



# sucimaPix

A software suite for pixel  
detector data analysis

Antonio Bulgheroni (INFN – Roma III)

# Contents

- A series of questions: what's that sucimaPix?  
When will it be ready? What do I need to run it?  
Where can I read more? How difficult is to  
analyse my own pixel detector? How does it  
work? What's next?
- An example...
- Basic idea and working principle
- Code structure

# What's that?

- **sucimaPix** is data analysis **software framework** providing all the general and standard tools required by **pixel detector characterization**
- It's **C++ coded** and exploits the advantages of **Object Orientation**
- It's fully **embedded in Root**: can be used as a standalone program or runned interactively from a Root shell.
- It can be executed in batch non-interactive mode using console commands or via a newly implemented GUI.

# When will it be ready?

- **It's available right now!** Just download, compile and enjoy it!
- **It was born 2 years ago** within the SUCIMA collaboration (mainly developed and used in Como) to analyse data taken with M5, MimoRoma, Successor5, MimoTera and now also on **Mimo\* 2**
- It was the first attempt to move **from the standard Paw/Fortran** environment **toward the new (and future) Root/C++** one
- Since then, **it evolved very much** and pretty quickly becoming **a stable and unique software tools** in our labs.

# What do I need to run it?

- You need a **personal computer with Root** (better if  $> 5$ ) installed.
- sucimaPix can be compiled under any operating system where Root is compilable (**Linux, Windows, Mac Os**). It has been successfully ported under Windows in one night only!
- In principle you can download and use binaries but **compiling the source is recommended**.
- The development and the stable CVS snapshots can be downloaded from our server (address at the end).
- ... And of course you need a **sample of data to analyze**

# Where can I read more?

- sucimaPix is well documented and the documentation is available on the web:  
<http://sucimaweb.dipscfm.uninsubria.it/doc/sucimapix>
- The documentation is done using the Root style (THtml) and includes also Root classes.
- You are invited to report bugs to our BugZilla  
<http://sucimaweb.dipscfm.uninsubria.it/bugzilla>
- You can subscribe to the developers mailing list  
[sucimapix-dev@sucimaweb.dipscfm.uninsubria.it](mailto:sucimapix-dev@sucimaweb.dipscfm.uninsubria.it) to stay tuned with the latest release.

# How difficult is to run it with my sensor?

- If your **DAQ output data format is already compatible** with the sucimaPix one (the LEPSE DAQ is supported) then it's trivial: **just fill in the configuration file**
- sucimaPix is a **completely general environment**. The source has been written in a **dynamic way**, so there are no limitation in the size and number of pixels in your detector.
- If your **DAQ output is not supported**, a new 'converter' has to be developed.
- All the **"standard" histograms are provided** but, if you know a little of Root and some basics of C++, you can easily add histos and personalize them.

# How does it work?

## Initialization phase:

Data are converted in the sucimaPix format and detector/run information are permanently stored in the file

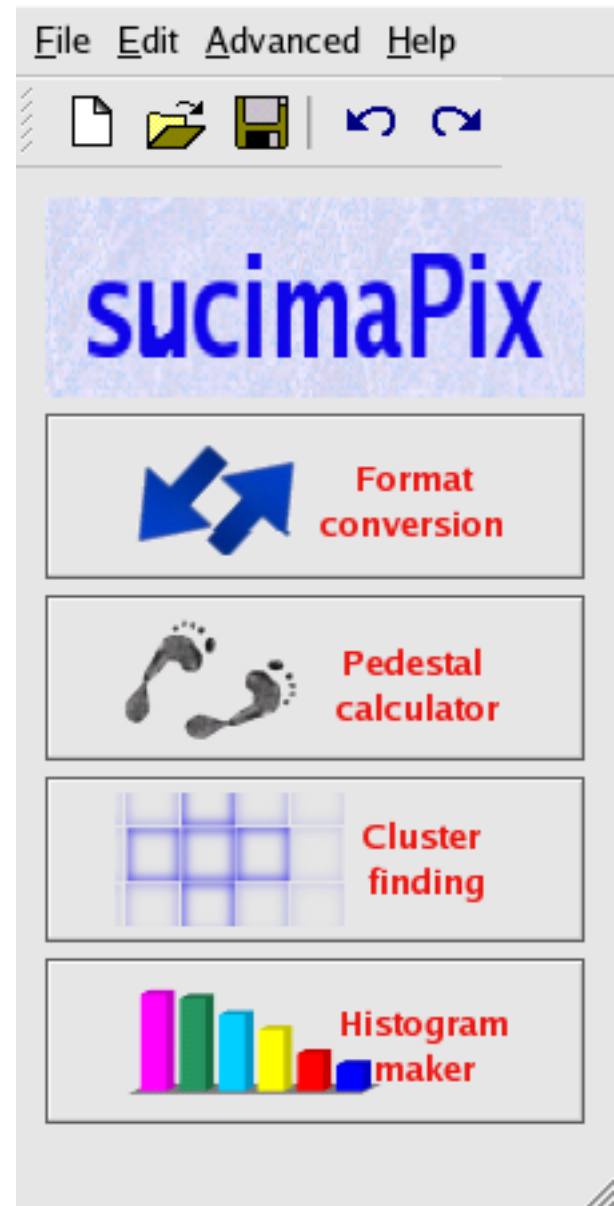
## Pedestal and noise calculation

## Data reduction phase

Input data are scanned looking for clusters and only useful information is stored on disk

## Histogramming

All the standard histograms are booked and filled





# Initialization phase

1

- Data coming from the DAQ are converted in a TTree of TPixelMatrix.
- Also detector and run information are saved in the file
- This is the slowest part of the software execution and may seem to be useless if only one kind of detector and one DAQ is used. Moreover it is compressing data! Real case: 3.2 GB of Mimo\*2 data compressed to 768 MB!
- It is providing a common entry point for all the rest of the analysis that can proceed in the same way regardless the specific detector and DAQ specifications.

# Initialization phase

1

LEPSI DAQ | SUCIMA DAQ | MIMOTERA | STRUCK DAQ

Run identifier

Run number	Description	First event	Last event	x [mm]	y [mm]	z [mm]
///						

Add run...  Use filename suffix

Remove run

Pixel detector

Pixel along X: 132 Pixel along Y: 128

Pixel pitch X: 30 Pixel pitch Y: 30

Total Number of entries: 16896

Signal polarity:  Positive  Negative

Use lookup table: star2LookUp.dat

DAQ and Post-processing

Operation mode: RAW\_CDS\_nm

Post processing: CDS

Linear combination options

Frame to combine: 4

Filename: shaping\_weight.conf

Oscilloscope options

Frame to skip: 2

Sample to skip: 13

Sub-matrix structure

Bottom left X	Bottom left Y	Top right X	Top right Y
---------------	---------------	-------------	-------------

New browser

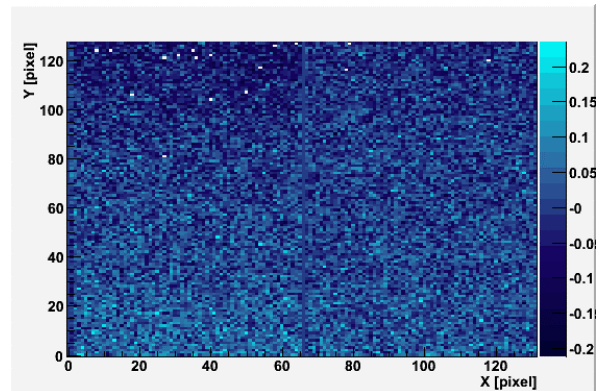
Close

# Pedestal and noise calculation

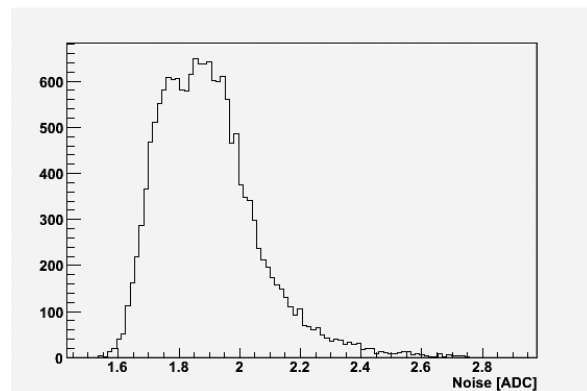
2

- For each pixel both the pedestal and noise are calculated from a run without source. For the future, also automatic pedestal estimation and tracking are foreseen.
- The noise of each pixel is crucial because all the following “cuts” are given in SNR units.

Pedestal map



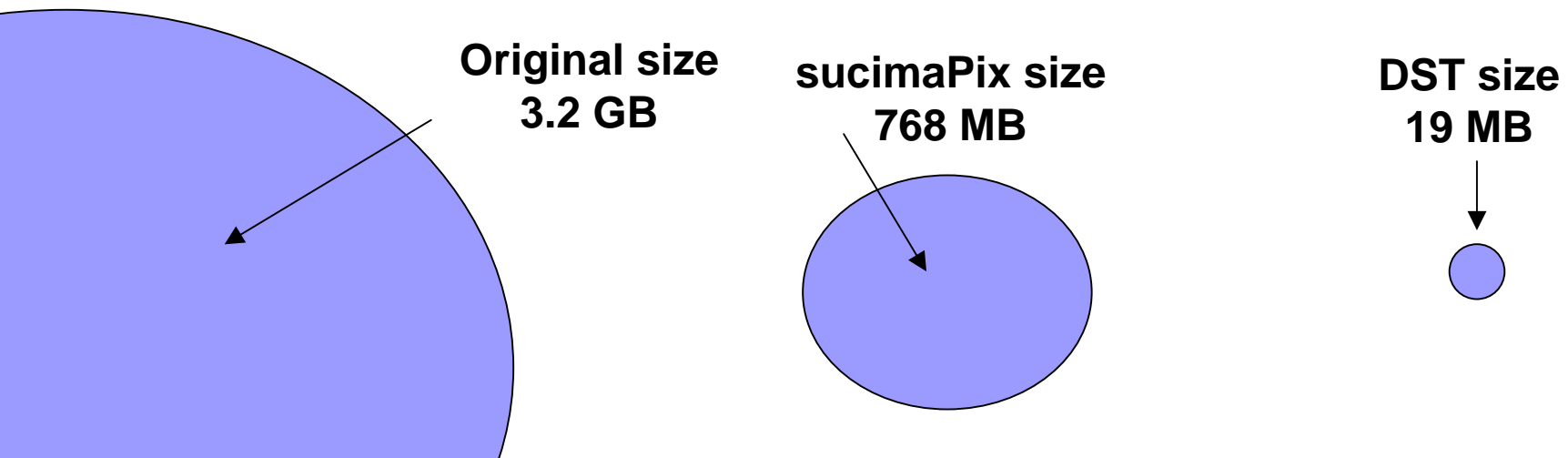
Noise distribution



# Data reduction and DST production

3

- Run data are pedestal sub'ed and common mode corrected.
- The frame is scanned for clusters. Two different algorithms are available: standard fixed frame ( $n \times m$  pixels) and a recursive one.
- The recursive algorithm is faster and more general...



# Data reduction and DST production

3

File identifier

	File name	Pedestal file	First event	Last event
1	run_313.root	run_312.ped	0	2000

Use filename suffix

Fill in the detector low level debug histos  
 Dump the DST into an exchange binary file  
 Pedestal update  
Update the pedestal every  events

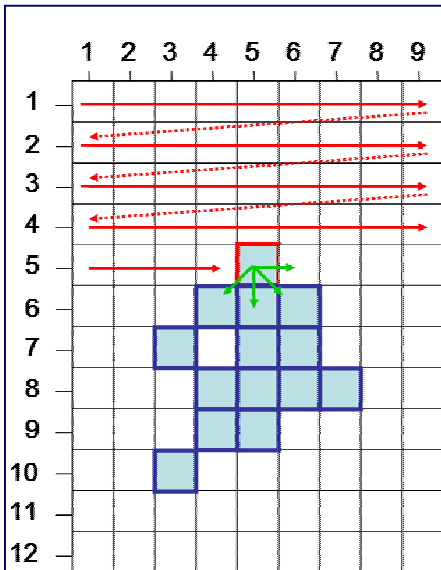
Single pixel spectrum

Common mode suppression  
Hit rejection cut in SNR   
Common mode limit

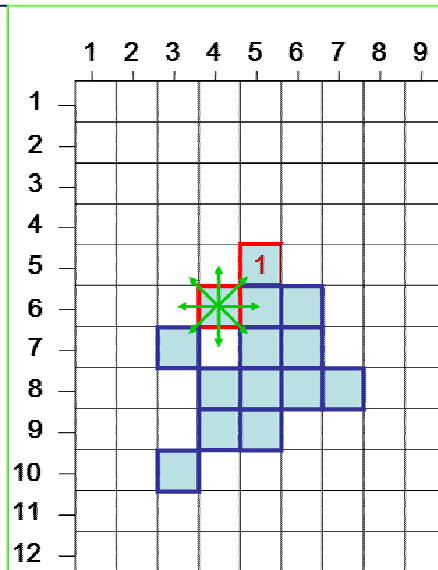
Cluster search options  
Seed pixel cut   
Search algorithm

Fixed frame size  
Along X   
Along Y

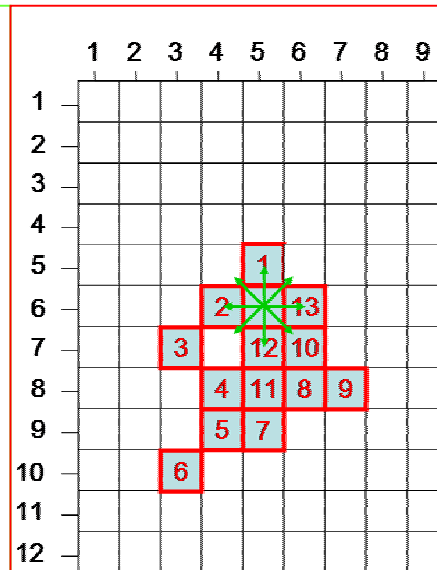
# Recursive cluster search



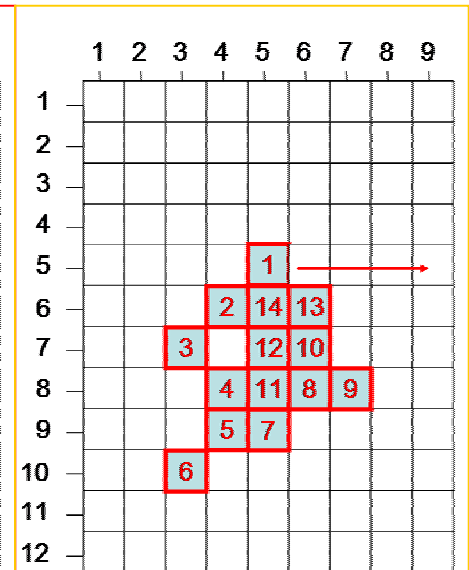
Matrix scanned until a first pixel above threshold is found. The first 4 copies of *Add&Check* are generated.



If the added pixel is found above threshold, *Add&Check* generates other 8 copies of itself for neighbouring pixels



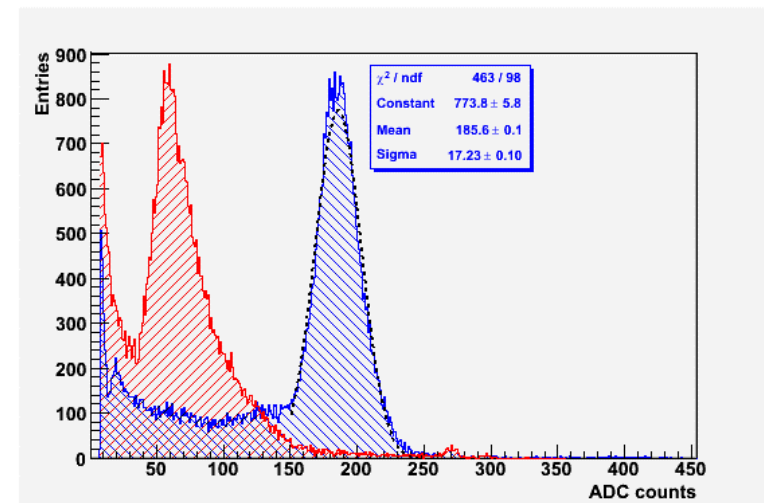
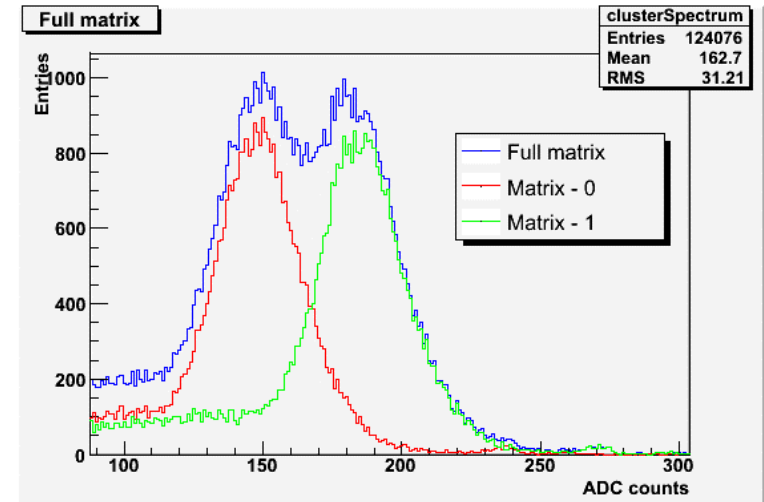
If *Add&Check* doesn't find anything else to add, it returns to a previous level until all the copies exit.



The matrix scanning can restart from where it was stopped.

# Histogramming 4

- With the information stored in the DST, producing histos is very simple and extremely quick. In this way you can easily compare the results of different cuts.
- This last phase can be easily personalized in order to build your favourite histos and distributions.




# Histogramming

4

Calculation status / log

- Analysing frame 0/36787
- Analysing frame 3678/36787
- Analysing frame 7356/36787
- Analysing frame 11034/36787
- Analysing frame 14712/36787
- Analysing frame 18390/36787
- Analysing frame 22068/36787



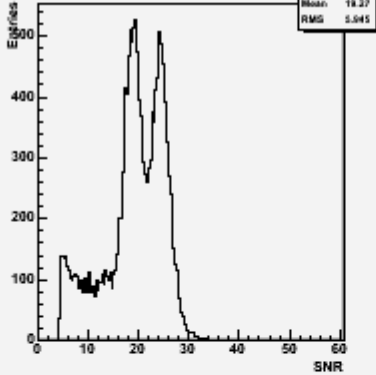
On line histos

Histo name

- clusterNoise
- clusterNumber
- clusterPHVsSize
- clusterSize
- clusterSNR
- clusterSpectrum
- hitsHisto
- hitsHistoMetric

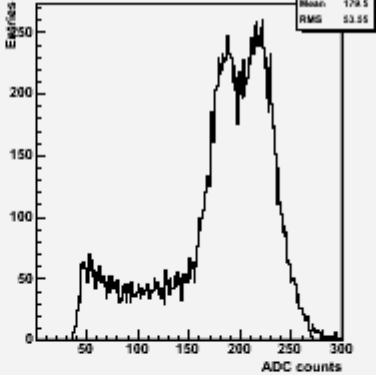
Draw option

Cluster Signal to Noise Ratio




ClusterSNR  
Entries: 11634  
Mean: 19.27  
RMS: 5.645


Source spectrum



ClusterSpectrum  
Entries: 24342  
Mean: 179.5  
RMS: 53.55

Calculation progress

Event per file  59%

File to be analyzed  0%

Save log

Browse results...

Done



# Source code structure

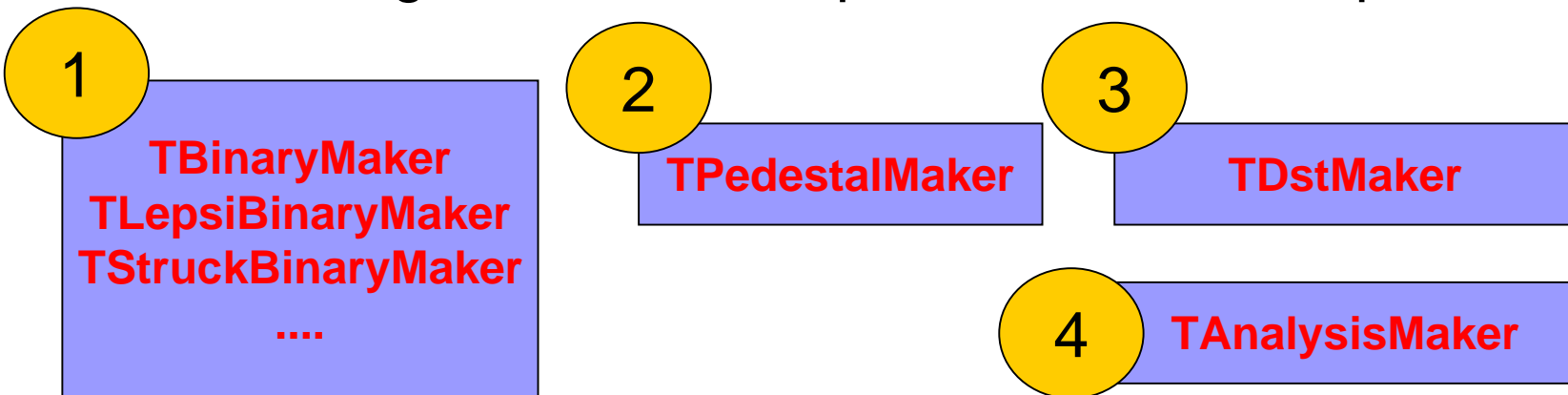
- Three shared libraries that can be also dynamically loaded into ROOT

**libEvent.so (.dll)**

**libAnalib.so (.dll)**

**libSpeciallib.so (.dll)**

- Among sucimaPix classes there are some of them inheriting from TMaker specialized in result production



# How does the GUI work?

- The Qt GUI has been prepared using the Designer tool freely provided by Trolltech and the integration of ROOT has been done according to the Chapter 9 of ROOT User's guide.
- The link among the TMaker classed and the GUI has been implemented using the SIGNAL/SLOT mechanism available in both Qt and ROOT
- The ROOT Canvas can be embedded into a Qt GUI!

# What's next?

- Integrate the sucimaPix framework in a telescope environment in order to deal with many sensors making available 3D coordinates for track reconstruction.
- Multi-threading for // processing
- As an intermediate step, integration of INFN strip telescope.
- On a best effort basis, your wish list...

# Anonymous CVS

CVSROOT=:pserver:anonymous@sucimalab.dipscfm.uninsubria.it/root

cvs logging (when prompted for password type sucimaPix)

cvs checkout -r v2-0-10 -P sucimapix

Latest stable release is v2-0-10.

The development release is v2-0-11.

Send me ([antonio.bulgheroni@roma3.infn.it](mailto:antonio.bulgheroni@roma3.infn.it)) to join the sucimapix-dev mailing list

# Conclusions

- sucimaPix, a software suite, Root-embedded has demonstrated itself to be stable and can profitable used for pixel detector characterization.
- The C++ OO code has been carefully written in order to be fully dynamic and portable.
- The documentation is produced while writing the code and available on line.
- The user and developer communities need to be enlarged.