
    day2sec = 86400.
    k       = int(fitsfile.split('_I')[-1][:3])

    fitsname      = fitsfile.split('/')[-1]
    fitsfile      = fits.open(fitsfile)
    startJD,endJD = get_julian_date_from_header(fitsfile[0].header)
    timeSpan      = (endJD - startJD)*day2sec/nGroupsBig
    time          = startJD  + timeSpan*(k+0.5) / day2sec - 2450000.

#     print '\nNEED to control for multiframe arrays; maybe request only SLP\n'
    dataframe     = fitsfile[0].data[2] - fitsfile[0].data[0]
    skybg         = np.median(dataframe)

    imagecenter   = 0.5*array(dataframe.shape)
    if guesscenter == None:
        guesscenter = imagecenter

    subframe      = dataframe[guesscenter[y]-subframesize[y]:guesscenter[y]+subframesize[y],
                              guesscenter[y]-subframesize[x]:guesscenter[y]+subframesize[x]].copy()

    # ysize, xsize  = fitsfile[0].data.shape
    yinds0, xinds0= indices(dataframe.shape)
    yinds         = yinds0[guesscenter[y]-subframesize[y]:guesscenter[y]+subframesize[y],
                           guesscenter[y]-subframesize[x]:guesscenter[y]+subframesize[x]]
    xinds         = xinds0[guesscenter[y]-subframesize[y]:guesscenter[y]+subframesize[y],
                           guesscenter[y]-subframesize[x]:guesscenter[y]+subframesize[x]]

    fitter        = fitting.LevMarLSQFitter()
    plane         = models.Linear1D
    gauss0        = models.Gaussian2D(amplitude = fitsfile[0].data.max(),
                                      x_mean    = guesscenter[x],
                                      y_mean    = guesscenter[y],
                                      x_stddev  = stddev0       ,
                                      y_stddev  = stddev0       ,
                                      theta     = zero)

    CCCenter      = cross_correlation_shifts(gauss0(xinds, yinds), subframe) + imagecenter
    CCCenter      = CCCenter[::-1] # need in order to associate y = 1, x = 0
    gauss1        = fitter(gauss0, xinds, yinds, subframe - skybg)

    circCenter     = gauss1.parameters[1:3][::-1] - imagecenter + subframesize

    circaper       = CircularAperture(circCenter, aperrad[0])
    aperphot       = aperture_photometry(data=subframe - skybg, apertures=circaper)
    del fitsfile[0].data
    fitsfile.close()
    del fitsfile

    return fitsname, float(aperphot['aperture_sum']), time, gauss1.amplitude.value, gauss1.y_mean.value,
            gauss1.x_mean.value, abs(gauss1.y_stddev.value), abs(gauss1.x_stddev.value), \
            CCCenter[1], CCCenter[0], skybg

#     return time, aperphot['aperture_sum'], gauss1, CCCenter, skybg
```

Test output using the first fits file name in the list from above

```
In [5]: load_fit_phot_time(nircam_data['fitsfilenames'][0], guesscenter = None)#[160,160]
```

```
/Users/jonathan/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:23: VisibleDeprecationWarn
ing: using a non-integer number instead of an integer will result in an error in the future
/Users/jonathan/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:28: VisibleDeprecationWarn
ing: using a non-integer number instead of an integer will result in an error in the future
/Users/jonathan/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:30: VisibleDeprecationWarn
ing: using a non-integer number instead of an integer will result in an error in the future
```

```
Out[5]: ('NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T18h00m43.red_I002.fits',
         59180.46450882219,
         7400.228726171423,
         12658.452268721961,
         160.05879372620817,
         162.68132082669806,
         0.8694059934665892,
         0.8649280606609967,
         162.74494690546683,
         160.07161319122716,
         74.0)
```

Wrapper function to cycle through each fits file name in the list of fits files from user input

Takes in a list of fits file names, loops over them in the crux function above, stores each entry (output from crux) into a dataframe for later storage and processing.

Input:

```
1. List of fits file names to be loaded
2. Initial guess location of star
3. Subframe size to compute centering and photometry within
4. Aperature radius to compute photometry over
5. Predicted with of PSF (nyquist sampling = 2)
```

Operation:

```
1. Loop over each file in the list of fits files
2. Send the fits file names to the crux function
3. Receive output list of aper phot, gauss centers/widths/amplitudes, cross-corr centers, sky background
4. Input the above computed values in the master data frame for stroage and later processing
```

Outputs:

```
1. Master dataframe containing list of aper phot, gauss centers/widths/amplitudes, cross-corr centers, sky bg
```

In [6]:

```
def loads_fits_phots_times(fitsfiles, guesscenter = None, subframesize = [10,10], aperrad = [5], stddev0
    '''
    'sky background'
    'cross correlation center'
    'gaussian center'
    'gaussian width'
    'gaussian ampitude'
    'aperture photometry dictionary' or 'aperture photometry dataframe'
    the keys to the aperture photometry dictionary or data frame will be the float values of the aperture
    'time' (in days?)
    '''

    print 'Need to add multiple aperture raddii usage'
    columnNames = ['filename'           , 'aperture phot %.1f' %aperrad[0],
                   'time'               , 'gaussian amplitude' ,
                   'gaussian y center'  , 'gaussian x center'  ,
                   'gaussian y width'   , 'gaussian x width'   ,
                   'cross corr y center', 'cross corr x center',
                   'sky background']

    nircam_master_df = DataFrame(columns=columnNames)
    for fitsfile in fitsfiles:
        columnInputs = load_fit_phot_time(fitsfile, guesscenter  = guesscenter,
                                                    subframesize = subframesize,
                                                    aperrad      = aperrad,
                                                    stddev0      = stddev0)
        #
        nircam_master_df.loc[len(nircam_master_df)] = columnInputs

    return nircam_master_df
```

Create JWST-NIRCam Master DataFrame and Print Out Table Thereof

The table below is the entire data set computed from the wrapper to the crux function

```
In [7]: nircam_master_df = loads_fits_phots_times(nircam_data['fitsfilenames'], guesscenter = None,
                                                  subframesize = [10,10], aperrad = [3], stddev0 = 2.0)
        nircam_master_df
```

```
Need to add multiple aperture raddii usage

/Users/jonathan/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:23: VisibleDeprecationWarn
ing: using a non-integer number instead of an integer will result in an error in the future
/Users/jonathan/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:28: VisibleDeprecationWarn
ing: using a non-integer number instead of an integer will result in an error in the future
/Users/jonathan/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:30: VisibleDeprecationWarn
ing: using a non-integer number instead of an integer will result in an error in the future
```

Out[7]:

	filename	aperture phot 3.0	time	gaussian amplitude	gaussian y center	gaussian x center	gaussian y width	gaussian x width	cross y cent
0	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12994.321388	7400.228726	12658.452269	160.058794	162.681321	0.869406	0.864928	162.74
1	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13147.855538	7400.228776	12810.408004	160.056269	162.677451	0.870740	0.847847	162.74
2	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13639.239849	7400.228825	12537.745624	160.055507	162.658742	0.875230	0.862240	162.72
3	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13751.872732	7400.228874	13027.911062	160.052331	162.648373	0.860581	0.841280	162.71
4	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13611.282800	7400.228924	12836.160231	160.044295	162.646101	0.847303	0.869884	162.70
5	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13363.330813	7400.228973	12929.235932	160.047626	162.655526	0.859026	0.852607	162.72
6	NRCN821CLRSUB1-6012172256_1_481_SE_2016-	13068.770177	7400.229022	12676.107372	160.030818	162.652132	0.865961	0.858885	162.71

	01-12T...								
7	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13159.304183	7400.229072	12871.447578	160.034013	162.654841	0.851660	0.866995	162.72
8	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12960.174343	7400.229121	12857.573207	160.020842	162.650891	0.866996	0.848764	162.72
9	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12744.905423	7400.229170	12893.242038	160.017789	162.653302	0.859876	0.851008	162.71
10	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12873.895574	7400.229220	12802.252877	160.025179	162.658336	0.848897	0.870504	162.72
11	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12834.567557	7400.229269	12900.947770	160.032254	162.667465	0.865694	0.851802	162.73
12	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12438.613385	7400.229318	12694.046422	160.042747	162.696909	0.875471	0.849927	162.75
13	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11897.847499	7400.229368	12838.690641	160.048959	162.720998	0.847700	0.863128	162.78
14	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11576.806835	7400.229417	13038.489219	160.046121	162.736462	0.859510	0.849435	162.79
15	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11442.848388	7400.229466	13050.065454	160.038647	162.730910	0.863404	0.842674	162.79
16	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10806.993884	7400.229516	13123.844177	160.042667	162.751519	0.842232	0.853624	162.81
	NRCN821CLRSUB1-								

17	6012172256_1_481_SE_2016-01-12T...	10856.942999	7400.229565	13005.152770	160.056109	162.767544	0.865305	0.840133	162.82
18	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10558.932133	7400.229614	13232.876302	160.064251	162.789383	0.839499	0.850197	162.84
19	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10855.517942	7400.229664	12992.070378	160.074949	162.788134	0.860828	0.843727	162.84
20	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10818.591770	7400.229713	12866.319711	160.075076	162.809214	0.849751	0.870247	162.86
21	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10812.918984	7400.229762	12908.305885	160.073880	162.798788	0.865253	0.851512	162.85
22	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11019.483385	7400.229812	13004.186150	160.076899	162.791395	0.841125	0.867212	162.85
23	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11166.871976	7400.229861	12935.353598	160.078405	162.781824	0.863327	0.851011	162.83
24	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11565.695812	7400.229910	12782.331252	160.075652	162.766516	0.853949	0.868250	162.82
25	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11937.316879	7400.229960	12847.663909	160.068158	162.753385	0.848217	0.872662	162.81
26	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11614.696058	7400.230009	12742.268553	160.067742	162.741402	0.851592	0.866125	162.80
27	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11939.794073	7400.230058	13012.911811	160.077287	162.746616	0.863392	0.845130	162.81

28	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11936.696475	7400.230108	13057.808207	160.081565	162.745003	0.857155	0.844511	162.80
29	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11712.705789	7400.230157	12836.226197	160.075255	162.745934	0.863081	0.849661	162.80
...	...	...	...	...	...	...	...	...	...
69	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12204.587567	7400.232131	12739.595338	160.051477	162.708341	0.861849	0.867077	162.77
70	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11703.772934	7400.232180	12871.785111	160.060233	162.733510	0.860634	0.853784	162.79
71	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11403.395843	7400.232229	13148.720445	160.060188	162.746760	0.858364	0.835703	162.80
72	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10865.056702	7400.232279	12979.540309	160.064675	162.765482	0.861921	0.843434	162.82
73	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11098.057273	7400.232328	13080.333740	160.063584	162.766822	0.855908	0.844867	162.82
74	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10486.655661	7400.232377	13064.726289	160.063177	162.790758	0.854631	0.846646	162.84
75	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10353.647884	7400.232427	12756.931313	160.071310	162.810802	0.866306	0.855264	162.86
76	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10561.564939	7400.232476	12821.474772	160.079942	162.812435	0.855043	0.865002	162.87
	NRCN821CLRSUB1-								

77	6012172256_1_481_SE_2016-01-12T...	10661.605344	7400.232525	13173.924490	160.086885	162.821067	0.853228	0.844156	162.87
78	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10407.743465	7400.232575	12993.772057	160.093372	162.820933	0.848512	0.853002	162.87
79	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10478.298657	7400.232624	12724.340461	160.091306	162.821655	0.852741	0.873718	162.87
80	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10390.893708	7400.232673	13041.244043	160.099057	162.841088	0.853856	0.848729	162.89
81	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10417.814805	7400.232723	12910.834050	160.102924	162.839072	0.846453	0.861852	162.89
82	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10558.163616	7400.232772	13126.894199	160.103053	162.823289	0.842255	0.853383	162.88
83	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11172.866986	7400.232821	12772.767036	160.098202	162.807260	0.860749	0.865230	162.86
84	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	10876.072138	7400.232871	13002.617183	160.104900	162.816163	0.848682	0.857585	162.87
85	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	11361.551440	7400.232920	12992.371612	160.115012	162.798269	0.849300	0.861067	162.85
86	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12218.505439	7400.232969	12854.022353	160.126174	162.781674	0.850332	0.867779	162.84
87	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12725.653768	7400.233019	12936.618205	160.130511	162.756163	0.861433	0.850686	162.81

88	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	12820.706358	7400.233068	12802.652323	160.118851	162.745703	0.861105	0.853766	162.80
89	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13326.964772	7400.233117	13096.575058	160.118802	162.719729	0.852235	0.844675	162.78
90	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13803.196983	7400.233167	13009.674979	160.126264	162.705127	0.846677	0.855529	162.76
91	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13964.934399	7400.233216	12989.444255	160.126506	162.701693	0.856749	0.847080	162.76
92	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	14369.344832	7400.233265	12994.084692	160.119595	162.693367	0.847253	0.859541	162.75
93	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	14637.153148	7400.233315	13049.890839	160.132912	162.695089	0.846007	0.856696	162.75
94	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	14061.764608	7400.233364	12529.757067	160.127878	162.706684	0.852561	0.885469	162.76
95	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13815.445577	7400.233413	12653.067748	160.115190	162.700402	0.875291	0.856938	162.76
96	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	13938.144777	7400.233463	12987.171834	160.112906	162.693415	0.851323	0.855642	162.76
97	NRCN821CLRSUB1-6012172256_1_481_SE_2016-01-12T...	14647.927508	7400.233512	12948.740063	160.125076	162.682580	0.862847	0.845084	162.75
98	NRCN821CLRSUB1-6012172256_1_481_SE_2016-	15013.923799	7400.233561	13015.218587	160.122350	162.663436	0.861886	0.844199	162.73

	01-12T...								

99 rows × 11 columns

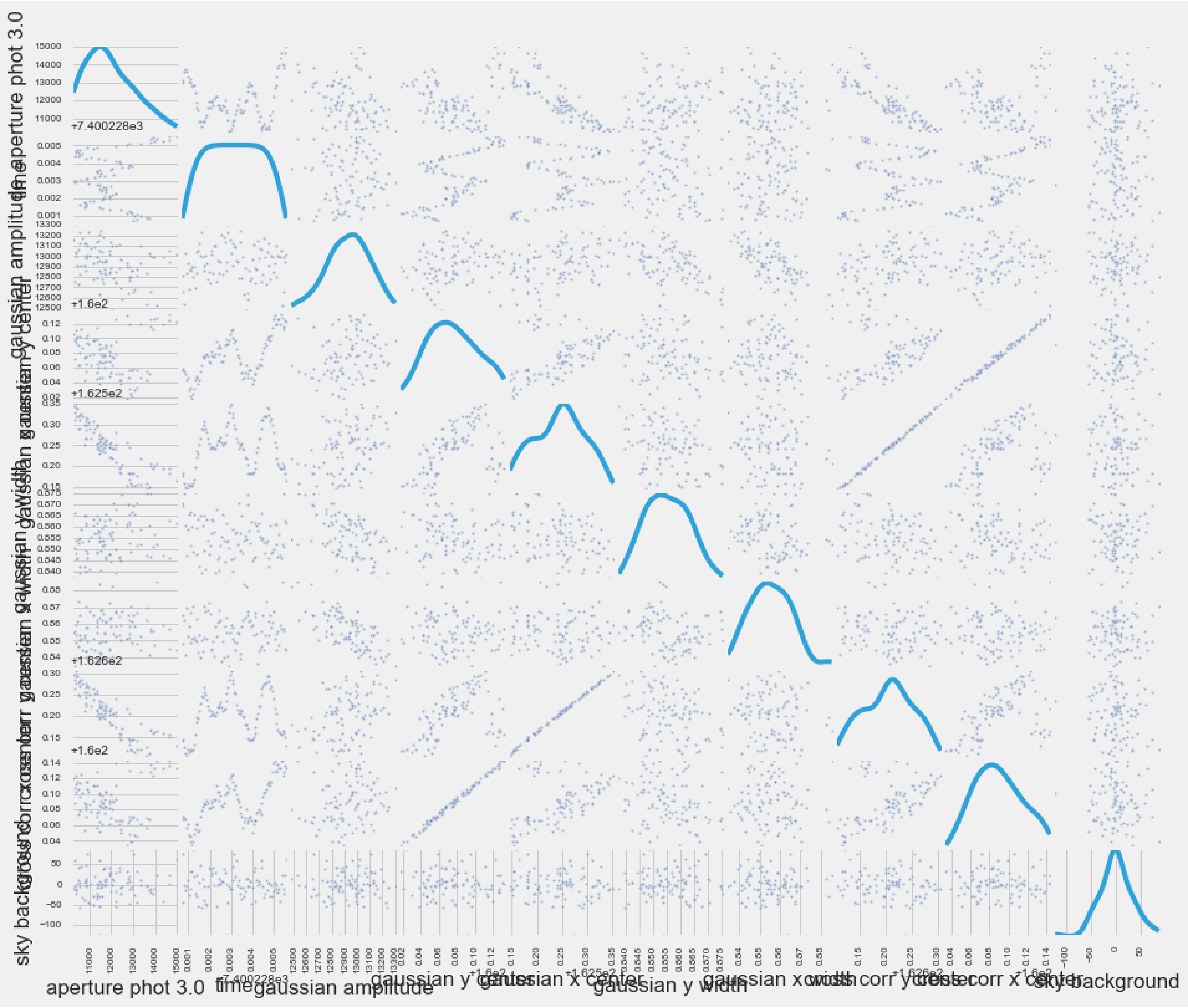
Generate Scatter Matrix to Cross Compare All Values with Eachother

The scatter matrix is a pandas data frame function that plots every column of the data frame against every other column of the data frame in a matrix format.

The diagonal is a kernel density estimator (default: histogram) as a metric on the specific column distribution.

In [8]:

```
scatter_matrix(nircam_master_df.drop('filename',1), diagonal='kde', figsize=(14,12));
```


aperture phot 3.0
time
gaussian amplitude
gaussian y center
gaussian x center
gaussian y width
gaussian x width
corr y center
corr x center
sky background
+7.400228e3
+1.6e2
+1.625e2
+1.626e2

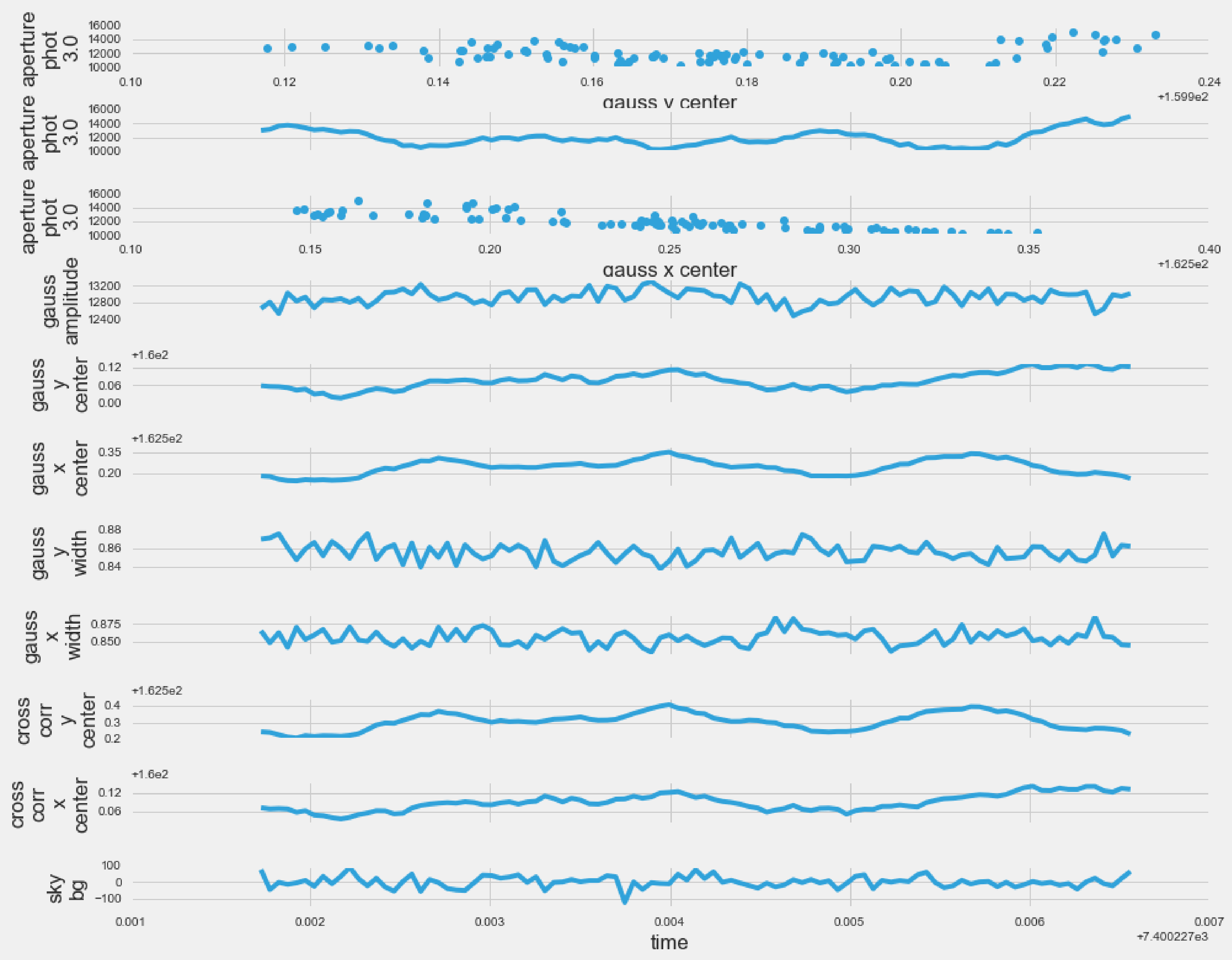
Plot All Values as Function of Time and Gaussian Centers

Cycle through all columns that have numerical data and plot them against time.

For special cases, plot the gaussian X and Y centers vs aperture photometry values

In [9]:

```
def renorm(arr):
    if arr.dtype == 'float64':
        return arr - median(arr)
    else:
        return arr

nircam_master_df.apply(renorm, axis=0)

fig = figure(figsize=(14,12))
for k, key in enumerate(nircam_master_df.keys()):
    ax  = fig.add_subplot(len(nircam_master_df.keys()), 1, k+1)
    if not key in ['time', 'filename']:
        ax.plot(nircam_master_df['time'], nircam_master_df[key])
        if k == len(nircam_master_df.keys()) - 1:
            ax.set_xlabel('time')
        else:
            ax.set_xticklabels([])

        ax.set_ylabel(key.replace('gaussian', 'gauss').replace('background', 'bg').replace(' ', '\n'))
        ax.yaxis.set_major_locator(MaxNLocator(nbins=3))

ax  = fig.add_subplot(len(nircam_master_df.keys()), 1, 1)
ax.plot(nircam_master_df['gaussian y center'], nircam_master_df['aperture phot 3.0'], 'o')
ax.set_ylabel('aperture phot 3.0'.replace(' ', '\n'))
ax.set_xlabel('gauss y center')
ax.yaxis.set_major_locator(MaxNLocator(nbins=3))

ax  = fig.add_subplot(len(nircam_master_df.keys()), 1, 3)
ax.plot(nircam_master_df['gaussian x center'], nircam_master_df['aperture phot 3.0'], 'o')
ax.set_ylabel('aperture phot 3.0'.replace(' ', '\n'))
ax.set_xlabel('gauss x center')
ax.yaxis.set_major_locator(MaxNLocator(nbins=3))

subplots_adjust( hspace=1 )
fig.canvas.draw()
```


aperture phot 3.0
16000
14000
12000
10000
0.10
0.12
0.14
0.16
0.18
0.20
0.22
0.24
+1.599e2
gauss y center
aperture phot 3.0
16000
14000
12000
10000
aperture phot 3.0
16000
14000
12000
10000
0.10
0.15
0.20
0.25
0.30
0.35
0.40
+1.625e2
gauss x center
gauss amplitude
13200
12800
12400
+1.6e2
gauss y center
0.12
0.06
0.00
+1.625e2
gauss x center
0.35
0.20
gauss y width
0.88
0.86
0.84
gauss x width
0.875
0.850
+1.625e2
cross corr y center
0.4
0.3
0.2
+1.6e2
cross corr x center
0.12
0.06
sky bg
100
0
−100
0.001
0.002
0.003
0.004
0.005
0.006
0.007
+7.400227e3
time

The Following is Strictly from My Python Routine

This routine is for later plotting with html interface. It is not useful for the above routines yet.

```
import numpy as np
from bokeh.plotting import figure, show, output_file

N = 10000

x = np.random.normal(0,np.pi, N)
y = np.sin(x) + np.random.normal(0,0.2,N)

output_file('test_bokeh2.html', title='scatter 10k points')

p = figure(webgl=False)
p.scatter(x,y,alpha=0.1)
show(p)
```