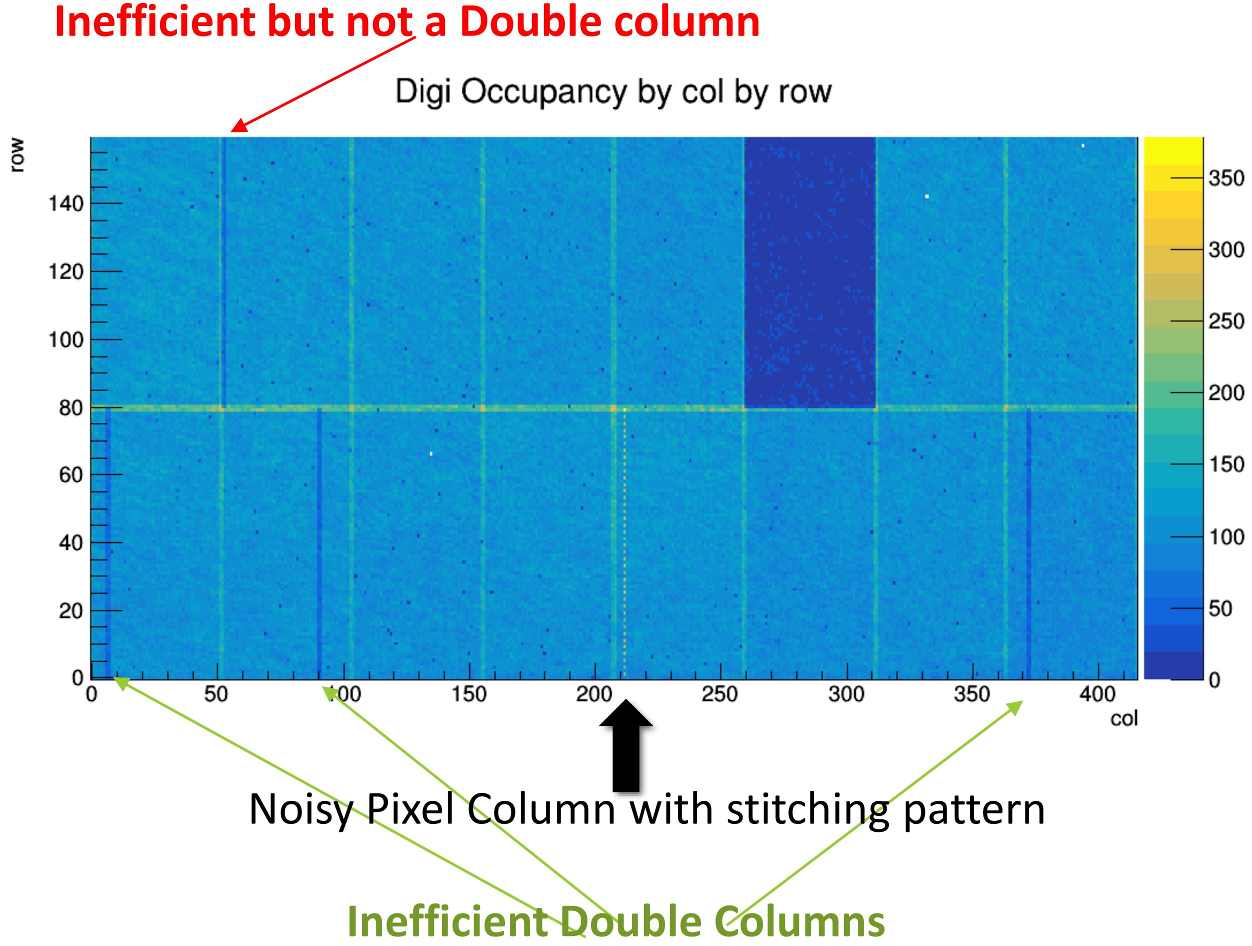
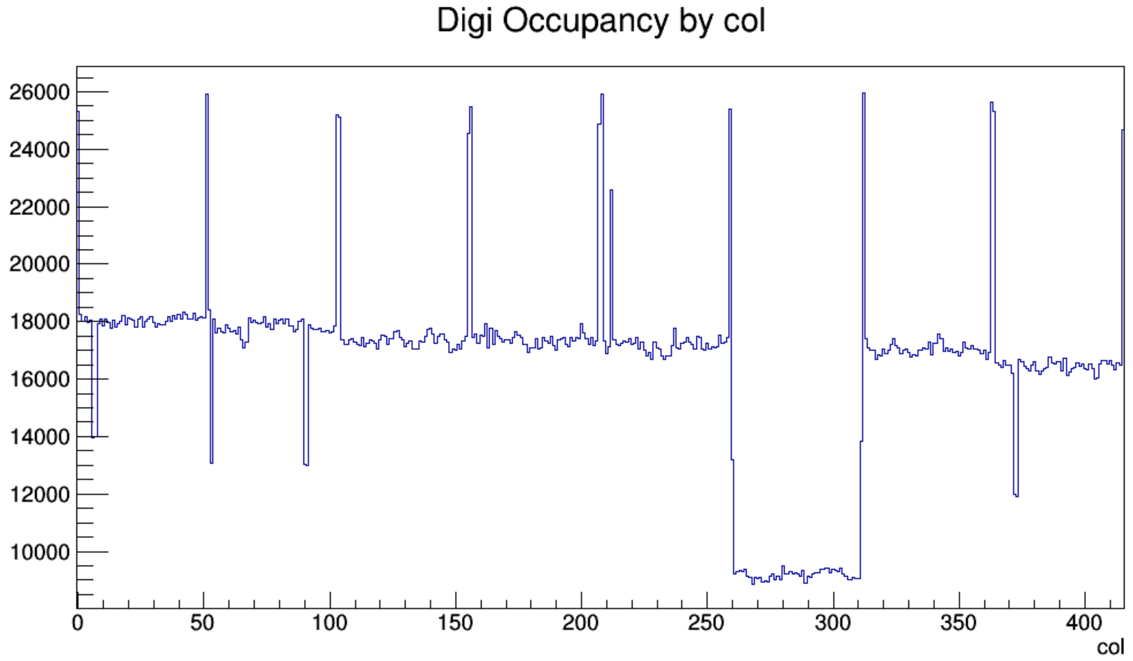
Inefficient Double ROC Columns - Tool

PAWEL JURGIELEWICZ (AGH UST, CRACOW)

The case: Inefficient Double Columns + extra effect





These effects are best visible in the Barrel Layer 1

The algorithm

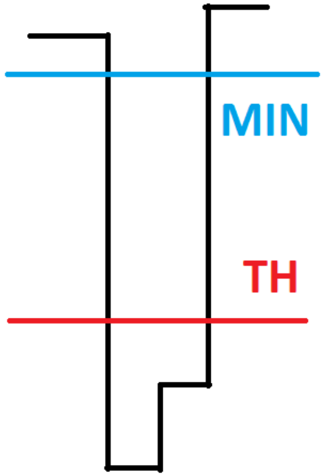
1. Take <digi_occupancy_per_col_per_row_> histograms (2D)
   - digi_occupancy_per_col_ (1D, row summed) insufficient, lots of information lost:
     - row number,
     - false inefficient double columns,
     - false column noise
     - and others...

2. Sum row data in each ROC column – do not take Big Pixels into account -> **pixelColArr**
   - Check for noisy pixels inside each column (TH = 6 * columnMean) – if there are noisy pixels in column `Column Noisyness` will not be checked
   - Smoothen sum column data using median filter (removes spikes) -> **medFiltRes**
     - kernel radius: 2,
     - repeat: 3

The Algorithm: Barrel

- Remove drops (down pointing spikes) from **pixelColArr** (**pixelColArr**[i] < **medFiltRes**[i] ? **medFiltRes**[i] : **pixelColArr**[i]) and normalize it -> **pixelColArrNorm**
  - Used for noisy column classification

- Look for inefficient double columns:
  - If two adjacent columns are and neighbours on left/right are not lower than current TH -> Inefficient Double Column
    - mean = <**medFiltRes**>
    - TH = sqrt(mean) * 8
    - min(**medFiltRes**) – **pixelColArr**[i] > TH



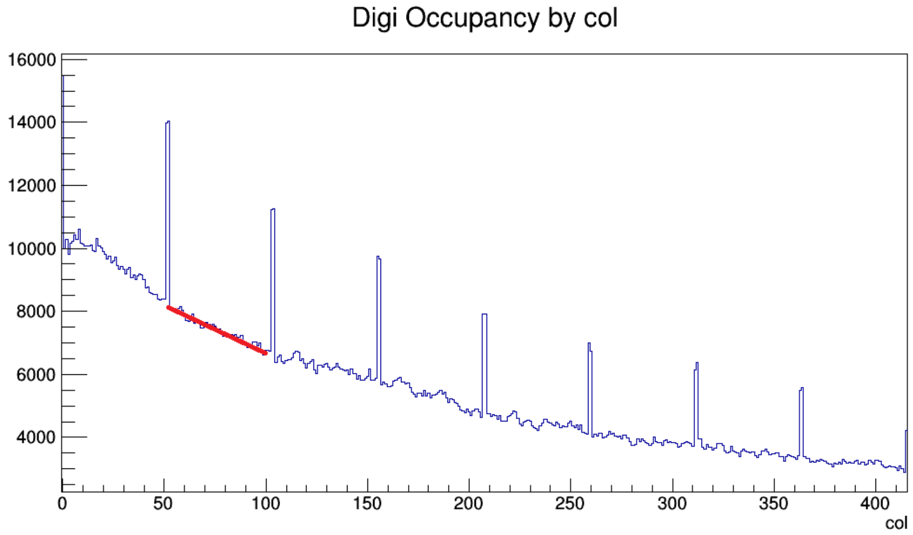
- Pixel Column Noisyness
  - Reject columns which already have very noisy pixels
  - Reject ROCs with low mean occupancy (TH = 200)
  - Column Noisyness TH = <**pixelColArrNorm**> * 4.5
    - **pixelColArrNorm**[i] > TH

The Algorithm: Endcap

- Fit the line to the **pixelColArr** using least mean squares method, since the distribution is not flat as in Barrel

- Look for inefficient double columns
  - trendVal(i) = a * i + b
  - TH = sqrt((trendVal(i) + trendVal(i + 1)) / 2) * 30
  - trendVal(i) – **pixelColArr**[i] > TH

- Pixel Column Noisyness
  - Reject pixel noise and low occupancy cases
  - TH = trendVal(i) * 1.5
  - **pixelColArr**[i] > TH



Inputs & outputs

- Repository: https://github.com/CMSTrackerDPG/PixelPhase1Scripts/tree/master/InefficientDoubleROC
- Call: python idr.py <Online DQM file>
- Two separate text files are created
  - inefficientDPixelColumns__XXXXXX.out
  - noisyPixelColumns_XXXXXX.out
    - Where XXXXXX is the runnumber deducted from the input file name
- Content of the files (divided in layer/disk sections):
  - Module Name
  - 2D histogram coordinates to ROC number mapping
  - Value which is above the TH, current TH

Use this tool wisely!

- This tool is not 100% efficient
    - But did my best to make it as good as possible (magic numbers in threshold calculation)
        - But if you think you can tune it even better (and have a lot of spare time) feel free to improve it
        - Or you think you have an idea how to improve it but have no time – share your idea with me and I will check it
- Best detection results are provided by high occupancy runs
- If it happens that you will manually find a problem that is not listed in logs or there are false positives
    - Switch to the run with higher module occupancy
    - See the first bullet