



The lineflt code for viewing a spectral line with two bandwidth filters

Petter Ström
***Department of Fusion Plasma Physics,
School of Electrical Engineering,
KTH Stockholm, Association VR***

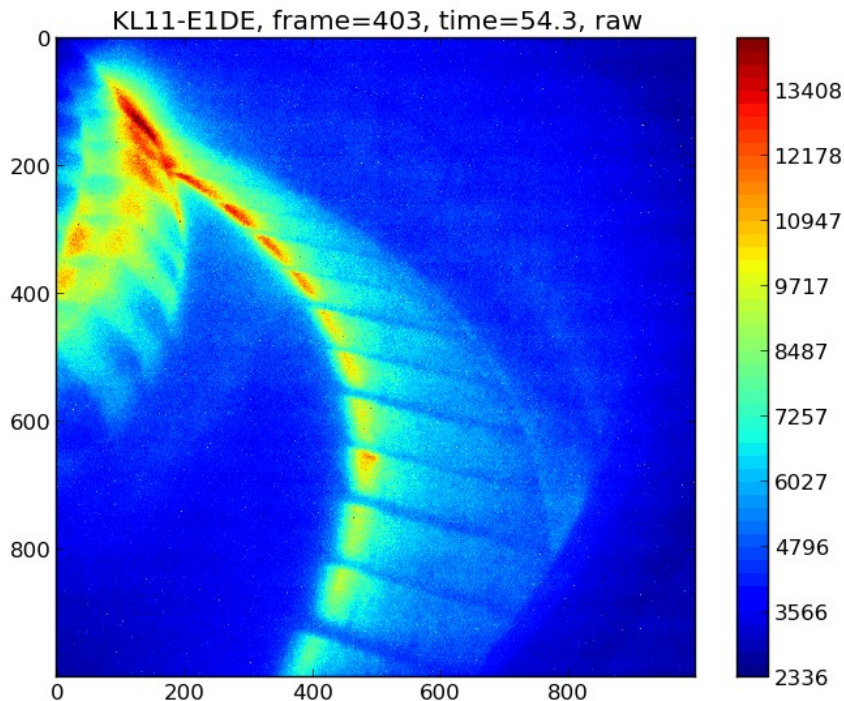
2016-02-05

- **Purpose:** what do we want to do?
- **Method:**
 - how to do it?
 - complications
- **Inputs, outputs**
 - images
 - processing parameters
- **Individual scripts and functions**
 - quick overview of data flow
 - required data files, format
- **Results so far**

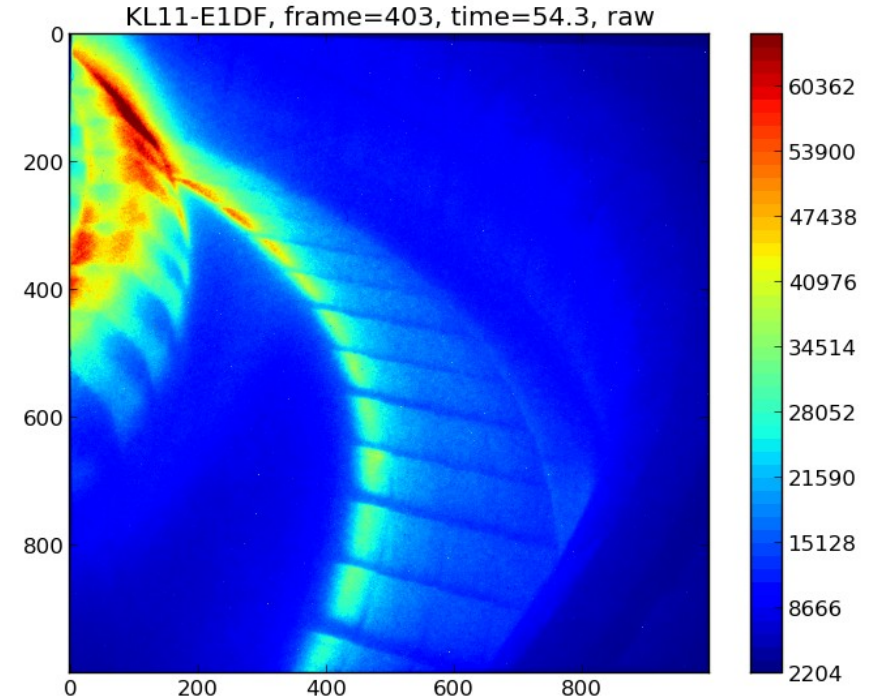
- Produce a clean KL11 image of a single spectral line!

Input: same image, different bandwidth filters

Example, WI line at 400.88 nm, pulse #84782

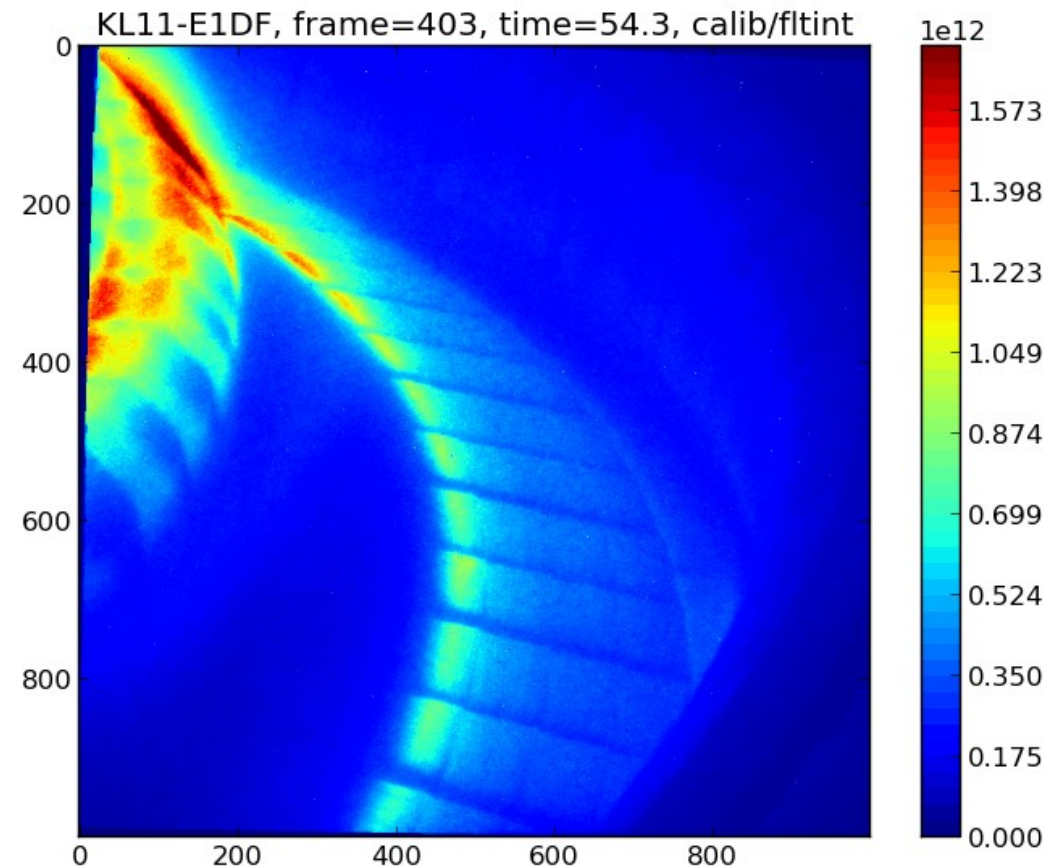


↑
Narrow filter



↑
Broad filter

- **Step 1:** Stretch one image so that it matches the other, pixel by pixel.
- **Step 2:** Calibrate for gain factor of cameras' electronics and image intensifier.
- **Step 3:** Divide by total integral of filter function (amplification factor for background), ND filter transmittance and exposure time.



- **Step 4:** Pixel-wise comparison of images.

$$B = \frac{M_2 F_1 S_{1,cal} - M_1 F_2 S_{2,cal}}{M_2 F_1 - M_1 F_2}$$

$$WI = \frac{S_{2,cal} - S_{1,cal}}{\frac{M_2}{F_2} - \frac{M_1}{F_1}}$$

$S_{1,cal}$ and $S_{2,cal}$ are here exactly the calibrated pixel values from the last slide. M_i and F_i are the transmittance at the line of interest and total integral respectively of filter i . For derivation: see lineflt user manual!

- **Manually select points to generate parameters for “stretching” image**

```
python stretchfunction.py [cam1] [cam2] [pulse] [time]
```

a) Fit 6 parameters for two general linear functions.

$$\begin{cases} x_{Bi} = C_f x_{Ai} + D_f y_{Ai} + K_f \\ y_{Bi} = C_g x_{Ai} + D_g y_{Ai} + K_g \end{cases}$$

b) Fit 3 parameters for rigid transformation (rot+trans):
angle and x,y-translations.

- **Automatically generate rigid transformation matrix**

```
python makerigid.py [angle, degrees] [tr. x] [tr. y]
```

- **Output file: sfacs.dat contains transformation parameters**

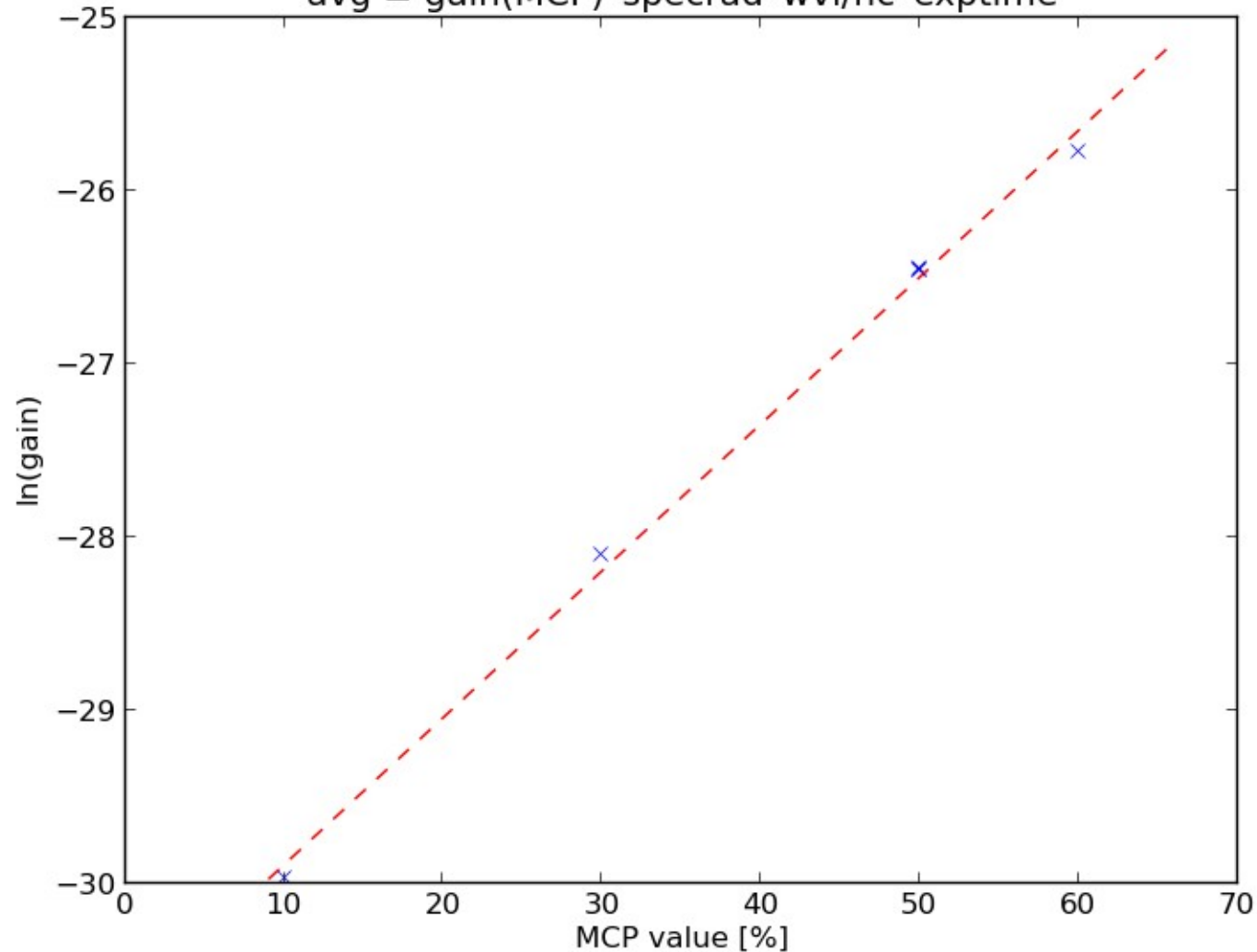
Check mask matrix and rotation angle by

```
python tiltangle.py
```

- **For next step, we need filter functions**
 - files: `cam1flt.dat` and `cam2flt.dat`
 - generate gaussians by
`python gaussgen.py`
- **Generate calibration parameters**
`python calibcurve.py [cam1] [cam2] [pulse]`
$$\text{gain(MCP)} = C \cdot e^{(k \cdot \text{MCP} + m)}$$
 - based on cf values in `calibs.dat`
 - parameters saved in file `cfacs.dat`
 - k and m fit to data
- **Custom calibration factor, C, selected by repeatedly prompting user until condition of continuous background is met**

KL11_E1DF (WI-400.8nm): $\text{gain}(\text{MCP}) = \exp(0.084851 \cdot \text{MCP} + (-30.739940))$

$\text{avg} = \text{gain}(\text{MCP}) \cdot \text{specrad} \cdot \text{wvl} / \text{hc} \cdot \text{exptime}$



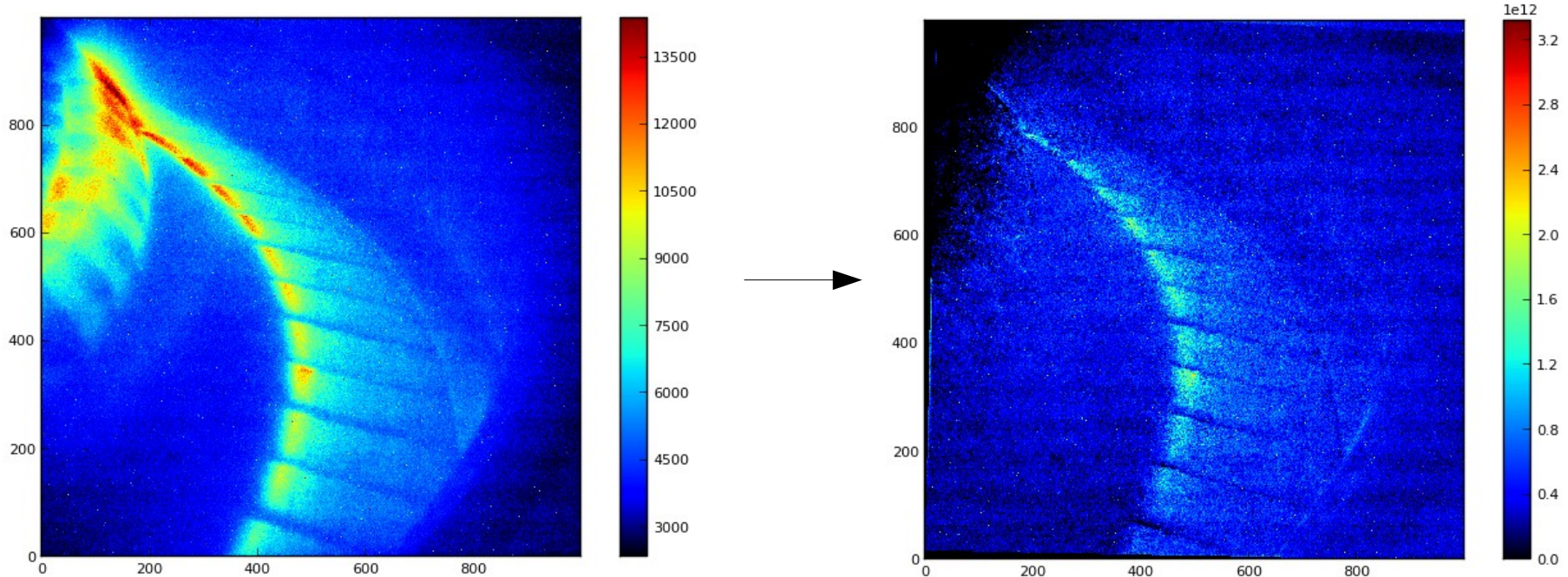
- **Contains the functions needed to produce a clean frame**
 - calibrate (single frame)
 - image_stretch (single frame)
 - getCalData (two frames, reads parameter files)
 - getClrData (based on output from the above)
- **For input/output lists of each function**
 - see lineflt user manual!

- **Run**

```
python showKL11_multi_main.py [cam1] [cam2] [pulse] [time]
```

- uses all previous files and functions.
- plots single frame through all stages of calibration.

- Result example 84782



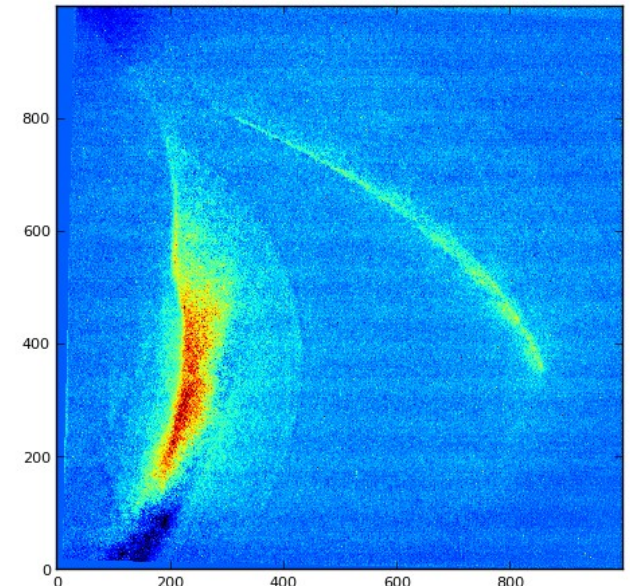
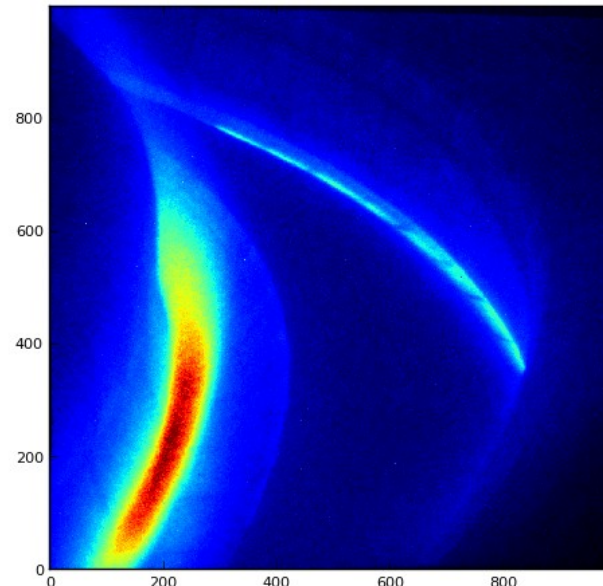
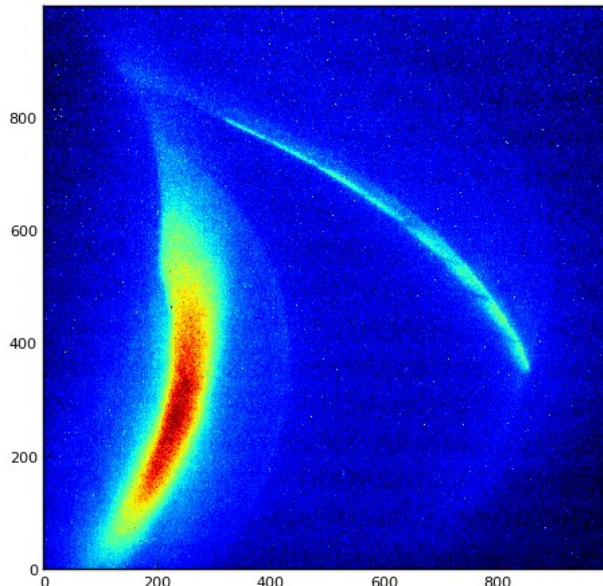
- It is very hard to find image transformation parameters such that the whole image fits well
- Certain regions tend to be inconsistent, i.e. qualitatively different between the two cameras (KL11-E1DE and KL11-E1DF)

- Example: 89463

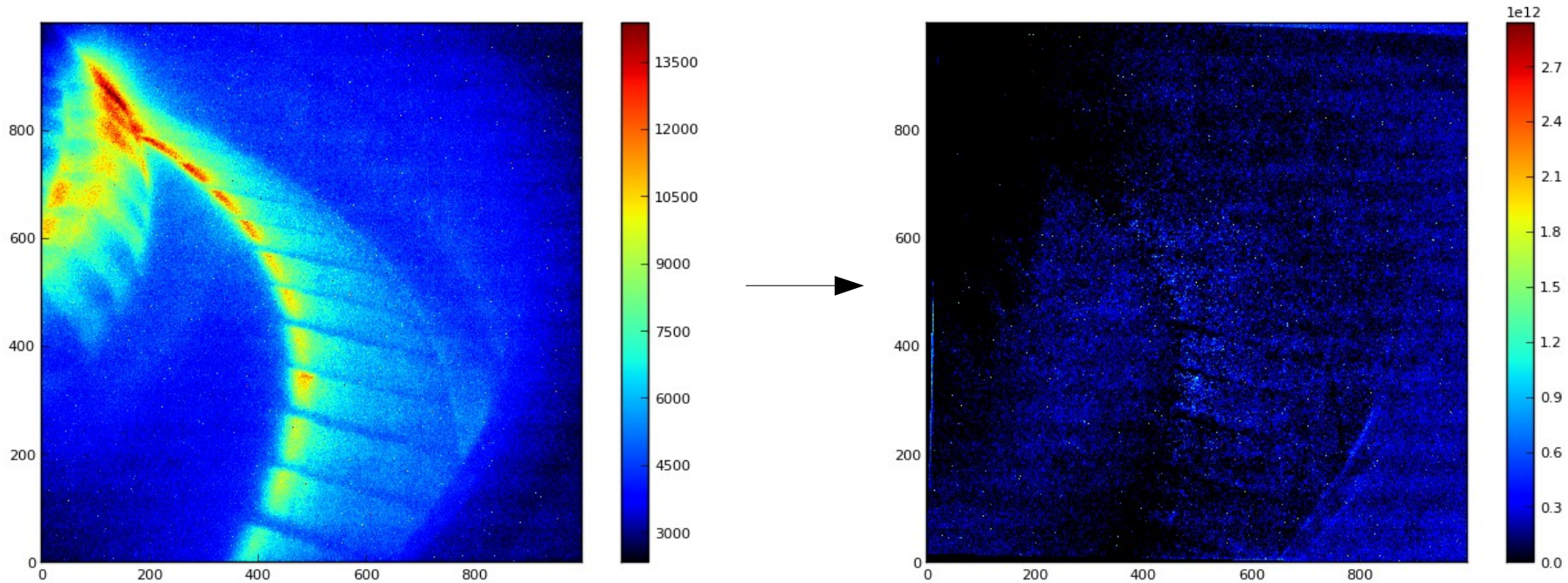
E1DE

E1DF

“Clean”



- The result is highly dependent on the arbitrary custom calibration factor!



- Detected via studying the background frame
 - Indicators of bad custom calibration factor:
 - Negative pixels
 - Structure (discontinuities)