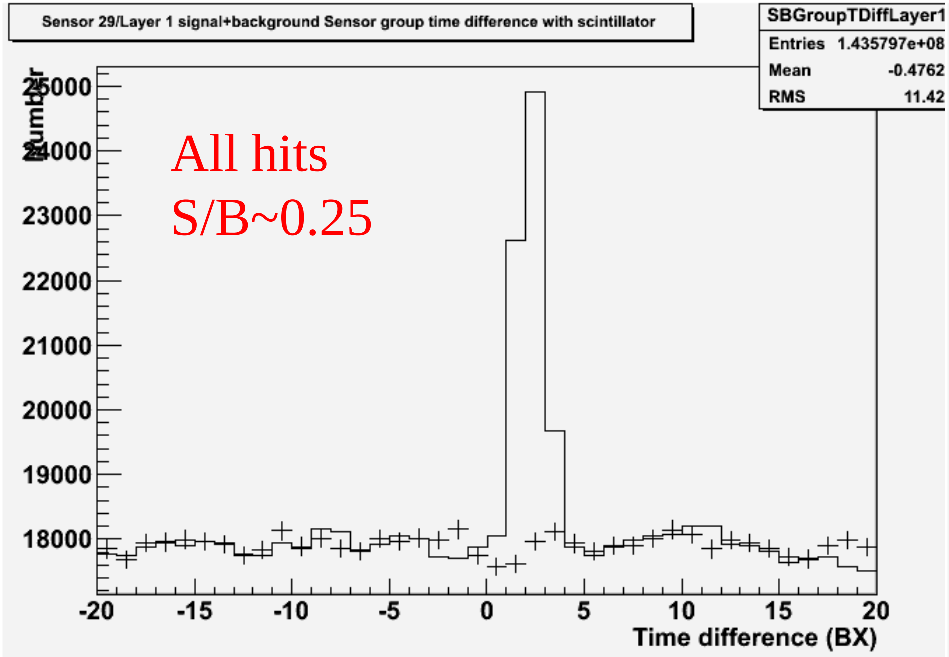
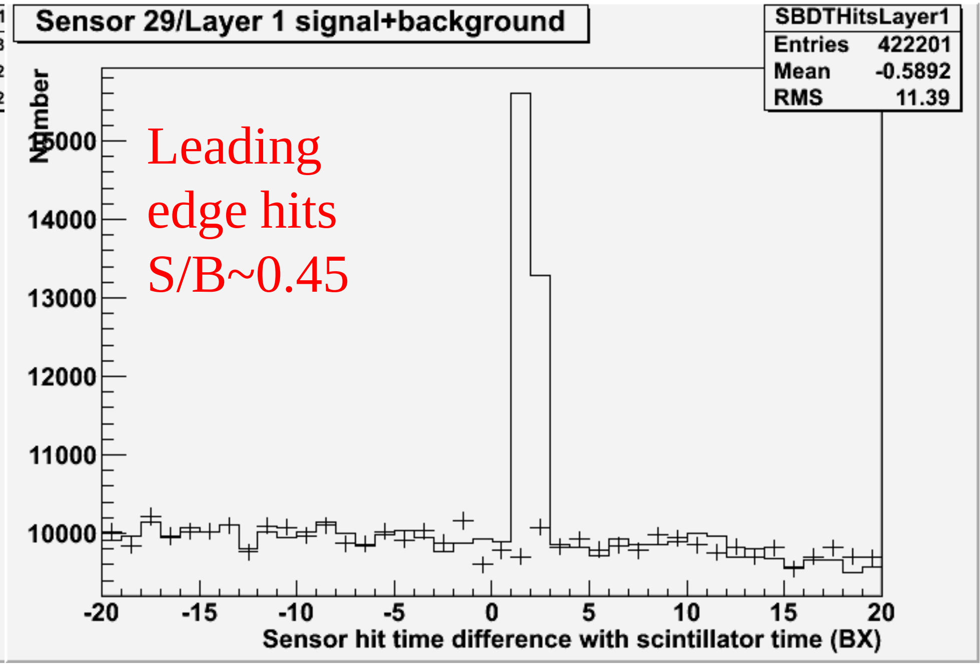
Status of 2D efficiency study

Paul Dauncey

Timing

- Previously showed hit BX distribution relative to scintillator
  - Signal peaks at 2BX, range is 1-3BX
- But now know many pixels have sequential hits in time
  - Use only first (“leading edge”) hit for each pixel
  - Signal now peaks at 1BX, range is 1-2BX
  - Two bins includes less background; better rejection

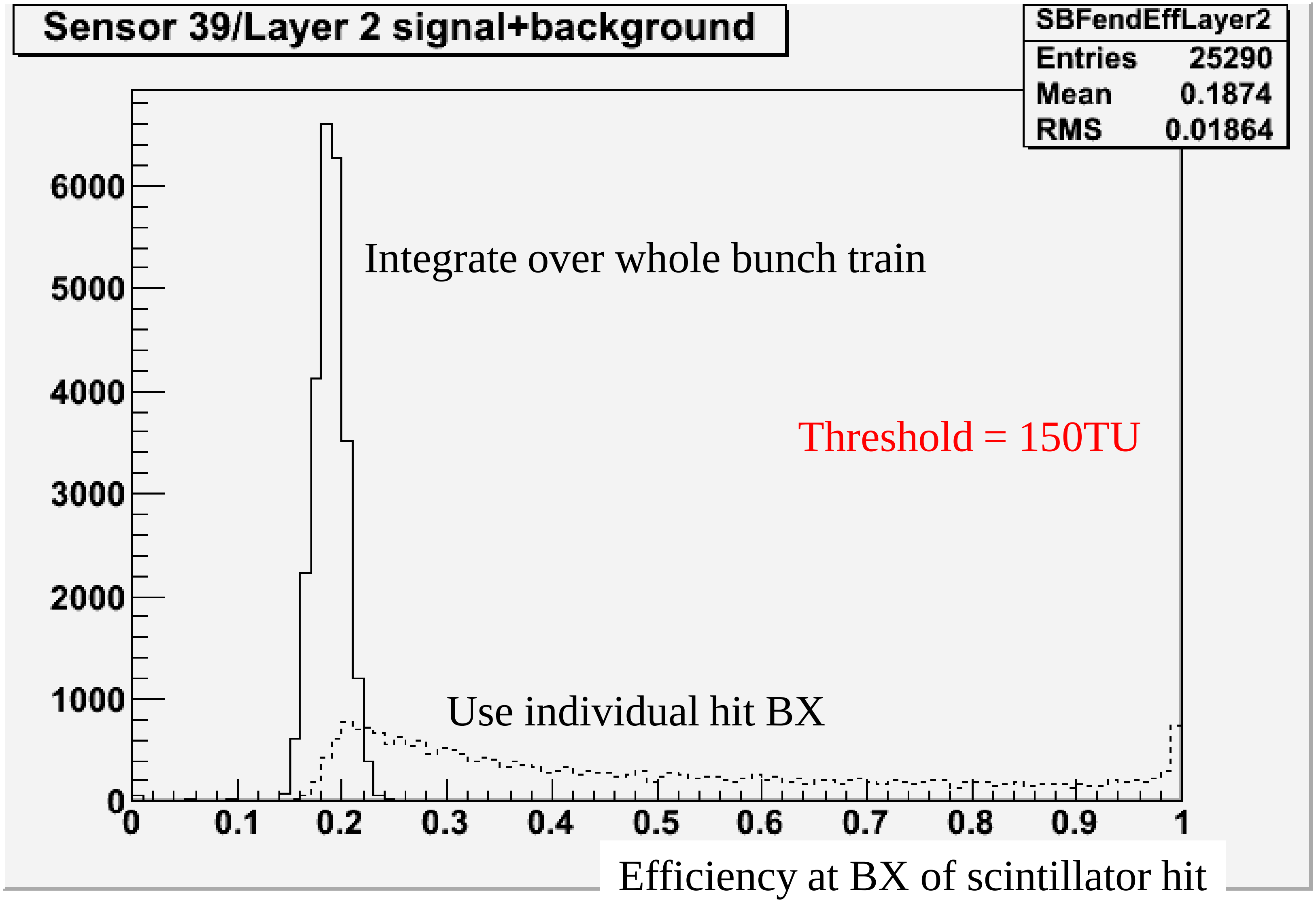




Full memory

- Storage for only 19 hits per row (per region = ¼ of width)
  - All hits after the BX of the 19th hit are lost
- Two possibilities discussed previously
  - Find which rows are full at the end of the bunch train and treat all pixels in these rows as bad for all BXs
  - Only treat pixels as bad for BXs after memory goes full for their row
- First is simpler but will throw away some good hits
  - How big a loss is this?
  - Will be threshold dependent; main effect is at low thresholds
  - Owen has code to do first method (see URL in previous minutes)
  - I wrote some code to do second method to compare

Efficiency due to full memory



Using the full memory code

- Define the objects to contain the lists

```
MpsFullMemory mfm[6];
```

- For each bunch train, find when memory goes full

```
MpsSensor1BunchTrainData *btd[6];
// Point btd to data from record
mfm[layer].setFull(*(btd[layer]))
```

- Find efficiency of a layer at a particular BX

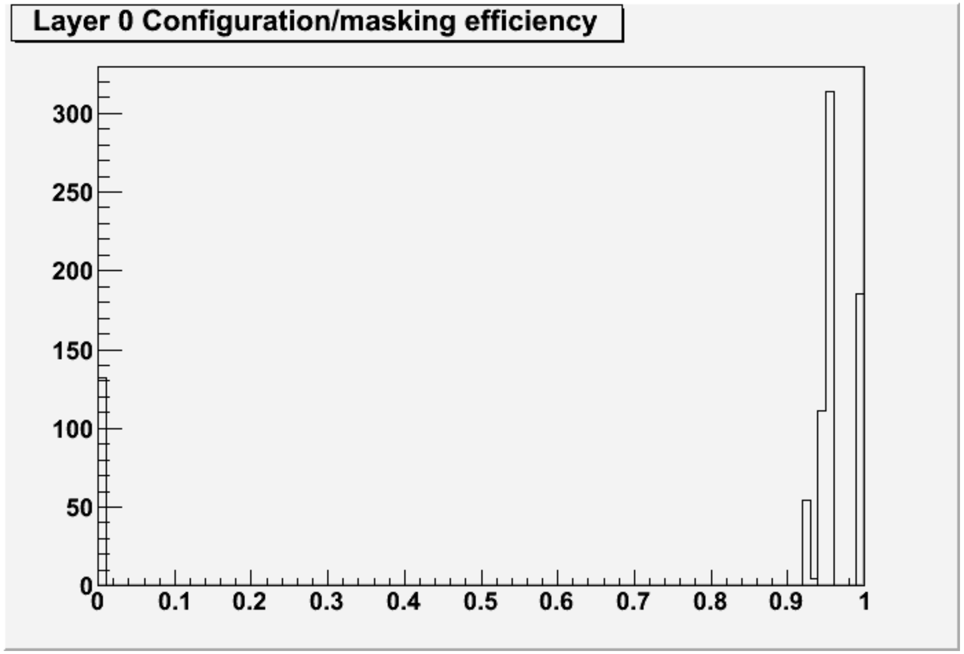
```
unsigned bx(1234); // Random BX value
double e=mfm[layer].efficiency(bx);
```

- For any pixel x<168 and y<168

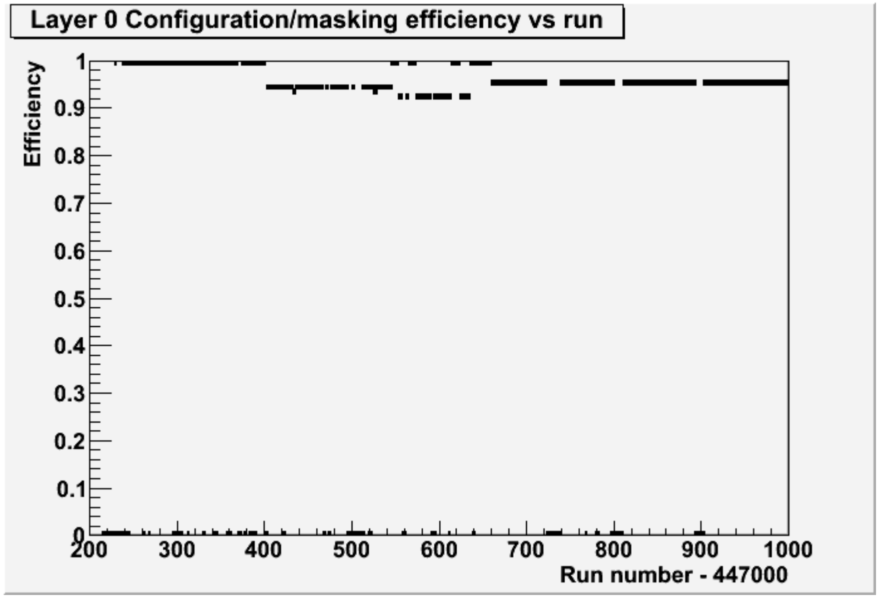
```
if(!mgp[layer].full(x,y,bx)) {
   // Use for analysis
```

- Check daquser/inc/mps/MpsFullMemory.hh for other useful methods

Efficiency due to bad config/masking

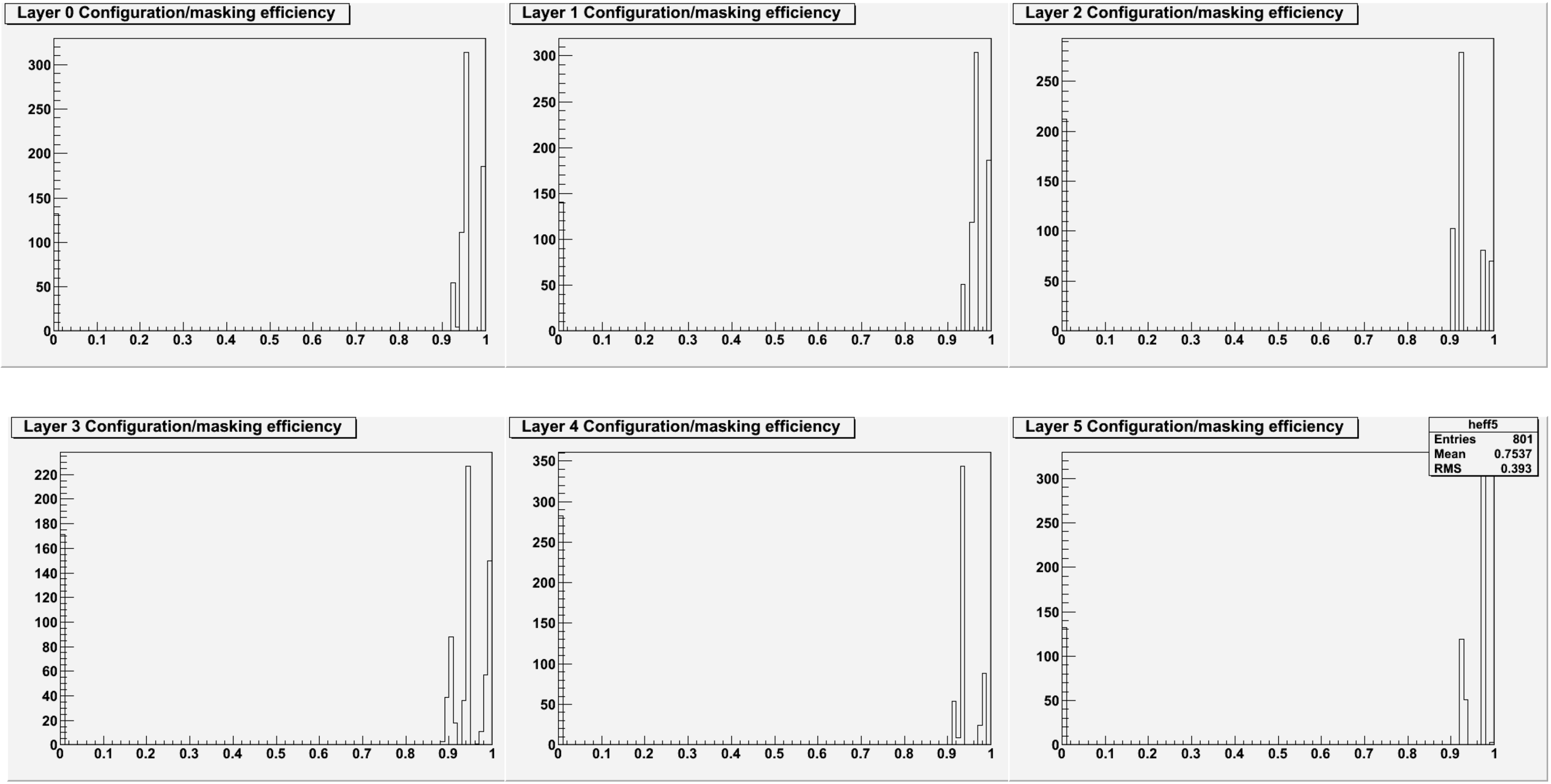


Efficiency for layer 0 per run
For good runs ~90%



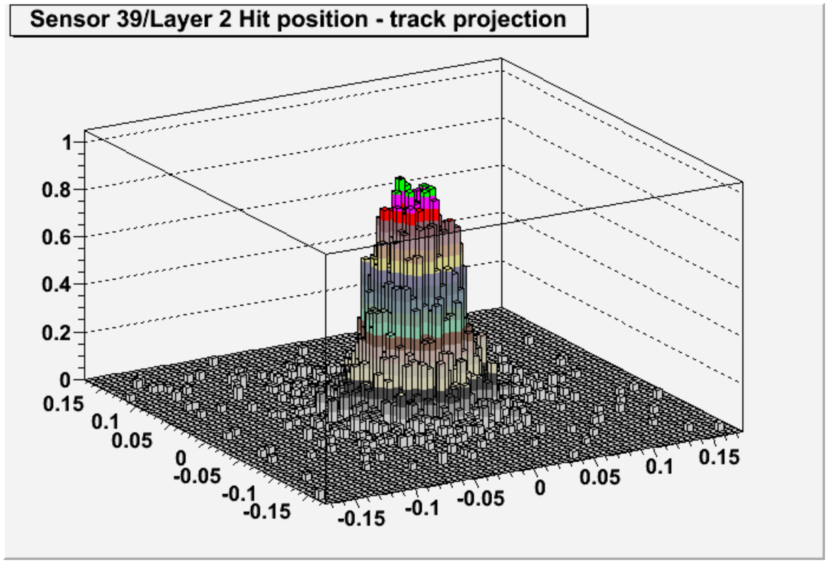
Evenly distributed
throughout run period

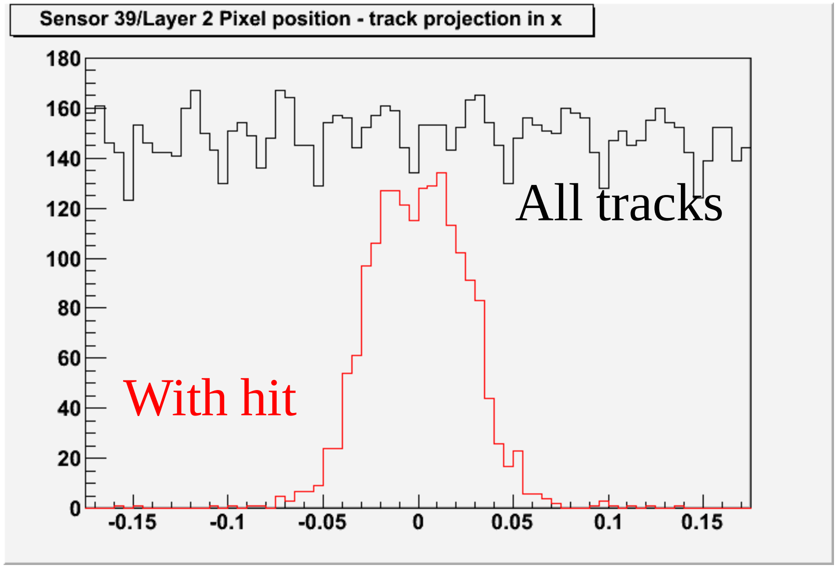
Bad config/masking efficiency per layer

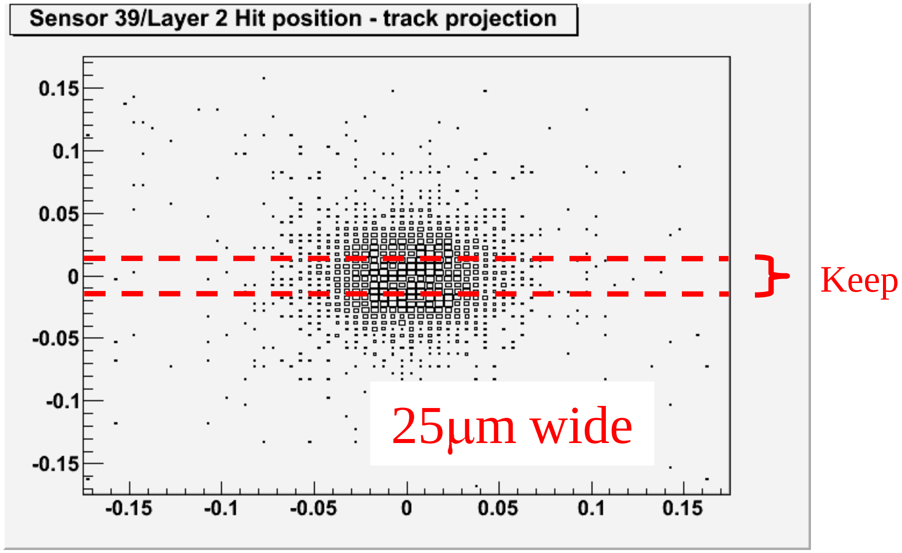


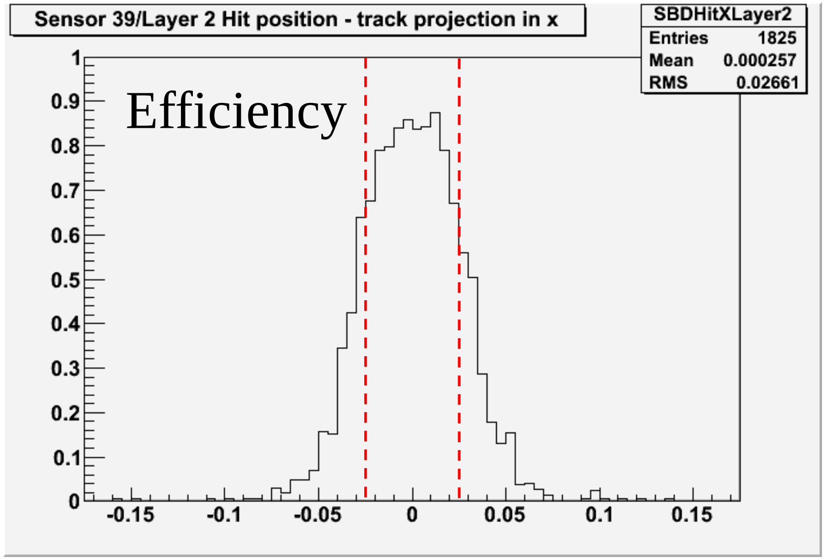
Same conclusion for all layers; for good runs efficiency ~90%

Projections in x and y (shown before)



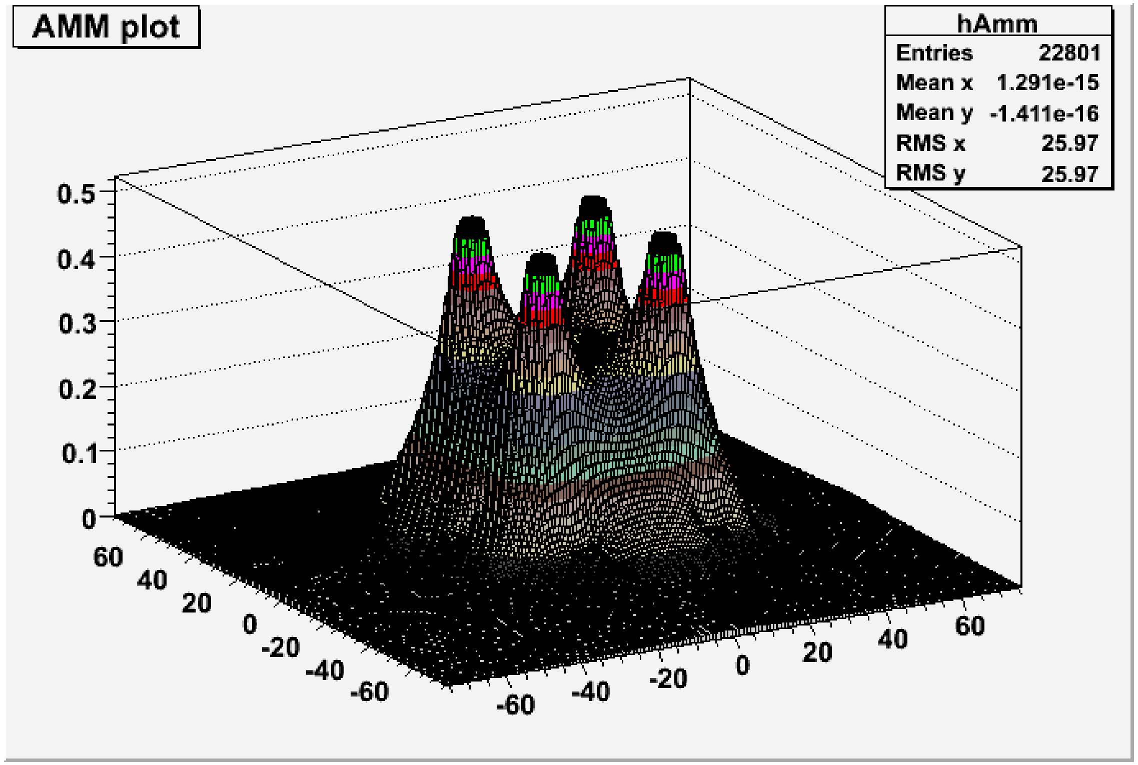






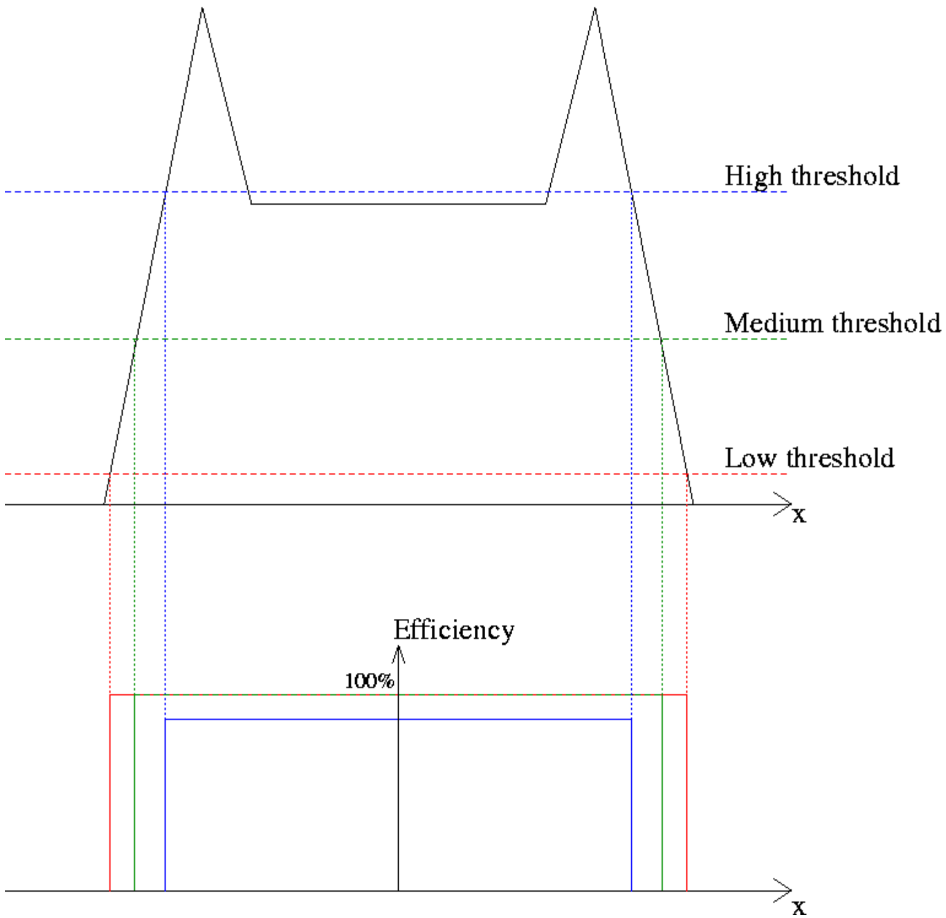
Expected efficiency in 2D

- Simulation plot of charge fraction vs position for a MIP
  - MIP ~1200e^- total, central plateau ~0.3 ~ 360e^-
  - Calibration 1TU ~ 3e^- so plateau ~ 120TU above pedestal ~ 220TU
  - Nominal threshold of 150TU is 50TU above pedestal, ~half plateau
  - Average noise ~7TU so nominal threshold is ~7σ above pedestal



Efficiency fit function

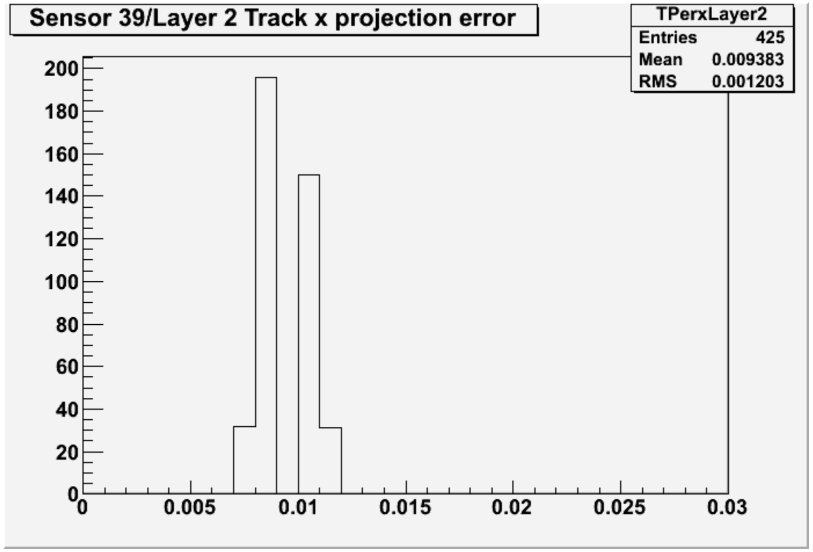
- Below plateau, pixel should be 100% efficient out to where charge fraction drops below threshold
  - Box ("top hat") function with width > 50µm



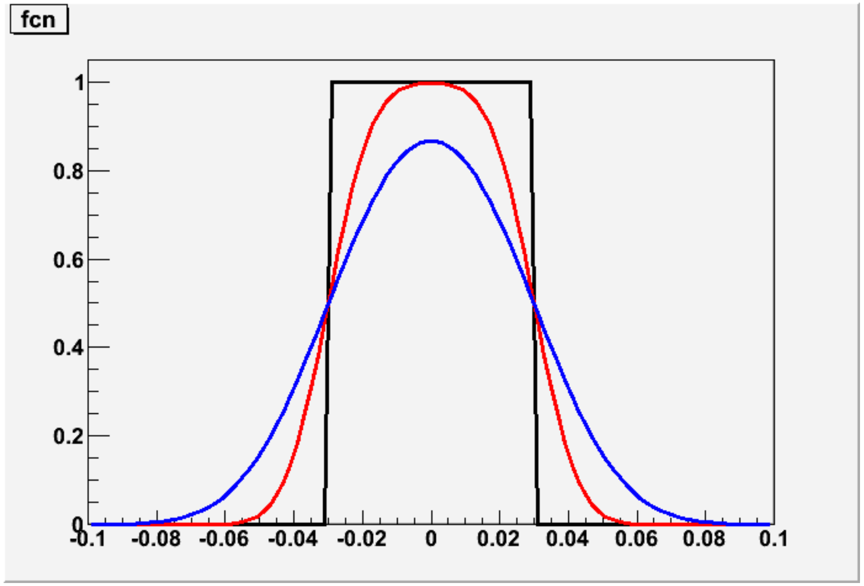
- Increasing threshold narrows box but efficiency within box stays at 100%
- With threshold ~ plateau, efficiency will drop from 100%

Efficiency fit function smearing

- In reality, box edges smeared by
  - Electronics noise, small?
  - Track resolution ~10μm for inner layers, more for outer
- Convolute box with Gaussian
  - Difference of two erfs

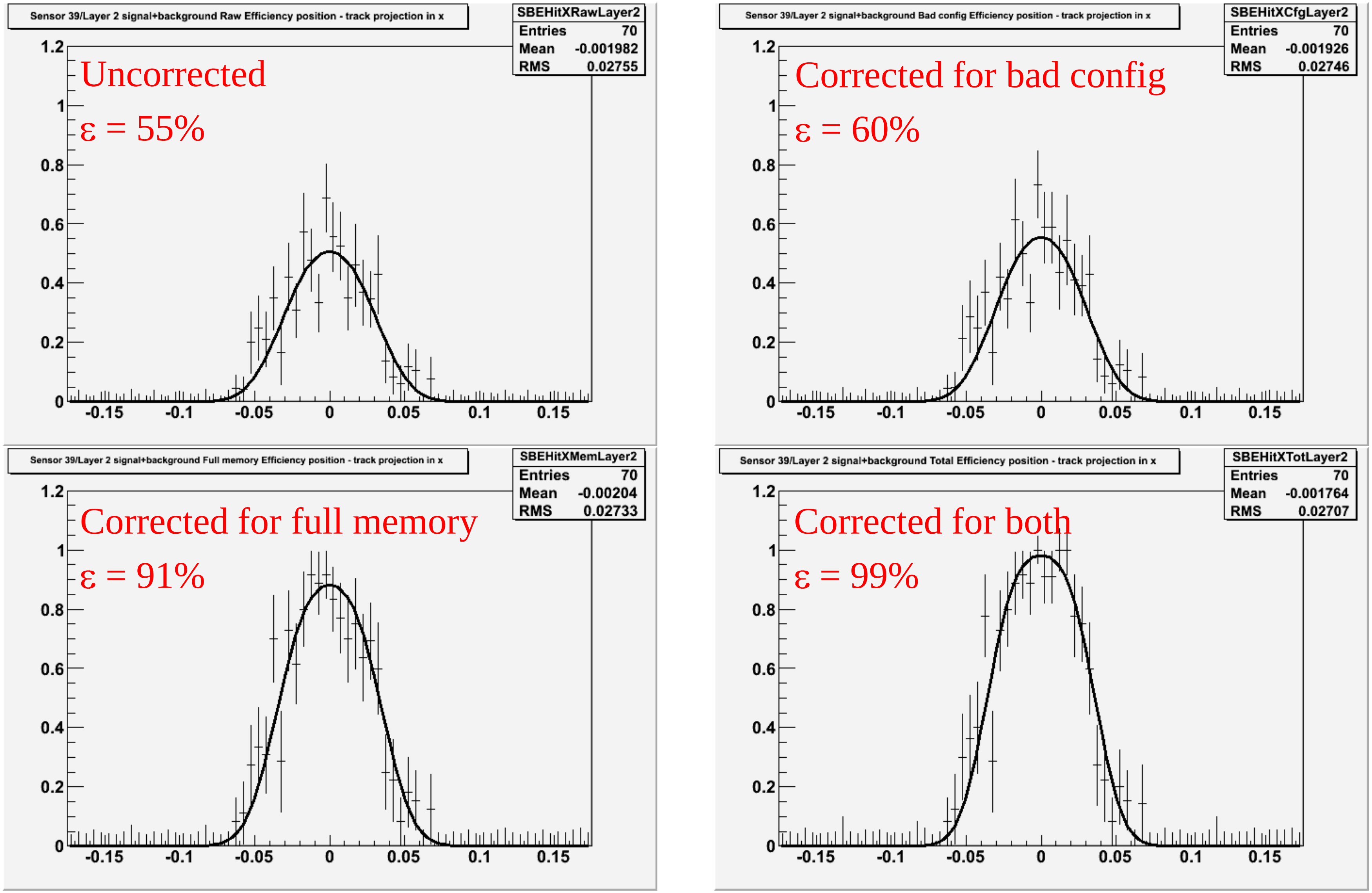


ε[TMath::Freq((0.5w−x)/σ)−TMath::Freq((−0.5w−x)/σ)]



- Note, 100% efficiency does not always give peak at 1.0
  - ε=1, w=0.06mm, σ=0.00mm
  - ε=1, w=0.06mm, σ=0.01mm
  - ε=1, w=0.06mm, σ=0.02mm

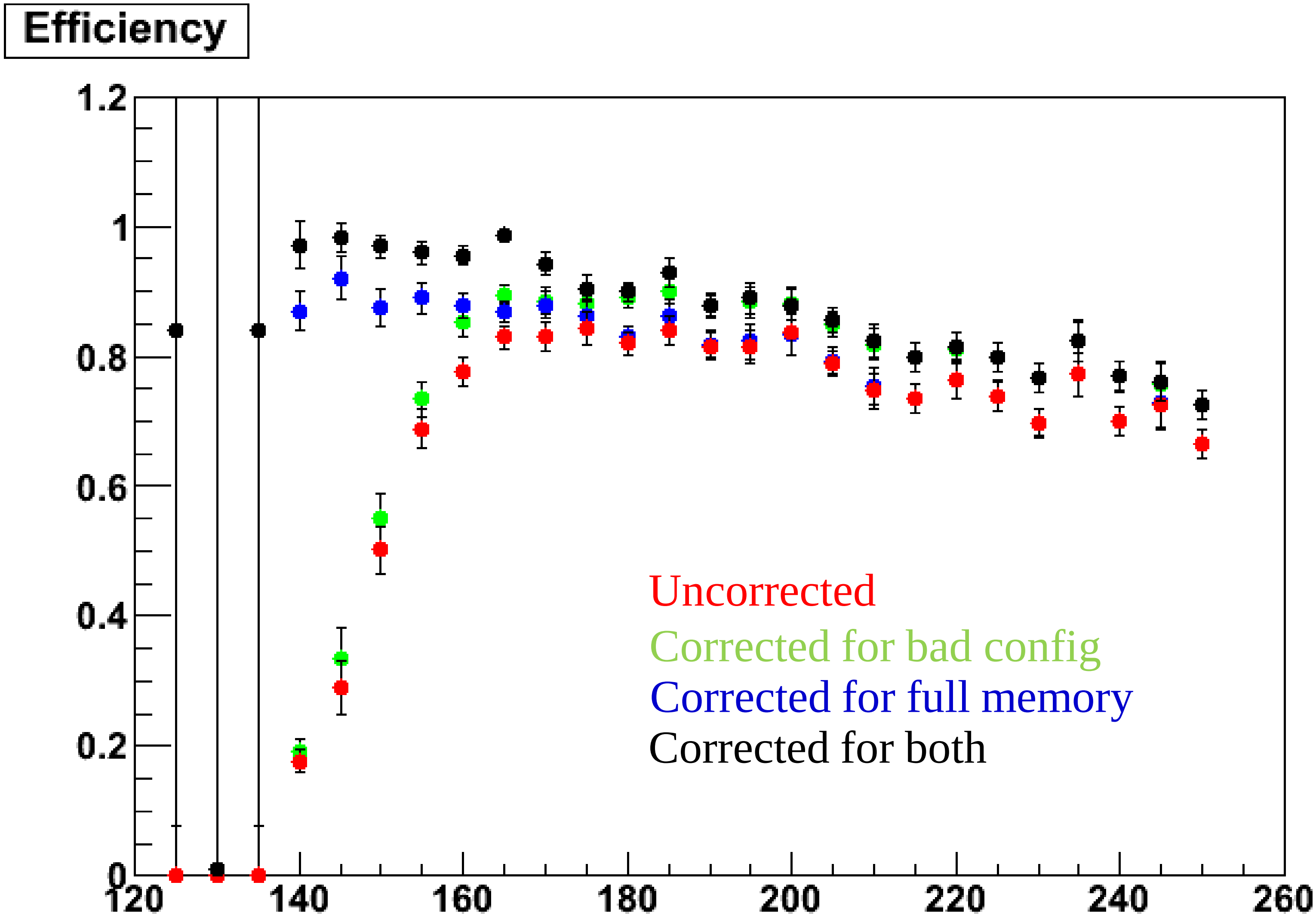
Fit to x projections: run 447825, layer 2



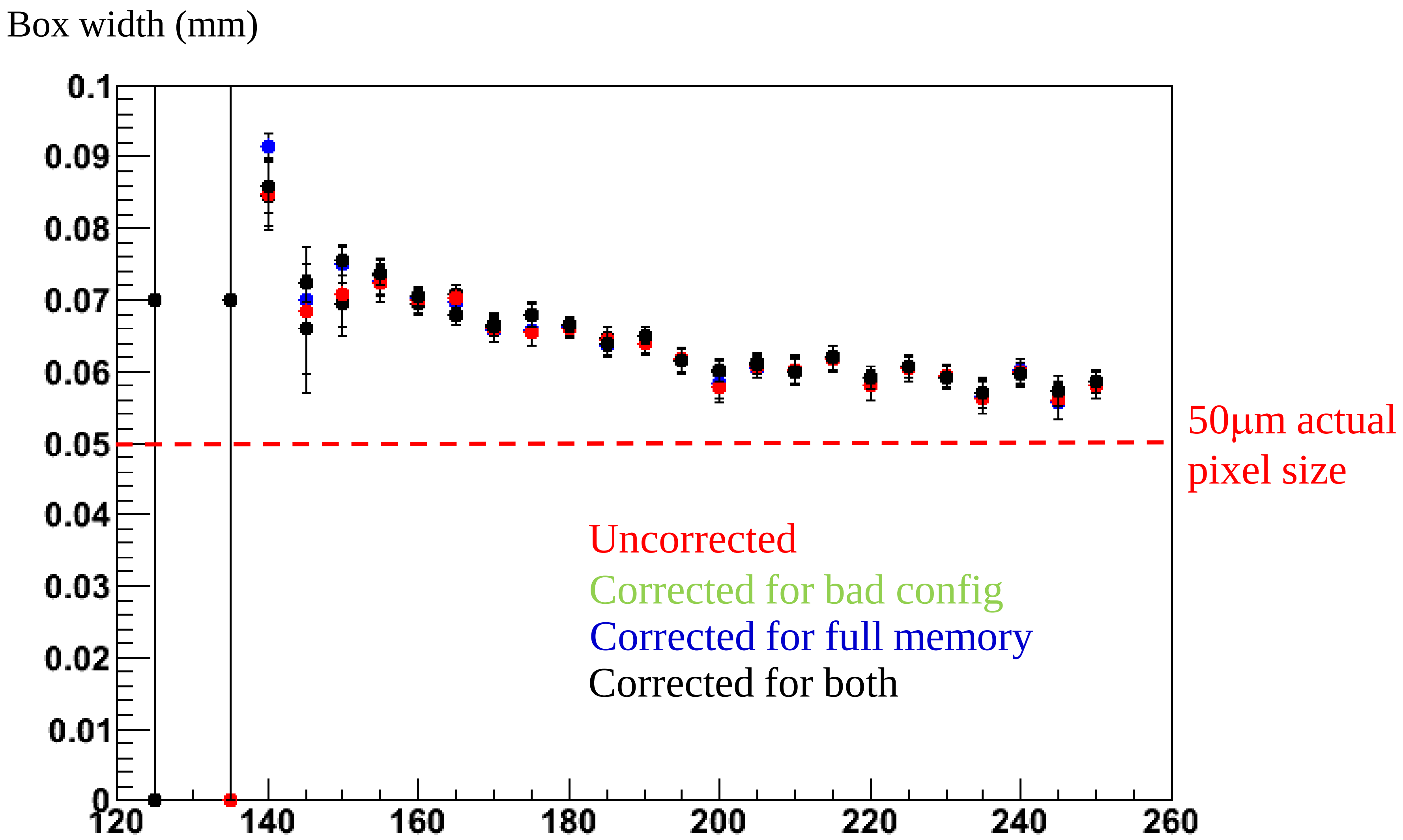
Run selection

- For each sensor
  - Sum data for all “good” runs/sensors with same threshold
  - Fit function to efficiency plot for that threshold
  - Repeat for all thresholds used for that sensor
- Good runs defined as
  - Number of bunch trains >= 1000
  - Number of scintillator coincidences >=500
- For good runs, good sensors defined as
  - Sensor id reads OK
  - Threshold in range 125-250
  - Number of good config pixels >=20000 (~71%)
- Results shown for x fit only
  - 2D xy fit gives similar results

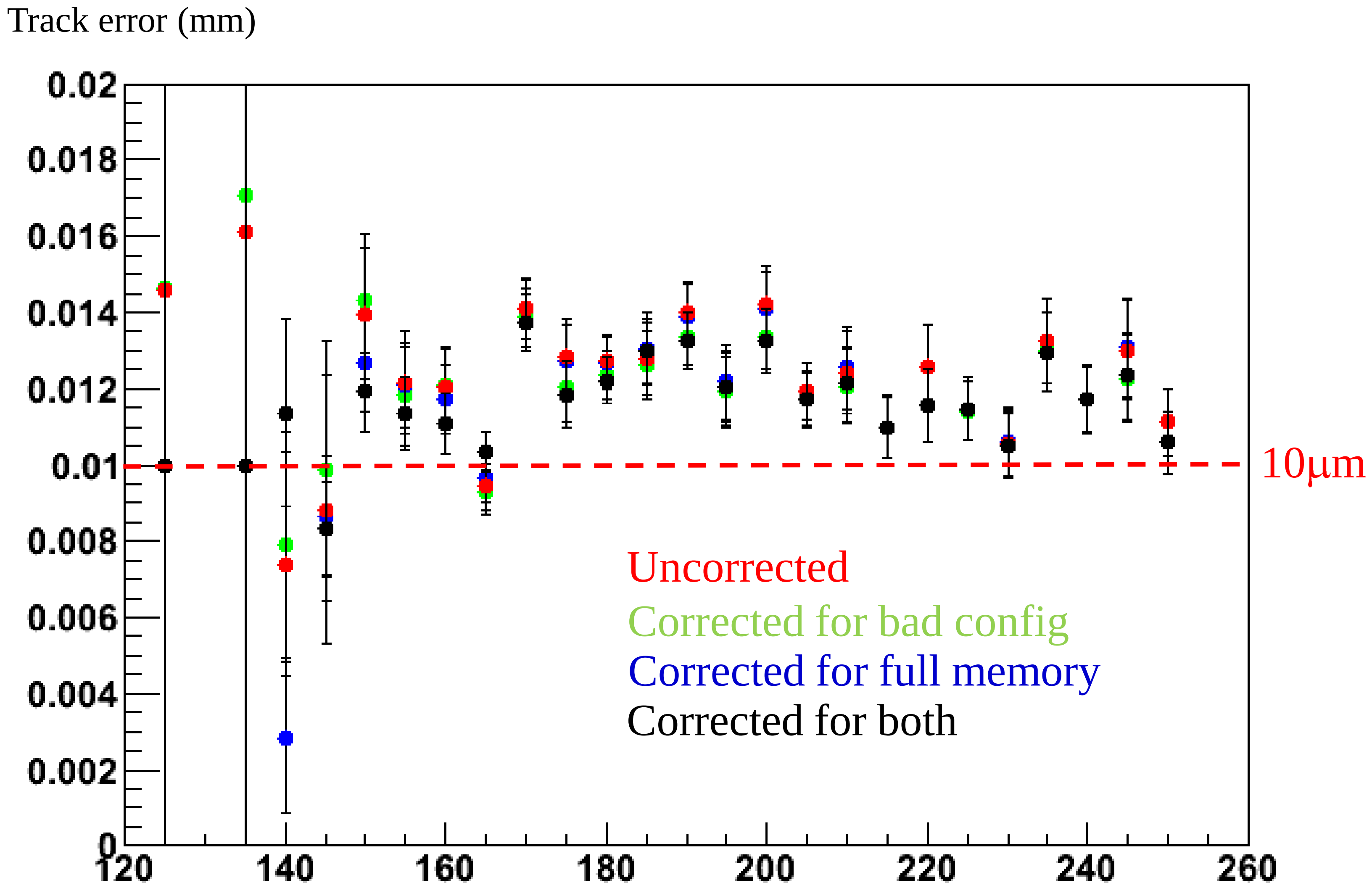
Fitted efficiencies; all runs with sensor 39



Fitted box widths; all runs with sensor 39

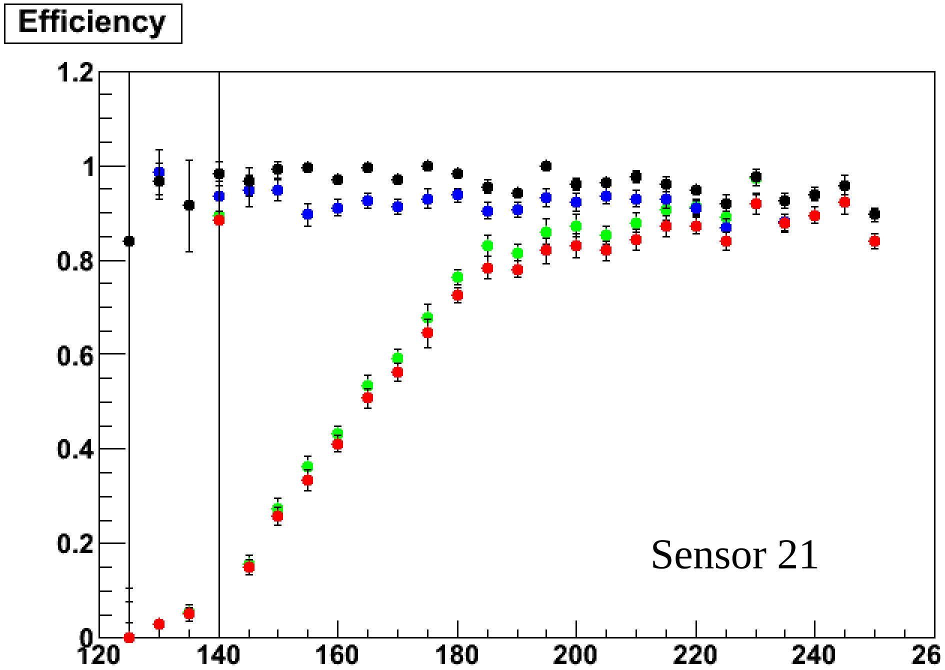


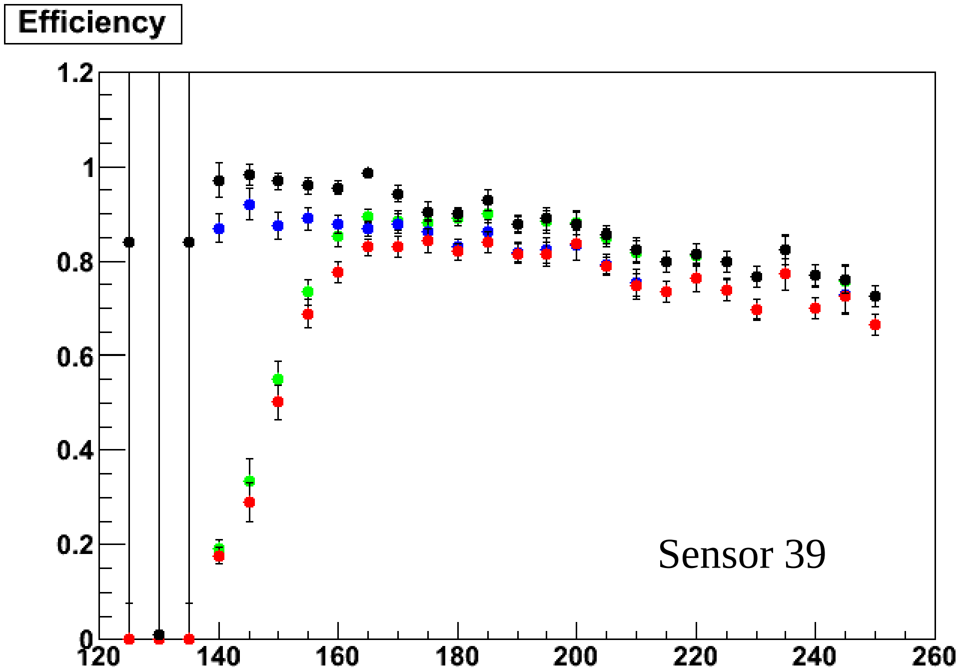
Fitted track errors; all runs with sensor 39



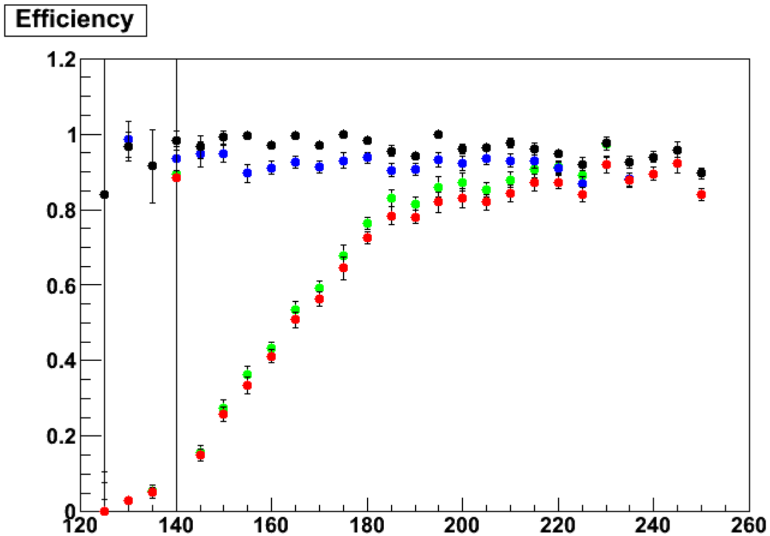
Sensors 21 and 39

- The two inner sensors with the “best” data
  - All thresholds from 125 to 250 in steps of 5
  - Sensor 21 is 12μm hi-res, sensor 29 is 12μm standard

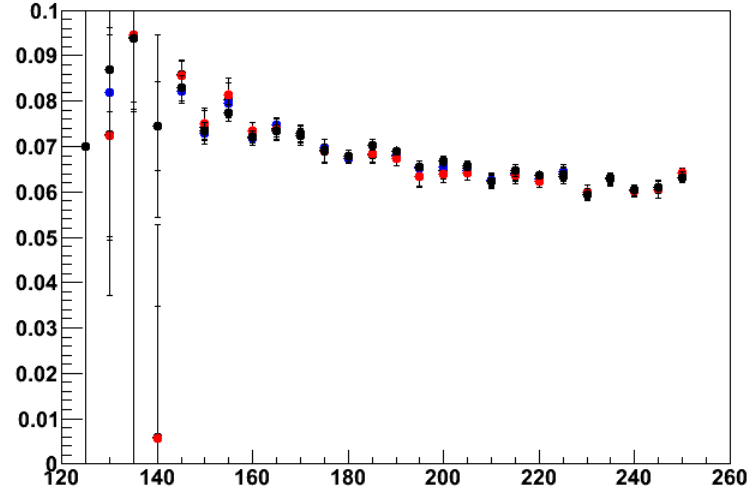




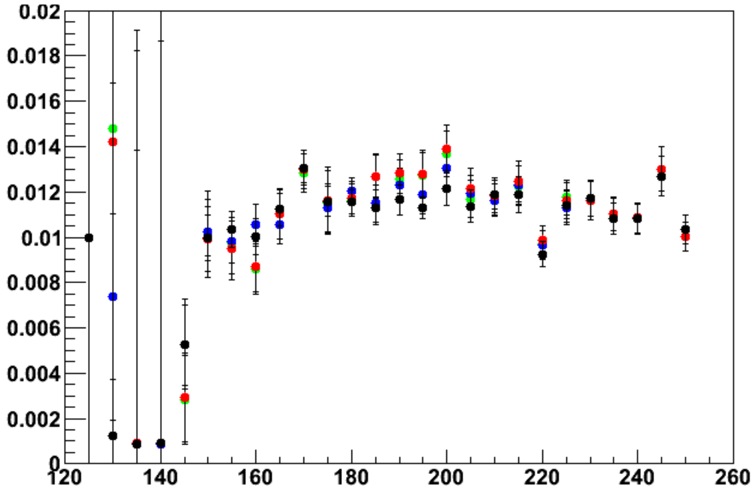
Sensor 21, layer 3 (12μm hi-res)



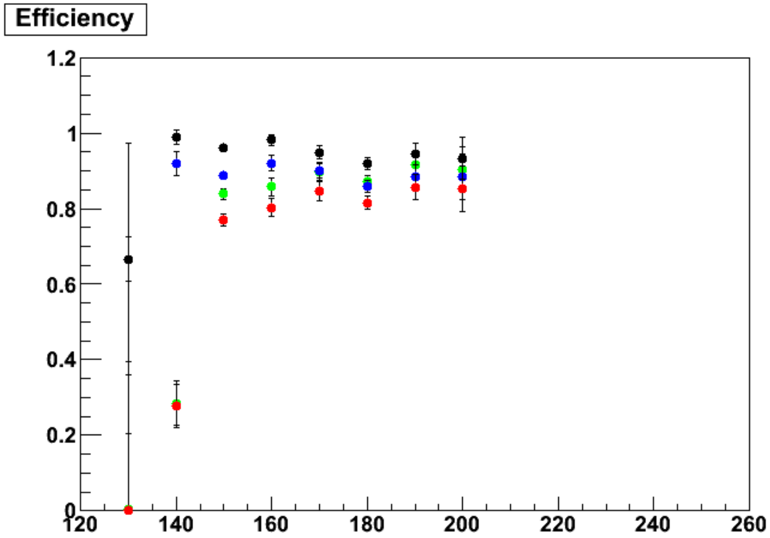
Box width (mm)



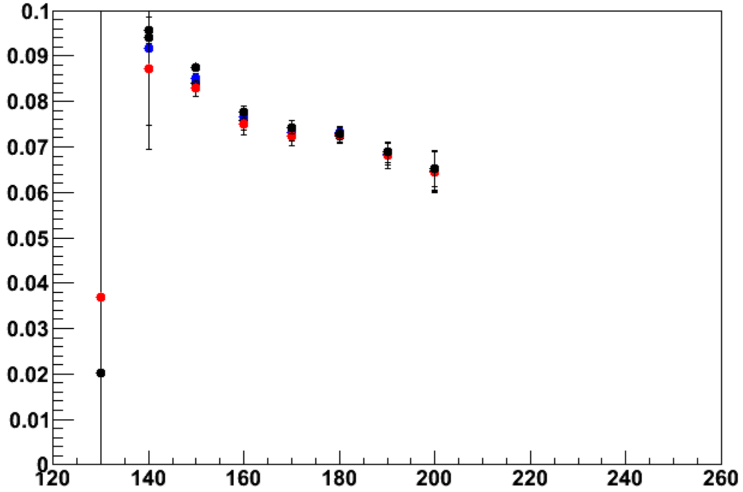
Track error (mm)



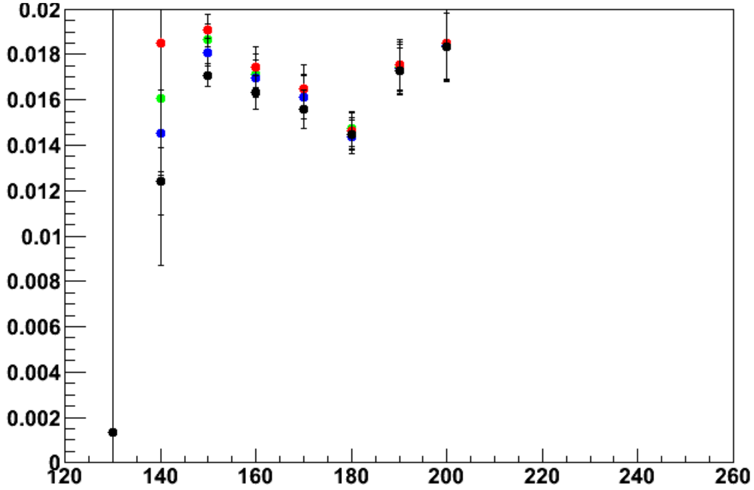
Sensor 26, layer 3 (18μm hi-res)



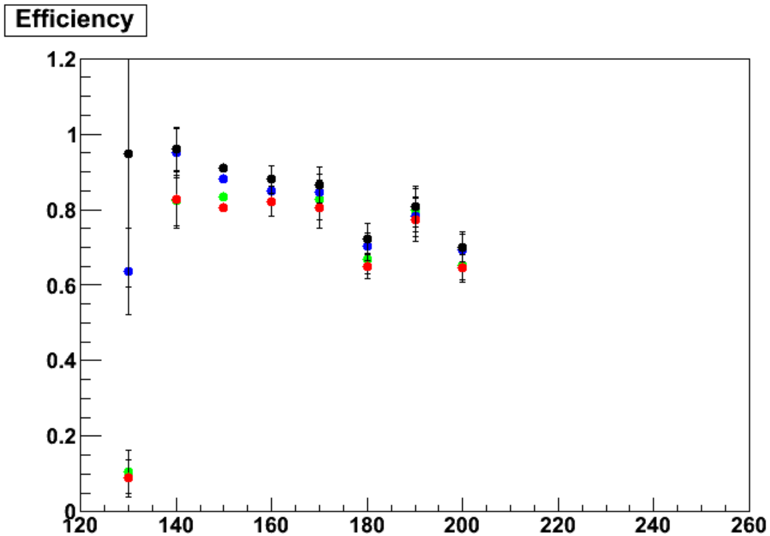
Box width (mm)



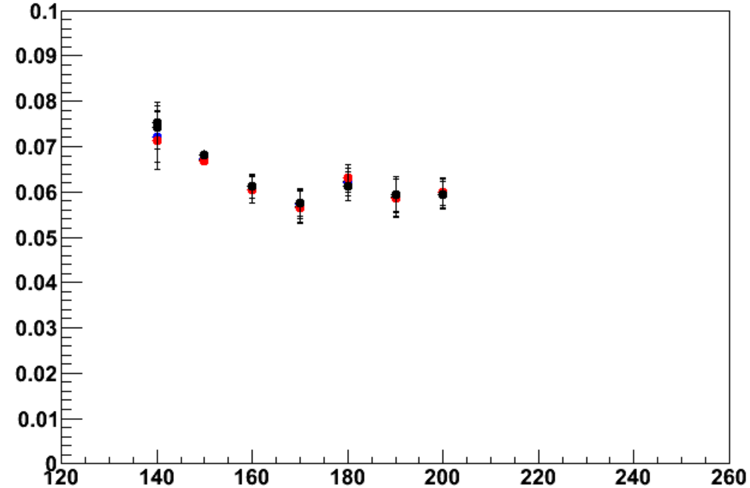
Track error (mm)



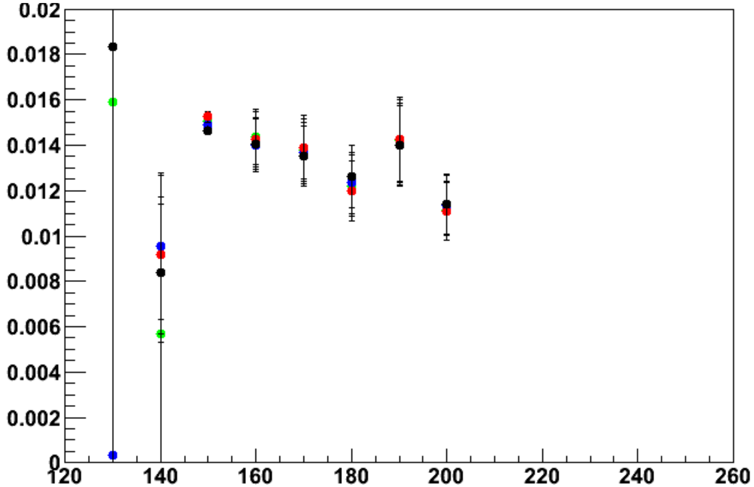
Sensor 29, layer 1



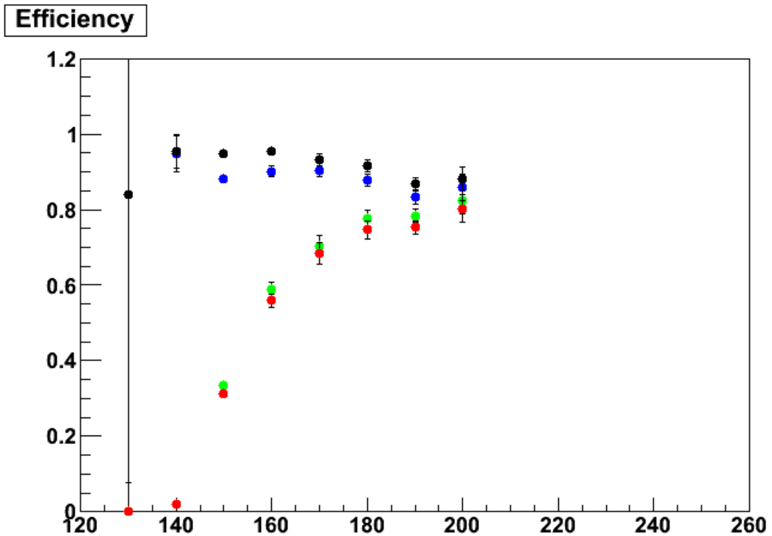
Box width (mm)



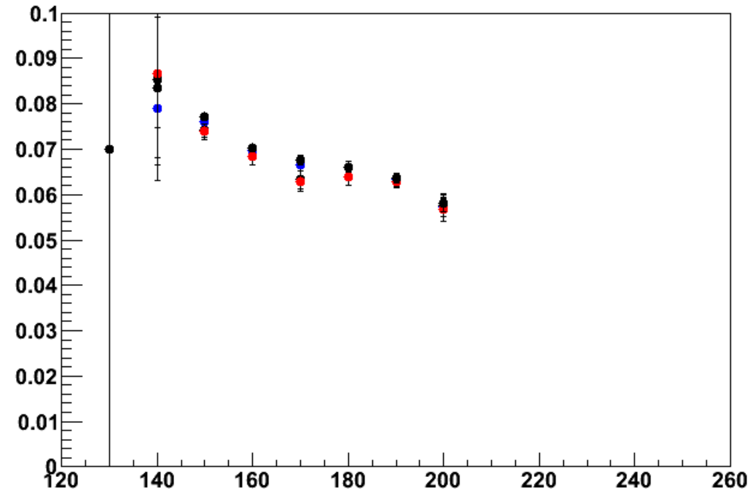
Track error (mm)



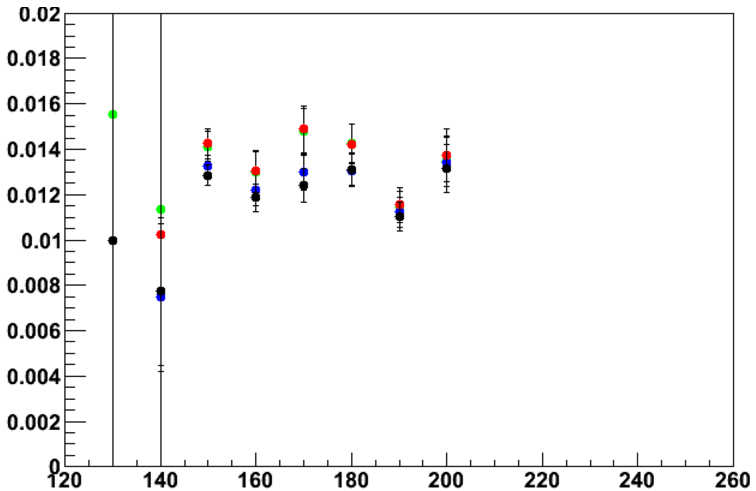
Sensor 32, layer 2



Box width (mm)

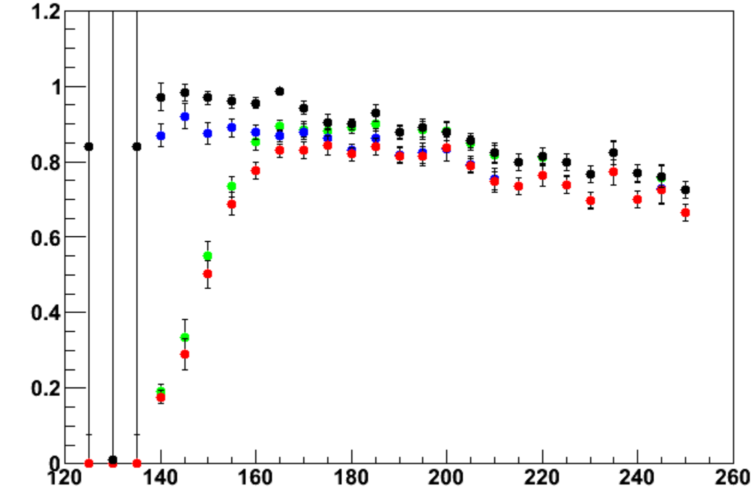


Track error (mm)

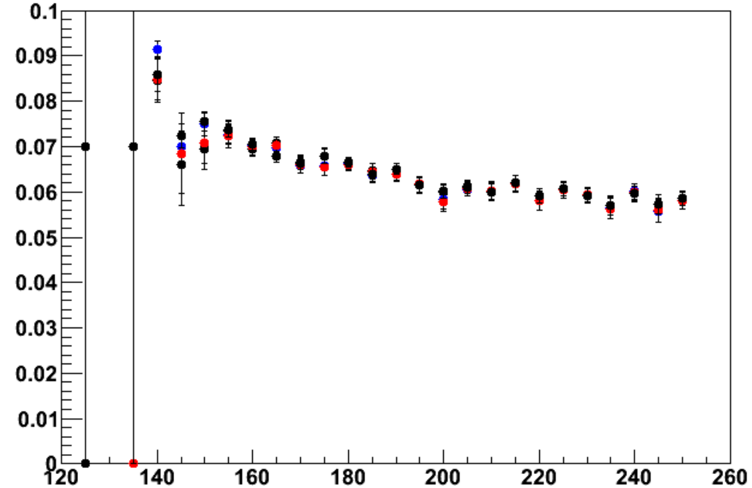


Sensor 39, layer 2

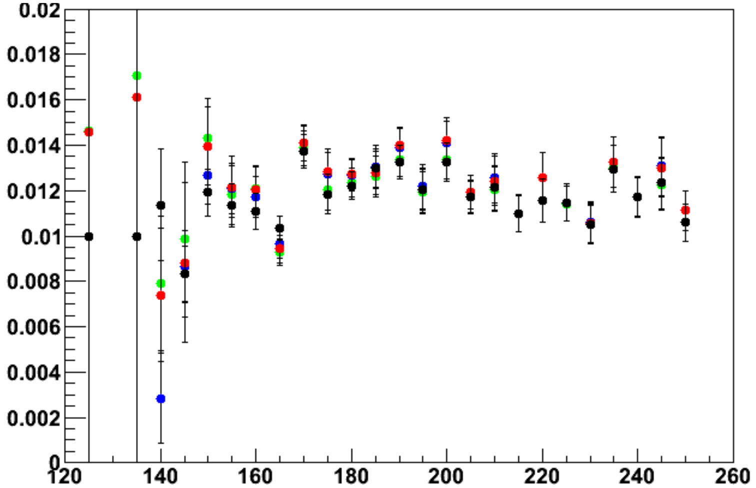
Efficiency



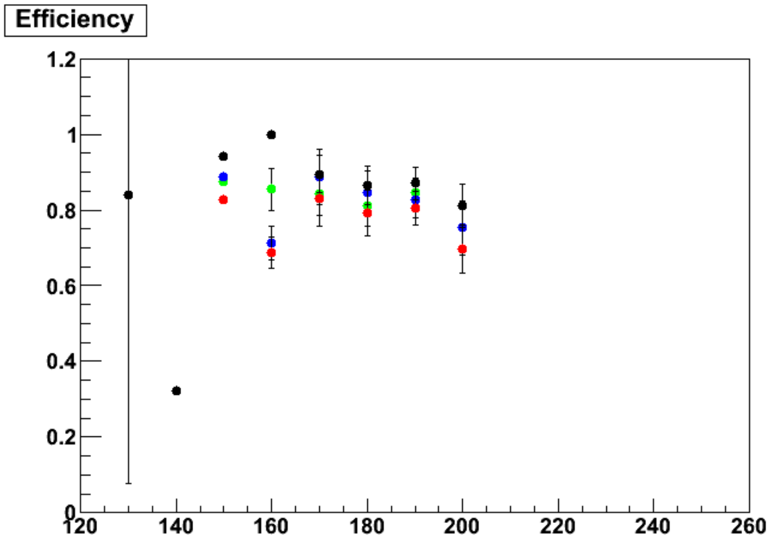
Box width (mm)



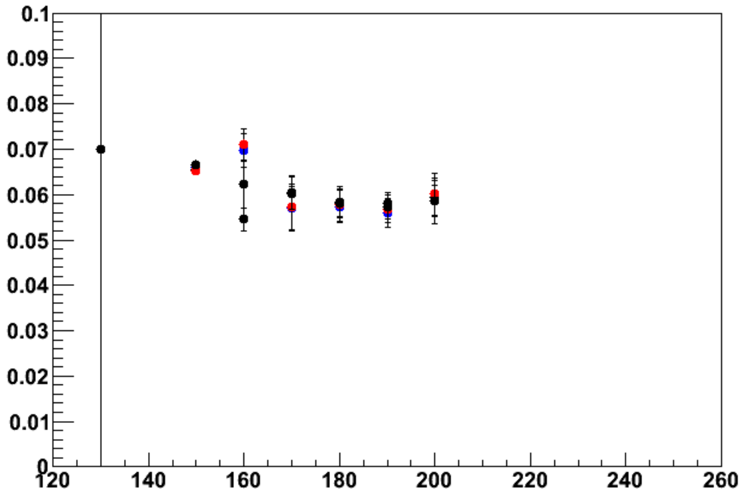
Track error (mm)



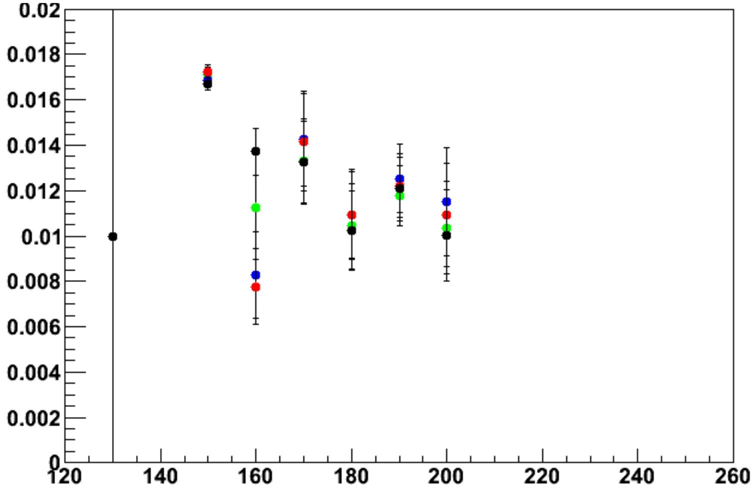
Sensor 41, layer 4



Box width (mm)

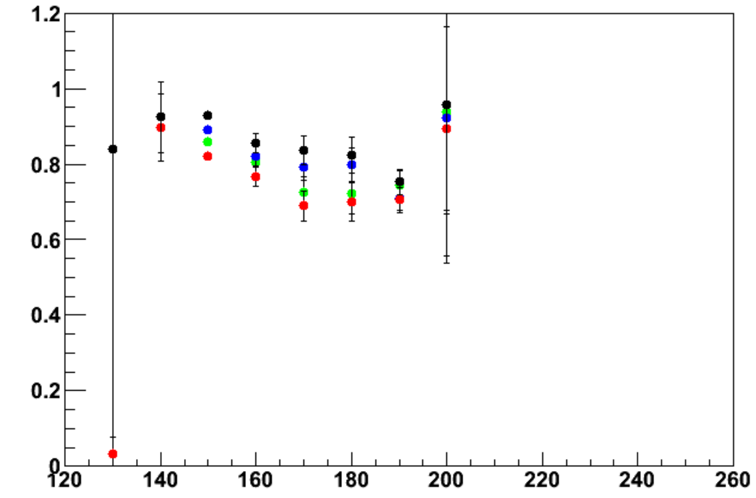


Track error (mm)

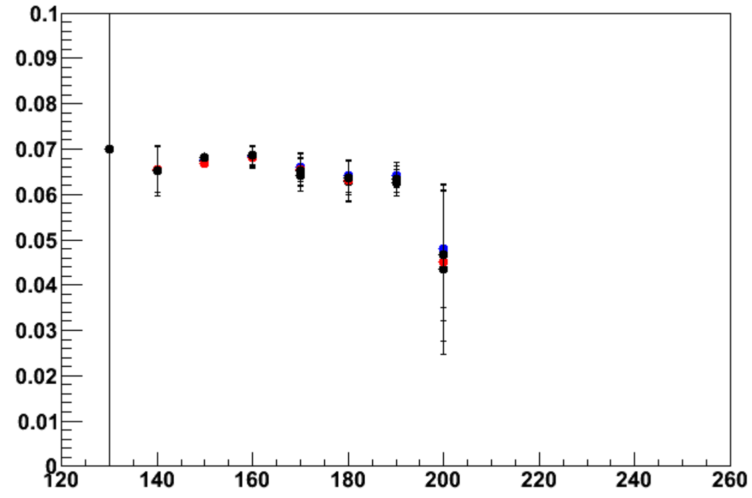


Sensor 43, layer 0

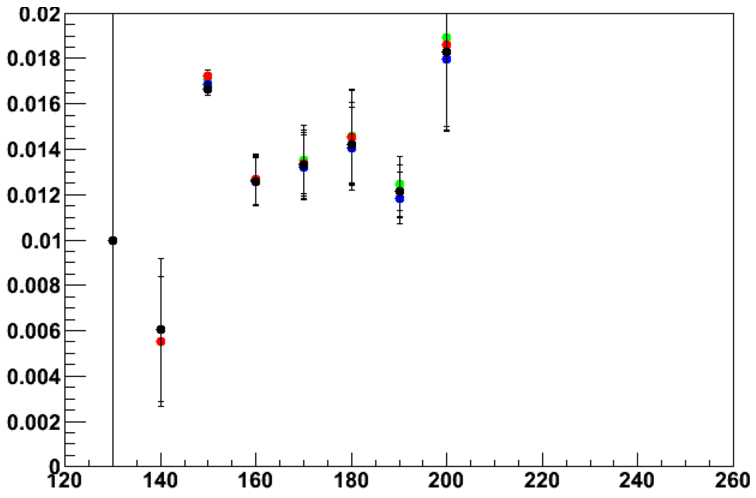
Efficiency



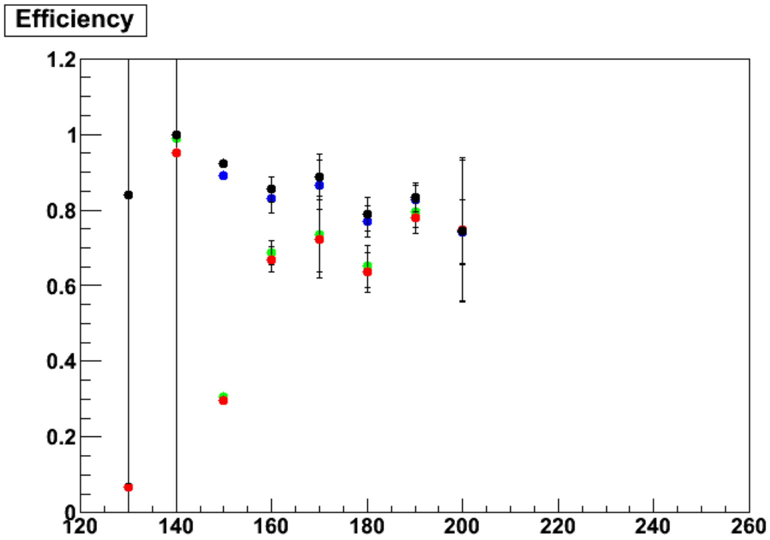
Box width (mm)



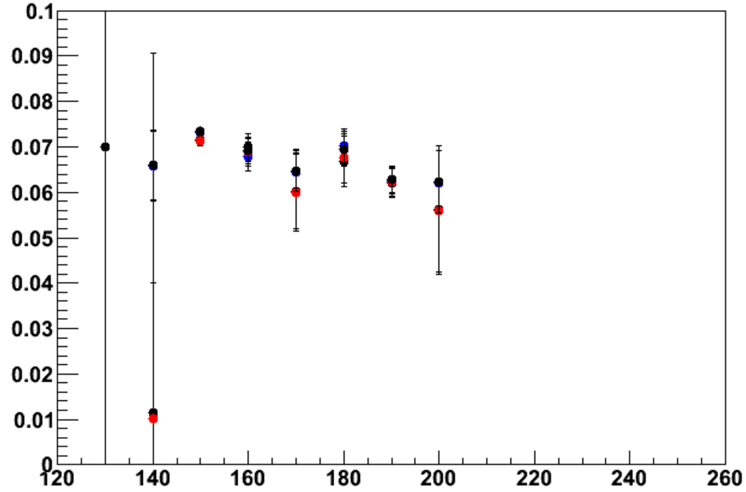
Track error (mm)



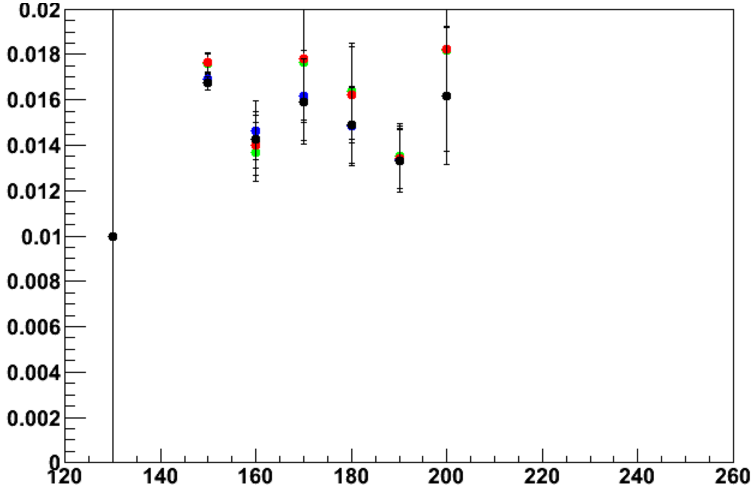
Sensor 48, layer 5



Box width (mm)



Track error (mm)



Conclusions

- Cuts on time difference of hits from scintillators should use leading edge, not all times
- Integrating over a full bunch trains for memory full bad pixels will not make good use of the statistics at low thresholds
- Preliminary conclusions on 2D efficiency
  - Fit is stable for box width and track error parameters; these give sensible values
  - Efficiency stays above 80% out to 200TU
  - The hi-res sensor seems more efficient at high thresholds than the standard sensor used for the last set of runs