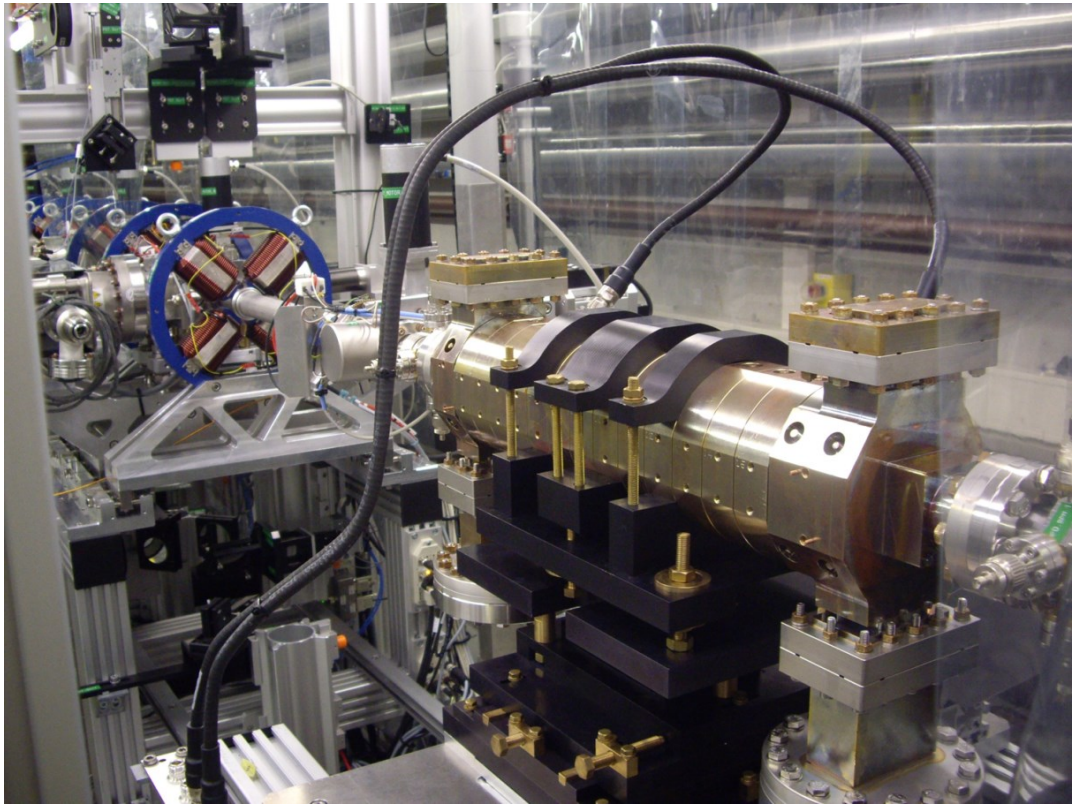
Image Filtering for PITZ

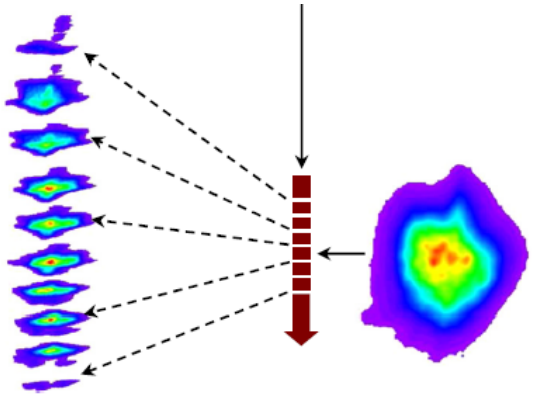
- Motivation: first SLEM@PITZ
- Systematic errors
- **Noisecut** methods
- Experimental results
- Conclusions & Outlook



Holger Huck
12.04.2018, PPS

Emittance Measurement: Slit Scan vs. Quad Scan

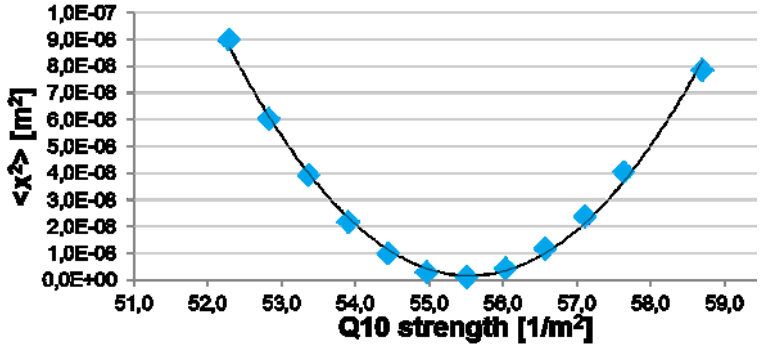
Slit scan



- Preferable technique at high space charge and low energy (PITZ case)
- **Significant intensity losses due to slit**

Both techniques can be done temporally resolved (SLEM) by using a vertically-deflecting TDS inbetween and scanning/analyzing horizontally…
…even more severe intensity losses!

Quad scan



$$\langle x^2 \rangle = R_{11}^2 \langle x_0^2 \rangle + 2R_{11}R_{12}\langle x_0 x_0' \rangle + R_{12}^2 \langle x_0'^2 \rangle$$

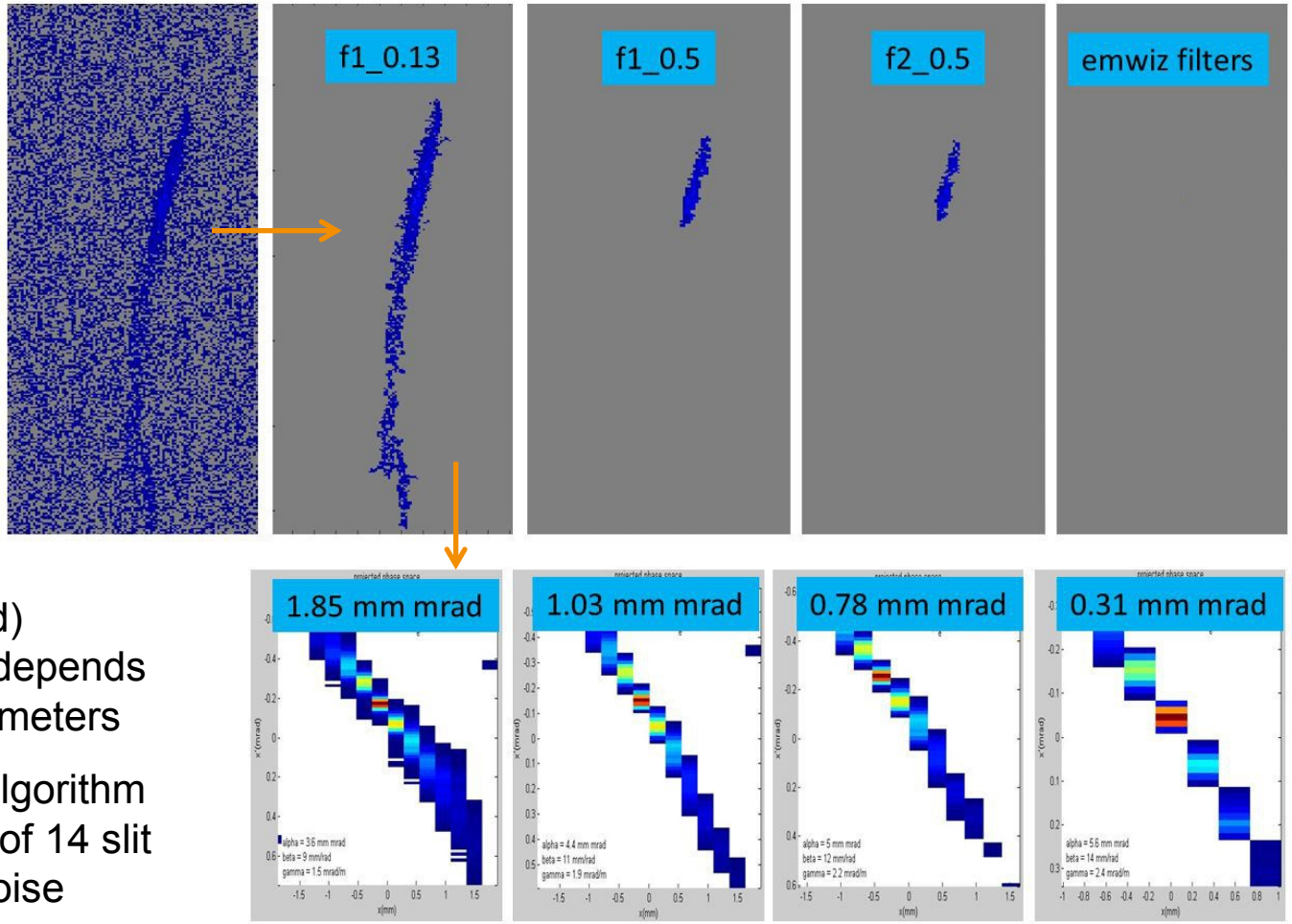
- Standard technique for **high energy** and **well known beam optics**
- **@PITZ we observed significant discrepancies between optics model(s) and optics measurements (i.e. effect of steerer kick)!**

Motivation: First SLEM Analysis (Slit Scan)

Example of slit-scan with **very low intensity**

(500 pC, TDS on, manual image taking & averaging x30)

$$\mathrm{avg}(img) - \mathrm{avg}(bg)$$

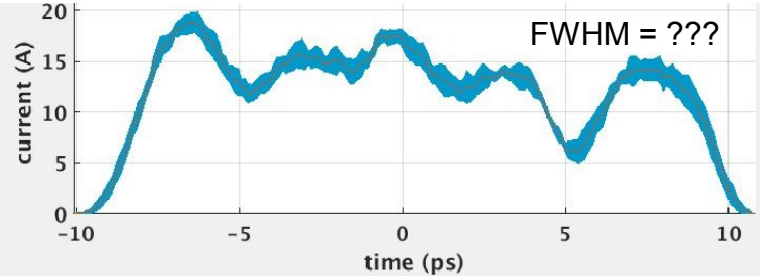


- Resulting (projected) emittance strongly depends on image filter parameters
- Standard „emwiz“ algorithm even regards 7 out of 14 slit positions as pure noise

Fit – FWHM – rms

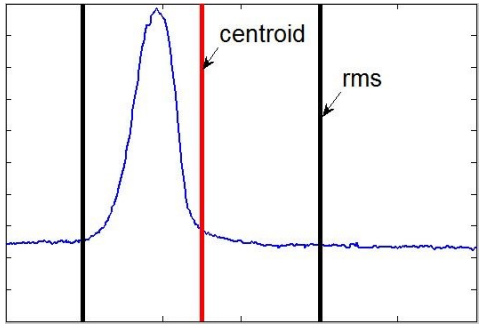
Fitting the distribution

- Noise hardly matters when fitting
- Systematic errors for Gaussian fits usually come from wrong baseline
- So just ensure proper background subtraction (or baseline offset as fit parameter)
- **Only works for well-known, esp. Gaussian distributions**
  - **…which we usually do not have at linacs**
  - **…and certainly not in slit scans**



Statistical methods

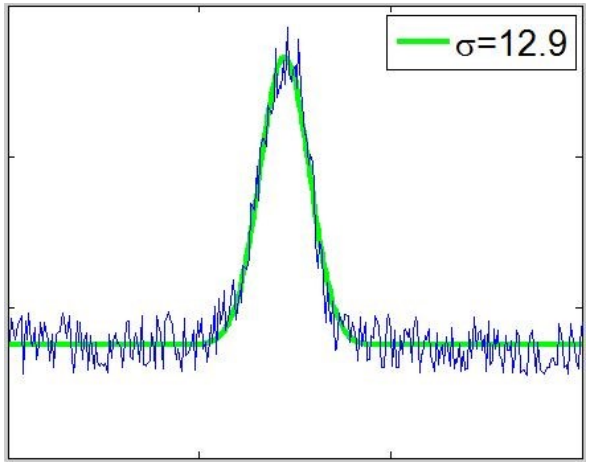
- FWHM and absolute maximum strongly depend on peak noise
  - **Should be combined with fit or smoothing**
- rms deviation and centroid strongly depend on noise (and background) outside the actual beam
  - **Noisecut important (i.e. use a mask of interest)**



☹

Background Subtraction

Correct baseline

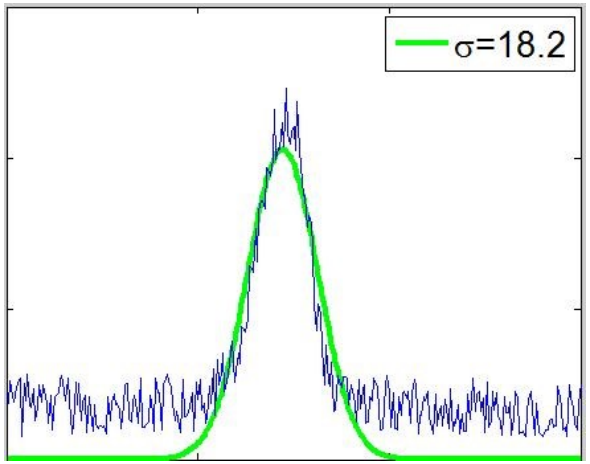


$$\text{avg}(img) - \text{avg}(bg)$$

~~$$\max(0, \text{avg}(img) - \text{avg}(bg))$$~~

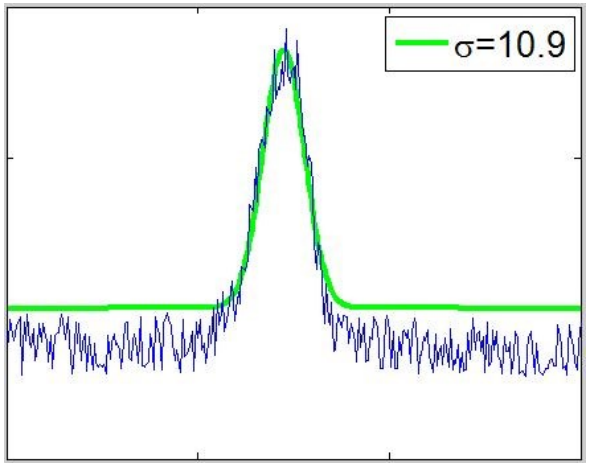
~~$$\text{avg}(img) - \max(bg)$$~~

~~$$\text{avg}(img) - (\max(bg) + 3\text{std}(bg))$$~~



Overestimation

- Forgot to subtract background
- Or didn‘t allow negative pixel values



Underestimation

- Too much background subtracted, e.g…
- „envelope“ (=max)
- „env+3rms“

Why do we need noisecut…

…shouldn‘t pos./neg. noise cancel out with correct baseline?

- Simulation: 50 shots of the same signal (2d Gauss) within a large image of random noise (Gaussian distributed, mean=0)
- Calculate rms and centroid as function of ROI size

$$S2N = \max(signal) \,/\mathrm{STD}(noise)$$

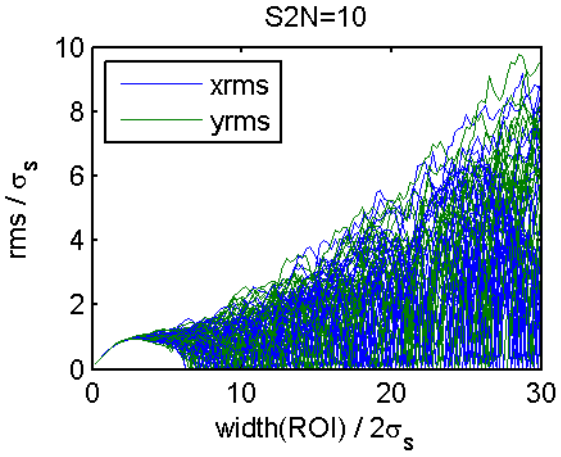
σ_S = signal width (“beam size”)

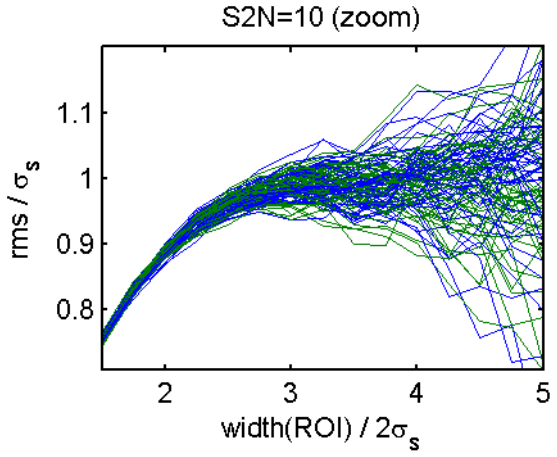
$$dEX = centroid - EX$$

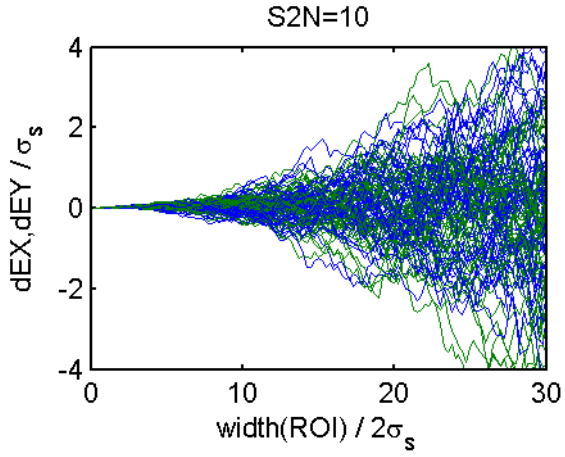
(„3000“ criterion at PITZ translates to an S2N of 50 – 100)

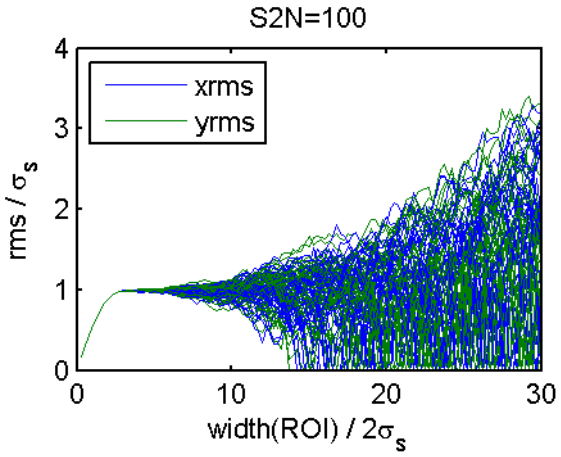
Why do we need noisecut…

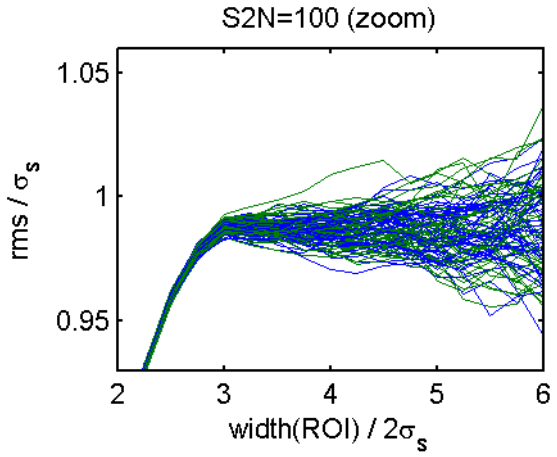
…shouldn‘t pos./neg. noise cancel out with correct baseline?

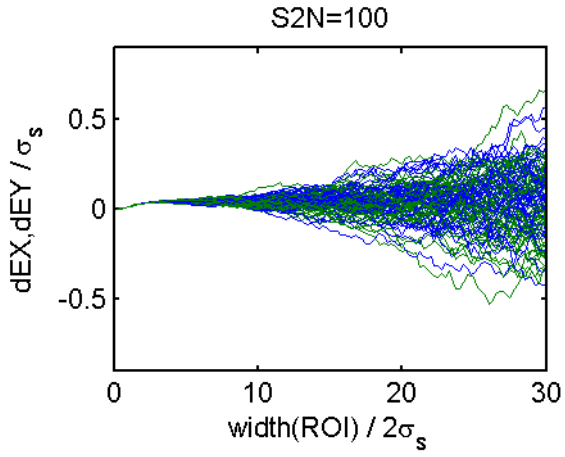










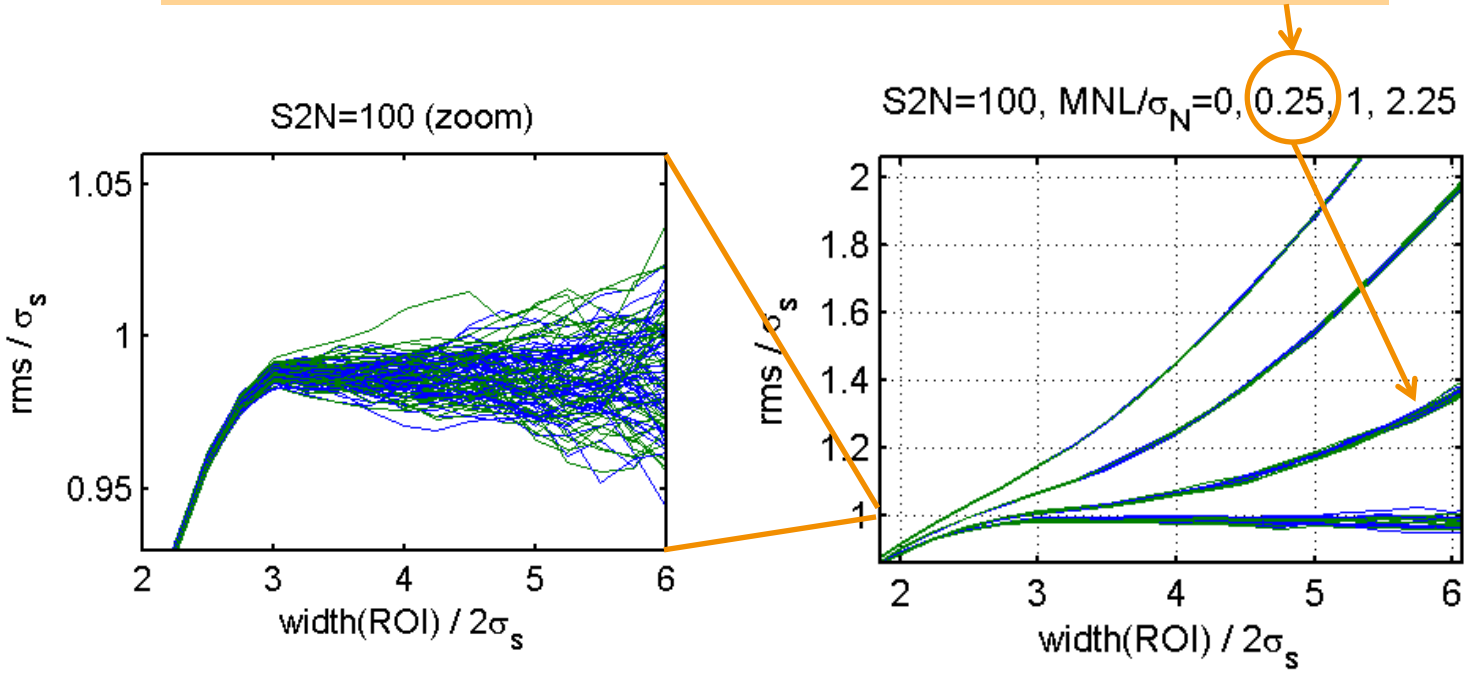


No, not even ideal, homogenous Gaussian noise cancels out!

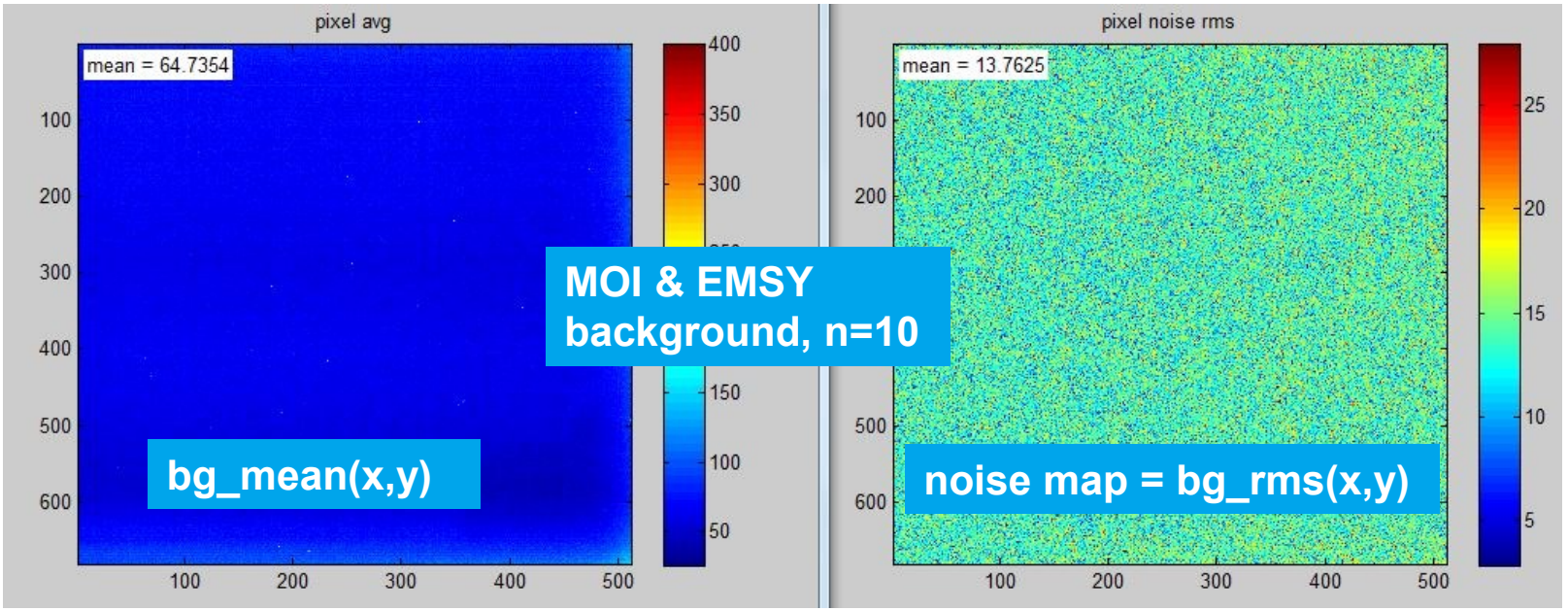
To keep errors <5%, we need a ROI of 3-6 sigma @S2N~100

Why do we need noisecut…

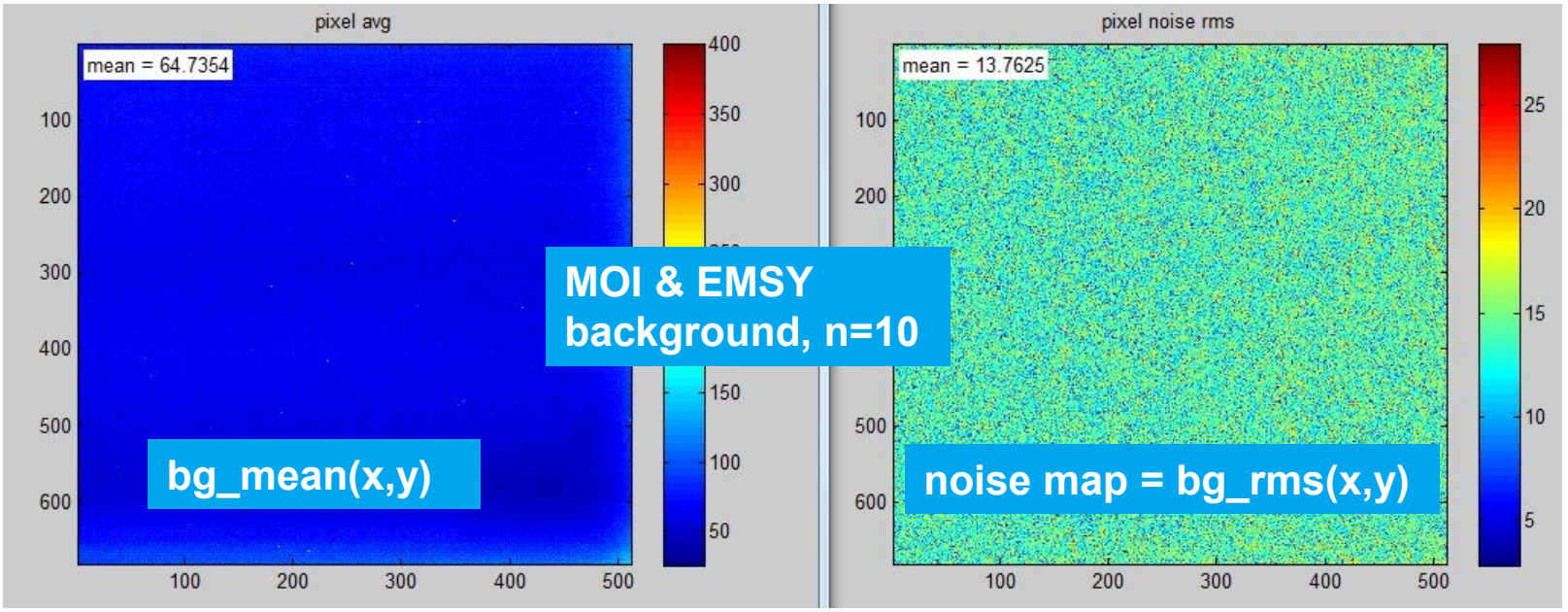
effect of incorrect baseline (mean noise level > 0)

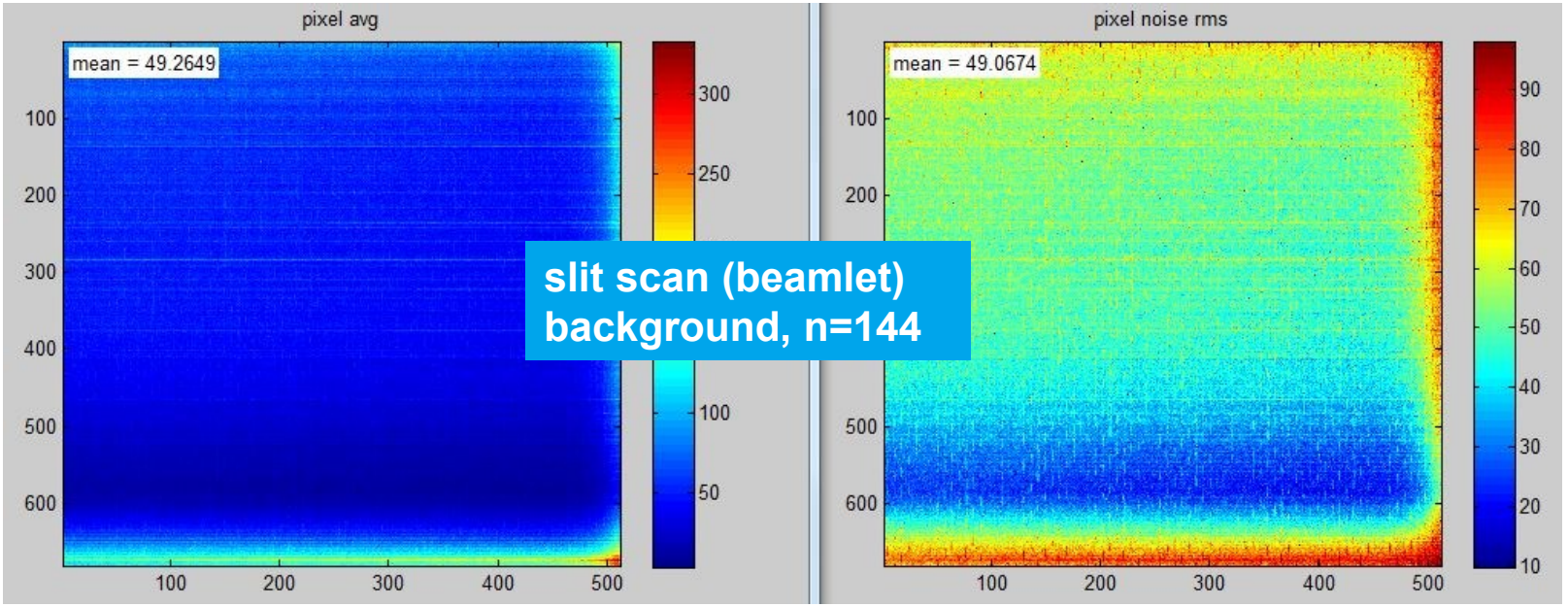


Noise inhomogeneity

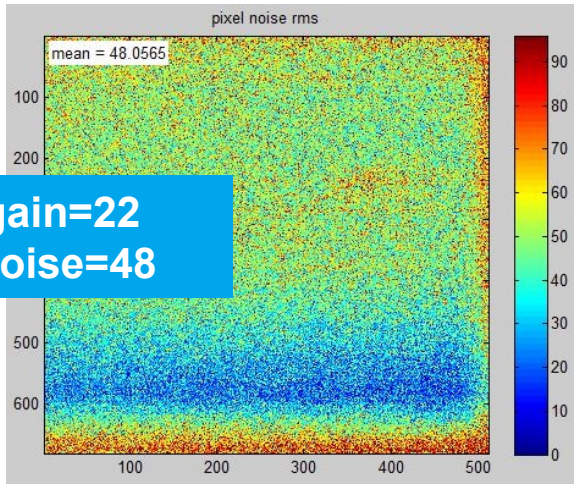


Noise inhomogeneity

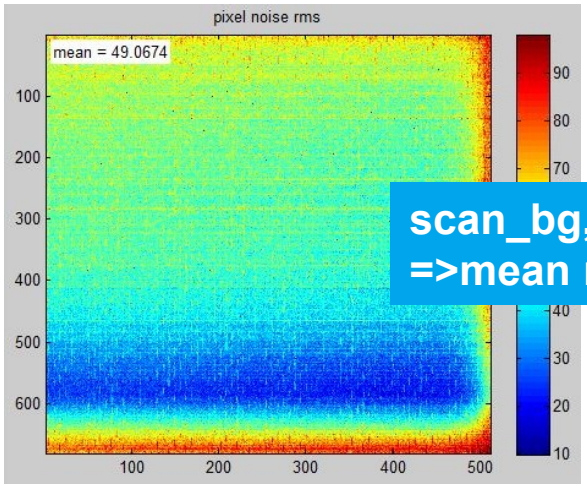




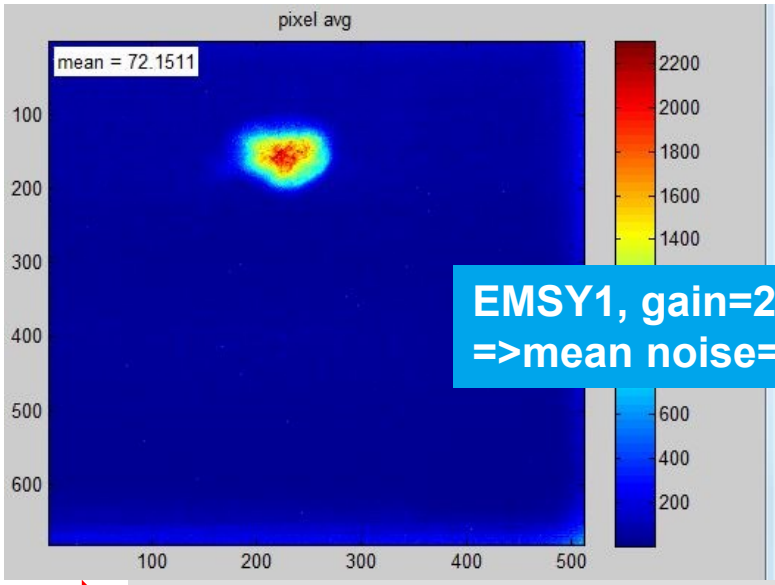
Noise inhomogeneity

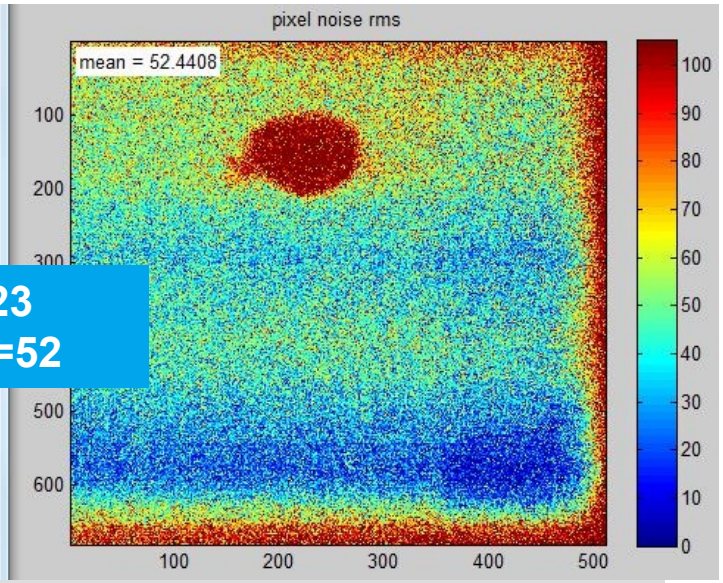


MOI_bg, gain=22
=>mean noise=48



scan_bg, gain=22
=>mean noise=49





EMSY1, gain=23
=>mean noise=52

Noise depends on gain and on intensity!

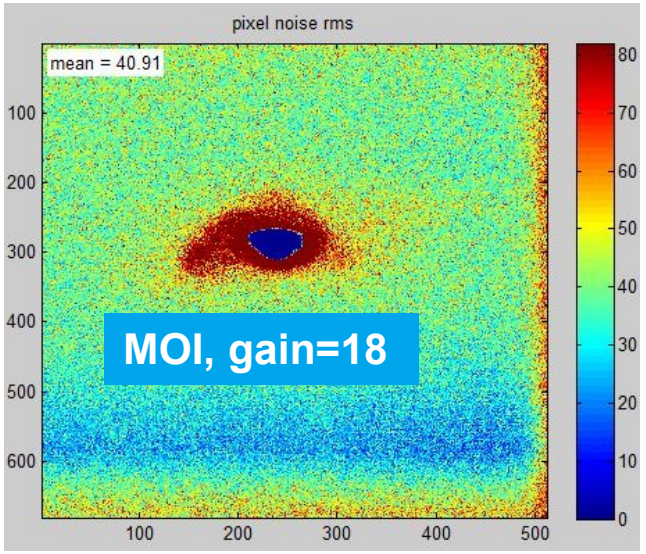
>> mean noise ~2x“gain“ (at least for YAG@H1S1,S4,PST1)

>> local (signal) noise ~2x mean noise

Noise inhomogeneity

Conclusions

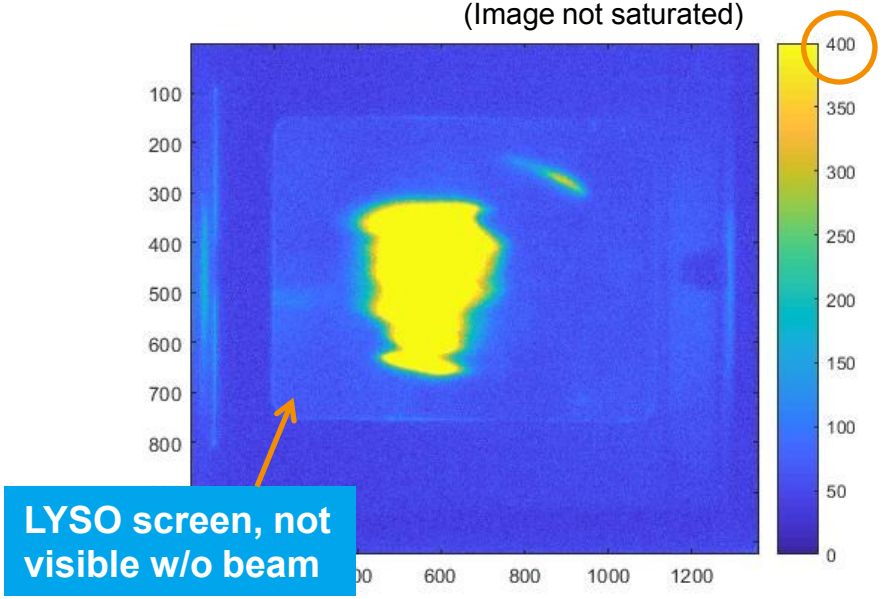
- PITZ screen stations (YAG+ProSilica) exhibit strong noise inhomogeneity at higher gain
  - Open questions: Does sum-of-pixels (beam signal) depend on screen position as well? Should we exchange cameras?
  - Update: acc.to SW, new cameras do not have this problem!
- Noise depends on gain and intensity
- Mean noise level roughly „2.2x gain“, independent on screen and #images (10<n<450)
- local (signal) noise ~2x mean noise
- Fabric / raindrop structure on all slitscan bg_rms(x,y) images, but not on MOI / EMSY
  - Open question: due to high n or slit movement?



Background issues

- **All** analyzed slit scans yield **larger** emittance values when cutting less noise (should be randomly larger and smaller)
- Reason might be background being not independent on beam signal, which can have many causes:
  - Real beam halo
  - Optics: stray light
  - Screen: scattering
  - CCD: bleeding
- Very obvious on HTR @LYSO images
  - Need to check new LYSO for this effect!

(Image not saturated)



- Motivation: first SLEM@PITZ
- Systematic errors
- **Noisecut** methods
- Experimental results
- Conclusions & Outlook

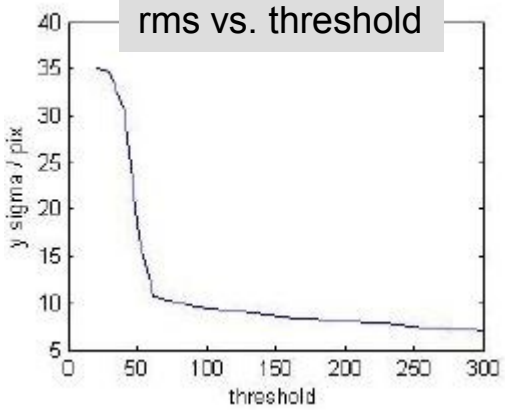
- CK
- emwiz / emcalc
- TDS_tool
- MYan
- HH

ck_centroid2d*

1. Find maximum of image
2. Find the region of connected pixels around maximum that are above threshold. Crop.
3. (optional) Check if region has enough pixels in them. If not: set region to zero and start over
4. Calculate Centroid

- Very fast
- No smoothing
- Threshold=?

rms vs. threshold



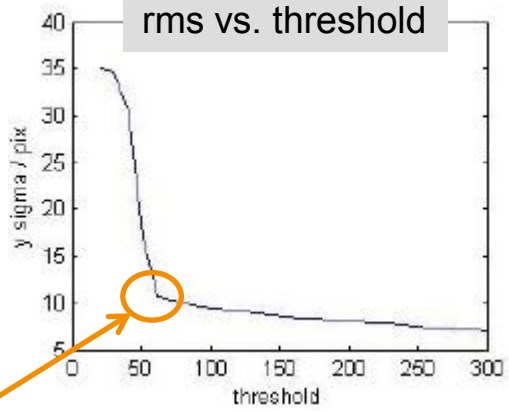
*C. Koschitzki,
A coherent definition for the width and position of 2 dimensional objects and how to calculate them introducing threshold masked centroids, PPS 29.08.2017

ck_centroid2d*

1. Find maximum of image
2. Find the region of connected pixels around maximum that are above threshold. Crop.
3. (optional) Check if region has enough pixels in them. If not: set region to zero and start over
4. Calculate Centroid

- Very fast
- No smoothing
- Threshold=?

rms vs. threshold



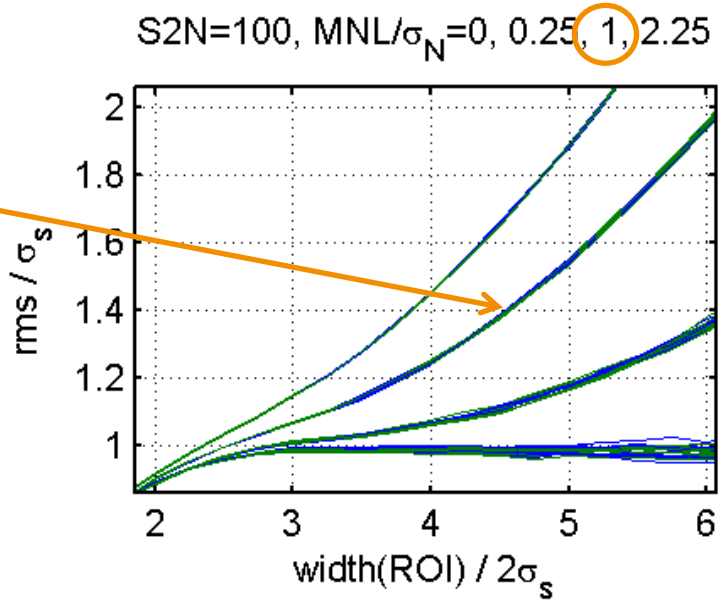
"Performance depends ~~slightly~~ on threshold. It can however be considered consistent for a set of data analyzed with the same threshold", same s2n and same signal shape.

- Usually NOT a clear corner
- Arbitrary definition of correct threshold, rms, emittance…
- Problem of ALL methods!

*C. Koschitzki,
A coherent definition for the width and position of 2 dimensional objects and how to calculate them introducing threshold masked centroids, PPS 29.08.2017

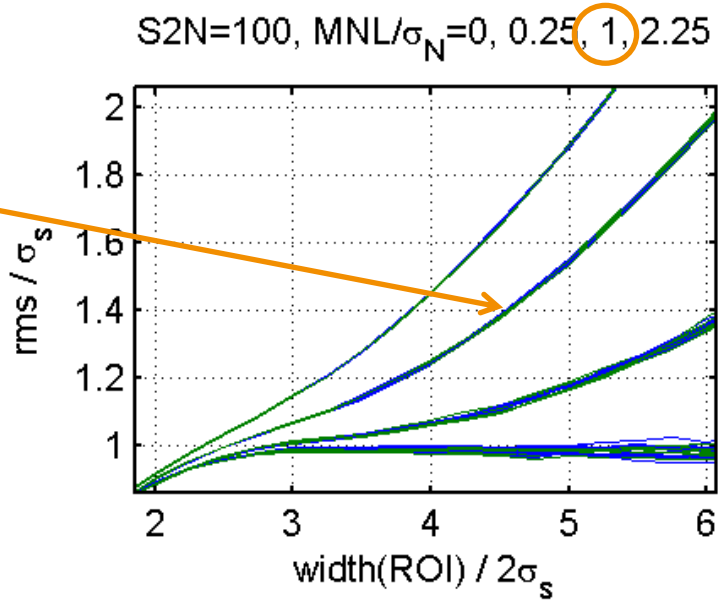
Why we need smoothing…

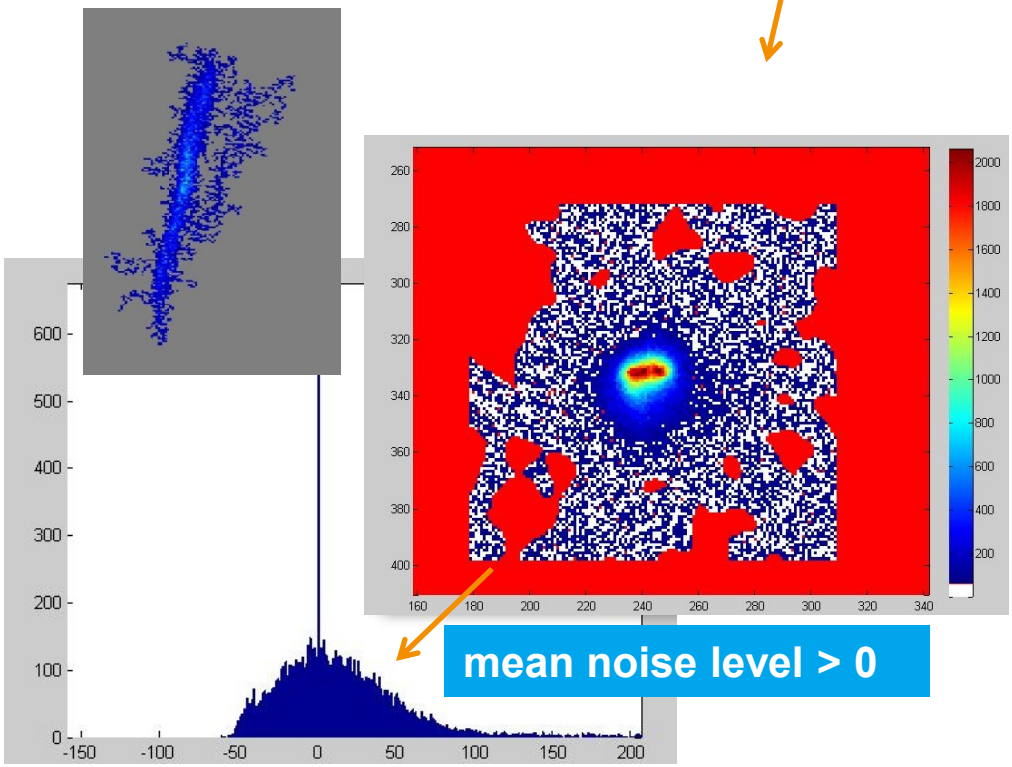
- Cutting noise simply at a threshold pixel value favors positive noise pixels!
- Extreme example: Threshold = 0 has same effect as not allowing negative pixel values
- Solution: Create a mask of interest and restore original pixel (incl. negative) values inside



Why we need smoothing…

- Cutting noise simply at a threshold pixel value favors positive noise pixels!
- Extreme example: Threshold = 0 has same effect as not allowing negative pixel values
- Solution: Create a mask of interest and restore original pixel (incl. negative) values inside
  - But mask outline must NOT depend on local noise levels!



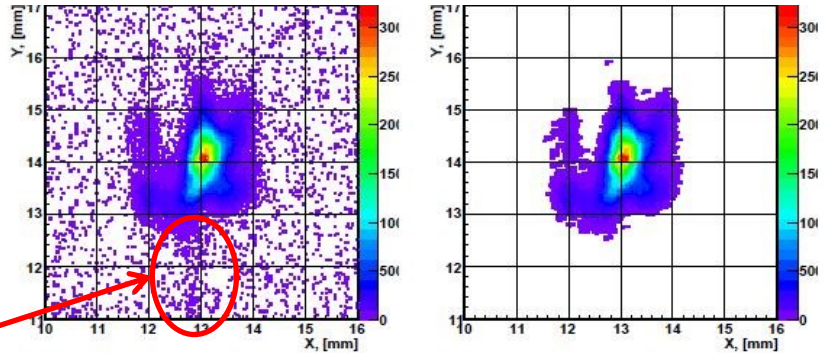


- Smoothing before cutting
- Identify main island(s)
- Generally work on small ROI

emwiz / emcalc

1. *raw(x,y) = avg(img)-avg(bg)*
2. *bg_rms(x,y)* = pixel-wise STD of bg images
3. Noisecut: *raw(x,y) < 1 * bg_rms(x,y)* ? => pixel = 0
4. „Smoothing“: any neighbor == 0 ? => pixel = 0
   - For MOI/EMSY 3 instead of 8 neighbors
5. Around 1 pixel of what remains after (4.), restore *raw(x,y)*
   - For MOI/EMSY 7x1 L-Shape instead of 8 neighbors

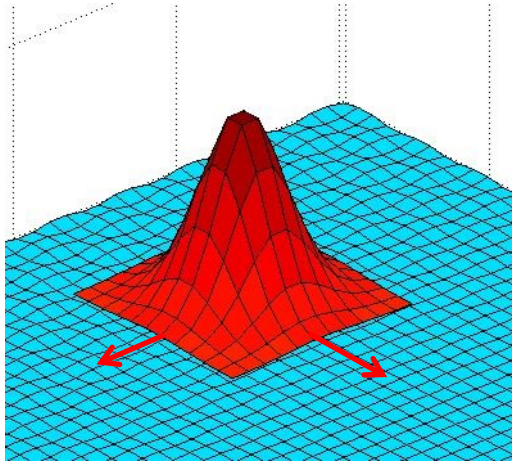
- Very fast matlab version is now available (different MOI, see p21)…
  - 5 sec loading fastscan data (150 frames)
  - 10 sec filtering & drawing & Twiss calculation
- MOI issues
  - L-Shape questionable
  - standard MOI might still cut too much from beamlets due to much lower camera gain
- Completely eradicates thin or noisy structures (no matter the intensity!)
- Consistent emittance definition through s2n control („3000-criterion“)



L. Staykov, PhD thesis, Hamburg 2008, p.125 (App.A)

Noisecut2.m (by BB) / TDS_tool

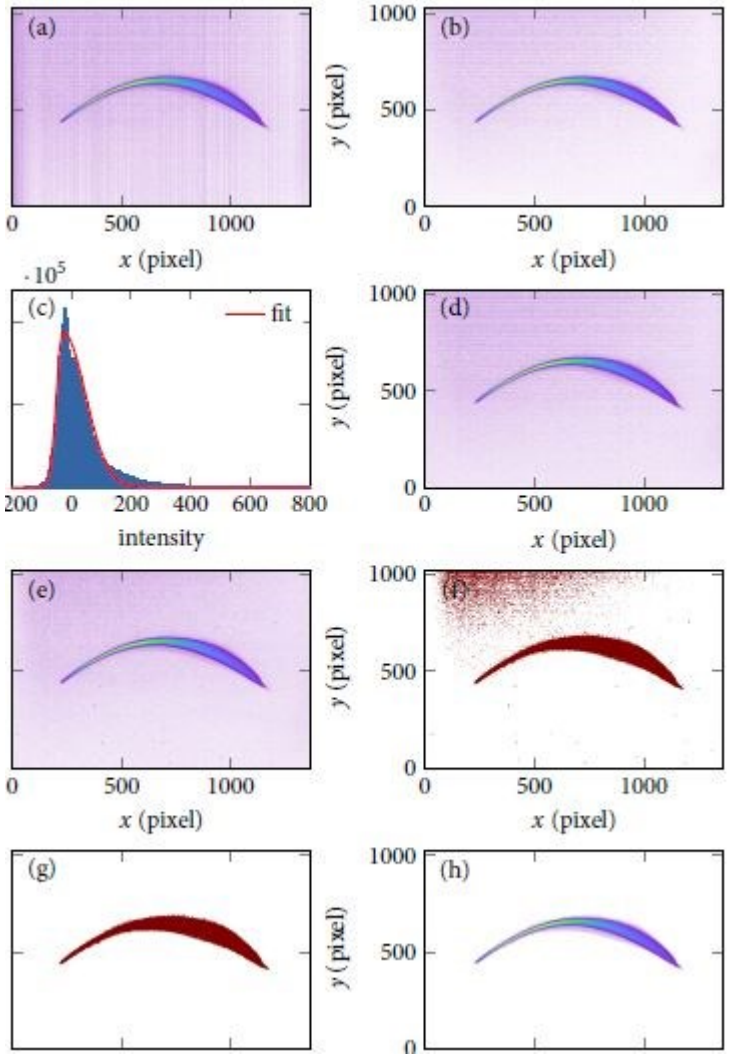
1. (assumes: *raw(x,y) = avg(img)-avg(bg)*)
2. Determine beam size by (Gauss-) fitting x- and y-profile, ***s***=max(sx, sy)
3. Smooth/blur image (convolute *raw(x,y)* with 2D-Gaussian filter of size ***s***)
4. Gauss fit of histogram of original image (assumes Gaussian noise distribution, and noise area >> beam)
5. Define noisecut threshold ***t*** relative to (4.)
6. *mask(x,y) = (blur(x,y) < **t**)*
7. Keep only the one connected area with the largest pixel value in *mask(x,y)*
8. Restore *raw(x,y)* inside mask, set to 0 outside
9. subtract mean noise = avg(*raw*(outside mask))



- universal approach…
- …that often fails
- many fits, very slow
- parts unused/unfinished

MYan*

1. (b) = (a) – background
2. (c) = Split-Gauss fit of histogram of (b), to get peak position μ_0
3. (d) = (b) - μ_0
4. (e) = Smooth (d) with 7x7 Gaussian filter
5. Split-Gauss fit of histogram of (e) => μ, σ
6. (f) = ((e) < $(\mu + 2\,\sigma)$)
7. (g) = Keep only the one connected area with the largest pixel value in mask (f)
8. (h) = Restore (b) inside mask, set to 0 outside



*Minjie Yan, PhD thesis, Hamburg 2015, p.143-144 (App.D)

- refined version of noisecut2
- (2,3) have no effect
- assumes homog.BG
- threshold still arbitrary

HH („f4_1“ filter)

1. Automatic rectangular ROI from MOI.imc with generous smoothing (~75 pixel running average), also cut screen edges
2. *raw(x,y)* = *avg(img)-avg(bg)*, ROI sized
3. *bg_rms(x,y)* = pixel-wise STD of bg images
4. Smooth/blur image (convolute *raw(x,y)* with 2D-Gaussian filter of size ***s***)
5. *mask(x,y)* = *(blur(x,y) < **t** * bg_rms(x,y))*
6. Keep only the ***n*** connected area(s) with largest pixel value in *mask(x,y)*
7. Restore *raw(x,y)* inside mask, set to 0 outside
8. Optionally: subtract mean noise level (i.e. avg(img outside ROI)) (effect <<1% with proper bg)
9. Optionally: apply median filter to remove salt-and-pepper noise (effect ~1%)

- > very fast (no fits, no x/y loops, uses ROI)
- > safe MOI/ROI size
- > smoothing
- > threshold based on local noise levels

Preliminary name of filter: „f**s**_**t**“ (e.g. f4_1)

Comparison of noisecut methods

	Description	PRO	CONTRA
CK	ck_centroid2d	fast	No smoothing
emwiz	Used in fastscan3	fast, PITZ standard, uses bg_rms(x,y)	Cuts too much
TDS_tool	Noisecut2.m (BB)		Slow; often fails for small/asym. beams; parts unfinished/unused
MYan	Advanced version of noisecut2 (XFEL, PSI)		Slow, and assumes homogenous bg
HH	f4_1 promising candidate for new PITZ standard?	fast, tunable, uses bg_rms(x,y)	No fixed settings that work for all s2n (true for all noisecut methods)

- Motivation: first SLEM@PITZ
- Systematic errors
- **Noisecut** methods
- Experimental results (filtering first SLEM data)
- Conclusions & Outlook

Slit scan + TDS @PITZ: First Setup

- Setup: EMSY2 slit, 5 m drift length to PST.Scr1 with TDS inbetween
- Manual slit scan in order to average 10 images at each slit position for higher signal-to-noise ratio
  - *fastscan3* **and** *emcalc* **do not support image averaging at discrete slit positions!**
  - „manual“: image taking using video client, manual slit positioning
- Matlab tools written for offline image filtering and analysis
  - Main idea: cut beamlet images in vertical slices and analyse those separately
- 500 pC, 1 nC, short and long Gaussian laser pulses
- 10-30 slit positions, 50 µm slit
- Also fastscans and manual scans w/o TDS done

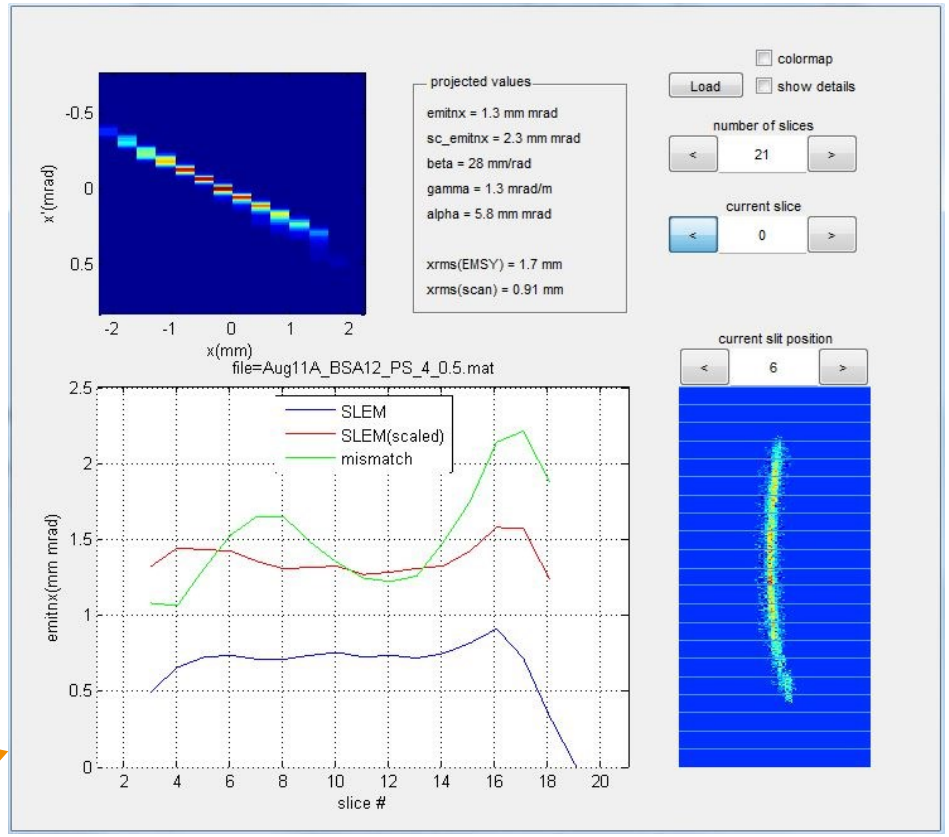
Matlab tools

1. SlitScanner.m (measurement)

- Move slit, acquire & save n images at each slit position
- Also write text/log file understandable by SlitScan.m
- WIP...: improve GUI, combine with SlitScan, final goal: obsolete fastscan & emcalc

2. SlitScan.m (image filtering)

- Load fastscan or manual scan data
- Various tunable filter methods
- Calculate projected PS & emittance
- „auto“ button for 1-click analysis of full scan
- Save filtered images (for SlemCalc.m)



3. SlemCalc.m (SLEM analysis)

- Load filtered images from SlitScan.m
- Tunable vertical slicing
- GUI shows PS, SLEM & mismatch to projected values, also filtered images

$$mismatch = (\beta_0\gamma - 2\alpha_0\alpha + \gamma_0\beta)/2$$

Results of first experiments (selection)

Setup		emcalc	f_emwiz	f4_2	f4_1	f4_0.5	f2_0.25	f2_0.13	emcalc	f_emwiz	f4_2	f4_1	f4_0.5	f2_0.25	f2_0.13
500pC (#1)	fastscan	1	1,03						5,64	5,8					
	TDS off		0,88		1,06	1,4	1,76			4,33		5,3	6,37	7,62	
	TDS on		0,3			0,42	1,28	1,75		1			1,32	2,41	3,15
	cSLEM		0,2			0,3	0,6	0,8							
500pC (#2)	fastscan	1,9	1,85		2,26		2,95		2,3	2,36		2,54		3,02	
	TDS off		1,6		1,7	1,76	1,76	1,9		2,12		2,24	2,42	2,25	2,48
	TDS on		0,6		0,55	1,29	1,83	2,2		1,2		1,23	2,22	2,82	3,31
	cSLEM		0,37		0,5	0,75	0,8	1,2							
500pC (#3)	fastscan	1,49	1,46		1,64		1,88		1,52	1,53		1,69		1,92	
	TDS off		1,65		1,76	1,82	1,84	1,94		1,75		1,87	1,88	1,82	1,96
	TDS on		0,81		0,82	1,36	1,65	2,04		1,13		1,24	1,67	1,91	2,1
	cSLEM		0,5		0,6	0,8	1	1,2							
1nC SG BSA2.4	fastscan1	2,9							3,1						
	fastscan2	1,7							2,6						
(12 ps@TDS)	TDS off		2,19	2,03	2,28	2,6				3,03	2,94	3,2	3,46		
Q5/6=3.3/-4.1	TDS on+		1,44	0,87	1,56	2,13				2,16	1,55	2,4	3,02		
n_slit=10	TDS on-		1,29	0,7	1,44	2,28				2,07	1,47	2,3	3,17		
	cSLEM+		1	0,7	1,2	1,7									
	cSLEM-		0,9	0,5	1,2										
1nC SG BSA2.2	fastscan	1,8							2,3						
Q5/6=+4/-4	TDS off		1,64	1,51	1,91	2,23				2,58	2,55	2,89	3,14		
	TDS on+		1,28	0,77	1,52	2,11				2,01	1,68	2,38	2,95		
n_slit=30	TDS on-		1,26	0,83	1,5	2,03				2,05	1,68	2,43	2,85		
	cSLEM+		1,1	0,7	1,4	1,8									
	cSLEM-		1,2	0,8	1,4	1,8									

Conclusions

- Manual slitscan with 10-30 slit positions work fine
- Image averaging at fixed slit positions works fine (better s2n)
- Adjusted FFT/noisyframe filter works fine
- emcalc_filter.m works fine with good s2n
  - (and very fast, 15s for 150 beamlets)
- f4_1 filter looks promising as new standard noisecut filter
  - slightly slower (+1s) (+1s for median filter)
  - slightly higher emittance values (~10-20%)
- Slicing of filtered images works fine
  - non-scaled center slice emittances are, as expected, roughly between 50% and 100% of the non-scaled projected emittance
- Systematic errors from finite slit width under investigation, probably not critical

- When turning on TDS, intensity dropped so much that even with image averaging s2n was too bad for emcalc and f4_1
- Other filters, e.g. f2_0.25 might be used here, but results are very sensitive to noise-cut options, thus questionable
- PITZ EMSY scaling helps to reduce discrepancies, but not enough
- Must not have waist/focus between slit and screen

Outlook

- LYSO@PST.Scr1 finally installed!
  - Will enable lower charge measurements and smaller slit (10 vs. 50 µm)
  - Better signal-to-noise ratio, robust filtering
  - Larger scan range
  - Maybe we can even use continous fastscan (no manual avg.)
  - But also need to check LYSO resolution and stray light (background vs. signal)
- Already during gun conditioning, spend 1-2 shifts each week for testing & bug fixing
- Combine the 3 matlab tools, shrink & speed up saving/loading
- More extensive measurement GUI (might fully replace emwiz suite…)
- ? Find reasonable noisecut threshold as function of s2n?
- ? Define a „core SLEM“ (charge-cut in 3D)? instead of arbitrary „100%“ claim?
- Systematic scans (laser shape, solenoid, charge, BSA,…thesis RN)

SoP(EMSY) preservation in solenoid scan?

EMSY1	378 A	379 A	380 A	381 A	382 A	383 A	384 A	385 A	386 A	387 A	388 A
gain	5	5	5	7	9	11	14	18	22	21	22
NoP	1	1	1	1	1	1	1	1	1	2	2
ewiz_8	1,05E+06	1,06E+06	1,04E+06	1,27E+06	1,53E+06	1,94E+06	2,74E+06	4,16E+06	6,37E+06	1,08E+07	1,05E+07
emwiz_L	1,06E+06	1,07E+06	1,05E+06	1,28E+06	1,54E+06	1,97E+06	2,77E+06	4,20E+06	6,44E+06	1,09E+07	1,06E+07
4_2	1,05E+06	1,06E+06	1,04E+06	1,26E+06	1,52E+06	1,93E+06	2,73E+06	4,15E+06	6,36E+06	1,08E+07	1,05E+07
4_1	1,06E+06	1,07E+06	1,05E+06	1,28E+06	1,55E+06	1,97E+06	2,78E+06	4,22E+06	6,47E+06	1,10E+07	1,06E+07
4_0.5	1,07E+06	1,09E+06	1,07E+06	1,30E+06	1,57E+06	2,01E+06	2,82E+06	4,27E+06	6,57E+06	1,11E+07	1,08E+07
4_0.25	1,08E+06	1,10E+06	1,08E+06	1,32E+06	1,60E+06	2,05E+06	2,88E+06	4,34E+06	6,67E+06	1,13E+07	1,09E+07
4_0.13	1,09E+06	1,11E+06	1,09E+06	1,34E+06	1,61E+06	2,10E+06	2,93E+06	4,41E+06	6,77E+06	1,17E+07	1,12E+07
1_1	1,06E+06	1,07E+06	1,05E+06	1,28E+06	1,55E+06	1,97E+06	2,77E+06	4,21E+06	6,46E+06	1,10E+07	1,06E+07
2_1	1,06E+06	1,07E+06	1,05E+06	1,28E+06	1,55E+06	1,97E+06	2,77E+06	4,21E+06	6,46E+06	1,10E+07	1,06E+07
8_1	1,07E+06	1,08E+06	1,06E+06	1,29E+06	1,56E+06	1,99E+06	2,79E+06	4,24E+06	6,51E+06	1,10E+07	1,07E+07
sum	1,06E+07	1,08E+07	1,06E+07	1,29E+07	1,56E+07	1,99E+07	2,80E+07	4,24E+07	6,51E+07	5,53E+07	5,35E+07
vs_B14		101,47%	99,58%	1,213775	1,466127	1,872224	2,630693	3,98918	6,122695	5,199473	5,031991
sum/dB21.5	6,22E+06	6,31E+06	6,20E+06	6,10E+06	5,94E+06	6,13E+06	6,24E+06	6,17E+06	6,17E+06	5,83E+06	5,07E+06
vs_B16		1,014678	0,995766	0,979749	0,955268	0,984664	1,00338	0,991362	0,991391	0,937073	0,814783

82% of expected value

- For the three gain=5 measurements, everything stays within a few percent fluctuation
  - Accuracy of method not good enough to prefer one filter over another
  - But it shows that for high s2n filter choice does not matter!
- Meaning of gain parameter unclear…measurements suggest it is not dB (10*log10), but close to 21.5*log10 (or 10*ln?)
- SoP(2 pulses) = 0.82*2*SoP(1 pulse)
  - Operator mistake? Wrong gain values?

FFT/noisyframe filter

```
% a is bad frame if rating > 0
% offset adjusts rating
function rating = bad_frame( a, offset )
if ~exist('offset','var')      %noise threshold (pixel value)
   offset = 0.0;
end
        proj = (sum(a,find(size(a) == max(size(a))))); %proj
        fftproj = log10(abs(fftshift(fft(proj)))); %get fft

%         if fftproj(min(size(a))/4+1) > mean(fftproj)
%             rating = 1;
%             disp('noisy frame')
%         end

        f0=fftproj(min(size(a))/4+1);
        f=mean(fftproj);
        rating = -offset+(f0-f)/(f0+f);
```

old code (getimage.m), often removes ~30% of all images

- new code (bad_frame.m) uses normalized FFT-rating and allows offset
- works fine for 2x2 binning (PITZ standard) and offset of 5%
- other binnings might need other offset
- …should we integrate this to getimage.m?

Filtering out spatial noise

Compared for High2.Scr1, only background (no real signal)

- Current implementation (by James): log10(FFT of 1-d projection of image)
  - if the amplitude of the noise frequency higher than mean – noisy frame

```
if fftproj(129) > mean(fftproj)
```

  - Holger mod.: if normalized value larger than an offset – noisy frame

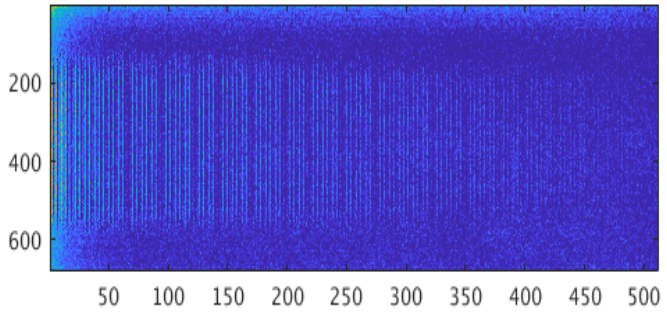
$$\text{if } \frac{\texttt{fftproj(129)-mean(fftproj)}}{\texttt{fftproj(129)+mean(fftproj)}} > \texttt{offset}$$

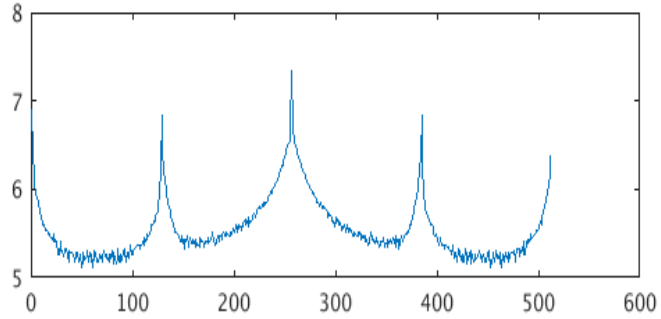
- Previous filter (also by James): log10(2-d FFT projection of image)
  - If the amplitude of noise frequency is higher than mean + 4σ – noisy frame

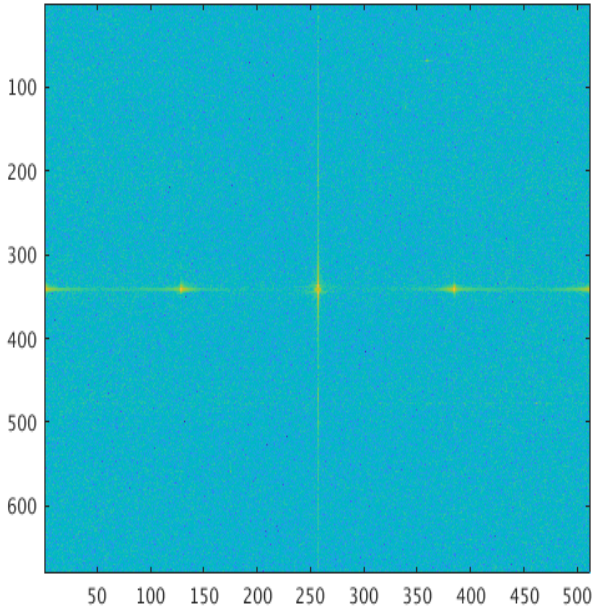
```
if val > Jmean+(4*Jstd)
```

  - Slow on full resolution images

1000 frames	1-d (James)	1-d (Holger)	2-d (James)
2x2 binning	23	3	22
Full res.	613	38	38





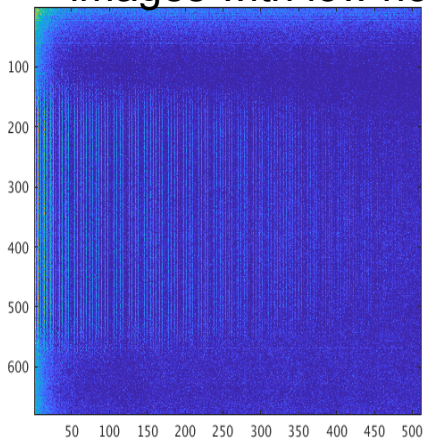


Filtering out spatial noise

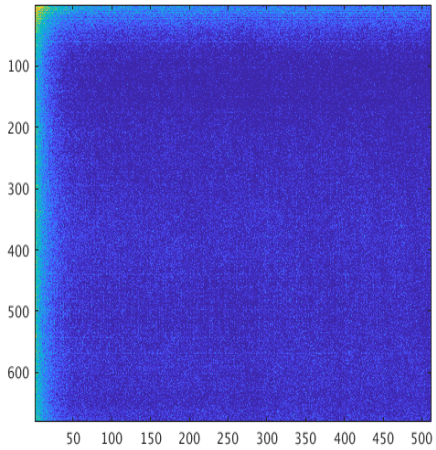
1000 frames	1-d (James)	1-d (Holger)	2-d (James)
2x2 binning	23	3	22
Full res.	613	38	38

Conclusion

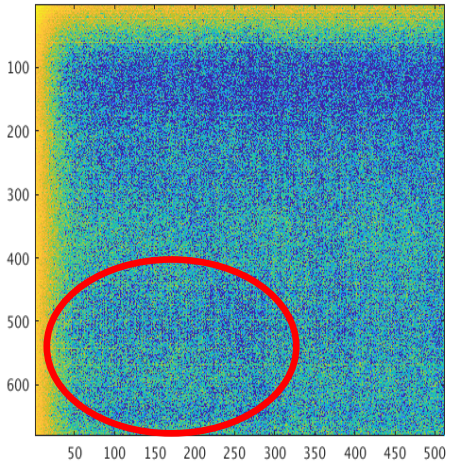
- For 2x2 images, Holger's mod filters out images with a strong noise, however keeps images with low noise



Strong noise



Low noise



Low noise (log scale)

- For Full res. Images, Holger's mod filters the same frames as 2-d FFT (some are really low noise), while being much faster. 1-d implementation by James does not work properly
- Tests on other cameras needed
- Tuning of the offset parameter?
- If agreed on the filter – a week by week degradation monitoring of cameras could be included into the start up checklist (a script is needed)