Hands-On: Corryvreckan

The Maelstrom for Your Test Beam Data

Finn Feindt, on behalf of the Corryvreckan developers

11th Beam Telescopes and Test Beams Workshop
April 2023, DESY, Hamburg

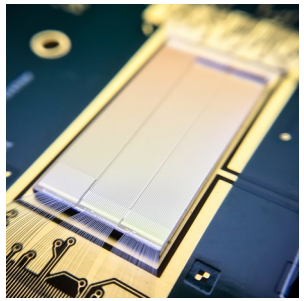
HELMHOLTZ

Based on the work from: Jens Kröger, formerly Heidelberg University & CERN

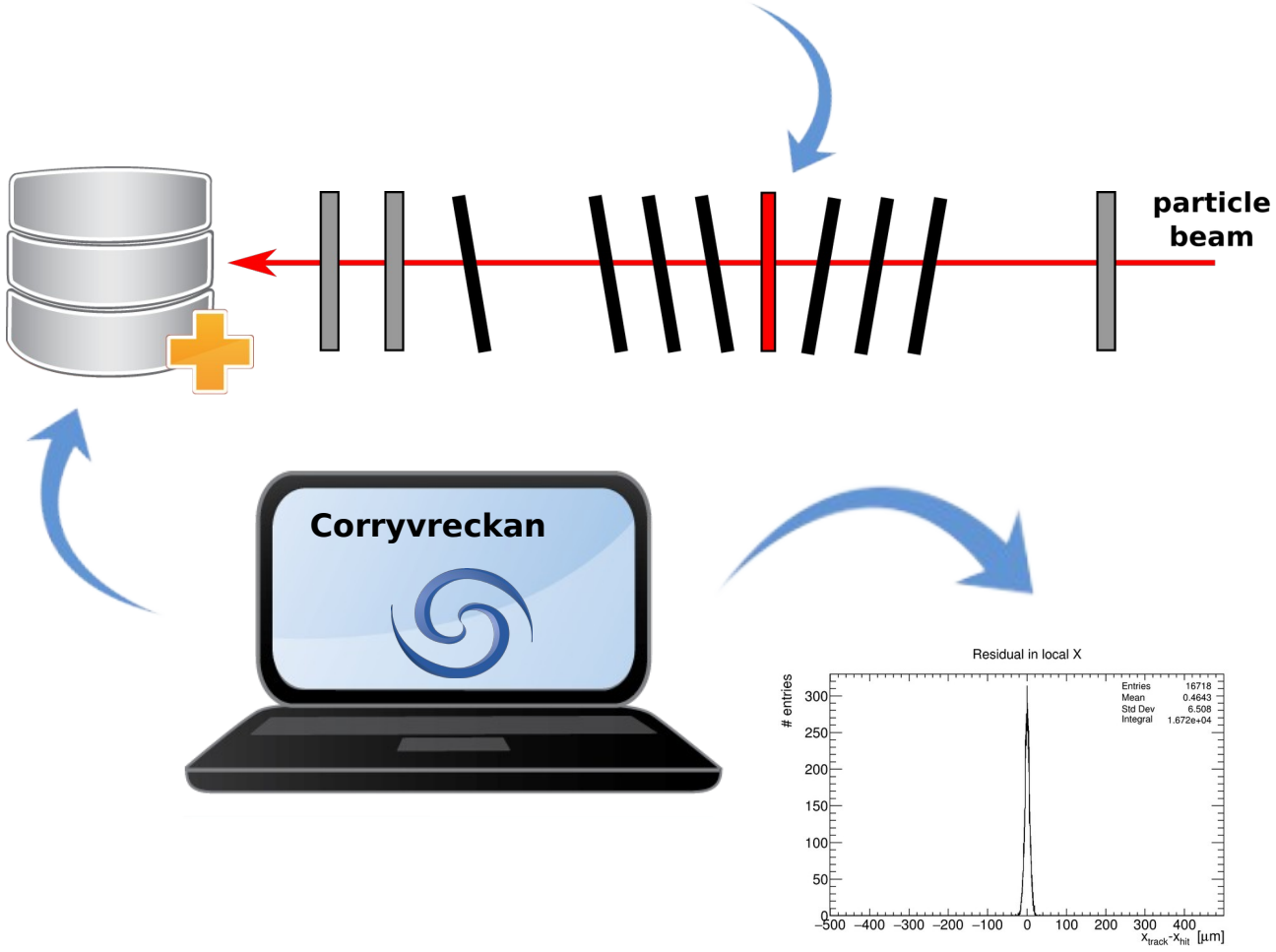
DESY.

Introduction

What are we going to learn today?

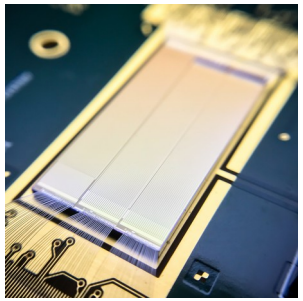


- Introduction
  - What is Corryvreckan?
  - What's new since BTTB10?
- Hands-on:
  - Installation options
  - Setting up an analysis step-by-step
- Please ask questions at any time!

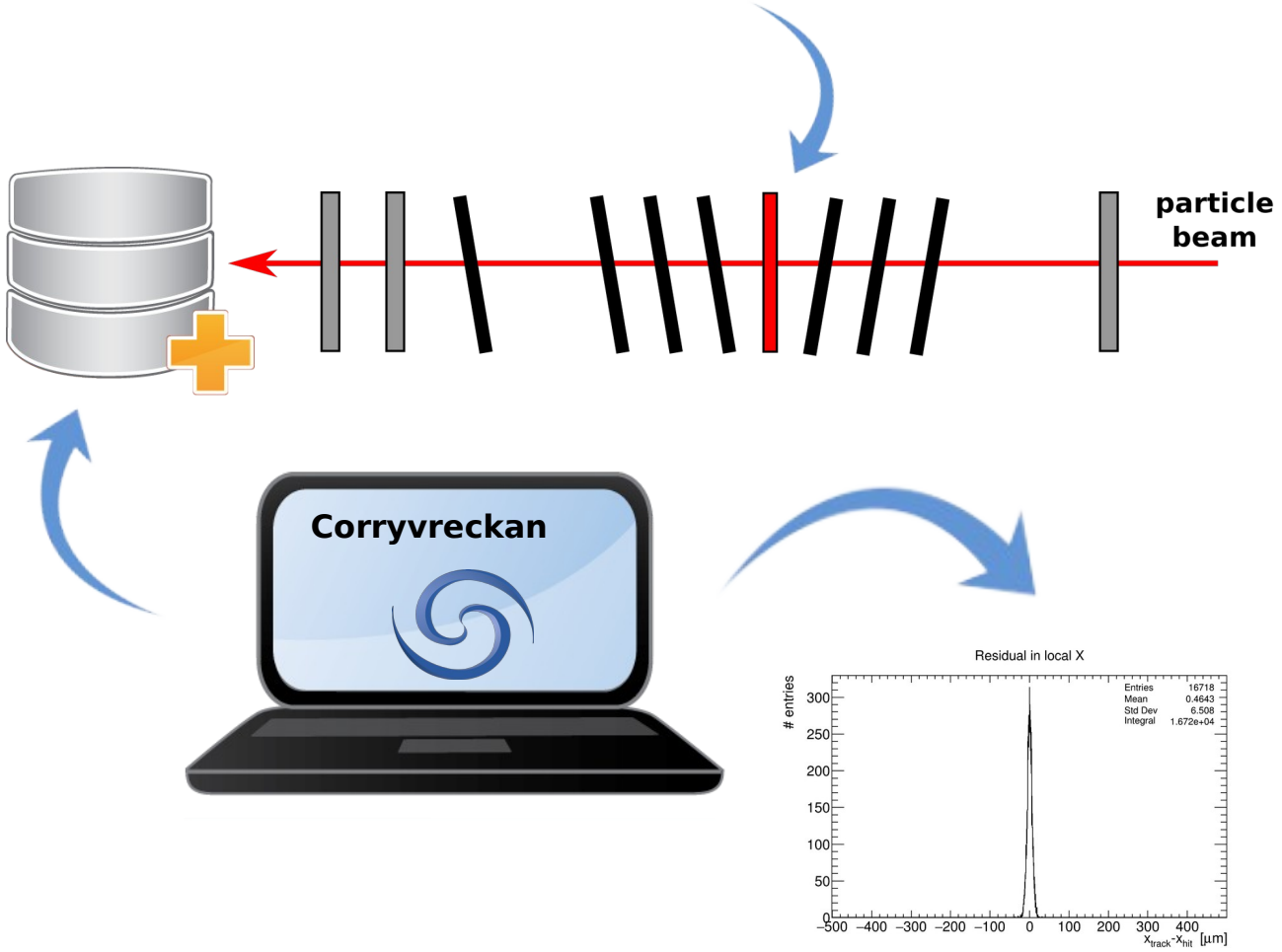


What is Corryvreckan?

A Reconstruction and Analysis Tool for Pixel Sensor Test-Beam Data



- Modular structure
  - Framework core
  - Modules for specific tasks
- Highly flexible and configurable
  - Easy to understand
  - Written in modern C++
  - Comprehensive documentation (> 120 pages!)

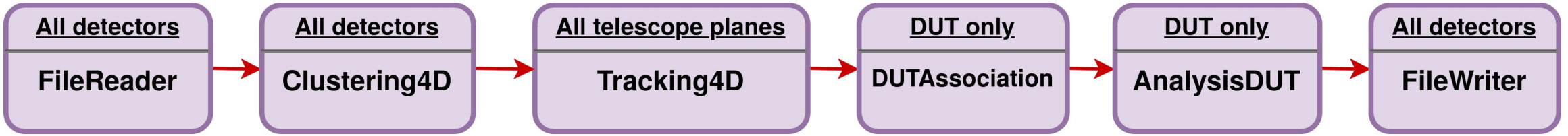


The Modular Approach

Reconstruction Step by Step

- Modular structure:
  - Framework core
  - Implementation of algorithms `[Modules]`
- Modules:
  - Algorithms for specific tasks (can be user defined)
  - Objects are stored temporarily:
    - Events, pixels, clusters, tracks

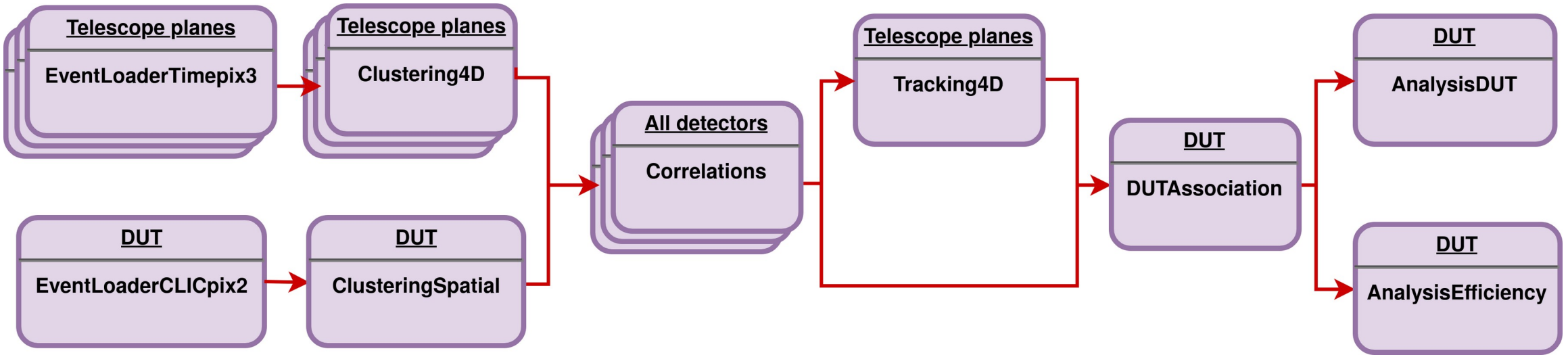
- Select suitable modules for:
  - Event building
  - Clustering
  - Tracking
  - Analysis (also multiple DUTs)
  - …
- Quick to set up and easy to configure



The Modular Approach

Reconstruction Step by Step

- Can create **complex reconstruction chains**
- Apply **different modules to different devices** in the same reconstruction
  - We will do this in **Example 2**:



The Corryvreckan Website

The First Place to Go

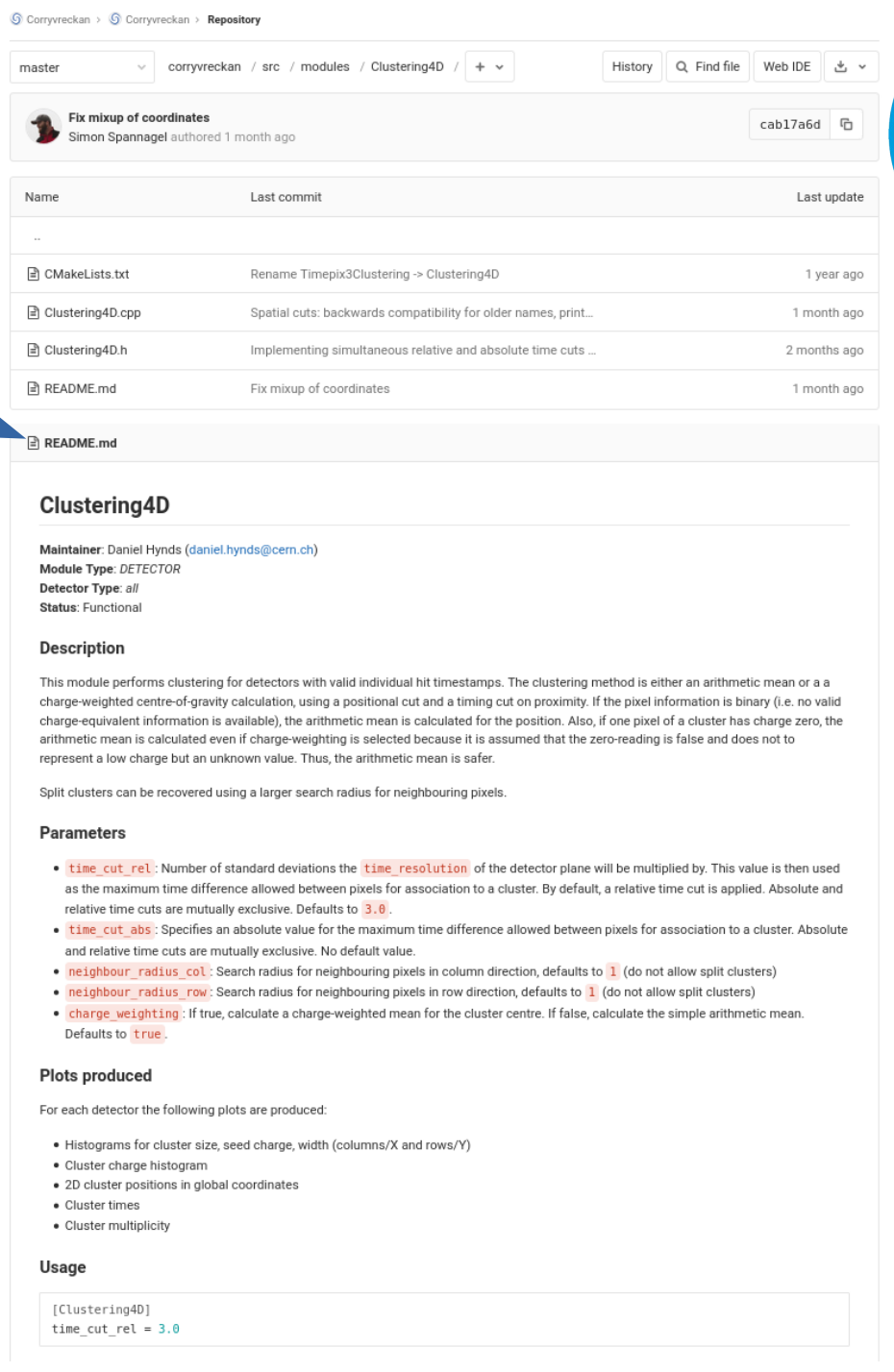
- https://cern.ch/corryvreckan
  - News on releases
  - Installation/Getting Started
- Links:
  - Code repository
  - Issue tracker
  - Forum
  - ...

Documentation

Documentation

Gitlab README, User Manual, Doxygen

- Online documentation in git repository
  - https://gitlab.cern.ch/corryvreckan/corryvreckan
  - Every modules has a README
- Extensive user manual
  - corryvreckan-manual-v2.0.1.pdf
  - Full description of framework
  - Installation instructions
  - “Getting started”, FAQs
  - Module descriptions (fetched from repo)
- Doxygen code reference
  - https://cern.ch/corryvreckan/reference/
  - More details on code



Corryvreckan › Corryvreckan › Repository

master | corryvreckan / src / modules / Clustering4D / | History | Find file | Web IDE

Fix mixup of coordinates
Simon Spannagel authored 1 month ago — cab17a6d

Name	Last commit	Last update
..		
CMakeLists.txt	Rename Timepix3Clustering -> Clustering4D	1 year ago
Clustering4D.cpp	Spatial cuts: backwards compatibility for older names, print…	1 month ago
Clustering4D.h	Implementing simultaneous relative and absolute time cuts …	2 months ago
README.md	Fix mixup of coordinates	1 month ago

README.md

Clustering4D

Maintainer: Daniel Hynds (daniel.hynds@cern.ch)
Module Type: *DETECTOR*
Detector Type: *all*
Status: Functional

Description

This module performs clustering for detectors with valid individual hit timestamps. The clustering method is either an arithmetic mean or a a charge-weighted centre-of-gravity calculation, using a positional cut and a timing cut on proximity. If the pixel information is binary (i.e. no valid charge-equivalent information is available), the arithmetic mean is calculated for the position. Also, if one pixel of a cluster has charge zero, the arithmetic mean is calculated even if charge-weighting is selected because it is assumed that the zero-reading is false and does not to represent a low charge but an unknown value. Thus, the arithmetic mean is safer.

Split clusters can be recovered using a larger search radius for neighbouring pixels.

Parameters

- `time_cut_rel`: Number of standard deviations the `time_resolution` of the detector plane will be multiplied by. This value is then used as the maximum time difference allowed between pixels for association to a cluster. By default, a relative time cut is applied. Absolute and relative time cuts are mutually exclusive. Defaults to `3.0`.
- `time_cut_abs`: Specifies an absolute value for the maximum time difference allowed between pixels for association to a cluster. Absolute and relative time cuts are mutually exclusive. No default value.
- `neighbour_radius_col`: Search radius for neighbouring pixels in column direction, defaults to `1` (do not allow split clusters)
- `neighbour_radius_row`: Search radius for neighbouring pixels in row direction, defaults to `1` (do not allow split clusters)
- `charge_weighting`: If true, calculate a charge-weighted mean for the cluster centre. If false, calculate the simple arithmetic mean. Defaults to `true`.

Plots produced

For each detector the following plots are produced:

- Histograms for cluster size, seed charge, width (columns/X and rows/Y)
- Cluster charge histogram
- 2D cluster positions in global coordinates
- Cluster times
- Cluster multiplicity

Usage

```
[Clustering4D]
time_cut_rel = 3.0
```

Documentation

Gitlab README, User Manual, Doxygen

- Online documentation in git repository
  - https://gitlab.cern.ch/corryvreckan/corryvreckan
  - Every modules has a README
- Extensive user manual
  - corryvreckan-manual-v2.0.1.pdf
  - Full description of framework
  - Installation instructions
  - “Getting started”, FAQs
  - Module descriptions (fetched from repo)
- Doxygen code reference
  - https://cern.ch/corryvreckan/reference/
  - More details on code

Corryvreckan User Manual

Morag Williams (morag.williams@cern.ch)
Simon Spannagel (simon.spannagel@cern.ch)
Jens Kröger (jens.kroeger@cern.ch)

January 27, 2021

Version v2.0

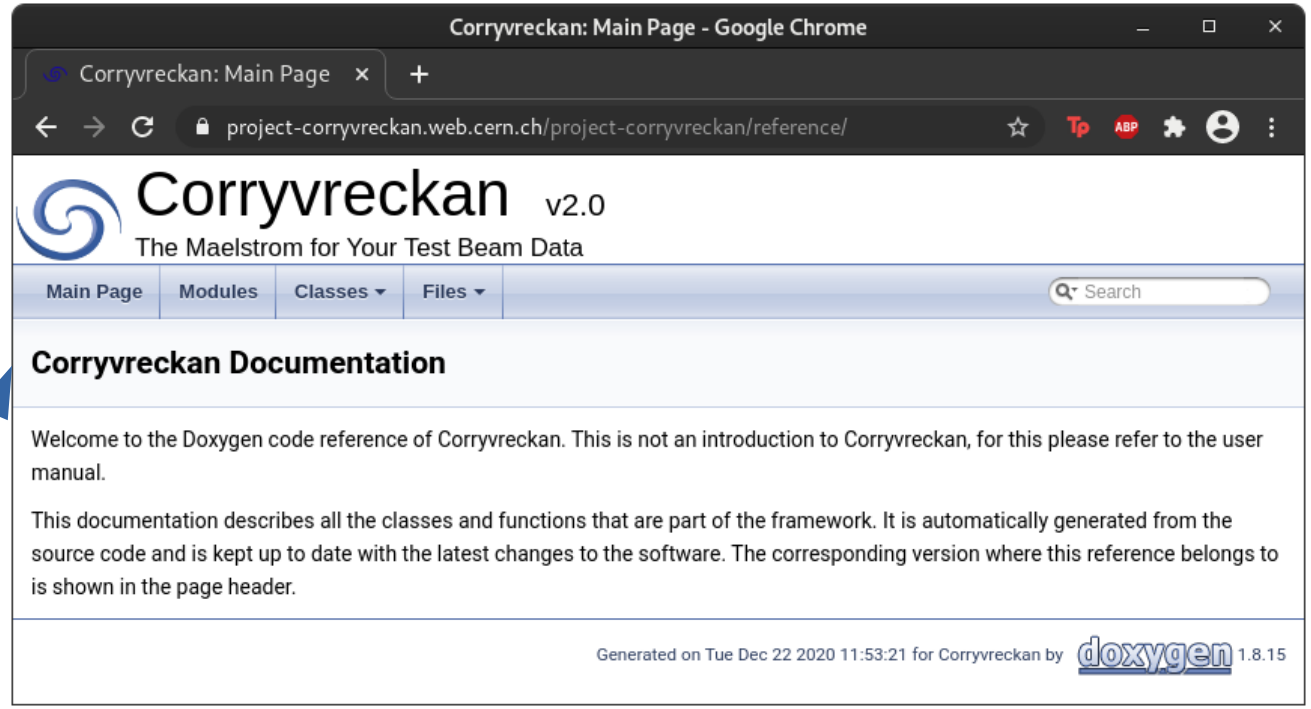
Contents

v

Documentation

Gitlab READMEs, User Manual, Doxygen

- Online documentation in git repository
  - https://gitlab.cern.ch/corryvreckan/corryvreckan
  - Every modules has a README
- Extensive user manual
  - corryvreckan-manual-v2.0.1.pdf
  - Full description of framework
  - Installation instructions
  - “Getting started”, FAQs
  - Module descriptions (fetched from repo)
- Doxygen code reference
  - https://cern.ch/corryvreckan/reference/
  - More details on code

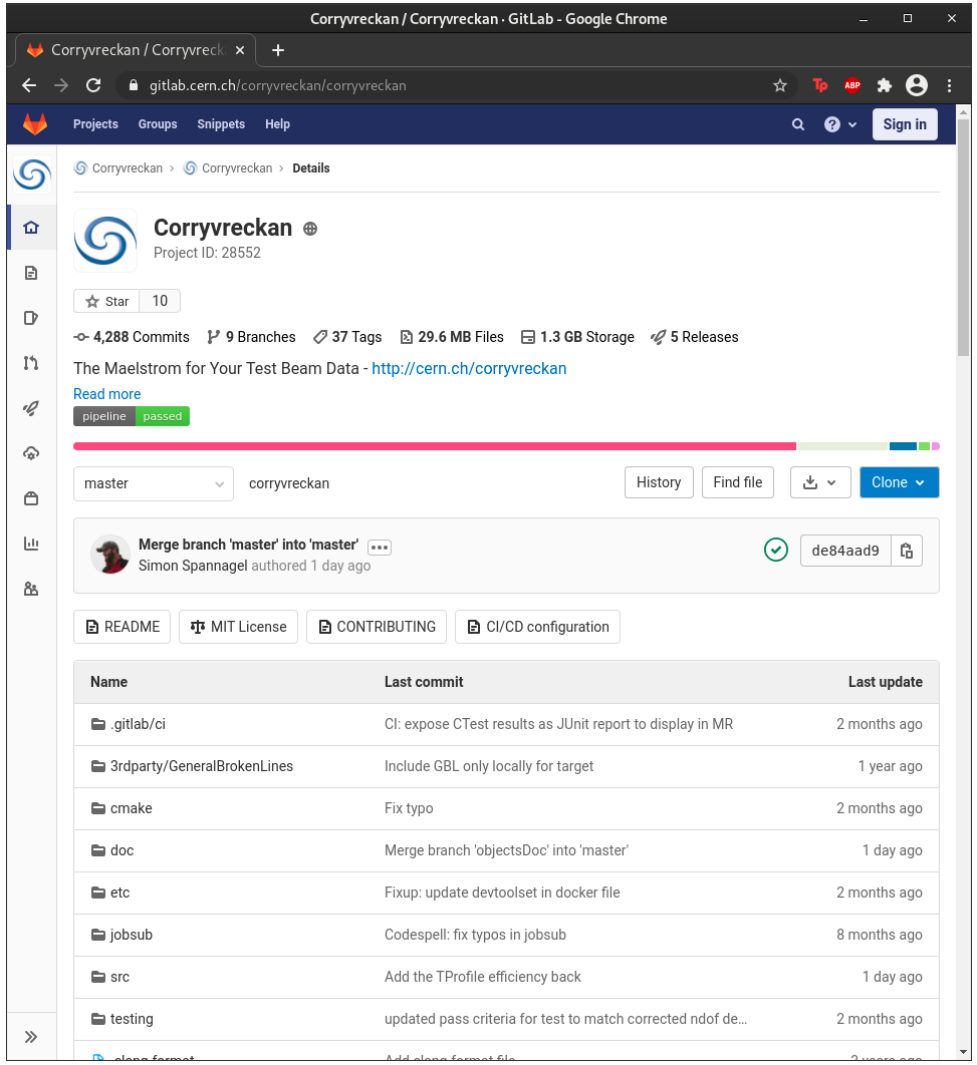


The Corryvreckan Repository

Let's Explore the Repository

https://gitlab.cern.ch/corryvreckan/corryvreckan

- **`doc/`** documentation/user manual
- **`etc/`** lxplus setup script, etc.
- **`jobsub/`** job submission tool + README
- **`src/`** all the source code
  - **`modules/`** code and READMEs of all modules
- **`testing/`** CI test configs
- **`README.md`** Short introduction/getting started
- **`Contributing.md`** contribution guidelines



What's new since BTTB10?

Official Release v2.0 in December 2020 (and first patch release v2.0.1 on 4th February)

- **Reference paper** accepted by JINST:
  - Corryvreckan: A Modular 4D Track Reconstruction and Analysis Software for Test Beam Data arXiv:2011.12730
  - Zenodo entry: https://zenodo.org/record/4384186
- Many updates since then! Checkout the newest version on git! **New Features**:
  - New Module: **`ClusteringAnalog`**: Configurable threshold definitions and clustering algorithms. Especially for analog detectors or detectors with fine charge measurements
  - New Module: **`EventLoaderALiVBaVa`**: Reading data from the ALiBaVa readout system
  - Correcting for DUT movement w.r.t the beam telescope (mismatch of thermal expansion coefficients) by **introducing** time dependent alignment constants
  - **`EventLoaderEUDAQ2`**: Now supports waveforms
  - Major fix: DUT Alignment: Refit track to correctly align DUTs when a GBL is used.
- Work in progress
  - Hexagonal pixels and eta-correction for them, alignment with non-Gaussian residuals, alignment in time

Installation

Let’s Get Started

Installing Corryvreckan

1. **Virtual machine**
2. Download b**inary**
3. Compile from s**ource**
4. Use on **LXPLUS**
5. **Docker** image

Installation

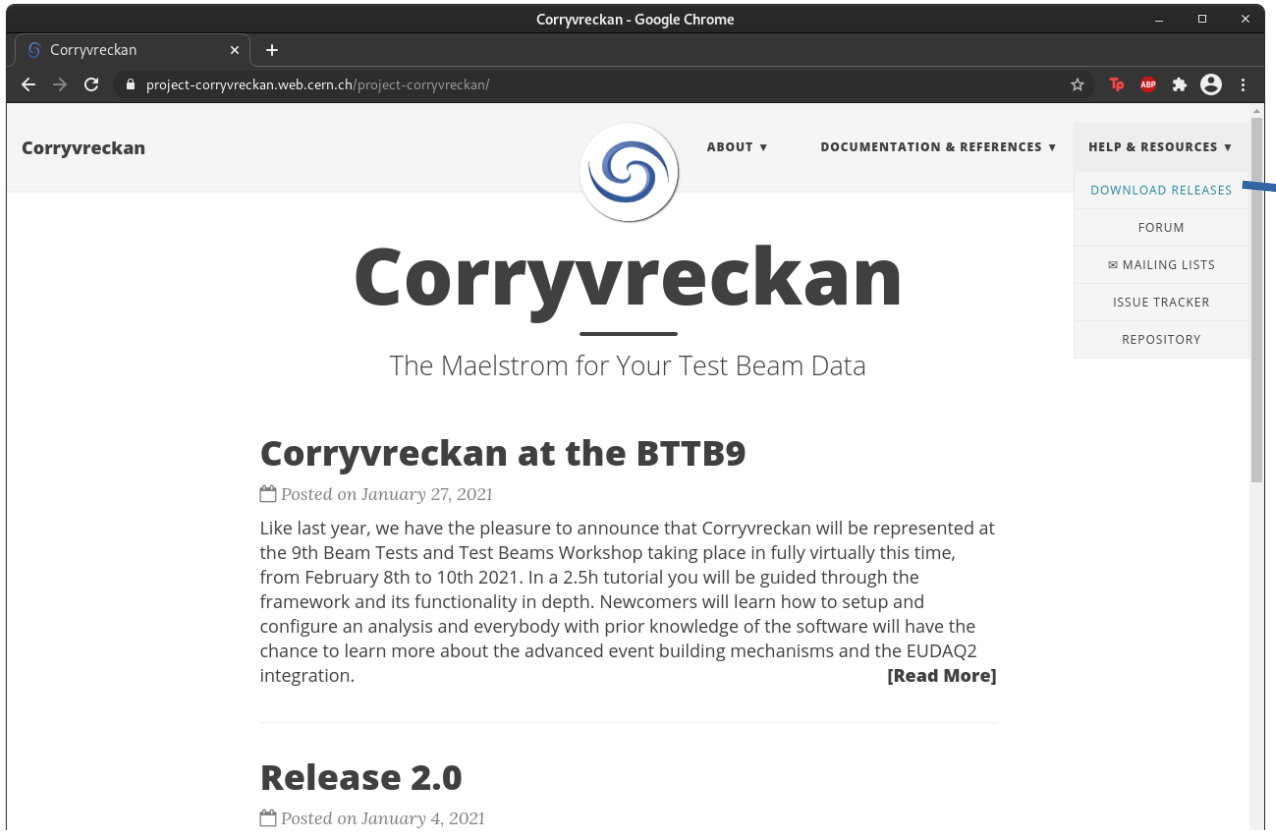
1. Download the Virtual Machine

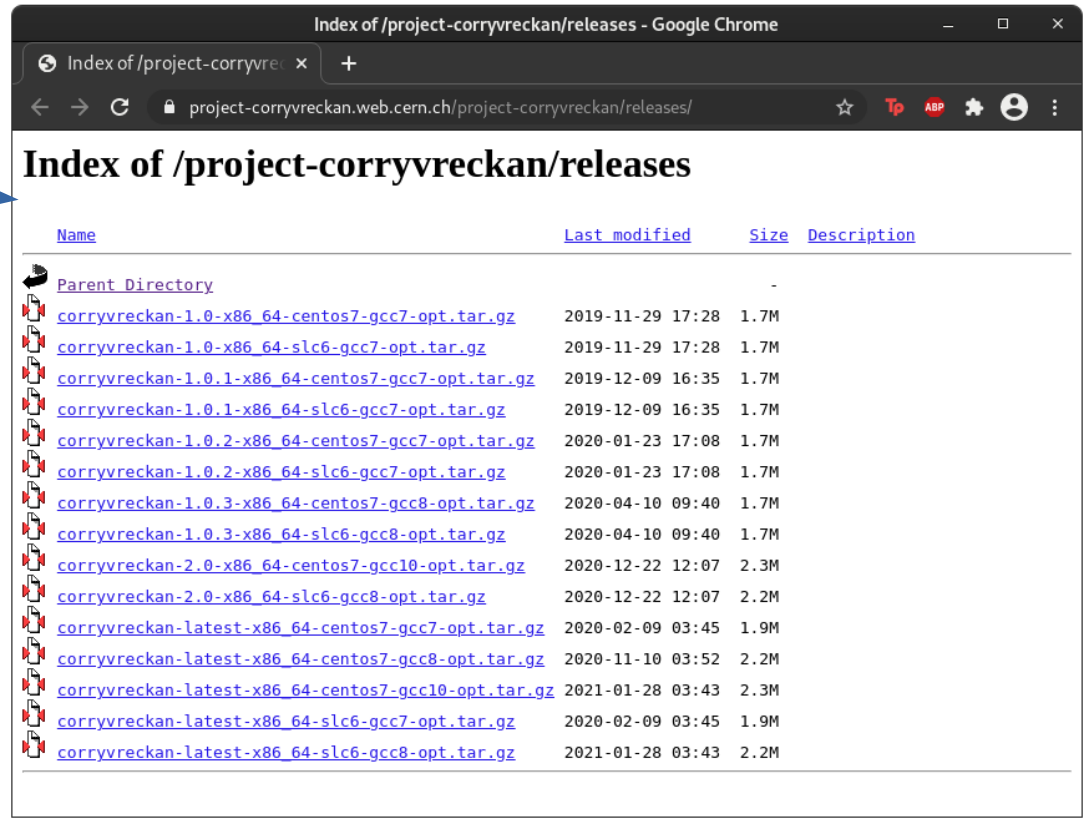
- Works on any operating system with Virtual Box 6.1
- https://www.virtualbox.org/
- Setup instructions here
- Virtual machine image contains everything needed for this tutorial:
  - Corryvreckan v2.0.1
  - All dependencies
  - Data
- **Ideal to get started!**

Installation

2. Download the binary

- For **Centos7** and **SLC6**: Download binary release tarballs
- https://cern.ch/corryvreckan → Help & Resources → Download Releases





Installation

3. Compile From Source Code

- Requires local **ROOT 6** installation
- Recommended if you want to **develop your own code**

```
$ git clone https://gitlab.cern.ch/corryvreckan/corryvreckan.git
$ cd corryvreckan
$ mkdir build && cd build/
$ cmake ..        # change CMake flags as needed
$ make install -j<number_of_cores>
```

- **Corry executable** available in installation directory:

```
$ path/to/corryvreckan/bin/corry
```

Installation

4. Use Corryvreckan on LXPLUS

- Requires **CERN LXPLUS** account
- Two options: **Source CVMFS version** or **install on LXPLUS**

```
$ source
/cvmfs/clicdp.cern.ch/software/corryvre
ckan/<version>/x86_64-<slc6/centos7>-
gcc7-opt/setup.sh
```

```
$ git clone
https://gitlab.cern.ch/corryvreckan/corryvre
ckan.git
$ cd corryvreckan
$ source etc/setup_lxplus.sh
$ # continue as for local installation
```

- If “sourcing”, **note**:
  - Only default modules are available
  - E.g. no **`[EventLoaderEudaq2]`** (which is needed in example 3)

Installation

5. Use Docker Image

- Run software in container including all dependencies
  - Like “light-weight virtual machine”
  - Needs “docker” executable
  - Install via: `$ sudo snap install docker`
- start docker container:
  - `$ sudo ./start_docker_container.sh`
  - Then use like usual terminal
- Note: only default modules are available (like CVMFS version)
  - E.g. not **`[EventLoaderEUDAQ2]`** (needed for example 3)
- Docker does not support graphics; needs external **ROOT** installation for **`TBrowser t`**

Setting Up an Analysis

Download the Example Data and Configuration Files

3 Examples to Go Through

Example configuration files and scripts: (not needed if using the virtual machine)

- `$ git clone https://gitlab.cern.ch/ffeindt/bttb10_tutorial_corryvreckan.git`
- In this repository we've got 3 examples we'll go through:
  - **`01_atlaspix/`** ← simple
  - **`02_clicpix2/`** ← simple
  - **`03_desy/`** ← advanced
- Go to **`data/`** and download the example data files:
  - **`$ cd data/`**
  - **`$ ./download_example_data_01.sh`** ← only this if connection is slow
  - **`$ ./download_example_data_02.sh`**
  - **`$ ./download_example_data_03.sh`**

Reminder: Configuring Corryvreckan

TOML Style Configuration Files

- TOML style = easy to read
- Support of physical units (e.g. 25um)
- **Configuration** file:
  - Global parameters
  - Specifies modules and parameters
  - Sets input/output paths
  - Sefines log level
- **Geometry** file:
  - Number, types and positions/rotations of sensors
  - Role of sensors (DUT, reference, auxiliary)

example ***configuration***

```
[Corryvreckan]
log_level = "WARNING"
detectors_file = "geometry.conf"
number_of_tracks = 900000

[EventLoaderEUDAQ2]
file_name = "data/run0000456.raw"
inclusive = false
buffer_depth = 1000
shift_triggers = -1
```

example ***geometry***

```
[W0013_D04]
number_of_pixels = 256, 256
orientation = 10.9deg, 17.2deg, -1.3deg
orientation_mode = "xyz"
pixel_pitch = 55um, 55um
position = 886.5um, 270um, 0
spatial_resolution = 4.8um,4.8um
time_resolution = 1.56ns
type = "Timepix3"
role = "reference"
```

Example 1

Example 1

Data and Configuration Files

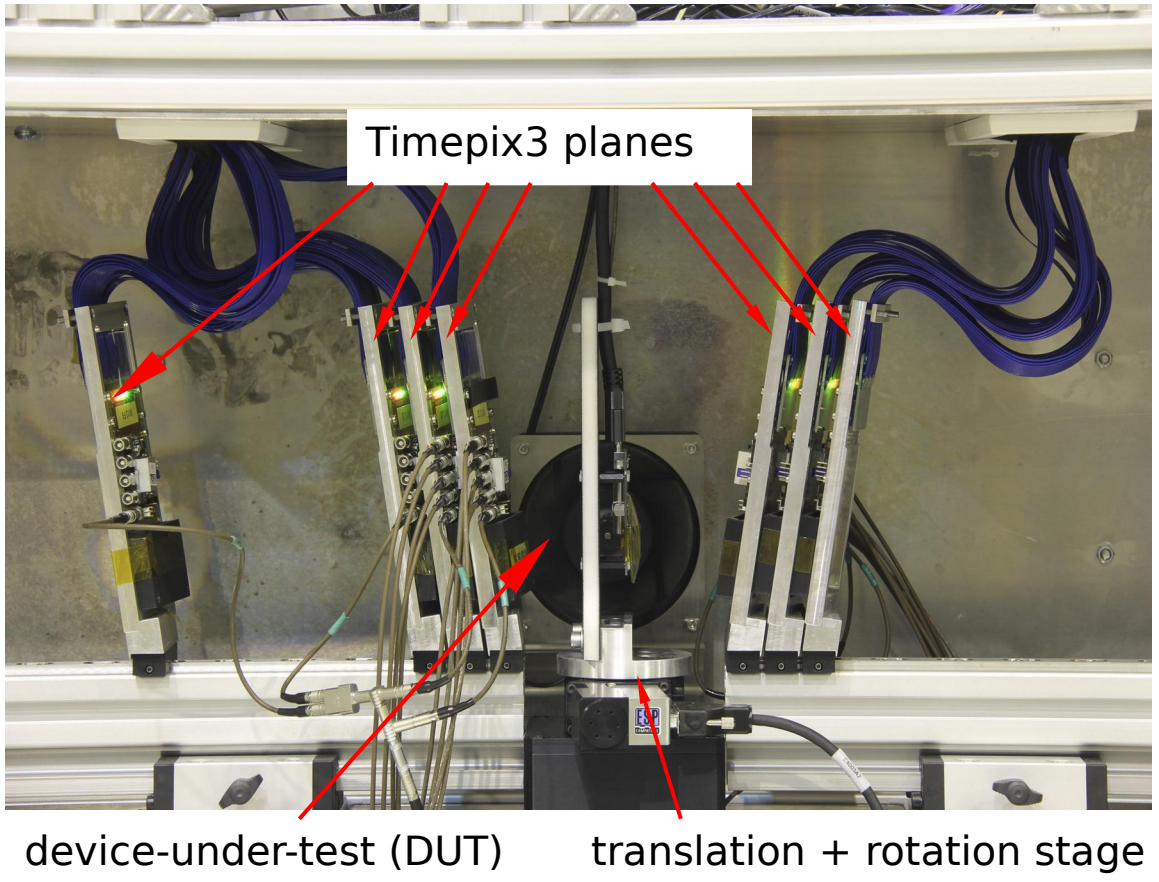
- All **configuration** and **geometry** files in
- `01_atlaspix/`
  - `geometries/`
  - `01_analysis_telescope.conf`
  - `02_analysis_withDUT.conf`
  - `03_online_monitor.conf`
  - `04_filewriter.conf`
  - `05_filereader.conf`
- The data is in
  - `data/`
    - `01_data_atlaspix/`
    - `01_data_telescope/`

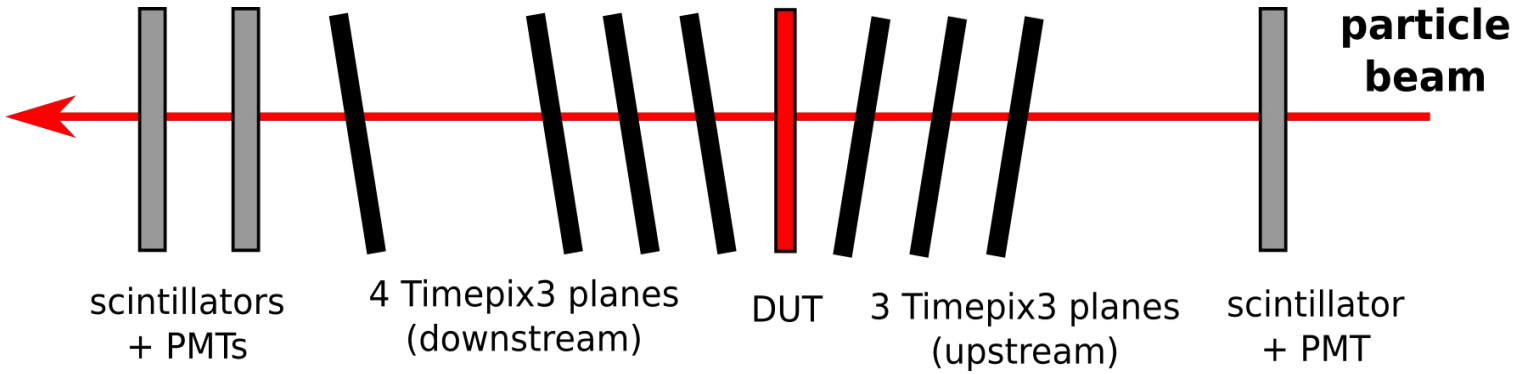
Disclaimer:
The shown analyses are chosen to be instructive and illustrate the framework functionality.
Please don't infer any sensor performance estimates.

Example 1

The Setup

- Timepix3 **telescope**:
  - Data-driven readout
  - Pixel-by-pixel timestamp
- ATLASpix as **device-under-test**:
  - Data-driven readout
  - Pixel-by-pixel timestamp





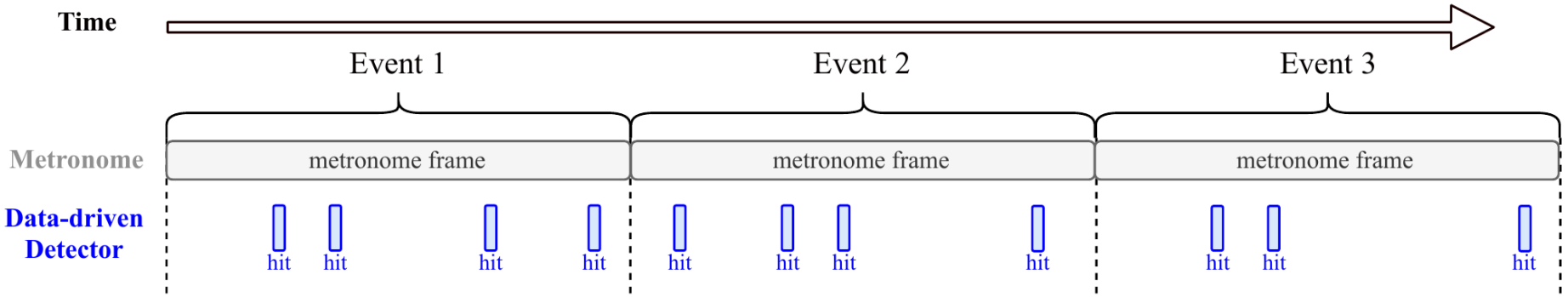
Example 1.1

Starting With the Telescope Only

- **Configuration** file:
  - Global parameters
  - Specifies **[Modules]** and **parameters**
  - Sets input/output paths
  - Defines log level
- All detectors are data-driven
  - Need **[Metronome]** to define event

01_analysis_telescope.conf

```
[Corryvreckan]
output_directory = “output”
detectors_file =
      “geometries/01_alignment_telescope.geo”
histograms_file =
      “01_histograms_analysis_telescope.root”
[Metronome]
event_length = 20us
[EventLoaderTimepix3]
input_directory = “../data/01_data_telescope”
[Clustering4D]
time_cut_abs = 200ns
[Correlations]
[Tracking4D]
min_hits_on_track = 6
spatial_cut_abs = 200um, 200um
time_cut_abs = 50ns
```



Example 1.1

Checking the Geometry File

- **Geometry** file:
  - Number/size of pixels
  - Detector type
  - Position/rotation
  - Role of particular sensors:
    - DUT
    - Reference
    - Auxiliary

geometries/01_alignment_telescope.geo

```
[Timepix3_0]
number_of_pixels = 256,256
orientation = 9.94299deg,187.403deg,-1.36501deg
orientation_mode = “xyz”
pixel_pitch = 55um,55um
position = 939.743um,285.05um,0
spatial_resolution = 4um,4um
time_resolution = 1.5ns
type = “Timepix3”
[Timepix3_1]
...
[Timepix3_2]
...
role = “reference”
```

Example 1.1

Running the Analysis

- Run analysis:
  - `$ corry -c 01_analysis_telescope.conf`
- Event loop is updated continuously:
  - You can interrupt gently by hitting **Ctrl+C**
- Alternatively use
  - `$ corry -c 01_analysis_telescope.conf -o number_of_events=25000`
  - `$ corry -c 01_analysis_telescope.conf -o number_of_tracks=250000`

We'll look at this in more detail in a minute!

```
|16:24:49.775|  (STATUS) ========================| Event loop |========================
|16:25:12.339|  (STATUS) Ev: 564.9k Px: 310.4k Tr: 17.6k (0.0312/ev) t = 11.298s
```

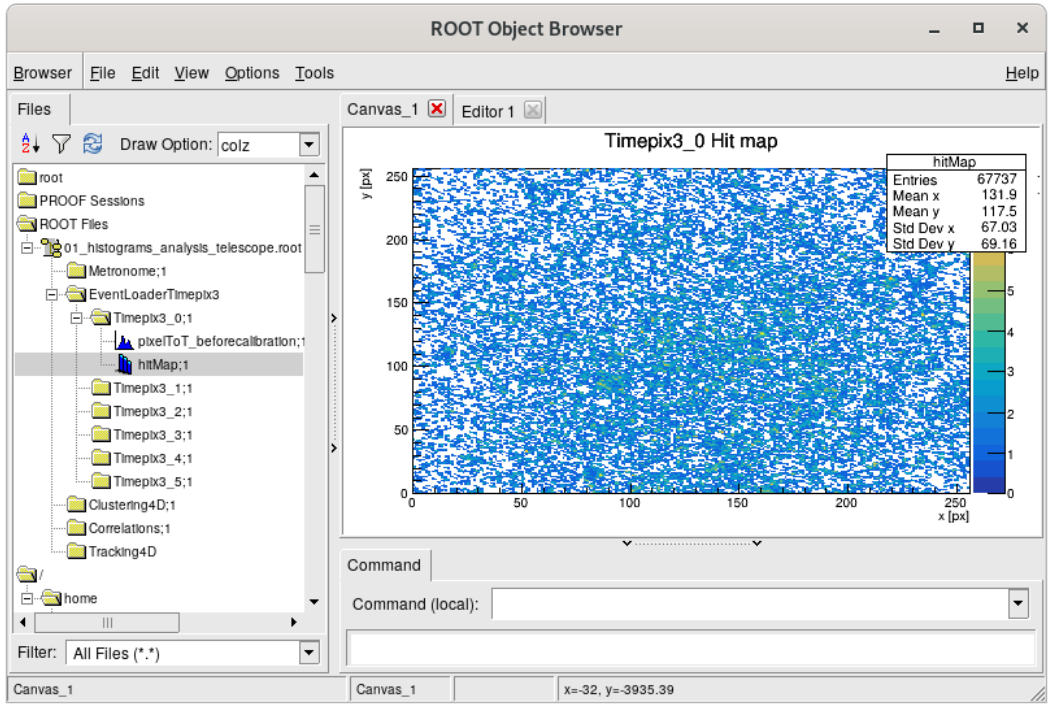
Example 1.1

Checking the Results

- Terminal output:

```
|16:37:52.337|  (STATUS) ==================| Finalising modules |==================
|16:37:53.317|  (STATUS) Wrote histogram output file to /home/jens/Documents/bttb8_tutorial_corryvreckan/01_atlaspix/output/01_histograms_analysis_telescope.root
```

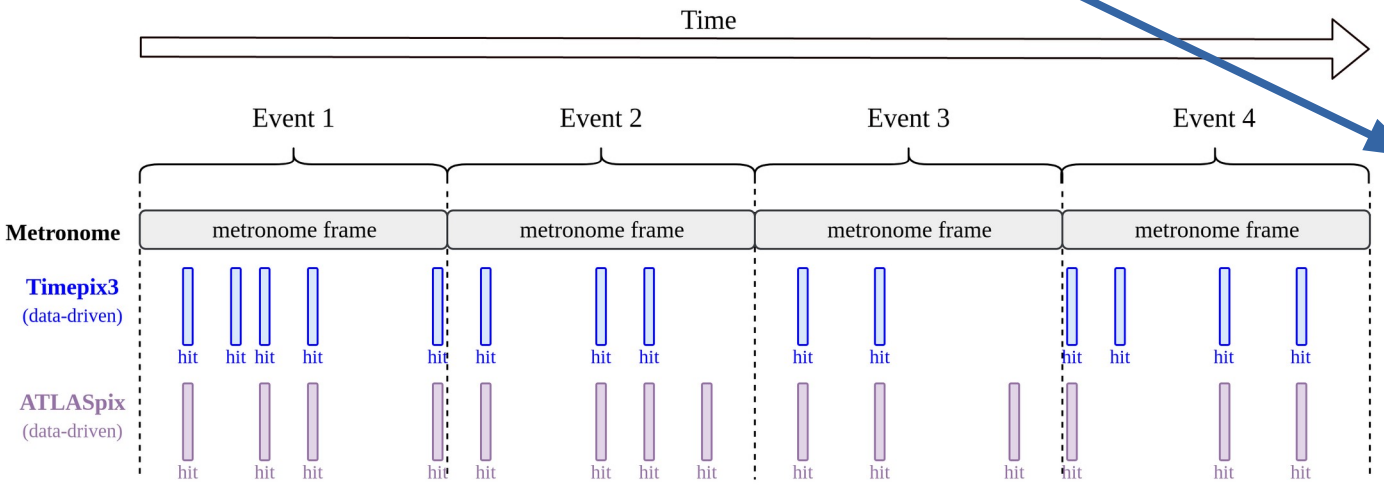
- Output histograms of all modules:



Example 1.2

Adding a DUT

- Still all detectors are data-driven
  - Keep **[Metronome]** to define event
- In **configuration** file:
  - Add **[EventLoaderATLASpix]**
  - Add **[DUTAssociation]** to associate tracks to
  - Hits on the device-under-test
  - Add analysis module(s)



02_analysis_telescope_withDUT.conf

```
[EventLoaderTimepix3]
input_directory =
"../data/01_data_telescope"
[EventLoaderATLASpix]
input_directory =
"../data/01_data_atlaspix"
[Clustering4D]
[Correlations]
time_cut_abs = 200ns
[Tracking4D]
min_hits_on_track = 6
spatial_cut_abs = 200um, 200um
time_cut_abs = 50ns
[DUTAssociation]
time_cut_abs = 200ns
spatial_cut_abs = 300um,150um
[AnalysisDUT]
[AnalysisEfficiency]
```

Example 1.2

Adding a DUT

- In **geometry** file
  - Add **`[ATLASpix_0]`**
  - Mark as device-under-test (“**`dut`**”)
  - Optionally define region-of-interest (“**`roi`**”) for analysis modules

geometries/02_alignment_telescope_withDUT.geo

```
[Timepix3_2]
...

[ATLASpix_0]
number_of_pixels = 25,400
orientation = -0.0124905deg,0.0684685deg,-0.52609deg
orientation_mode = "xyz"
pixel_pitch = 130um,40um
position = 335.278um,-1.55396mm,105mm
spatial_resolution = 40um,10um
time_resolution = 200ns
role = "dut"
type = "ATLASpix"
roi = [1,1], [1, 398], [23, 398], [23, 1]

[Timepix3_3]
...
```

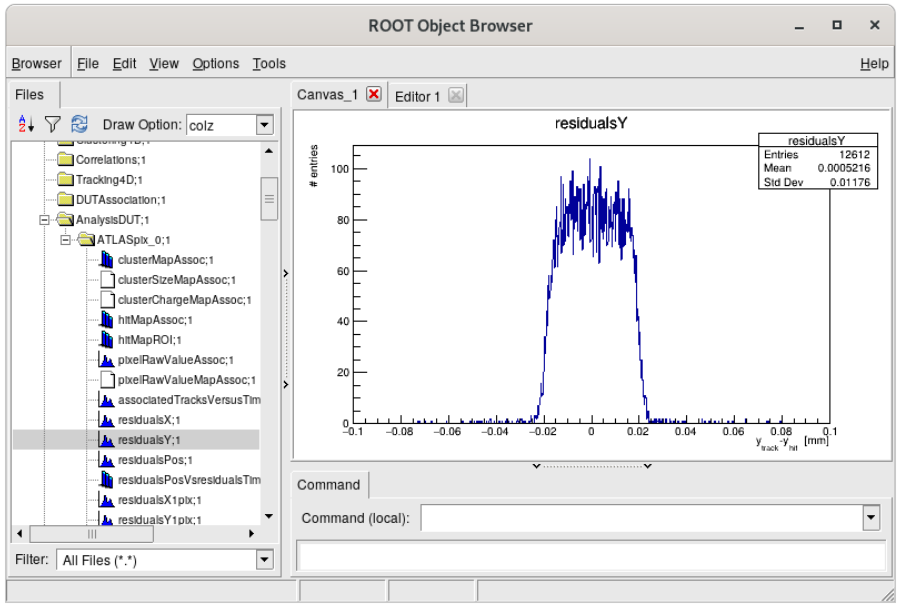
Example 1.2

Checking the Results Again

- Now there is more information on the **terminal** output

```
|17:21:35.912|  (STATUS) ==================| Finalising modules |==================
|17:21:38.781|  (STATUS) [F:DUTAssociation:ATLASpix_0] In total, 28385 clusters are associated to 28323 tracks.
|17:21:38.781|    (INFO) [F:DUTAssociation:ATLASpix_0] Number of tracks with at least one associated cluster: 28323
|17:21:40.187|  (STATUS) [F:AnalysisEfficiency:ATLASpix_0] Track selection flow:       86598
                                                          * rejected by chi2          -45801
                                                          * track outside ROI         -529
                                                          * track outside DUT         -28712
                                                          * track close to masked px  -0
                                                         Accepted tracks:            12073
|17:21:40.187|  (STATUS) [F:AnalysisEfficiency:ATLASpix_0] Total efficiency of detector ATLASpix_0: 99.9255%, measured with 12064/12073 matched/total tracks
|17:21:40.329|  (STATUS) Wrote histogram output file to /home/jens/Documents/bttb8_tutorial_corryvreckan/01_atlaspix/output/02_histograms_analysis_telescope_withDUT.root
```

- Output histograms of all modules:



Example 1.2

Ways to Quit the Framework

- **Ctrl+C** (SIGINT):
  - **Gently** quit corry
  - Finish processing current event and finalize properly
  - Recommended option
- **Ctrl+** (SIGQUIT):
  - **Forcefully** terminate corry
  - Leads to loss of all generated data
  - Generally not recommended

```
|11:12:18.931|  (STATUS) ========================| Event loop |=======================
|11:12:21.607|  (STATUS) Ev: 122.1k Px: 0.3k Tr: 0.0k (0/ev) t = 2.442s^C
|11:12:21.610|  (STATUS) Interrupted! Finishing up current event...
|11:12:21.610|  (STATUS) Ev: 122.2k Px: 0.3k Tr: 0.0k (0/ev) t = 2.444s
|11:12:21.610|  (STATUS) ===================| Finalising modules |==================
|11:12:22.077|  (STATUS) Wrote histogram output file to /home/jens/Documents/bttb8_tutoria
nalysis_telescope.root
|11:12:22.077|  (STATUS) ===============| Wall-clock timing (seconds) |===============
|11:12:22.077|  (STATUS)            Metronome                --  0.04757s = 0.000389ms/evt
|11:12:22.077|  (STATUS)  EventLoaderTimepix3 : Timepix3_0   --  0.28141s = 0.002303ms/evt
|11:12:22.077|  (STATUS)  EventLoaderTimepix3 : Timepix3_1   --  0.27699s = 0.002267ms/evt
|11:12:22.077|  (STATUS)  EventLoaderTimepix3 : Timepix3_2   --  0.30479s = 0.002494ms/evt
|11:12:22.077|  (STATUS)  EventLoaderTimepix3 : Timepix3_3   --  0.27372s = 0.002240ms/evt
|11:12:22.078|  (STATUS)  EventLoaderTimepix3 : Timepix3_4   --  0.27328s = 0.002236ms/evt
|11:12:22.078|  (STATUS)  EventLoaderTimepix3 : Timepix3_5   --  0.27130s = 0.002220ms/evt
|11:12:22.078|  (STATUS)         Clustering4D : Timepix3_0   --  0.06246s = 0.000511ms/evt
|11:12:22.078|  (STATUS)         Clustering4D : Timepix3_1   --  0.05449s = 0.000446ms/evt
|11:12:22.078|  (STATUS)         Clustering4D : Timepix3_2   --  0.05425s = 0.000444ms/evt
|11:12:22.078|  (STATUS)         Clustering4D : Timepix3_3   --  0.05352s = 0.000438ms/evt
|11:12:22.078|  (STATUS)         Clustering4D : Timepix3_4   --  0.05330s = 0.000436ms/evt
|11:12:22.078|  (STATUS)         Clustering4D : Timepix3_5   --  0.06170s = 0.000505ms/evt
|11:12:22.078|  (STATUS)         Correlations : Timepix3_0   --  0.05661s = 0.000463ms/evt
|11:12:22.078|  (STATUS)         Correlations : Timepix3_1   --  0.05492s = 0.000449ms/evt
|11:12:22.078|  (STATUS)         Correlations : Timepix3_2   --  0.05488s = 0.000449ms/evt
|11:12:22.078|  (STATUS)         Correlations : Timepix3_3   --  0.05459s = 0.000447ms/evt
|11:12:22.078|  (STATUS)         Correlations : Timepix3_4   --  0.05364s = 0.000439ms/evt
|11:12:22.078|  (STATUS)         Correlations : Timepix3_5   --  0.05346s = 0.000437ms/evt
|11:12:22.078|  (STATUS)           Tracking4D               --  0.10156s = 0.000831ms/evt
|11:12:22.078|  (STATUS) ============================================================
```

```
|11:12:46.911|  (STATUS) [I:Tracking4D] Initialising "Tracking4D"
|11:12:46.911|  (STATUS) ========================| Event loop |=======================
|11:12:49.362|  (STATUS) Ev: 113.4k Px: 0.3k Tr: 0.0k (0/ev) t = 2.268s ^\
|11:12:49.362|   (FATAL) Aborting!
Aborted (core dumped)
```

Example 1.2

Playing With the Command Line

- Parameters from config file can be **overwritten** in the command line:
  - `corry ...`
  - `-c config.cfg` → path to configuration file
  - `-l log.txt` → write terminal output to file
  - `-v “FATAL”,“STATUS”,“ERROR”`, → logging verbosity level
  - `-o MyModule.log_level=”DEBUG”` → also per module, for **`“WARNING”,”INFO”,”DEBUG”`**
  - `-o histogram_file=”hists.root”` → global framework parameter
  - `-o AnalysisDUT.chi2ndof_cut=3` → module conf., specified by dot(.) between module and key
  - `-g mydut.orientation=0deg,5deg,0deg` → detector geometry parameter

Play around for a few minutes!

Example 1.3

The Online Monitor

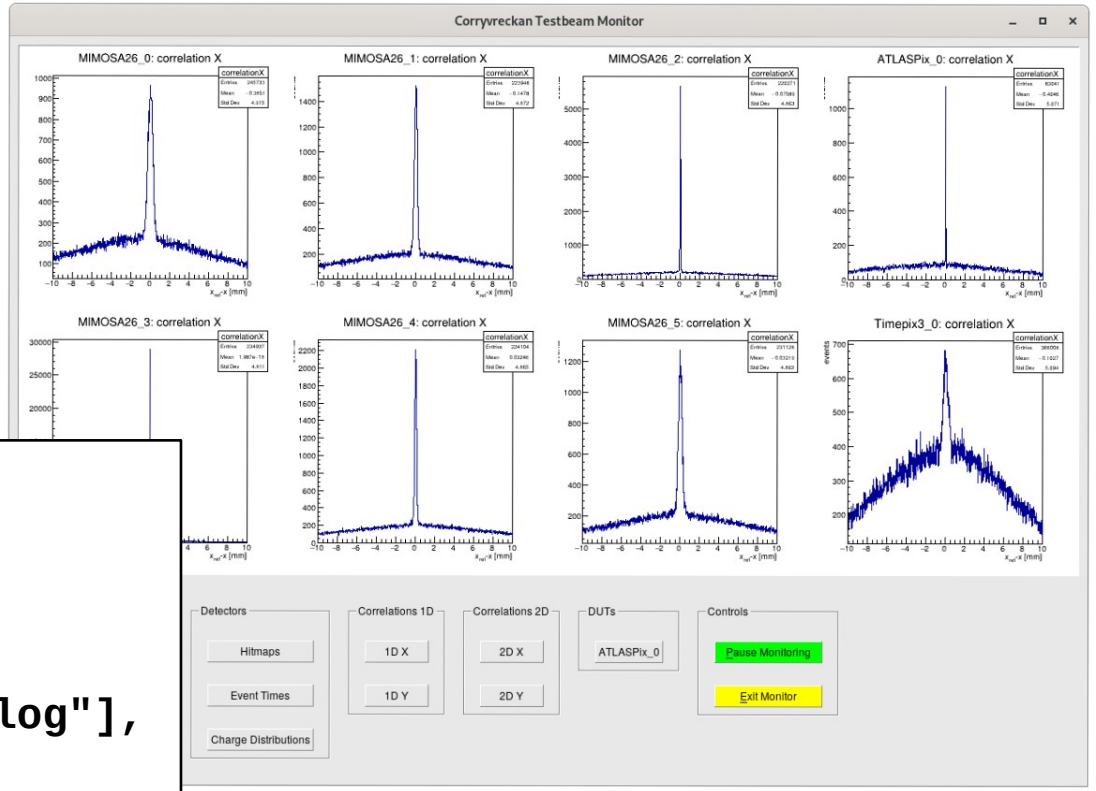
- We can also see the results **while** the analysis is running
  - **Configuration** file:
  - Add `[OnlineMonitor]`
  - Specify which plots you want to see
  - Supported keywords: **%DETECTOR%, %DUT%, %REFERENCE%**

03_online_monitor.conf

```
[OnlineMonitor]
update = 200
dut_plots =
   [["EventLoaderATLASpix/%DUT%/hitMap", "colz"],
    ["EventLoaderATLASpix/%DUT%/pixelToT"],
    ["EventLoaderATLASpix/%DUT%/pixelMultiplicity","log"],
    ["EventLoaderATLASpix/%DUT%/hPixelTimes_long"]]
```

opens interactive window

(Note: does not work in docker container!)



Example 1.4

The FileWriter

- We can **write out reconstructed data** at **any stage** during the reconstruction:
  - In **configuration** file:
  - Add **[FileWriter]**
  - Objects of **all previous modules** will be written to file
  - Explore output file with ROOT TBrowser
  - Read in for further analysis via the **[FileReader]** (next slide)

04_filewriter.conf

```
[Tracking4D]
min_hits_on_track = 6
spatial_cut_abs = 200um, 200um
time_cut_abs = 50ns
[FileWriter]
file_name = “04_output_tuples.root”
[DUTAssociation]
time_cut_abs = 200ns
spatial_cut_abs = 300um,150um
[AnalysisDUT]
[AnalysisEfficiency]
```

Example 1.5

The FileReader

- We can **read in** reconstructed data and **continue** the reconstruction
  - In **configuration** file:
  - Add **[FileReader]**
  - All objects from modules above will be written to file
- We can also read in **simulated** data from **Allpix²**
  - Talk:
    - A Semiconductor Detector Simulation Framework
  - Hands-on tutorials:
    - Tuesday afternoon
    - Thursday afternoon
  - Users workshop
    - 22-23 May 2023

05_filereader.conf

```
# [Tracking4D] ← not needed anymore!
# min_hits_on_track = 6
# spatial_cut_abs = 200um, 200um
# time_cut_abs = 50ns
[FileReader]
file_name = “output/04_output_tuples.root”
[DUTAssociation]
time_cut_abs = 200ns
spatial_cut_abs = 300um,150um
[AnalysisDUT]
[AnalysisEfficiency]
```

Example 2

Example 2

Data and Configuration Files

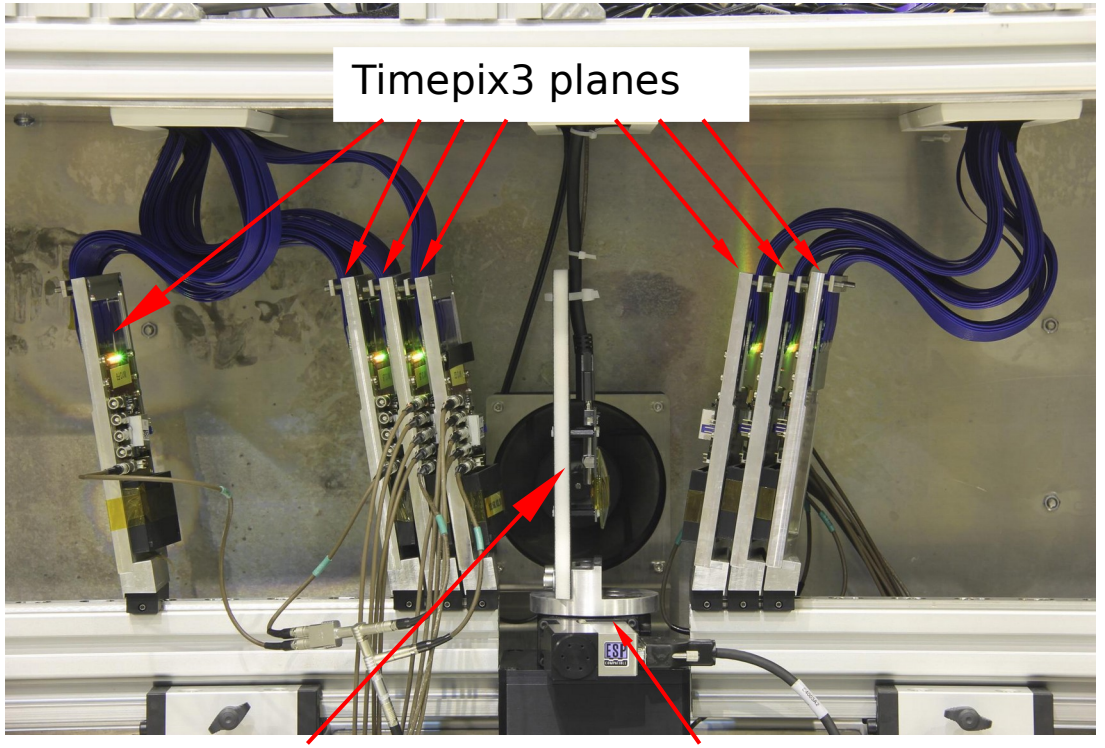
- All **configuration** and **geometry** files in:
  - `02_clicpix2/`
    - `geometries/`
    - `01_analysis_telescope.conf`
    - `02_analysis_withDUT.conf`
- The **data** is in:
  - `data/`
    - `02_data_clicpix2/`
    - `02_data_telescope/`

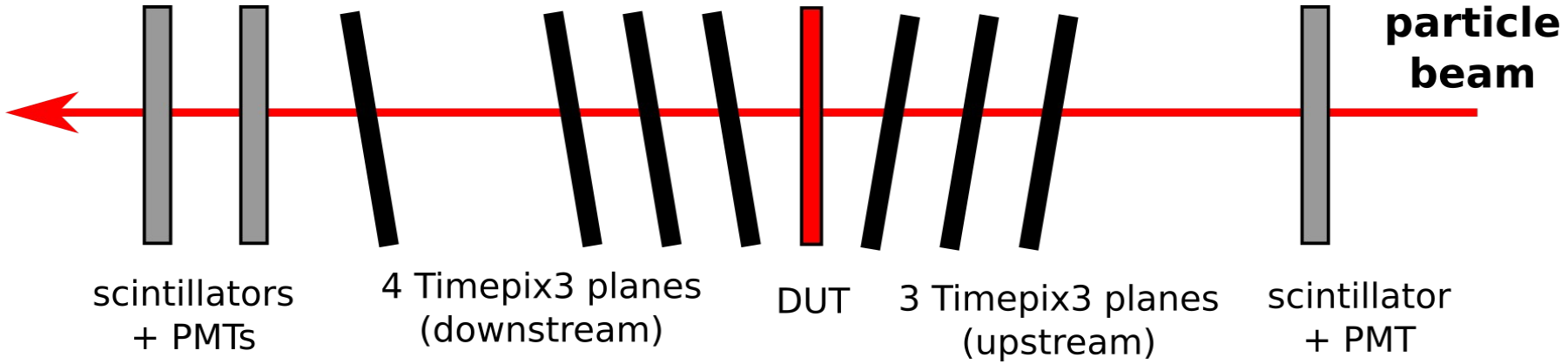
Disclaimer:
The shown analyses are chosen to be instructive and illustrate the framework functionality.
Please don't infer any sensor performance estimates.

Example 2

Looking at the Setup

- Timepix3 telescope, same as in example 1
  - Data-driven readout
  - Pixel-by-pixel timestamp
- CLICpix2 as device-under-test:
  - Frame-based readout → shutter open/close
  - No pixel-by-pixel timestamp (in this operation mode)





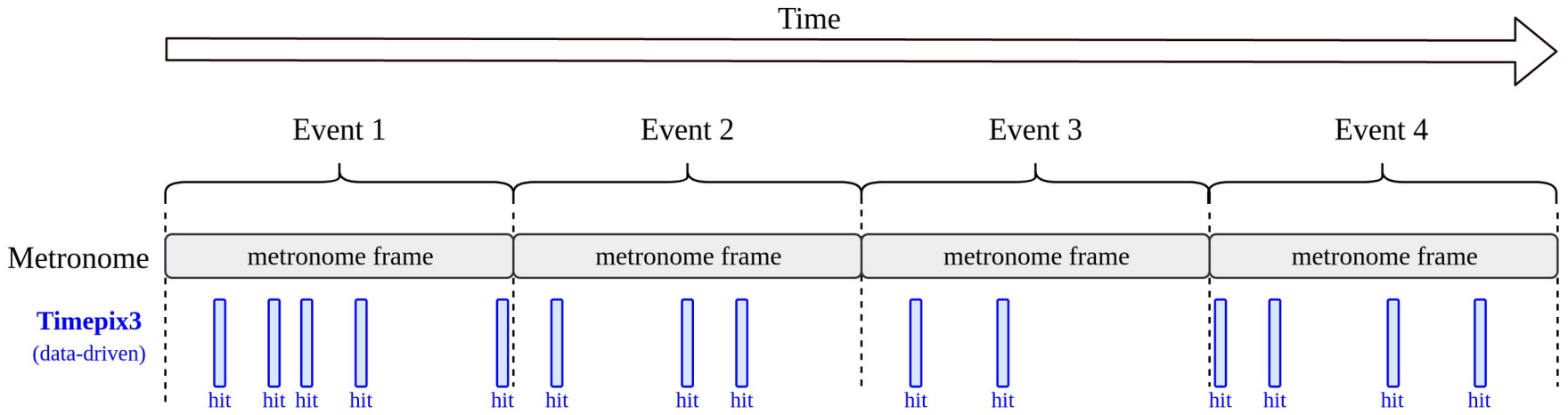
Example 2.1

Again Starting with The Telescope

01_analysis_telescope.conf

```
[Corryvreckan]
detectors_file =
      “geometries/01_alignment_telescope.geo”
histograms_file =
      “01_histograms_telescope.root”
[Metronome]
event_length = 100us
[EventLoaderTimepix3]
input_directory = “../data/02_data_telescope”
[Clustering4D]
[Correlations]
[Tracking4D]
min_hits_on_track = 6
spatial_cut_abs = 200um, 200um
time_cut_abs = 60ns
```

- Telescope detectors are data-driven
  - Need **[Metronome]** to define event
- **Configuration** file as before (only path/to/data is updated)



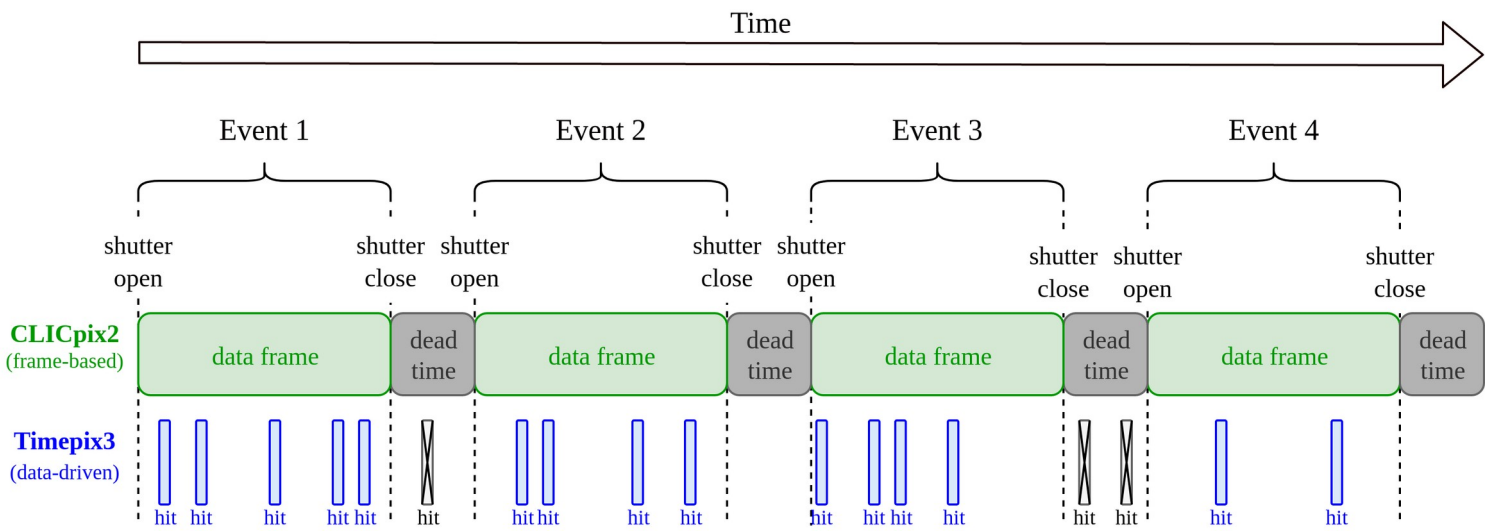
Example 2.2

Adding the DUT

- CLICpix2 is frame-based
  - This detector needs to define the events
  - Data from Timepix3 are filled in
- **Configuration** file:
  - Remove `[Metronome]`
  - Place `[EventLoaderCLICpix2]`
  - Above Timepix3
  - Remember: The 1st module defines the event!

02_analysis_telescope_withDUT.conf

```
# Remove this module:
# [Metronome]
# event_length = 20us
# Now the CLICpix2 defines the frames:
[EventLoaderCLICpix2]
input_directory = "../data/01_data_clicpix2"
[EventLoaderTimepix3]
input_directory = "../data/01_data_telescope"
```



Example 2.2

Adding the DUT

- CLICpix2 does not have pixel timestamps (in this operation mode)
- **Configuration** file:
  - Add `[ClusteringSpatial]`
  - Now need to specify which clustering for which detector:
  - Hierarchy: name > type > nothing
  - Also add DUT association + analysis module

02_analysis_telescope_withDUT.conf

```
[Clustering4D]
type = “Timepix3”
[ClusteringSpatial]
name = “CLICpix2_0”
[Correlations]
[Tracking4D]
[DUTAssociation]
[AnalysisDUT]
```

Example 3

Example 3

Data and Configuration Files

- All **configuration** and **geometry** files in:
  - `03_desy/`
  - `geometries/`
    - `01_tlu_mimosa26_timepix3.conf`
    - `02_tlu_mimosa26_timepix3_atlaspix.conf`
    - `03_EventDefinitionM26.conf`
    - `04_clictd_tlu_mimosa26_timepix3.conf`
- The **data** is in:
  - `data/`
    - `03_data_desy/`

Note:
This example uses the modules
- **`[EventLoaderEUDAQ2]`**
- **`[EventDefinitionM26]`**

which require EUDAQ2:
https://github.com/eudaq/eudaq
Use:
→ VirtualBox (already prepared)
→ Compilation from source

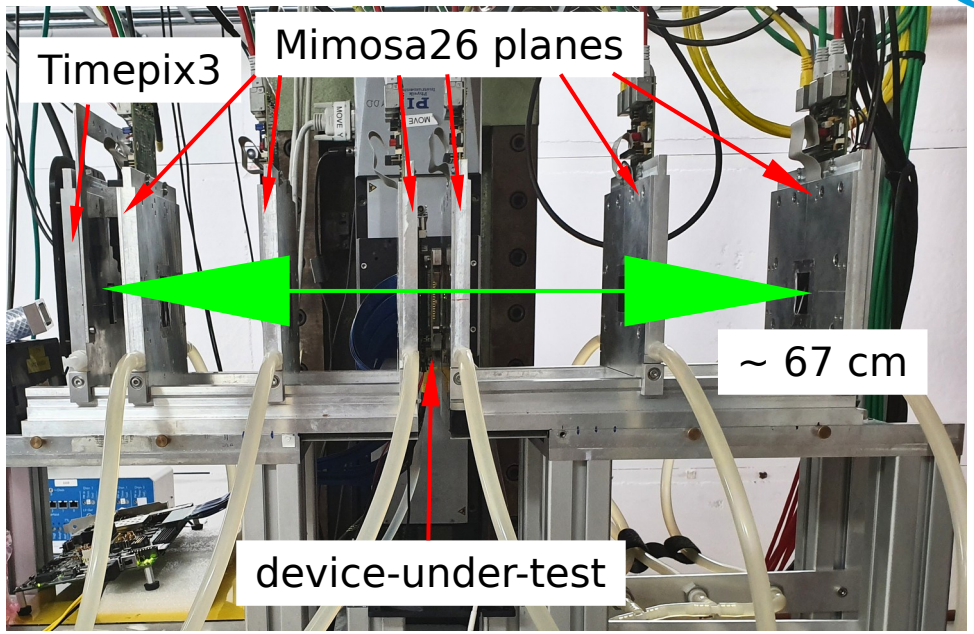
Disclaimer:
The shown analyses are chosen to be instructive and illustrate the framework functionality.
Please don't infer any sensor performance estimates.

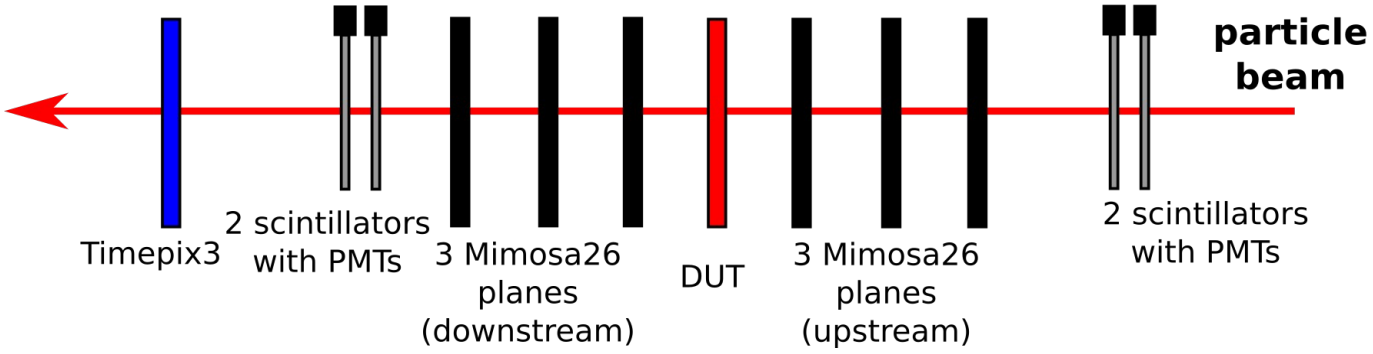
Example 3

Looking at The Setup

Setup at DESY using EUDAQ2:

- https://github.com/eudaq/eudaq
- **AIDA TLU** (“Trigger Logic Unit”)
  - Generates trigger signal from scintillator coincidence
- Mimosa26 **telescope**:
  - Triggered by TLU
  - Rolling-shutter readout
  - No pixel-by-pixel timestamp
- **Additional** Timepix3 plane:
  - Data-driven
  - Pixel-by-pixel timestamp



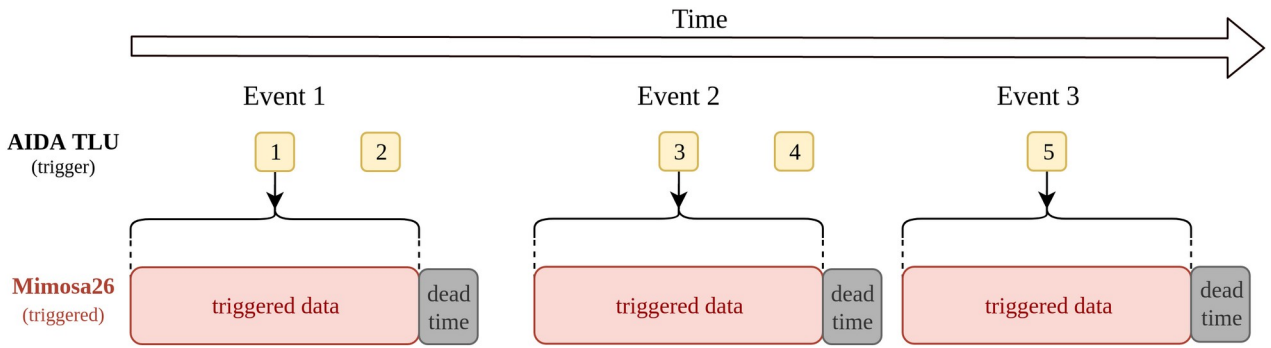


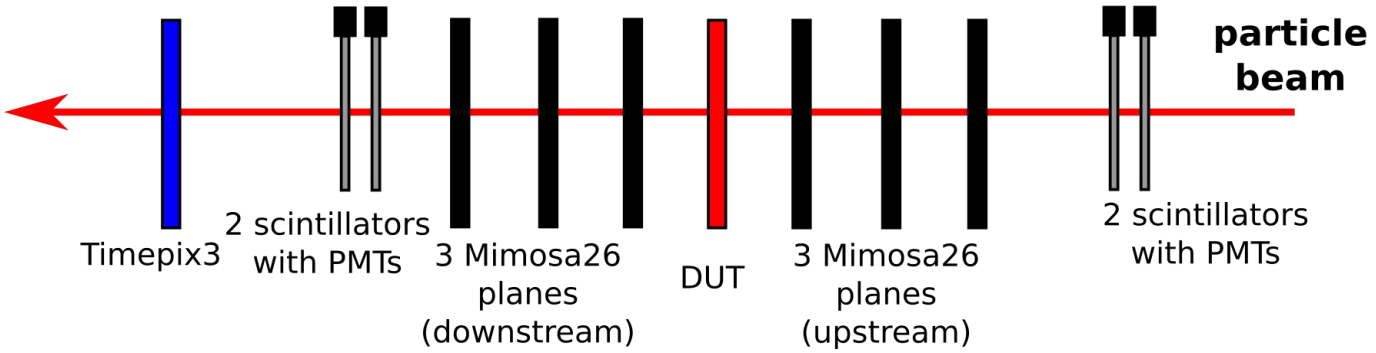
Example 3

Looking at The Setup

Setup at DESY using EUDAQ2:

- https://github.com/eudaq/eudaq
- **AIDA TLU** (“Trigger Logic Unit”)
  - Generates trigger signal from scintillator coincidence
- Mimosa26 **telescope**:
  - Triggered by TLU
  - Rolling-shutter readout
  - No pixel-by-pixel timestamp
- **Additional** Timepix3 plane:
  - Data-driven
  - Pixel-by-pixel timestamp



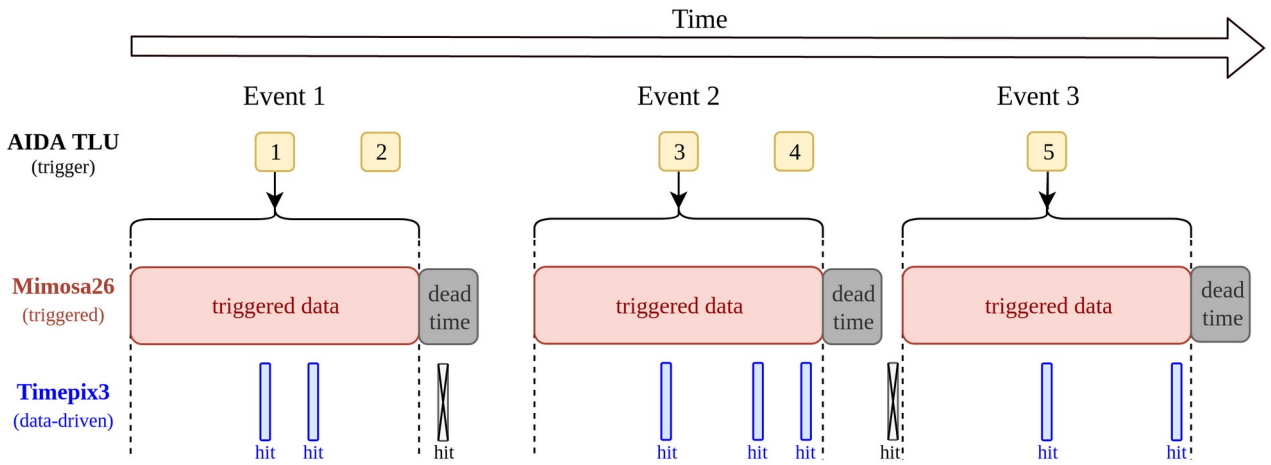


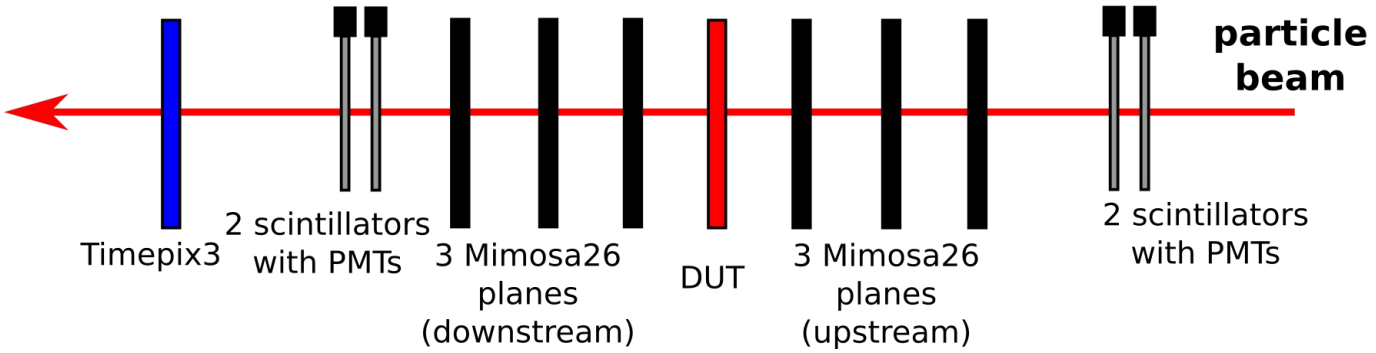
Example 3

Looking at The Setup

Setup at DESY using EUDAQ2:

- https://github.com/eudaq/eudaq
- **AIDA TLU** (“Trigger Logic Unit”)
  - Generates trigger signal from scintillator coincidence
- Mimosa26 **telescope**:
  - Triggered by TLU
  - Rolling-shutter readout
  - No pixel-by-pixel timestamp
- **Additional** Timepix3 plane:
  - Data-driven
  - Pixel-by-pixel timestamp





Example 3.1

Looking at The Geometry

New situation:

- AIDA TLU is **not a pixel detector**
  - Use as “auxiliary” detector
- **Geometry** file:
  - No pixels/spatial resolution
  - But can have position, time resolution
  - Define its role as “**auxiliary**” or “**aux**”

alignment_tlu_mimosa26_timepix3_atlaspix.geo

```
[TLU_0]
...
type = “TLU”
role = “aux”
[Mimosa26_0]
...
```

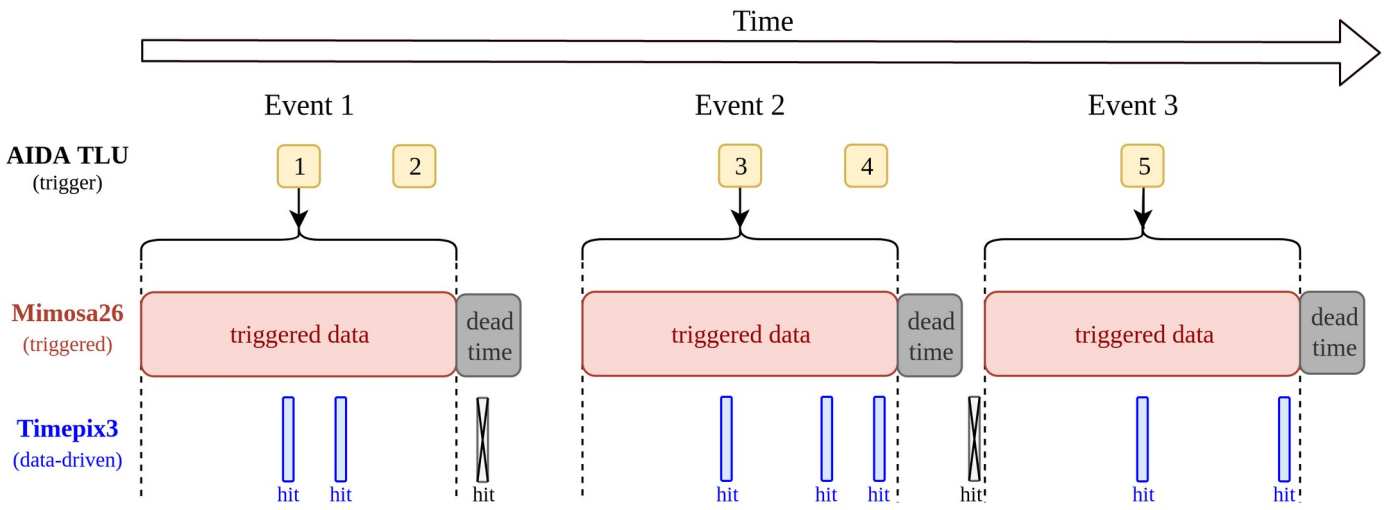
Example 3.1

Looking at The Configuration File

- Configuration file:
  - Start with the TLU and “expand” the event to match time slice of triggered data
  - Add data from Mimosa26 based on **trigger ID**
  - Add hits from Timepix3 based on **pixel timestamp**
- Remember: The first module defines the event!

01_tlu_mimosa26_timepix3.conf

```
[EventLoaderEUDAQ2]
name = “TLU_0”
adjust_event_times =
 [[“TluRawDataEvent”,-115us,+230us]]
[EventLoaderEUDAQ2]
type = “Mimosa26”
[EventLoaderEUDAQ2]
type = “Timepix3”
```



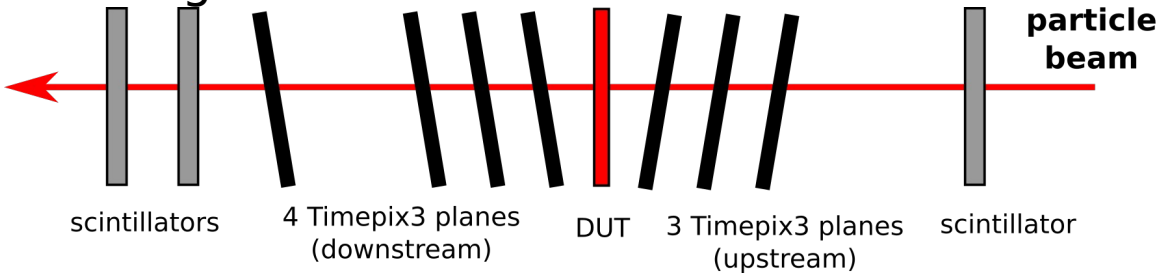
Example 3

Differences in Tracking

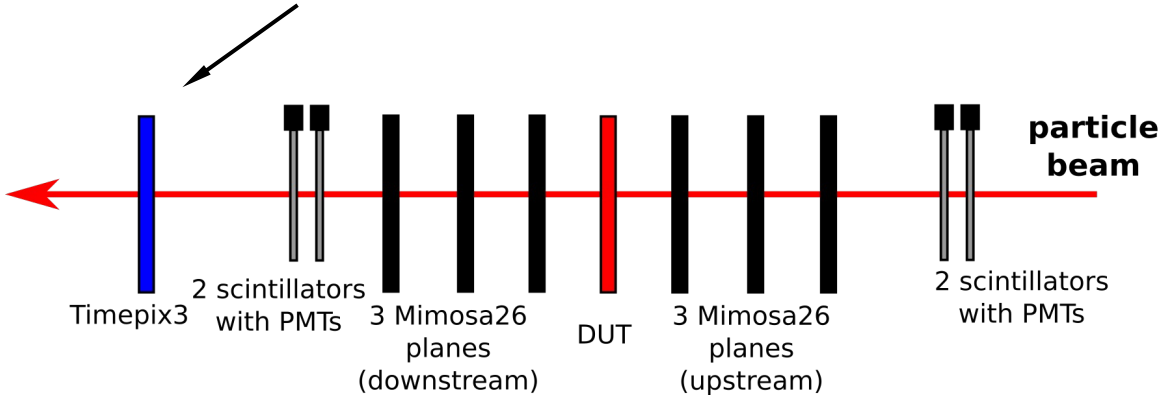
- Examples 1-2: **SPS**
  - 7 Timepix3 hits with precise timestamp
  - Track timestamp = average TPX3 timestamp

- Example 3: **DESY**
  - Mimosa26 hits (3x 115µs) with multiple trigger timestamps
  - Require Timepix3 for **unambiguous track time**
  - Track timestamp = TPX3 timestamp

SPS:
all sensors provide hit timestamps for tracking



DESY:
unambiguous track timestamp only with TPX3

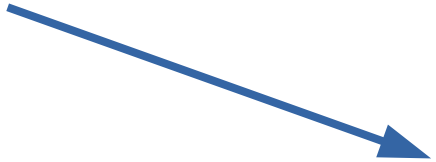


Example 3

Differences in Tracking

DESY

- **Mimosa26** hits (3x 115µs) with multiple trigger timestamps
- Require **Timepix3** for unambiguous track time
- Track timestamp = TPX3 timestamp



01_tlu_mimosa26_timepix3.conf

```
[ClusteringSpatial]
type = “Mimosa26”
use_trigger_timestamp = true

[Clustering4D]
type = “Timepix3”

[Tracking4D]
require_detector = “Timepix3_0”
timestamp_from = “Timepix3_0”
```

Example 3

Differences in Tracking

- One more major difference:
  - SPS beam: 120 GeV pions
  - DESY beam: 5.4 GeV electrons

    → Much lower momentum

    → Use **general-broken-line** tracking instead of straight tracks
- In **configuration**:
  - Add momentum and track model
- In **geometry**:
  - Add material budget in (X/X0)

01_tlu_mimosa26_timepix3.conf

```
[Tracking4D]
require_detector = “Timepix3_0”
timestamp_from = “Timepix3_0”
momentum = 5.4GeV
track_model = “gbl”
```

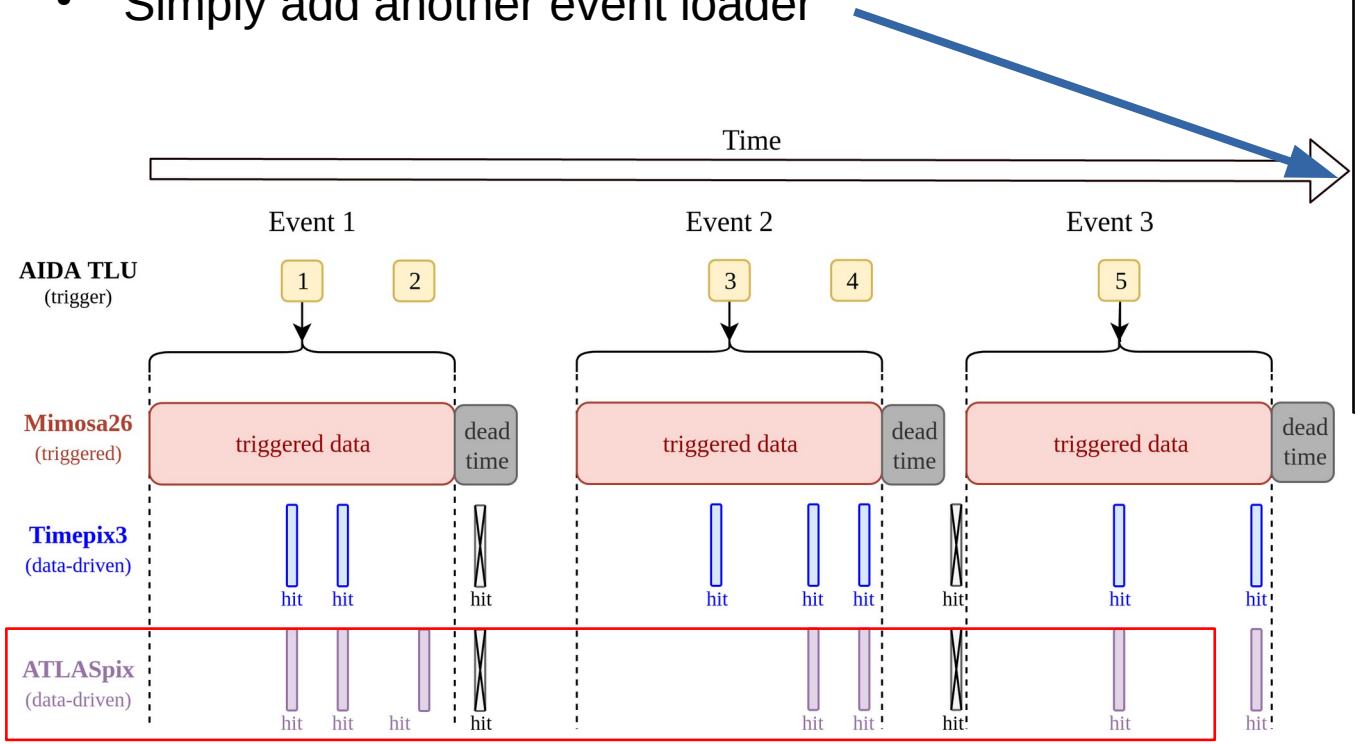
alignment_tlu_mimosa26_timepix3_atlaspix.geo

```
[Mimosa26_0]
material_budget = 0.00075
...
[Timepix3_0]
material_budget = 0.01068
...
```

Example 3.2

Adding a DUT

- ATLASpix as **device-under-test**:
  - Data-driven
  - Pixel-by-pixel timestamp
- Simply add another event loader



01_tlu_mimosa26_timepix3_atlaspix.conf

```
[EventLoaderEUDAQ2]  # the 1st event loader
name = "TLU_0"  # defines the event!
adjust_event_times =
 [["TluRawDataEvent",-115us,+230us]]
[EventLoaderEUDAQ2]
type = "Mimosa26"
[EventLoaderATLASpix]   # Here the order
type = "ATLASpix"    # doesn't matter!
[EventLoaderEUDAQ2]
type = "Timepix3"
```

Example 3.3

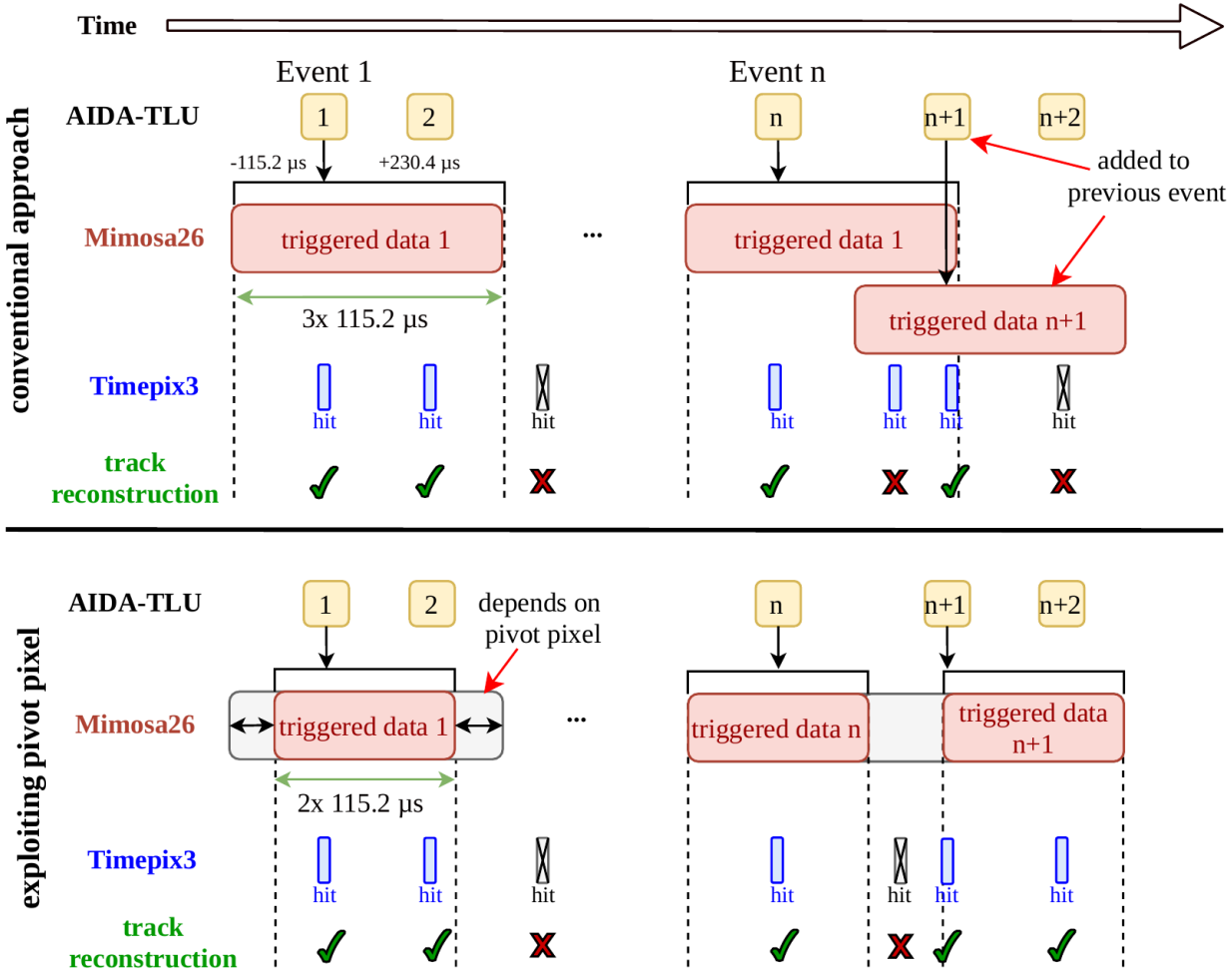
Using the EventDefinitionM26

- Module **EventDefinitionM26**
  - Use pivot pixel information for more precise event definition
  - Add before all event loaders
- Remember: The 1st module defines the event!

```
[EventDefinitionM26]   # the 1st module
detector_event_time = "TLU" # defines the event!
file_timestamp = "path/to/TLUdata"
file_duration = "path/to/M26data"

[EventLoaderEUDAQ2]
name = "TLU_0"
# Remove this parameter! Only add triggers now!
# adjust_event_times =
 [["TluRawDataEvent",-115us,+230us]]
[EventLoaderEUDAQ2]
type = "Mimosa26"
[EventLoaderATLASpix]
type = "ATLASpix"
[EventLoaderEUDAQ2]
type = "Timepix3"
```

03_EventDefinitionM26.conf



Example 3.3

Looking at the Event Building

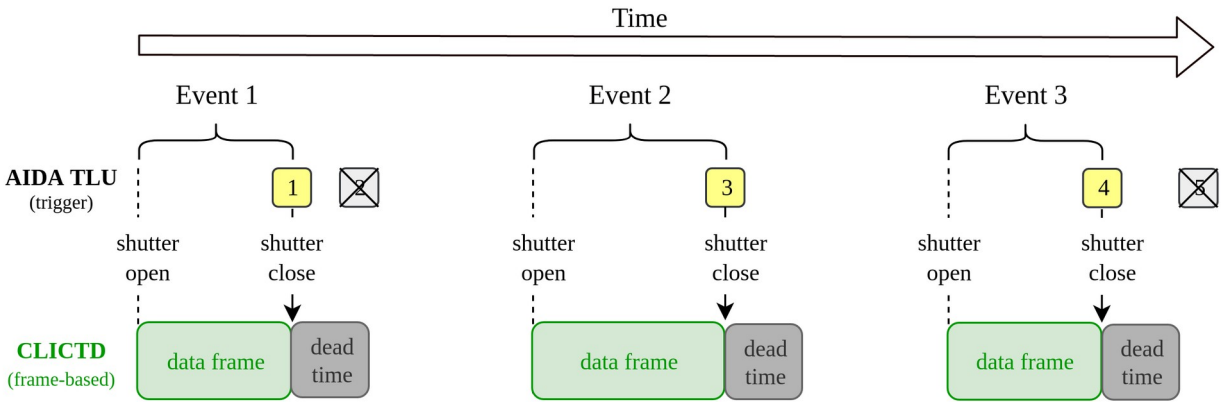
- CLICTD as **device-under-test**:
  - No pixel-by-pixel timestamp
  - Frame-based
    - → Shutter opened randomly
    - → Shutter closed by trigger
- CLICTD should define events:

The 1st module defines the event!

04_clicpix2_tlu_mimosa26_timepix3.conf

```
[EventLoaderEUDAQ2]
name = "CLICTD_0"

[EventLoaderEUDAQ2]
name = "TLU_0"
```



Example 3.3

Looking at the Event Building

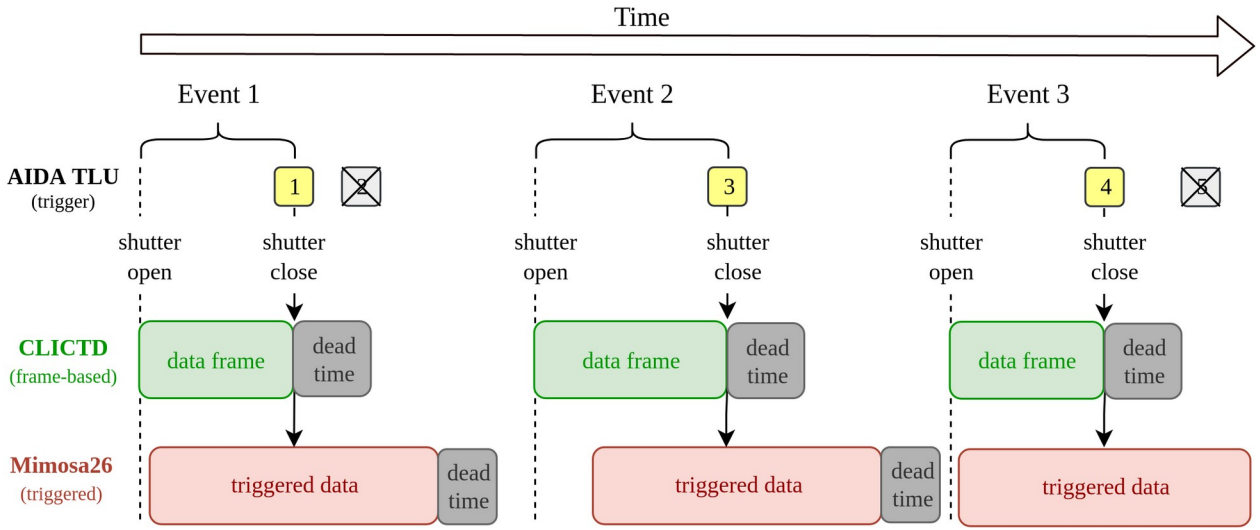
- CLICTD as **device-under-test**:
  - No pixel-by-pixel timestamp
  - Frame-based
    - → Shutter opened randomly
    - → Shutter closed by trigger
- CLICTD should define events:

The 1st module defines the event!

04_clicpix2_tlu_mimosa26_timepix3.conf

```
[EventLoaderEUDAQ2]
name = "CLICTD_0"

[EventLoaderEUDAQ2]
name = "TLU_0"
[EventLoaderEUDAQ2]
type = "Mimosa26"
```



Example 3.3

Looking at the Event Building

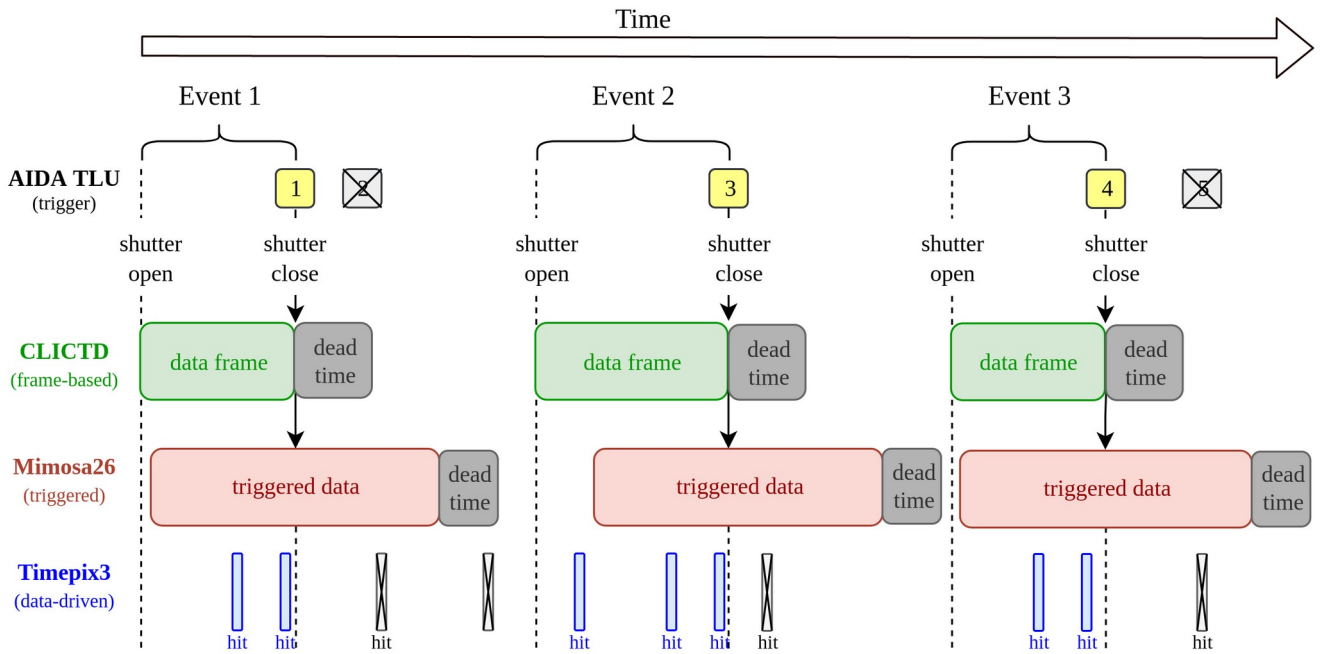
- CLICTD as **device-under-test**:
  - No pixel-by-pixel timestamp
  - Frame-based
    - → Shutter opened randomly
    - → Shutter closed by trigger
- CLICTD should define events:

The 1st module defines the event!

04_clicpix2_tlu_mimosa26_timepix3.conf

```
[EventLoaderEUDAQ2]
name = "CLICTD_0"

[EventLoaderEUDAQ2]
name = "TLU_0"
[EventLoaderEUDAQ2]
type = "Mimosa26"
[EventLoaderEUDAQ2]
type = "Timepix3"
```



Concluding

Corryvreckan In Summary

Reconstruction and Analysis Tool for Pixel Sensor Test Beam Data

- Highly flexible/configurable
  - Separate modules for each reconstructions/ analysis step
  - Many different event building options
  - Comprehensive documentation
  - Beginner-friendly tutorials
- Growing number of users/contributors

  → **Thanks to all of you!**

Learn more:

- Visit our website: https://cern.ch/corryvreckan
- Browse through our manual: Get the lastest version here
- Try our other tutorial: Get Started (no prior experience required)
- Check out the repository: https://gitlab.cern.ch/corryvreckan/corryvreckan
- Discuss in the forum: https://corryvreckan-forum.web.cern.ch/
- Contact us: corryvreckan.info@cern.ch https://mattermost.web.cern.ch/corryvreckan

Thank you

Contact

Deutsches Elektronen-
Synchrotron DESY

www.desy.de

Finn Feindt
FH, ATLAS
finn.feindt@desy.de
+49157 8734 3595