# RealDataTests

September 2, 2022

## 1 Tests of Novel Sky Subtraction on Real Data

Author: Aaron Watkins

Last Updated: 02/09/2022

[1]:
```
# Importing packages
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy.io import ascii as asc
```

[2]:
```
# Defining a function to make consistent images
def plotImage(ax, imageData, vmin, vmax):
    palette = plt.cm.gray.with_extremes(bad='k')
    maskedImage = np.ma.masked_array(imageData, mask=np.isnan(imageData))
    ax.imshow(maskedImage, origin='lower', vmin=vmin, vmax=vmax, cmap=palette)
```

### 1.0.1 Introduction

This notebook displays tests of a novel sky-subtraction algorithm under development for possible use with LSST images on a handful of datasets from past observing campaigns. The novel algorithm relies on first creating preliminary sky-subtracted image coadds, then using these coadds to remove stars, galaxies, and other objects persisting between the coadds and individual exposures. These coadd-subtracted images are then masked, binned, and smoothed to reduce the pixel-to-pixel noise, generating tailored sky maps for each exposure without the need for modeling and without contamination from low-surface-brightness astrophysical flux. The full method will be described in a paper (Watkins et al. in prep.).

I use images from two such campaigns, though this will be updated in the future with additional datasets. For now, these include:

- Nordic Optical Telescope narrow-band and r-band images
- Burrell Schmidt Telescope B-band images

Here I show a visual summary of the novel method, as well as visual comparisons between polynomial-modeled sky-subtracted images and sky-subtracted images using the novel method. I also show visual comparisons between the preliminary sky-subtracted coadds and second-generation coadds generated using the novel sky-subtration method, as well as quantitative comparisons between derived limiting surface brightnesses on each of these coadds.

Because the data rights for the images used for this experiment are ambiguous at the time of writing, I am publishing only this notebook, without the images attached. This may be updated in the future when the images themselves become, with certainty, publicly available.

### 1.0.2 Demonstrating visually the novel sky-subtraction algorithm

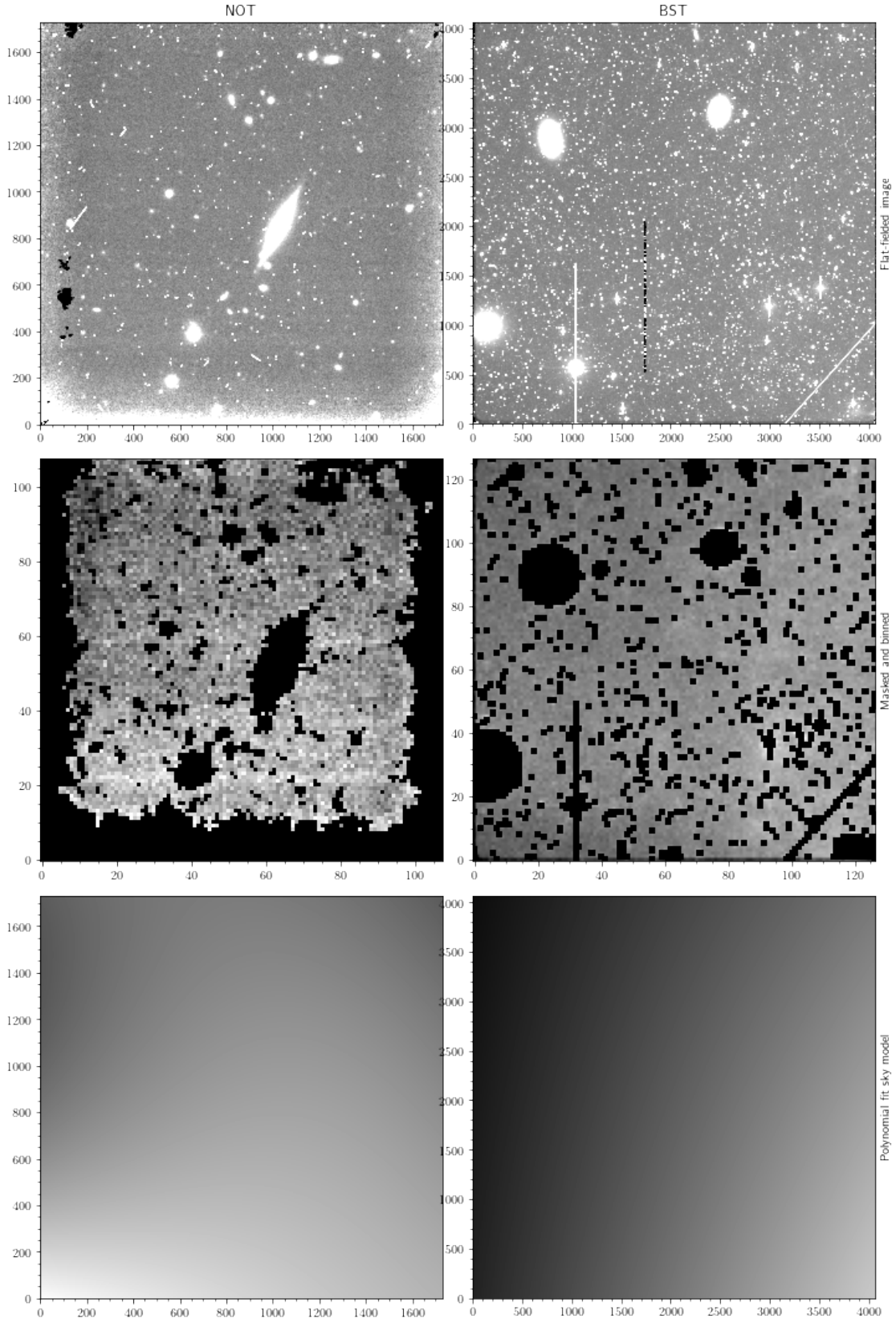
Here I demonstrate, step-by-step, how the novel sky-subtraction algorithm is intended to work, with example images of each important step.

The first step is to generate a preliminary sky-subtracted image coadd. We do this step, at the moment, by masking all flat-fielded images in the dataset, binning them, and then modeling the unmasked pixels with a 2D Legendre polynomial to generate smooth sky models for each image. We then subtract these models as the sky. The figure below shows example images from the Nordic Optical Telescope (NOT) and Burrell Schmidt Telescope (BST), their masked and binned counterparts, and the corresponding best-fit polynomial sky models. Masks for the NOT data were made using Noisechisel (Akhlaghi & Ichikawa 2015), while for the BST they were made using IRAF's OBJMASK task with additional by-hand masking (see Watkins et al. 2014). Extensive masking around the NOT image edge resulted from the images having strong vignetting due to the filter used.

[3]:
```
notIm = fits.getdata('ftzALDi270130.fits')
notMaskBin = fits.getdata('notDemoIm.fits')
notSkyModel = fits.getdata('notDemoSky.fits')

bstIm = fits.getdata('fwnlzobj0415037.fits')
bstMaskBin = fits.getdata('leoDemoIm.fits')
bstSkyModel = fits.getdata('leoDemoSky.fits')
```

[4]:
```
fig, ax = plt.subplots(3, 2, figsize=(10, 15))
plotImage(ax[0,0], notIm, vmin=600, vmax=900)
ax[0,0].set_title('NOT')
plotImage(ax[0,1], bstIm, vmin=950, vmax=1030)
ax[0,1].set_title('BST')
ax[0,1].yaxis.set_label_position('right')
ax[0,1].set_ylabel('Flat-fielded image')
plotImage(ax[1,0], notMaskBin, vmin=733, vmax=783)
plotImage(ax[1,1], bstMaskBin, vmin=980, vmax=1005)
ax[1,1].yaxis.set_label_position('right')
ax[1,1].set_ylabel('Masked and binned')
plotImage(ax[2,0], notSkyModel, vmin=733, vmax=783)
plotImage(ax[2,1], bstSkyModel, vmin=990, vmax=1000)
ax[2,1].yaxis.set_label_position('right')
ax[2,1].set_ylabel('Polynomial fit sky model')
plt.tight_layout()
```

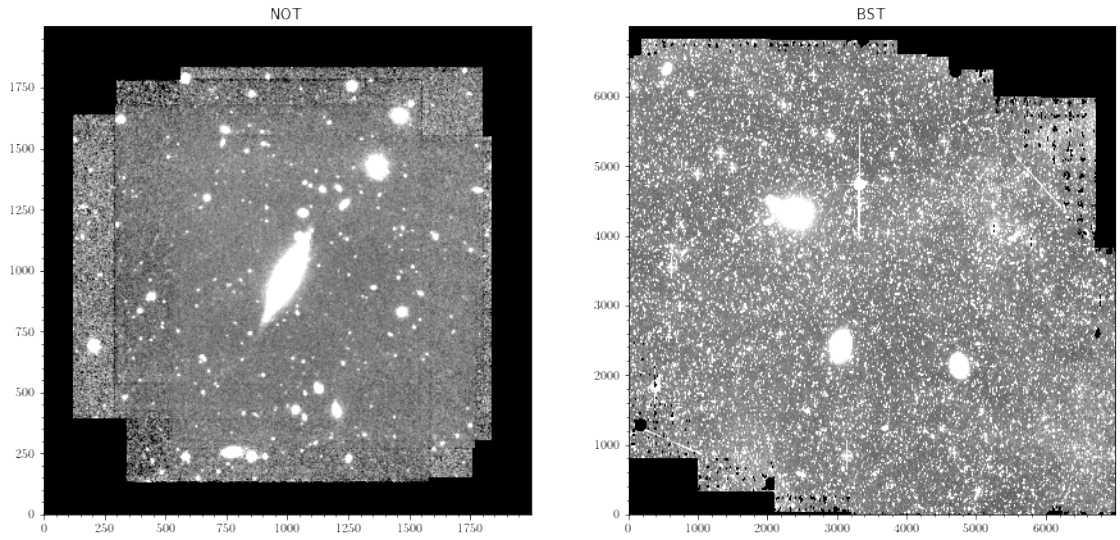

NOT
BST
Flat-fielded image
Masked and binned
Polynomial fit sky model

We subtract these polynomial models from each corresponding image, register all sky-subtracted images to a common world coordinate system, and stack the registered images together. This results in the following image coadds.

[5]:
```
notMos = fits.getdata('ESO544_027_ha_polysub.fits')
bstMos = fits.getdata('leoB_polysub.fits')
```

[6]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 7))
plotImage(ax[0], notMos, vmin=-50, vmax=50)
ax[0].set_title('NOT')
plotImage(ax[1], bstMos, vmin=-10, vmax=10)
ax[1].set_title('BST')
```

[6]:
```
Text(0.5, 1.0, 'BST')
```



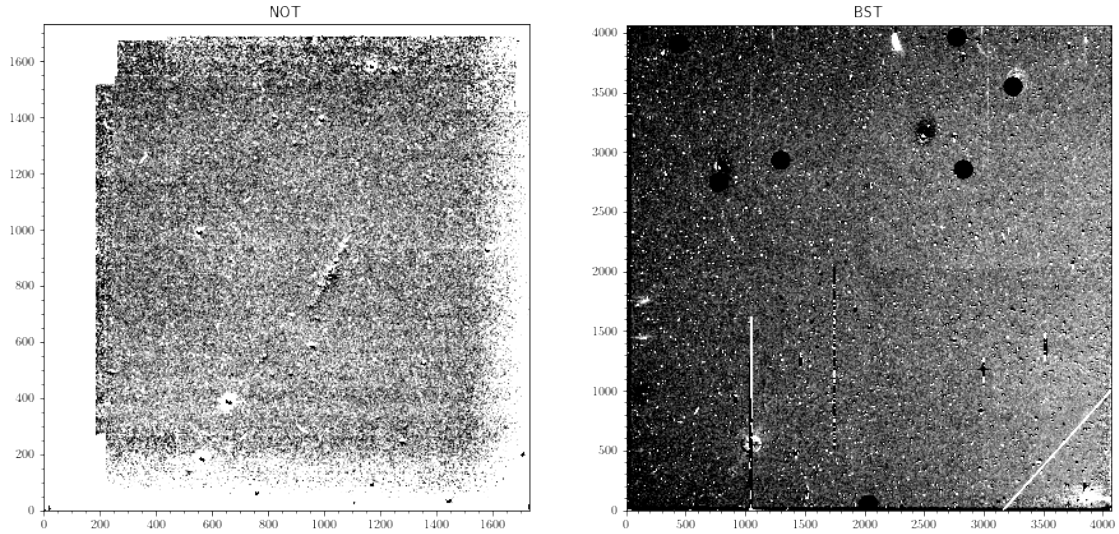

We then register these coadds back onto the individual flat-fielded exposures, scale them in flux to said exposures' photometric zeropoints (which we estimate by matching the target galaxies' fluxes between coadd and exposure), and subtract them from the exposures. This results in maps of the night sky, with artifacts from e.g. PSF differences between exposure and coadd, cosmic ray hits, bad columns, etc. We show examples of this for a single image from each dataset, chosen arbitrarily.

[7]:
```
notSkyMap = fits.getdata('ms_tzALDi270130.fits')
bstSkyMap = fits.getdata('Mfwnlzobj0415037.fits')
```

[8]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 7))
plotImage(ax[0], notSkyMap, vmin=733, vmax=783)
ax[0].set_title('NOT')
plotImage(ax[1], bstSkyMap, vmin=990, vmax=1000)
```

```
ax[1].set_title('BST')
```

[8]: `Text(0.5, 1.0, 'BST')`




The structure in these images appears at first glance to better represent the real backgrounds in the images, moreso than the polynomial fit models. The starkest example in the BST image, in which slight differences in pedestal levels between image quadrants are clearly visible. These boundaries mark the regions controlled by different amplifiers on the CCD; at the time this was first reduced, some effort was made to correct for quad-to-quad differences, but evidently some remains at a low level, which appears upon coadd-subtraction.

We reduce the noise in these sky maps via the following steps:

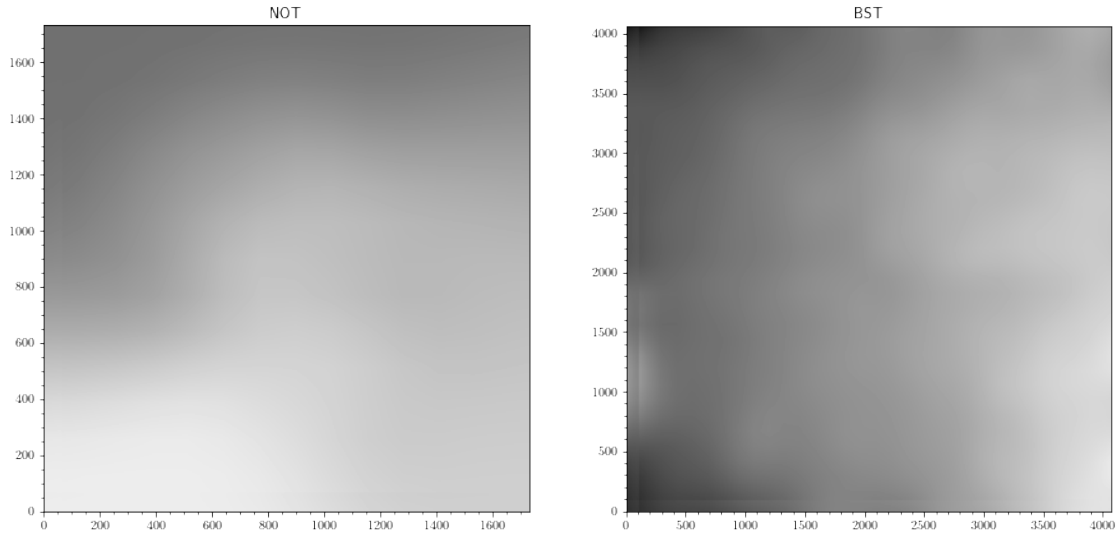
- First, we mask the images of residual artifacts
- Then we median-bin the masked images with a bin factor dependent on the polynomial order we used to fit the sky in the initial sky-subtraction (128 x 128 pixels for 2nd order, 256 x 256 pixels for 1st order)
- If masks remain after binning, we interpolate the image flux across these masks using in-painting (OpenCV), an art restoration technique
- We then smooth the restored binned images with a Gaussian kernel of width 1 px (i.e., a kernel with the same size as the bin factor used)
- Finally, we restore the binned, smoothed sky map to the proper resolution using bi-linear interpolation

Below, the corresponding smooth sky maps resulting from this full process, to be compared to the above noisy sky maps generated solely by coadd-subtraction.

[9]:
```
notSmoothSky = fits.getdata('sky_tzALDi270130.fits')
bstSmoothSky = fits.getdata('bnMfwnlzobj0415037.fits')
```

[10]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 7))
plotImage(ax[0], notSmoothSky, vmin=740, vmax=770)
ax[0].set_title('NOT')
plotImage(ax[1], bstSmoothSky, vmin=987, vmax=998)
ax[1].set_title('BST')
```

[10]:
```
Text(0.5, 1.0, 'BST')
```




Finally, we subtract these smooth sky maps from their corresponding images, and register and combine these new sky-subtracted images into second-generation coadds.

### 1.0.3 Comparing individual sky-subtracted exposures

Below, we show single exposures from the NOT campaign and the BST campaign, post-sky-subtraction. On the left, we show images using the initial polynomial-fit sky subtraction, while on the right we show images from which the coadd-subtraction sky models were subtracted. To enhance the residual structures in the backgrounds, we show masked, binned version of these sky-subtracted images.

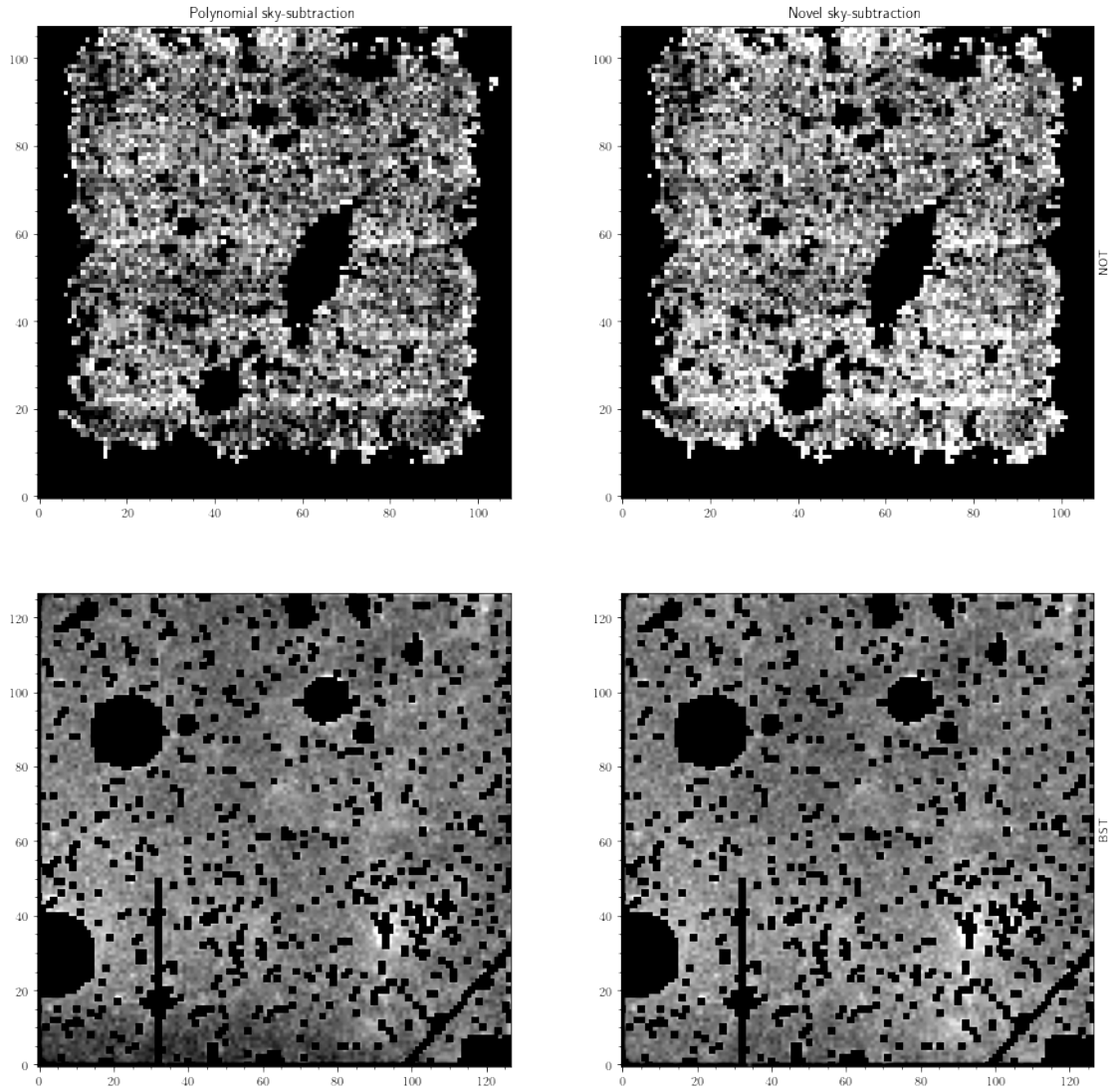
[11]:
```
notSsPoly = fits.getdata('notSkySubInit.fits')
notSsNovel = fits.getdata('notSkySubFinal.fits')

bstSsPoly = fits.getdata('leoSkySubInit.fits')
bstSsNovel = fits.getdata('leoSkySubFinal.fits')
```

[12]:
```
fig, ax = plt.subplots(2, 2, figsize=(15, 15))
plotImage(ax[0,0], notSsPoly, vmin=-10, vmax=10)
plotImage(ax[0,1], notSsNovel, vmin=-10, vmax=10)
ax[0,0].set_title('Polynomial sky-subtraction')
ax[0,1].set_title('Novel sky-subtraction')
```

```
ax[0,1].yaxis.set_label_position('right')
ax[0,1].set_ylabel('NOT')
plotImage(ax[1,0], bstSsPoly, vmin=-5, vmax=5)
plotImage(ax[1,1], bstSsNovel, vmin=-5, vmax=5)
ax[1,1].yaxis.set_label_position('right')
ax[1,1].set_ylabel('BST')
```

[12]: `Text(0, 0.5, 'BST')`




The novel method does appear to result in a slight improvement over the initial sky subtraction, insofar as the backgrounds in the sky-subtracted images are a touch flatter overall when using the novel method. The expectation, then, is that this novel method should result in improved backgrounds, yielding less noisy coadds. But we test this explicitly in the next section.

### 1.0.4 Comparing first- and second-generation coadds

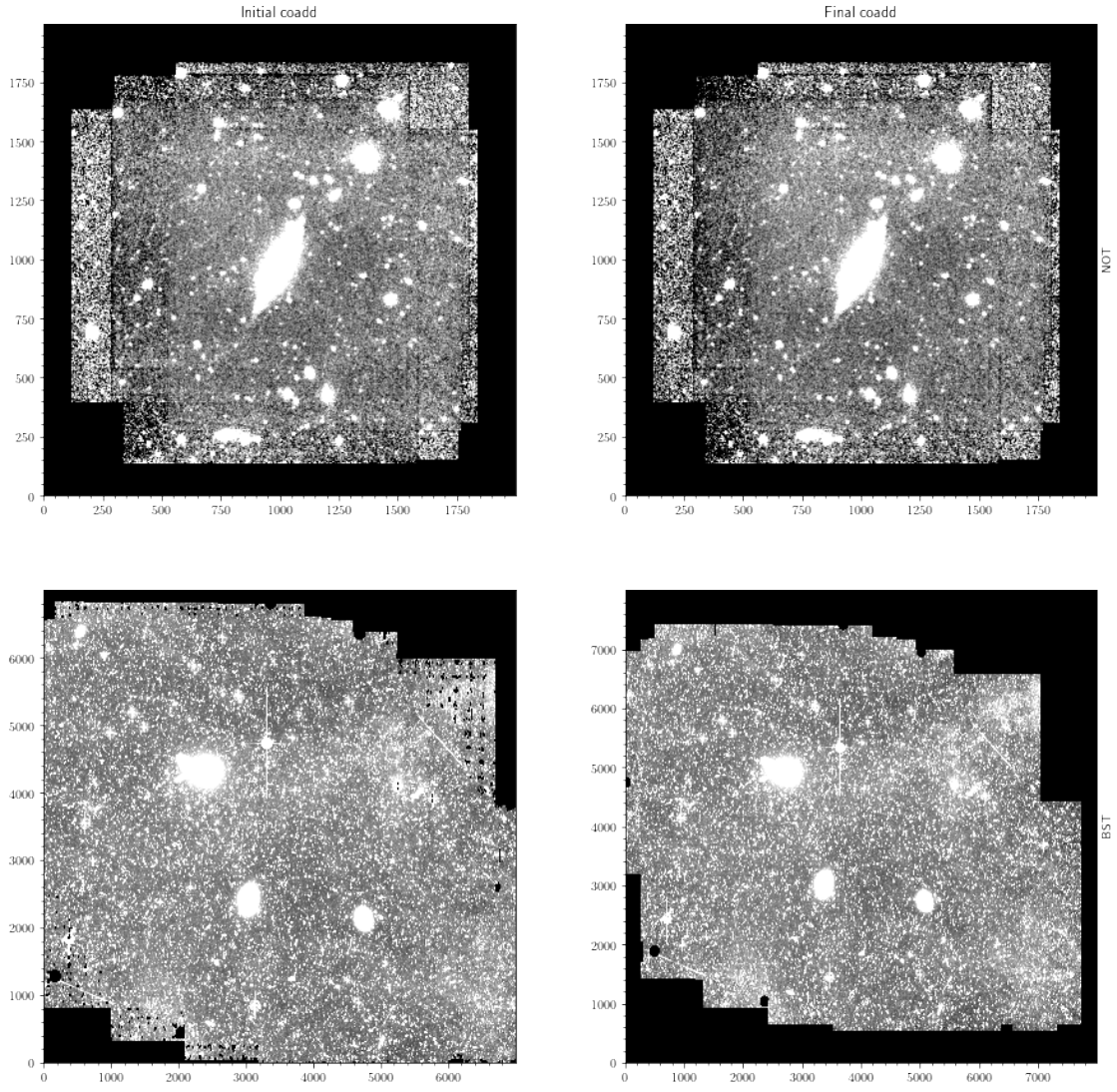
Below, we show a visual comparison between the initial and final-generation coadds for the NOT and BST campaigns tested here. Initial coadds are shown on the left, while final coadds are shown on the right.

[13]:
```
notCoaddInit = fits.getdata('ESO544_027_ha_polysub.fits')
notCoaddFinal = fits.getdata('ESO544_027_ha_mossub.fits')

bstCoaddInit = fits.getdata('leoB_polysub.fits')
bstCoaddFinal = fits.getdata('leoB_mossub.fits')
```

[14]:
```
fig, ax = plt.subplots(2, 2, figsize=(15, 15))
plotImage(ax[0,0], notCoaddInit, vmin=-20, vmax=20)
plotImage(ax[0,1], notCoaddFinal, vmin=-20, vmax=20)
ax[0,0].set_title('Initial coadd')
ax[0,1].set_title('Final coadd')
ax[0,1].yaxis.set_label_position('right')
ax[0,1].set_ylabel('NOT')
plotImage(ax[1,0], bstCoaddInit, vmin=-7, vmax=7)
plotImage(ax[1,1], bstCoaddFinal, vmin=-7, vmax=7)
ax[1,1].yaxis.set_label_position('right')
ax[1,1].set_ylabel('BST')
```

[14]:
```
Text(0, 0.5, 'BST')
```




Visually, both coadd versions look quite similar, with minimal obvious improvement coming from the novel sky-subtraction method. We therefore tested quantitatively how each coadd differs, by measuring a limiting surface brightness on each image. The standard method to measure a limiting surface brightness is to mask each coadd, randomly distribute equal-area boxes across the coadd centers (where the S/N is highest), and measure the median value within each box. The standard deviation of the medians is the limiting surface brightness in counts.

To estimate uncertainties, we adopted this strategy, but repeated it 200 times. For each iteration, we used 100 boxes, distributed within a circular region with a radius 1/3 of the image dimensions. At the end of this process, we measured 200 values of limiting surface brightness for each image, so we take as the true limiting surface brightness for each image the median among these 200 values, with the standard deviation in those values as the uncertainty on those estimates. We show the results below.

These are shown by image name. NOT imaging was of the galaxy ESO544_027, while BST imaging

was of the Leo I Group. The tag 'polysub' indicates the initial coadd with polynomial modeled skies, while 'mossub' indicates the final generation coadds made using the novel sky-subtraction algorithm. NOT imaging was done in two bands, an H-alpha narrow-band (ha) and r-band (r, not shown throughout this notebook).

[15]:
```
sbLimTab = asc.read('lim_sbs.tab')
print(sbLimTab)
```

```
        image                limSb              limSb_std
--------------------- ------------------- -------------------
 ESO544_027_ha_mossub   2.927180591487728  0.2214384566606363
ESO544_027_ha_polysub   2.147711396217346 0.18363235499997982
  ESO544_027_r_mossub  2.5206265938073305   0.286483430289972
 ESO544_027_r_polysub   2.141540994089185 0.18758326845162335
          leoB_mossub 0.46845194697380066 0.04570351541042328
         leoB_polysub  0.4691973328590393 0.04186811298131943
```

Evidently, the limiting surface brightness either increases (i.e. gets worse) or remains the same using the novel sky-subtraction method. Given that the backgrounds do look slightly flatter when using the novel method, we believe the reason behind this degradation in limiting surface brightness is the noise each coadd-subtraction-based sky map adds to each image upon subtraction.

While binning and smoothing reduces the noise in the coadd-subtracted images, it does not remove it entirely, so each image post sky-subtraction has this noise imprinted on it at the binning scale used. This is a somewhat insurmountable flaw–noise reduction techniques we have investigated are either imperfect and fast, or a bit less imperfect but very much slower. Therefore, we must consider whether model-free sky maps are the best approach; it may be better to re-model the skies with polynomials from the coadd-subtracted images, which will produce noiseless sky models which avoid bias from contamination by unmasked low-surface-brightness features. We will continue to experiment in consulation with the LSST Data Management Team to settle on the ideal solution.

[ ]:
```

```