The Proteus beam telescope reconstruction software

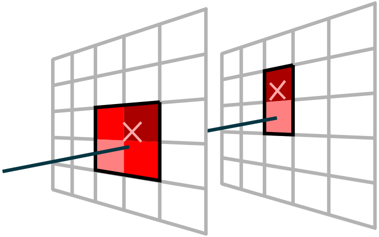
Moritz Kiehn (msmk@cern.ch)
Université de Genève
Département de physique nucléaire et corpusculaire

BTTB6, Zürich, 18.01.2018

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES

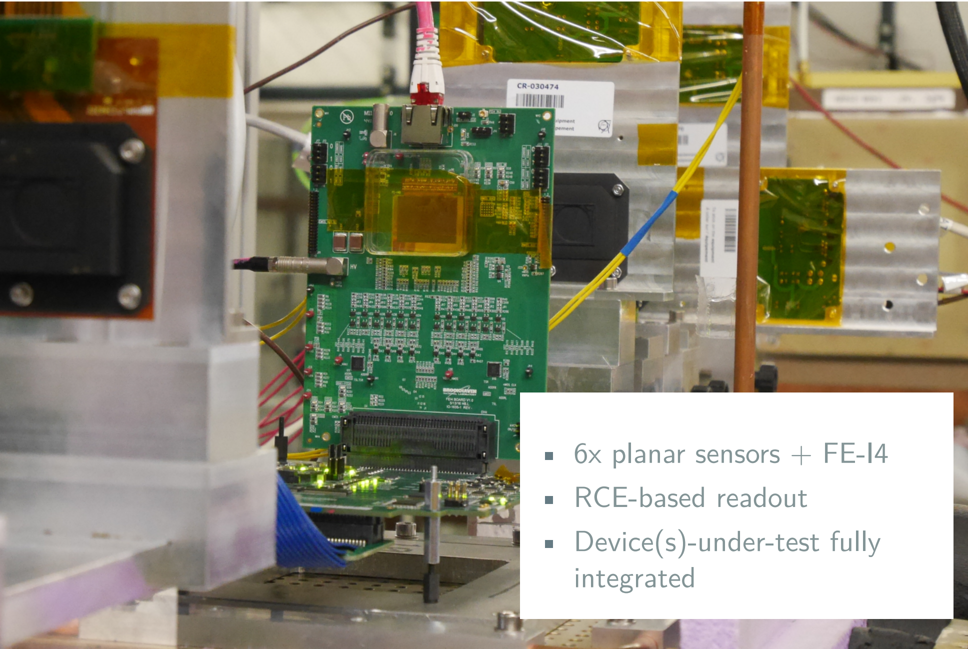
What is this for?

```
0xfa34 0xffaa
0x12ea 0x4223
0xabcd 0xfedc
0x00ff 0xff00
...
```



Raw data → Fully reconstructed events

Context: UNIGE FE-I4 telescope



- 6x planar sensors + FE-I4
- RCE-based readout
- Device(s)-under-test fully integrated

Overview

Goals

- Optimized for typical pixel telescope setups
- Raw data to reconstructed tracks
- Originally: Judith software

  `G. McGoldrick et al, NIM A765 140–145`

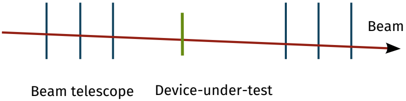
→ Simplified pipeline

→ Optimized data model

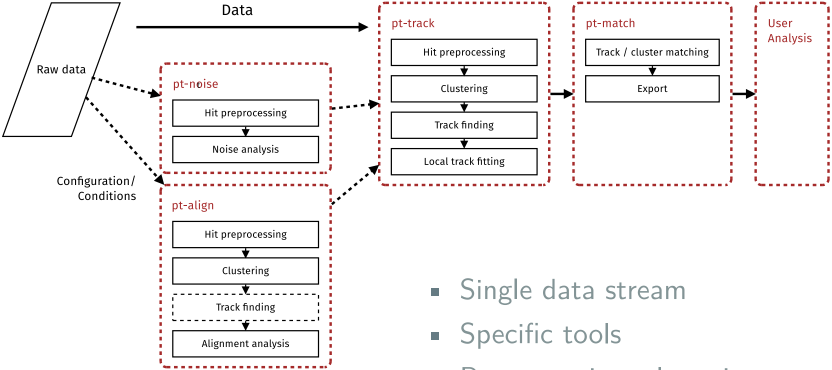
→ Reduced manual intervention

Non-goals

- General reconstruction framework
- Full analysis framework



Analysis flow



- Single data stream
- Specific tools
- Preprocessing, clustering, ...on-the-fly

Common configuration format

Device

```
[sensor_types.fei4-si]
cols = 80
rows = 100
```

Geometry

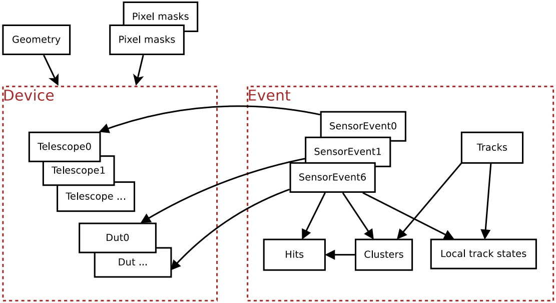
```
[[sensors]]
id = 2
offset = [0.0, 0.0, 100.3]
```

Analysis

```
[track]
sensor_ids = [0,1,2,3,4,5]
search_sigma_max = 5.0
```

- TOML file format, ini-like
- Separate static (device, analysis) and dynamic (pixel masks, alignment) configuration

Telescope-specific data model



Common process loop

1. Read event
2. Run all processors, e.g. clusterizer, tracker
3. Run all analyzers, e.g. occupancy, correlations
4. Store event

All information is local.

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES

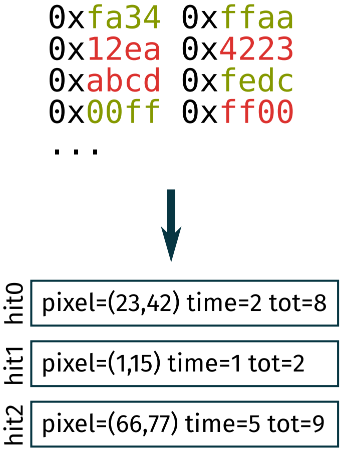
Raw input data

Supported formats

- RCE ROOT data
- Timepix3 SPIDR data

  `M. Williams (CLICdp)`
- EUDAQ raw data

  `via EUDAQ library decoder`



```
0xfa34 0xffaa
0x12ea 0x4223
0xabcd 0xfedc
0x00ff 0xff00
...
```

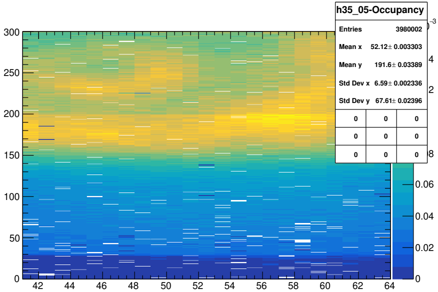
hit0 pixel=(23,42) time=2 tot=8

hit1 pixel=(1,15) time=1 tot=2

hit2 pixel=(66,77) time=5 tot=9

Noise scan

Occupancy



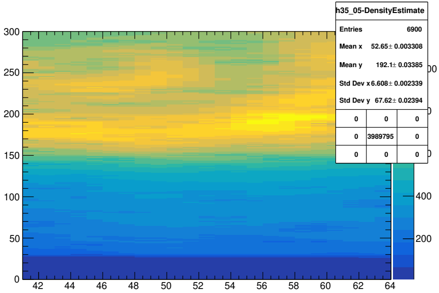
Occupancy inhomogeneity, e.g. due to beam profile

Local noise estimation

1. Calculate expected local occupancy via kernel density estimator
2. Calculate local significance
3. Cut

Noise scan

Expected occupancy



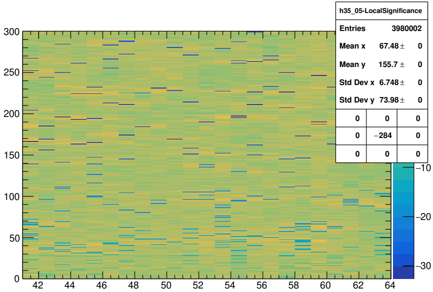
Occupancy inhomogeneity, e.g. due to beam profile

Local noise estimation

1. Calculate expected local occupancy via kernel density estimator
2. Calculate local significance
3. Cut

Noise scan

Local significance



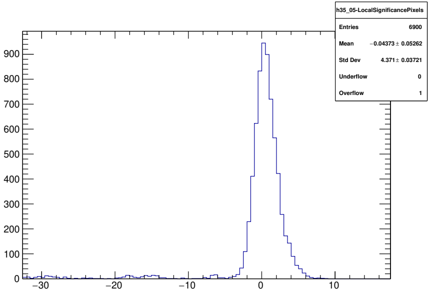
Occupancy inhomogeneity, e.g. due to beam profile

Local noise estimation

1. Calculate expected local occupancy via kernel density estimator
2. Calculate local significance
3. Cut

Noise scan

Local significance



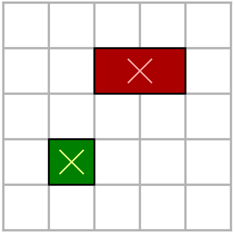
Occupancy inhomogeneity, e.g. due to beam profile

Local noise estimation

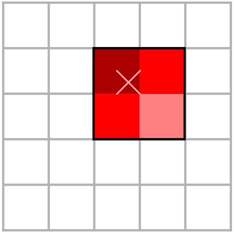
1. Calculate expected local occupancy via kernel density estimator
2. Calculate local significance
3. Cut

Clustering

Binary

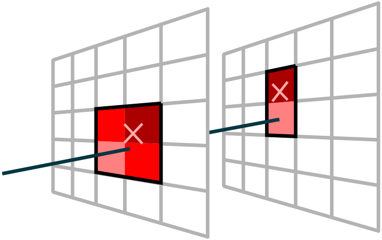


Center-of-gravity



- Greedy cluster finder
- Geometric cluster properties
- Time-over-threshold, binary, fastest hit
- Configurable per sensor

Geometry

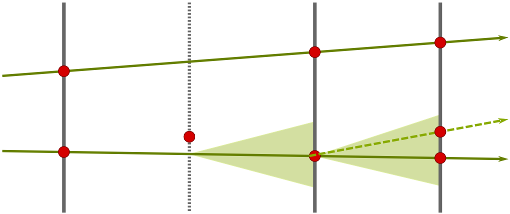


- Orientation defined via unit vectors

$$\vec{u} = (1, 0, 0)$$
$$\vec{v} = (0, 0.9, 0.1)$$

- Always reconstruct in local coordinates
- All transformations on-the-fly

Tracking



- Road-based track finder
- Global ambiguity solver based on χ^2, N_{hits}
- Linear track model
- Results in local coordinate system

Alignment

Track residual histograms

Measured (unbiased) residuals:

$$\begin{pmatrix}\epsilon_u\\ \epsilon_v\end{pmatrix} = \begin{pmatrix}1 & 0 & -v\\ 0 & 1 & u\end{pmatrix}\begin{pmatrix}\Delta u\\ \Delta v\\ \Delta\gamma\end{pmatrix}$$

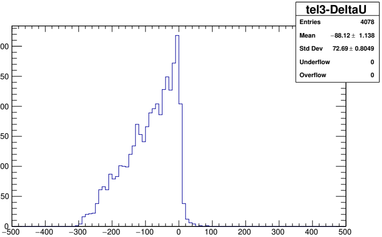
Solution in χ^2 sense:

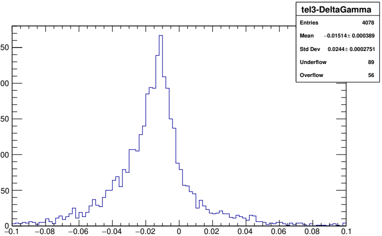
$$\Delta u \sim \epsilon_u(1-u^2)+\epsilon_v uv$$

$$\Delta v \sim \epsilon_v(1-v^2)+\epsilon_u uv$$

$$\Delta\gamma \sim \epsilon_v u-\epsilon_u v$$

Unaligned distributions





Alignment

Track residual histograms

Measured (unbiased) residuals:

$$\begin{pmatrix} \epsilon_u \\ \epsilon_v \end{pmatrix} = \begin{pmatrix} 1 & 0 & -v \\ 0 & 1 & u \end{pmatrix} \begin{pmatrix} \Delta u \\ \Delta v \\ \Delta \gamma \end{pmatrix}$$

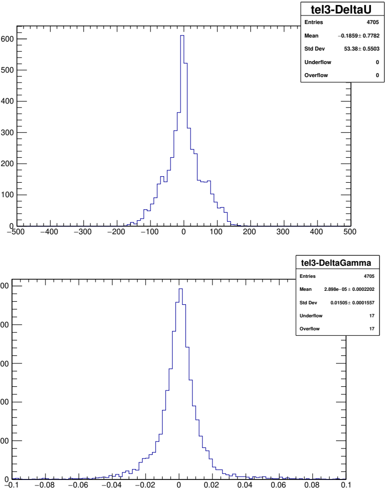
Solution in χ^2 sense:

$$\Delta u \sim \epsilon_u (1 - u^2) + \epsilon_v uv$$

$$\Delta v \sim \epsilon_v (1 - v^2) + \epsilon_u uv$$

$$\Delta \gamma \sim \epsilon_v u - \epsilon_u v$$

Aligned distributions



Alignment

Track residual histograms

Measured (unbiased) residuals:

$$\begin{pmatrix}\epsilon_u \\ \epsilon_v\end{pmatrix} = \begin{pmatrix}1 & 0 & -v \\ 0 & 1 & u\end{pmatrix}\begin{pmatrix}\Delta u \\ \Delta v \\ \Delta\gamma\end{pmatrix}$$

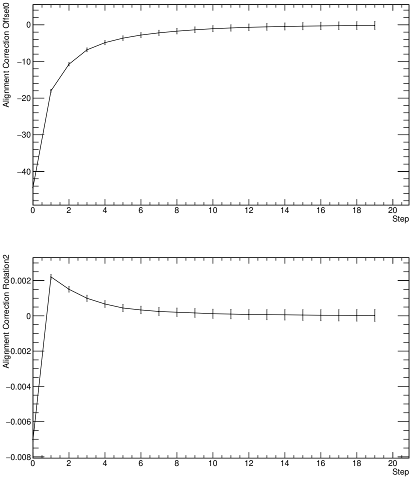
Solution in χ^2 sense:

$$\Delta u \sim \epsilon_u(1-u^2) + \epsilon_v uv$$

$$\Delta v \sim \epsilon_v(1-v^2) + \epsilon_u uv$$

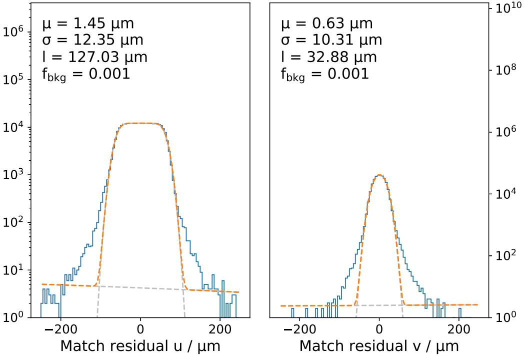
$$\Delta\gamma \sim \epsilon_v u - \epsilon_u v$$

Iterative procedure



Tracking performance

Unbiased residuals



- FE-I4 telescope, 6 planes, 250 μm × 50 μm pitch
- ATLASPix1 dut, 130 μm × 40 μm pitch
- SPS H8 beam line

Computational performance

Example: pt-track

Step	Time/µs/event
Input	15
Processing	190
Analyzers	53
Output	55
Total	313

- Example run with 100k events
- Total time $\mathcal{O}(1\,\mathrm{min})$
- Local machine:
  Core i5 CPU, solid state storage

→ Full campaign reprocessing:
$\mathcal{O}$(hours) on LXBATCH

Disclaimer: varies for occupancy, machine, ...

Upcoming Features

- Event stream merging, e.g. independent device-under-test, Timepix telescope

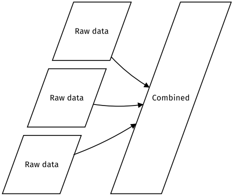
  `preliminary support already available`

- Full 3-D alignment

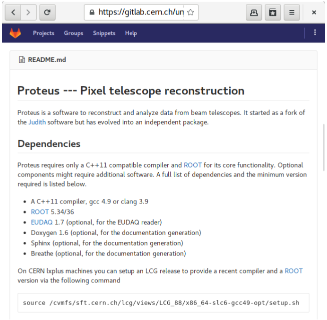
  `based on arXiv:physics/0306034`

- GeneralBrokenLines-based tracking

  `Summer student S. Maqbool`



Where to get it



Proteus --- Pixel telescope reconstruction

Proteus is a software to reconstruct and analyze data from beam telescopes. It started as a fork of the Judith software but has evolved into an independent package.

Dependencies

Proteus requires only a C++11 compatible compiler and ROOT for its core functionality. Optional components might require additional software. A full list of dependencies and the minimum version required is listed below.

- A C++11 compiler, gcc 4.9 or clang 3.9
- ROOT 5.34/36
- EUDAQ 1.7 (optional, for the EUDAQ reader)
- Doxygen 1.6 (optional, for the documentation generation)
- Sphinx (optional, for the documentation generation)
- Breathe (optional, for the documentation generation)

On CERN lxplus machines you can setup an LCG release to provide a recent compiler and a ROOT version via the following command

```
source /cvmfs/sft.cern.ch/lcg/views/LCG_88/x86_64-slc6-gcc49-opt/setup.sh
```

Conditions

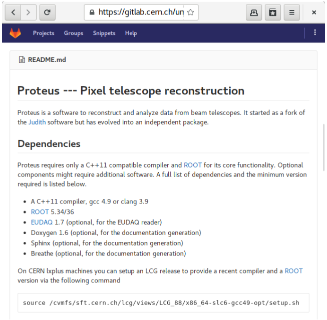
- Requirements: ROOT, CMake, C++11
- MIT/CC-BY-4.0 licensed
- Provided as-is

On Gitlab

- Current version: 1.3.0
- Issue tracker
- Continuous integration

https://gitlab.cern.ch/unige-fei4tel/proteus

Where to get it



Proteus --- Pixel telescope reconstruction

Proteus is a software to reconstruct and analyze data from beam telescopes. It started as a fork of the Judith software but has evolved into an independent package.

Dependencies

Proteus requires only a C++11 compatible compiler and ROOT for its core functionality. Optional components might require additional software. A full list of dependencies and the minimum version required is listed below.

- A C++11 compiler, gcc 4.9 or clang 3.9
- ROOT 5.34/36
- EUDAQ 1.7 (optional, for the EUDAQ reader)
- Doxygen 1.6 (optional, for the documentation generation)
- Sphinx (optional, for the documentation generation)
- Breathe (optional, for the documentation generation)

On CERN lxplus machines you can setup an LCG release to provide a recent compiler and a ROOT version via the following command

```
source /cvmfs/sft.cern.ch/lcg/views/LCG_88/x86_64-slc6-gcc49-opt/setup.sh
```

Conditions

- Requirements: ROOT, CMake, C++11
- MIT/CC-BY-4.0 licensed
- Provided as-is

On Gitlab

- Current version: 1.3.0
- Issue tracker
- Continuous integration

Thank You

https://gitlab.cern.ch/unige-fei4tel/proteus

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES