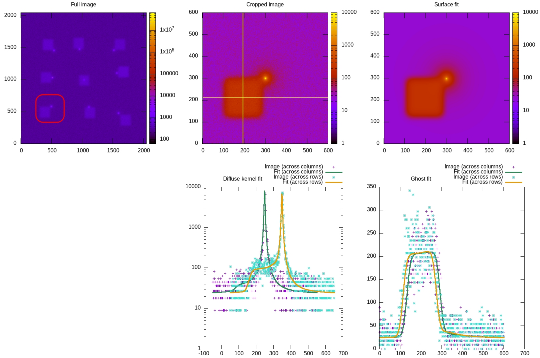
SPEXone L1A-L1B processor updates: stray light and binning

Raul Laasner

6 October 2022

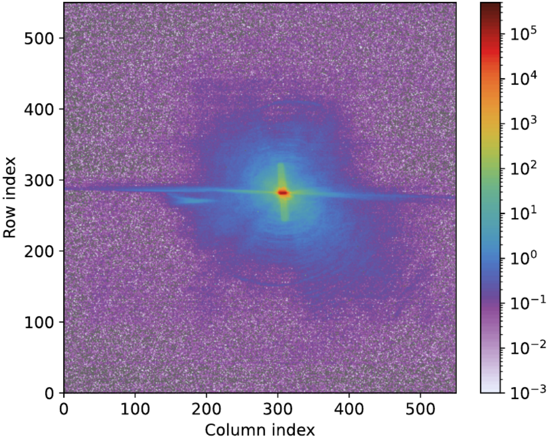
Previously

- The stray light model was based on analytical diffuse and ghost kernels.
- The kernels were determined from fits to (simulated) stray light calibration measurements.

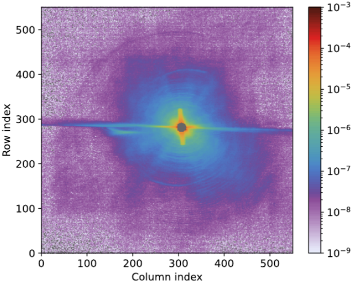


Real calibration measurements

- However, real calibration measurements have too much structure for an analytical formalism.

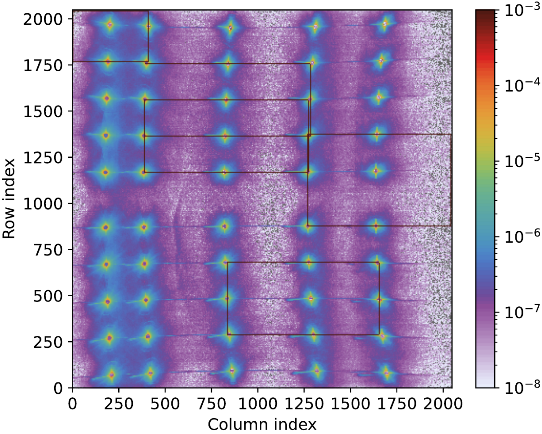


- Construct a stray light kernel from measurements of multiple exposure times and across track (ACT) angles. This increases the signal to noise (SNR) ratio.
- Normalize and set values within a radius of the image center to 0



However, the kernel looks different at different parts of the detector.

- Divide the detector into regions and derive a stray light kernel corresponding to each region.
- Each kernel $\boldsymbol{K}_k$ has associated weights $\boldsymbol{w}_k$ which define its region of influence or domain.

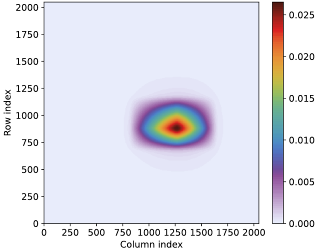


Procedure

- In order to activate a kernel k only within in its domain (box) we define a weight for it:

$$\begin{aligned} w_{k,ij} &= w_{k,i} w_{k,j}, \\ w_{k,i} &= \begin{cases} \frac{i - i^{\downarrow}}{i_0 - i^{\downarrow}} & i^{\downarrow} \leq i < i_0 \\ \frac{i^{\uparrow} - i}{i^{\uparrow} - i_0} & i_0 \leq i < i^{\uparrow}, \end{cases} \\ w_{k,j} &= \begin{cases} \frac{j - j^{\leftarrow}}{j_0 - j^{\leftarrow}} & j^{\leftarrow} \leq j < j_0 \\ \frac{j^{\rightarrow} - j}{j^{\rightarrow} - j_0} & j_0 \leq j < j^{\rightarrow}. \end{cases} \end{aligned} \tag{1}$$

- Outside the box $(i^{\downarrow}, i^{\uparrow}, j^{\leftarrow}, j^{\rightarrow})$ the kernel should have no effect.
- The box boundary is at the center of a neighboring kernel or a detector edge.



- Convolution with multiple kernels:

$$S_{ij}^{\text{conv}} = \sum_k \sum_{m=0,n=0}^{m=N,n=N} w_{k,mn} K_{k,ij,mn} S_{mn}^{\text{ideal}}, \tag{2}$$

  where $\boldsymbol{S}^{\text{ideal}}$ is an ideal signal, $\boldsymbol{K}_k$ is the kth kernel, $\boldsymbol{w}_k$ the corresponding weight, $\boldsymbol{S}^{\text{conv}}$ the convolved signal, and N is the detector dimension.

- The weights must be normalized:

$$w_{k,mn} \to w_{k,mn} / \sum_k w_{k,mn}, \tag{3}$$

  so that at each pixel the kernel weights add up to 1.

Deconvolution

- The standard Van Cittert deconvolution algorithm is

$$S_{ij}^{(v+1)} = \frac{S_{ij}^{(0)} - \sum_{mn} K_{ij,mn} S_{mn}^{(v)}}{1 - \sum_{mn} K_{ij,mn}}, \tag{4}$$

  where $\boldsymbol{S}^{(0)}$ is the convolved image, $\boldsymbol{K}$ is a kernel, and $\boldsymbol{S}^{(v+1)}$ is the updated image after $(v+1)$th iteration.
- The sum in the denominator is called the internal scattering factor:

$$\eta_{ij} \equiv \sum_{mn} K_{ij,mn}. \tag{5}$$

Deconvolution

- With multiple kernels the Van Citter algorithm is

$$S_{ij}^{(v+1)} = \frac{S_{ij}^{(0)} - \sum_k \sum_{mn} w_{k,mn} K_{k,ij,mn} S_{mn}^{(v)}}{1 - \sum_k \eta_{k,ij}}. \tag{6}$$

- The weights can be absorbed into the signal and thus there are no difficulties computing the convolutions:

$$\begin{aligned} \tilde{S}_{k,mn}^{(v)} &= w_{k,mn} S_{mn}^{(v)}, \\ \sum_k \sum_{mn} w_{k,mn} K_{k,ij,mn} S_{mn}^{(v)} \\ &= \sum_k \sum_{mn} K_{k,ij,mn} \tilde{S}_{k,mn}^{(v)} \\ &= \sum_k \boldsymbol{K}_k \otimes \tilde{\boldsymbol{S}}_{\boldsymbol{k}}^{(v)}. \end{aligned} \tag{7}$$

- In principle, each convolution $\boldsymbol{K}_k \otimes \tilde{\boldsymbol{S}}_{\boldsymbol{k}}$ operates on all pixels of the detector.
- If the kernel has “extent” r - meaning that $\boldsymbol{K}_k$ is 0 at r pixels from its center - we only need to consider a subimage $\tilde{\boldsymbol{S}}^s_{\boldsymbol{k}}$ with a box side of $W + 2r$ and a kernel with box side of $W + 6r$ where W is the length of a box side defined by the weight $\boldsymbol{w}_k$.
- We test two different kernel extents:
  - $r = 512$ pixels
  - $r = 256$ pixels
- We also test using a smaller number of kernels by skipping some wavelengths — from 50 to 30 kernels.

- In this delivery, the L1A product has been generated using the real flight binning table.
- This significantly speeds up noise progapation during the demodulation step in the L1A-L1B processor.

- The new delivery is located at `https://public.spider.surfsara.nl/project/spexone/PACE/L1A-L1C/2022_10_06/`
- `release_notes.pdf` explains the content of the delivery, including how to build and run the software.
- The objective is to have three successful runs:

```
# 50 kernels, kernel extent r = 512 pixels
mpirun -np <n> <spexone> L1B_full.yaml
# 30 kernels, kernel extent r = 512 pixels
mpirun -np <N> <spexone> L1B_30_kernels.yaml
# 50 kernels, kernel extent r = 256 pixels
mpirun -np <N> <spexone> L1B_reduced.yaml
```

Timings

Using an AMD Ryzen 9 5950X, 10 cores, 5000 images in L1A product:

`L1B_full.yaml`	134 min
`L1B_30_kernels.yaml`	84 min
`L1B_reduced.yaml`	70 min

(Using a better value of `[l1b][first_proc_rel_workload]` could save 5–10 min)

Processor output using `L1B_reduced.yaml`:

```
...
[14:51:28]          Dark correction:    0.548 s (900 calls)
[14:51:28]         Noise estimation:    0.720 s (450 calls)
[14:51:28]  Nonlinearity correction:   34.328 s (450 calls)
[14:51:28]          PRNU correction:    0.687 s (450 calls)
[14:51:28]   Stray light correction: 2499.970 s (450 calls)
[14:51:28] Spectra extraction (FOV):    9.004 s (91350 calls)
[14:51:28]  Radiometric calibration:   10.705 s (91350 calls)
[14:51:28]             Demodulation:  916.252 s (450 calls)
[14:51:28] Wall time for MPI process   0: 4188.550 s
[14:51:28] Wall time for MPI process   1: 4147.296 s
...
```