



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

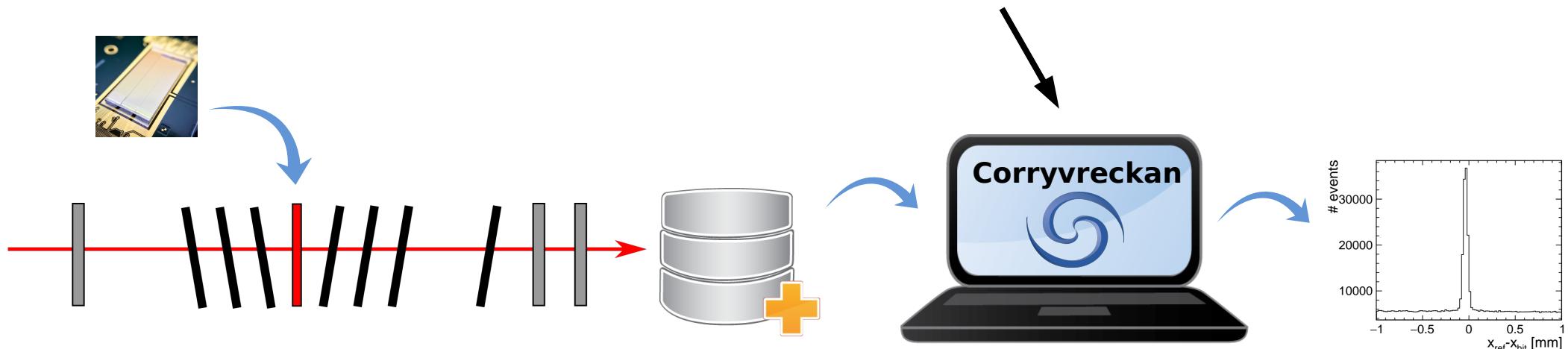


GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung



Efficient Analysis of Test-beam Data with **Corryvreckan**



Vertex 2020 Conference
October 6th, 2020

Jens Kröger, Heidelberg University & CERN
*on behalf of the CLICdp collaboration
and the Corryvreckan Developers*

Test-beam Analysis for Pixel Sensors

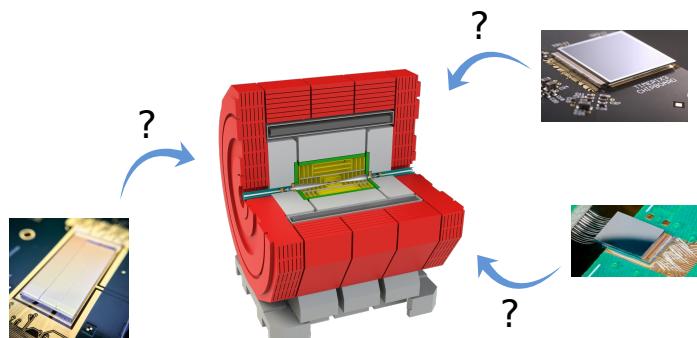
Challenge:

LHC upgrades + future HEP experiments:

- stringent detector requirements
- pushing limits of technology

→ R&D on vast range of pixel
detector technologies

- different readout concepts
- highly specialized to each use case

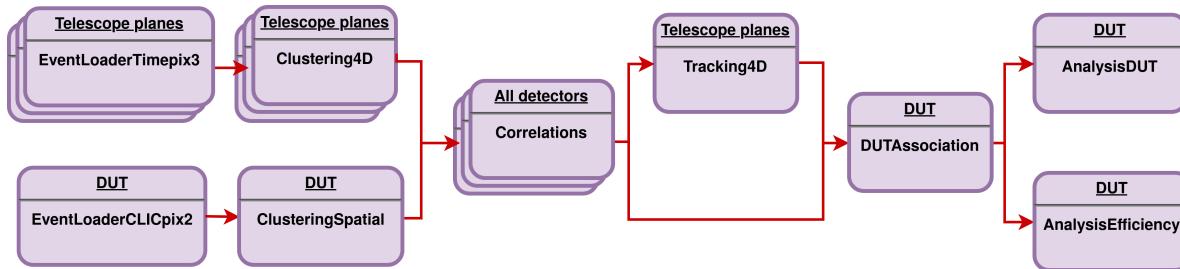


Our Motivation:

- maximize synergies
 - avoid numerous similar frameworks
 - "one framework fits all"
- flexible reconstruction
 - combine detectors with different read-out schemes
 - different analysis objectives
- easy to use, understand, contribute

The Corryvreckan Framework

- modular structure
 - framework core
 - (user) modules for specific tasks
- clean code & documentation
 - modern C++, code reviews, CI
 - comprehensive user manual (> 100 pages!)



- highly flexible and configurable
 - TOML style = easy to read
 - support of physical units (e.g. 25um)

```
1 [Corryvreckan]
2 log_level = "WARNING"
3 detectors_file = "geometry.conf"
4 number_of_tracks = 900000
5
6 [EventLoaderEUDAQ2]
7 file_name = "data/run0000456.raw"
8 inclusive = false
9 buffer_depth = 1000
10 shift_triggers = -1
```

```
1 [W0013_D04]
2 number_of_pixels = 256, 256
3 orientation = 10.9deg, 17.2deg, -1.3deg
4 orientation_mode = "xyz"
5 pixel_pitch = 55um, 55um
6 position = 886.5um, 270um, 0
7 spatial_resolution = 4.8um, 4.8um
8 time_resolution = 1.56ns
9 type = "Timepix3"
10 role = "reference"
```

Some Notable Features:

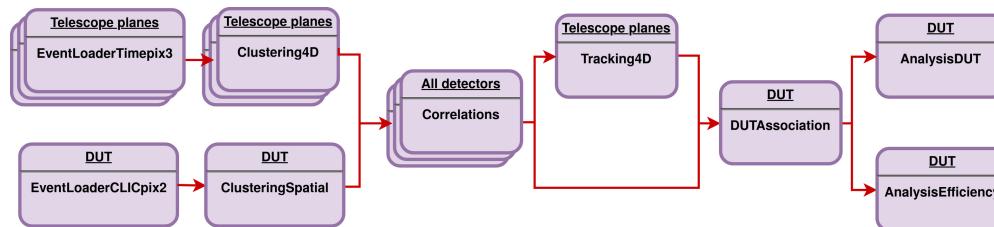
- 4D tracking (spatial + time cuts)
- various alignment methods (track χ^2 , Millepede)
- General Broken Line (GBL) track reconstruction (multiple Coulomb scattering)
- **EUDAQ** integration <https://github.com/eudaq/eudaq/>
 - include **AIDA TLU** as auxiliary device
 - process data recorded with **EUDAQ2 DAQ**
- job submission tool (HTCondor etc.)
- read in simulated data from **Allpix Squared** <https://cern.ch/allpix-squared>



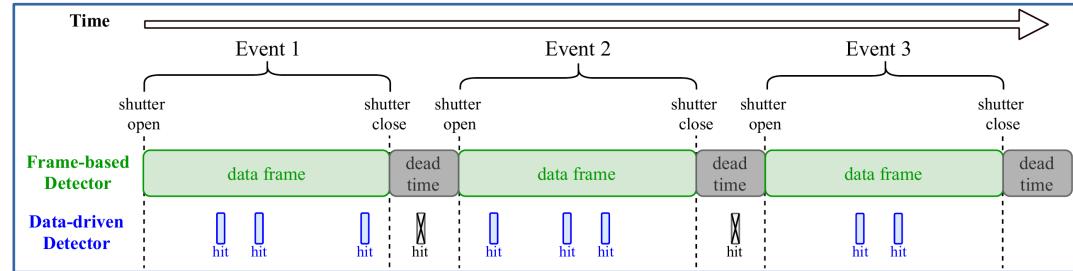
Combining Different Readout Schemes

- event building:
arrange data from different devices in
“time slices” (events) for reconstruction/analysis
- flexible:
combine devices with different readout schemes
 - frame-based,
 - data-driven,
 - triggered,
 - ...

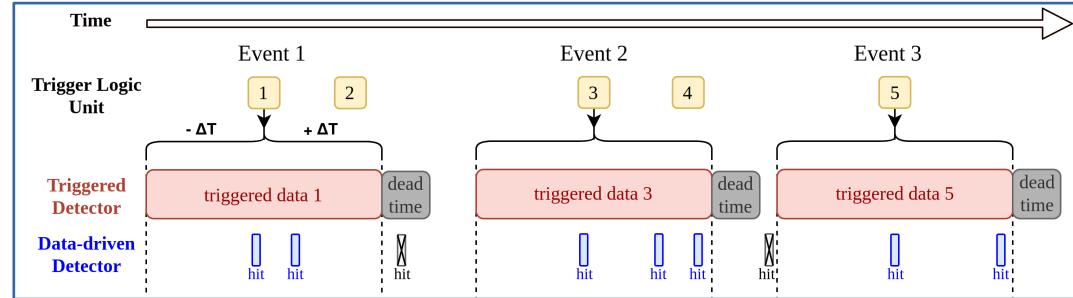
→ full analysis chain event-by-event



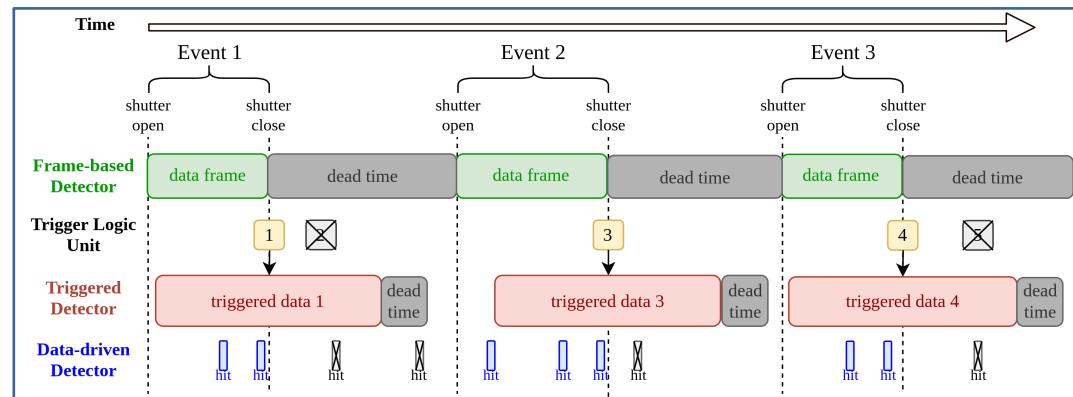
Example 1: frame-based + data-driven detectors



Example 2: triggered + data-driven detectors



Example 3: frame-based + triggered + data-driven detectors



Corryvreckan

in short
reconstruction and analysis tool
for pixel detector test beam data

- highly flexible/configurable
 - separate modules for each reconstruction/analysis step
 - many different event building options
- comprehensive documentation + beginner-friendly tutorials
- growing number of users + contributors

Learn more:

 Visit our website:

<https://cern.ch/corryvreckan>

 Browse through our manual:

→ [Get the lastest version here](#)

 Try our tutorials:

→ [Get Started](#) (no prior experience required)
→ [Advanced](#) (more complex use cases)

 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>

Backup

more details and examples...

Example Publications

for which Corryvreckan has been used

- Florian Pitters:
Time Resolution Studies with Timepix3 Assemblies with Thin Silicon Pixel Sensors
JINST 14 (2019) 05, P05022
DOI: 10.1088/1748-0221/14/05/P05022
- Mathieu Benoit:
Pixel detector R&D for the Compact Linear Collider
JINST 14 (2019) 06, C06003
DOI: 10.1088/1748-0221/14/06/C06003
- Magdalena Munker:
Vertex and tracking detector R&D for CLIC
Nucl.Instrum.Meth.A 980 (2020) 164475
DOI: 10.1016/j.nima.2020.164475
- Morag Williams:
R&D for the CLIC Vertex and Tracking detectors
JINST 15 (2020) 03, C03045
DOI: 10.1088/1748-0221/15/03/C03045
- Jens Kröger:
Silicon Pixel R&D for the CLIC Tracking Detector
JINST 15 (2020) 08, C08005
DOI: JINST 15 (2020) 03, C03045

PhD Theses

- Florian Pitters:
Silicon Detector Technologies for Future Particle Collider Experiments
<https://cds.cern.ch/record/2714709?ln=en>
- Thorben Quast:
Qualification, Performance Validation and Fast Generative Modelling of Beam Test Calorimeter Prototypes for the CMS Calorimeter Endcap Upgrade
<https://cds.cern.ch/record/2725040?ln=en>
- Morag Williams:
Evaluation of Fine-Pitch Hybrid Silicon Pixel Detector Prototypes for the CLIC Vertex Detector in Laboratory and Test-Beam Measurements
(work-in-progress)

Configuring Corryvreckan

- TOML style = easy to read
- support of physical units (e.g. 25um)
- need 2 files:
 - **configuration** file:
analysis parameters
 - **geometry** file:
detector description

example configuration

```
jens: ~/software/corryvreckan/testing
```

```
[Corryvreckan]
log_level = "WARNING"
log_format = "DEFAULT"

detectors_file = "geometries/geometry_timepix3_telescope.conf"
histogram_file = "histograms_run9444.root"

[Metronome]
event_length = 200us

[EventLoaderTimepix3]
input_directory = "data/timepix3tel_ebeam120"

[Clustering4D]
time_cut_abs = 100ns

[Tracking4D]
min_hits_on_track = 6
momentum=120GeV
time_cut_abs = 200ns
track_model = "gbl"
spatial_cut_abs = 200um, 200um
volume_scattering_length=304m

-UU-----F1 test_timepix3tel_ebeam120_gbl.conf
For information about GNU Emacs and the GNU system,
.
```

example geometry

```
jens: ~/software/corryvreckan/testing
```

```
[W0013_D04]
number_of_pixels = 256,256
orientation = 10.7471deg,186.437deg,-1.33797deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = 923.402um,296.86um,0
spatial_resolution = 4um,4um
material_budget = 0.01068
time_resolution = 20ns
type = "Timepix3"

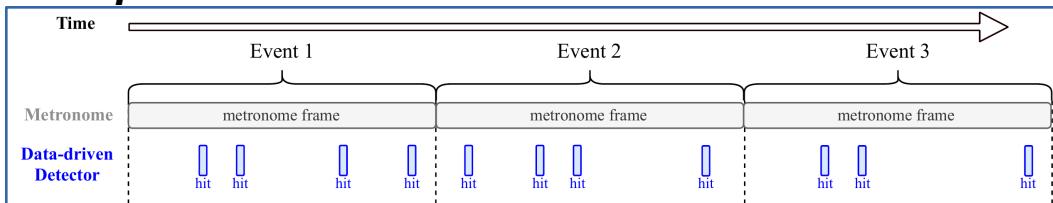
[W0013_E03]
number_of_pixels = 256,256
orientation = 11.0172deg,186.658deg,-1.06937deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = -249.154um,408.592um,21.5mm
spatial_resolution = 4um,4um
material_budget = 0.01068
time_resolution = 20ns
type = "Timepix3"

-UU-----F1 geometry_timepix3_telescope.conf Top L1 Git
For information about GNU Emacs and the GNU system, type C-h \
C-a.
```

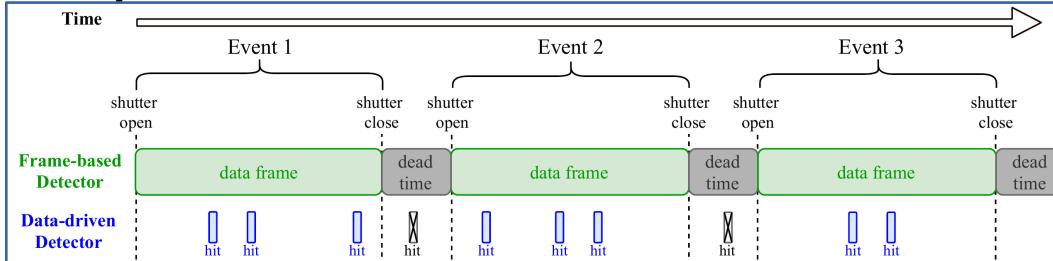
More Event Building Examples

- event building:
arrange data from different devices in
“time slices” (events) for reconstruction/analysis
 - flexible:
combine devices with different readout schemes
 - frame-based, data-driven, triggered, ...
- full analysis chain event-by-event

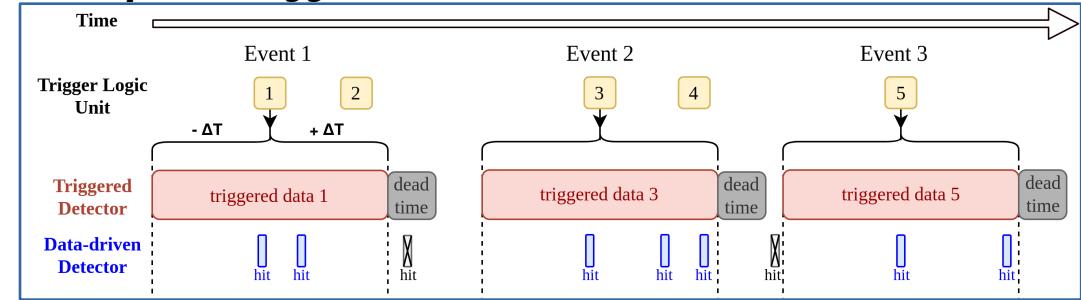
Example 1: data-driven detector



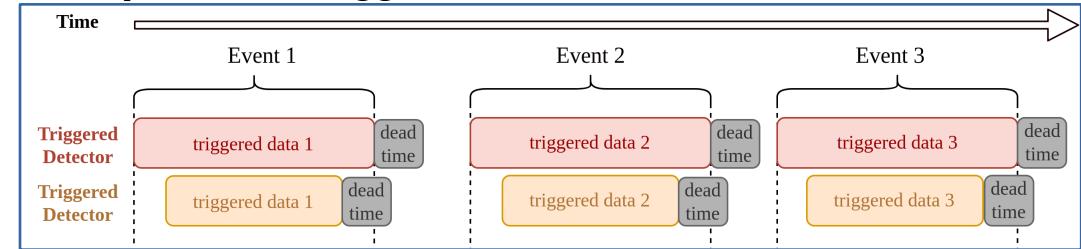
Example 2: frame-based + data-driven detectors



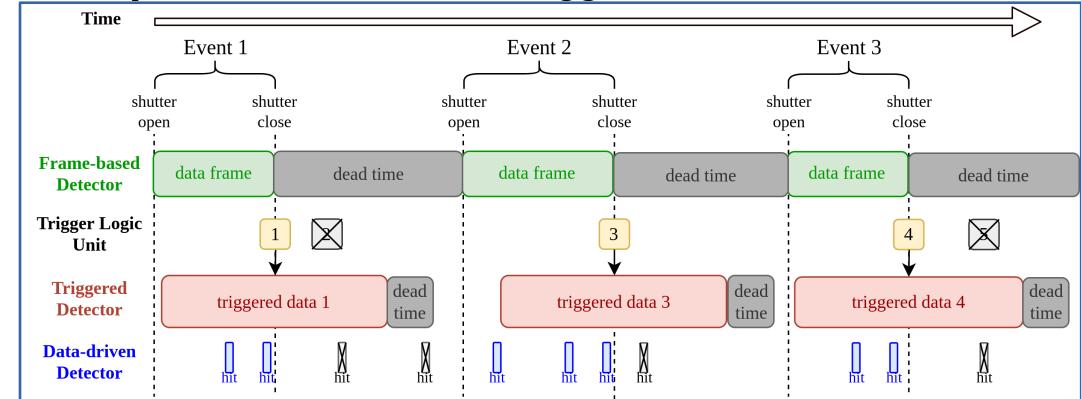
Example 3: triggered + data-driven detectors



Example 4: two triggered detectors



Example 5: frame-based + triggered + data-driven detectors



Documentation

- **online documentation** in repo

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

- every modules has a README

- **extensive user manual**

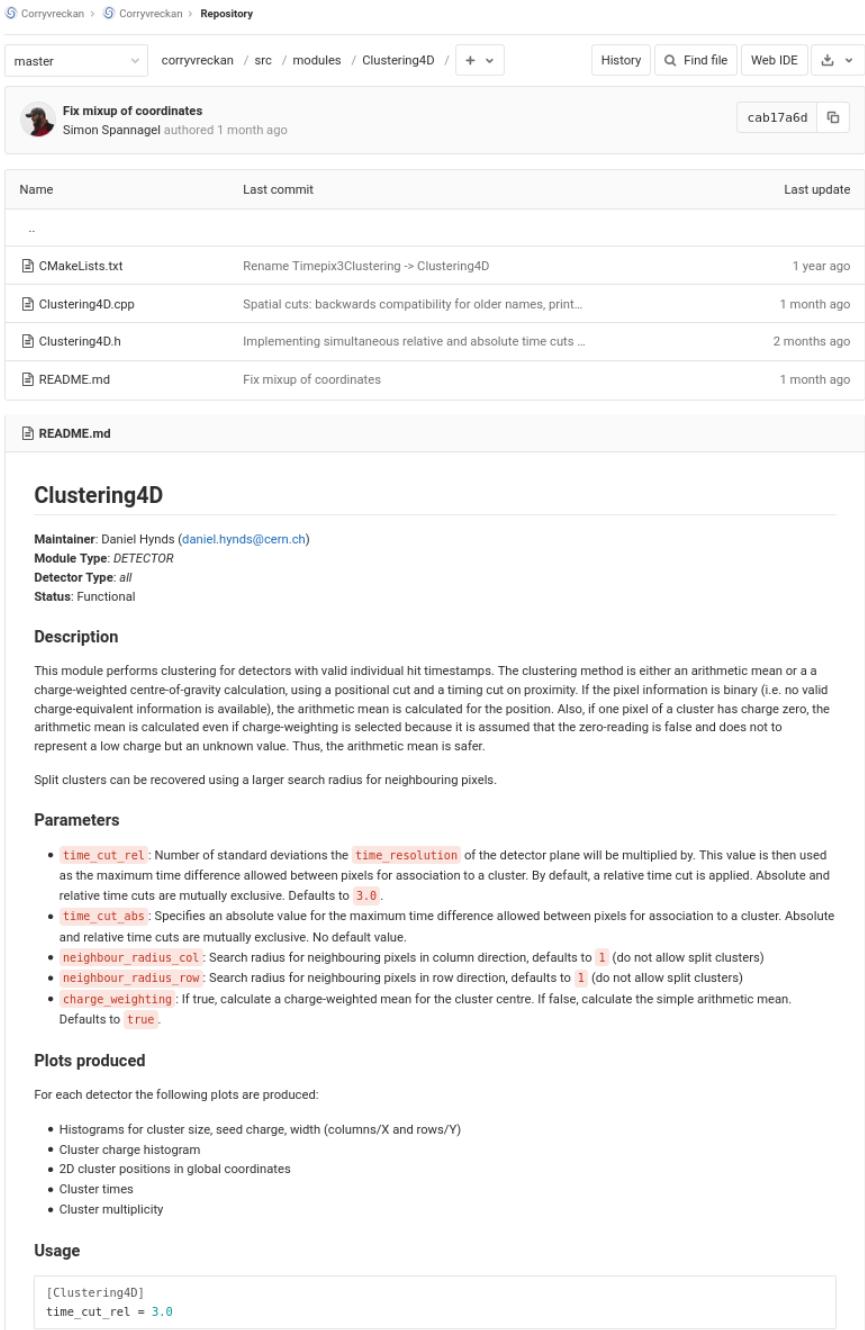
<https://cern.ch/corryvreckan/usermanual/corryvreckan-manual.pdf>

- full description of framework
 - installation instructions
 - “Getting started”, FAQs
 - module descriptions (fetched from repo)
 - published as CLICdp note:
<https://cds.cern.ch/record/2703012>

- **Doxygen code reference**

<https://cern.ch/corryvreckan/reference/>

- more details on code



The screenshot shows a GitLab repository interface for the 'corryvreckan' project, specifically the 'Clustering4D' module. The repository has a single commit titled 'Fix mixup of coordinates' by Simon Spannagel, dated 1 month ago. The commit hash is cab17a6d. The repository table lists several files: CMakeLists.txt, Clustering4D.cpp, Clustering4D.h, README.md, and README.md (a duplicate entry). The README.md file is expanded, showing its content which describes the Clustering4D module. It includes details about the maintainer (Daniel Hynds), module type (DETECTOR), detector type (all), and status (Functional). The 'Description' section explains the clustering method for detectors with valid timestamps. The 'Parameters' section lists configuration options like time cut relative and absolute, neighbour radius column and row, and charge weighting. The 'Plots produced' section describes histograms and cluster position plots. The 'Usage' section shows a snippet of code for the Clustering4D module.

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 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 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

- online documentation in repo
<https://gitlab.cern.ch/corryvreckan/corryvreckan>
 - every modules has a README
- **extensive user manual**
<https://cern.ch/corryvreckan/usermanual/corryvreckan-manual.pdf>
 - full description of framework
 - installation instructions
 - “Getting started”, FAQs
 - module descriptions (fetched from repo)
 - published as CLICdp note:
<https://cds.cern.ch/record/2703012>
- Doxygen code reference
<https://cern.ch/corryvreckan/reference/>
 - more details on code

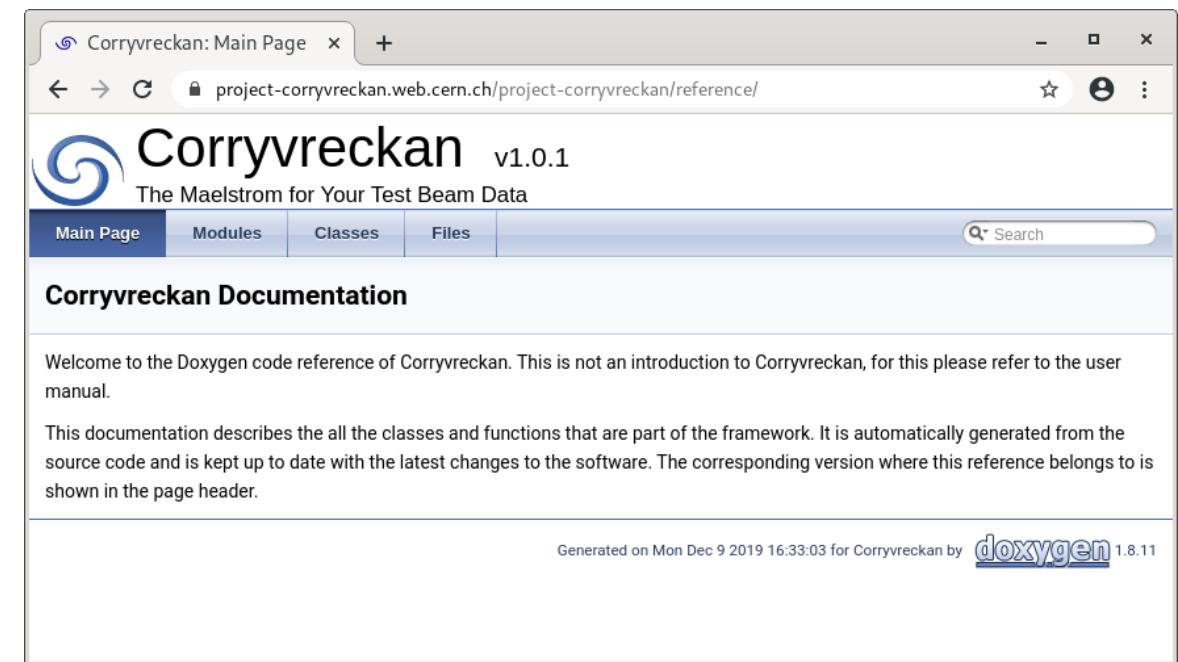


The screenshot shows the first page of the Corryvreckan User Manual. It features a large blue spiral logo at the top. Below the logo is the title "Corryvreckan User Manual". Under the title, the names and email addresses of the contributors are listed: Morag Williams (morag.williams@cern.ch), Simon Spannagel (simon.spannagel@cern.ch), and Jens Kröger (jens.kroeger@cern.ch). The date "December 9, 2019" and version "Version v1.0.1" are also present. To the right of the main content area, there is a "Contents" table of contents.

Section	Page
1 Introduction	1
1.1 Scope of this Manual	1
1.1.1 Getting Started	2
1.2 Support and Reporting Issues	2
1.3 Contributing Code	2
2 Installation	3
2.1 Supported Operating Systems	3
2.2 CMVFS	3
2.3 Docker	3
2.4 Binaries	4
2.5 Compilation from Source	4
2.5.1 Prerequisites	5
2.5.2 Downloading the source code	5
2.5.3 Configuration via CMake	5
2.5.4 Compilation and installation	6
3 The Corryvreckan Framework	7
3.1 The corry Executable	7
3.2 The Clipboard	8
3.2.1 The Event	9
3.2.2 Temporary Data Storage	9
3.2.3 Persistent Storage	9
3.3 Global Framework Parameters	9
3.4 Modules and the Module Manager	11
3.4.1 Module Status Codes	11
3.4.2 Execution Order	12
3.4.3 Module instantiation	12
3.5 Logging and Verbosity Levels	13
3.6 Coordinate Systems	14
4 Configuration Files	15
4.1 Parsing types and units	15
4.1.1 File format	18
4.1.2 Accessing parameters	19
4.2 Main configuration	20
4.3 Detector configuration	21
4.3.1 Masking Pixels Offline	24
4.3.2 Defining a Region of Interest	24

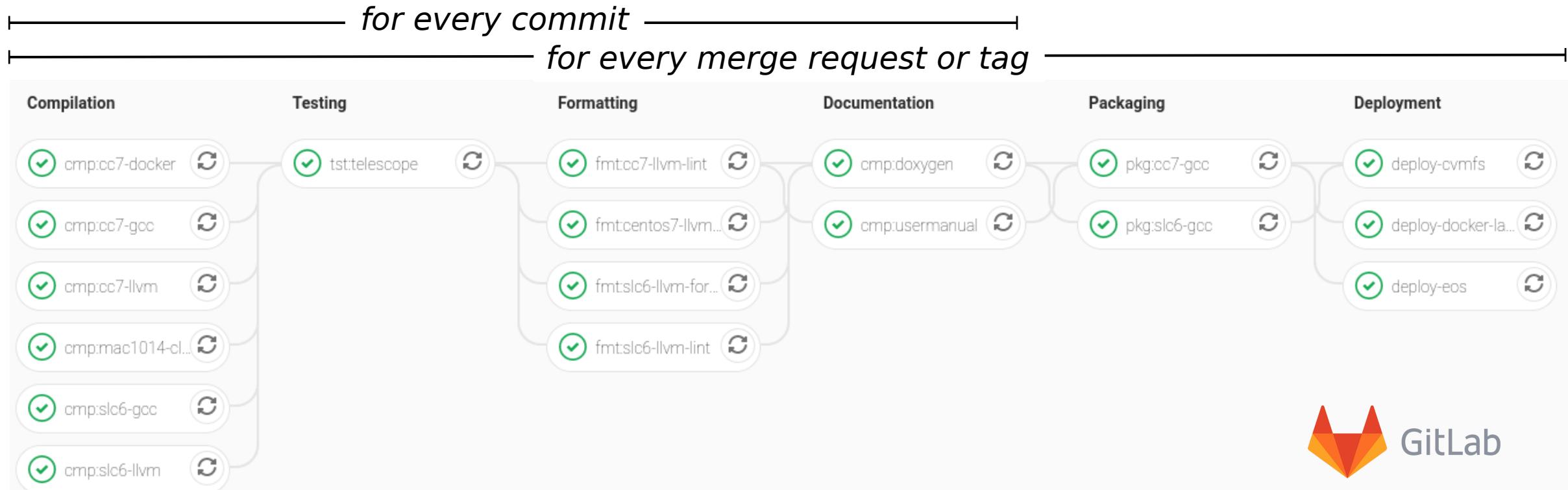
Documentation

- online documentation in repo
<https://gitlab.cern.ch/corryvreckan/corryvreckan>
 - every modules has a README
- extensive user manual
<https://cern.ch/corryvreckan/usermanual/corryvreckan-manual.pdf>
 - full description of framework
 - installation instructions
 - “Getting started”, FAQs
 - module descriptions (fetched from repo)
 - published as CLICdp note:
<https://cds.cern.ch/record/2703012>
- **Doxygen code reference**
<https://cern.ch/corryvreckan/reference/>
 - more details on code



GitLab Continuous Integration

- ensures compilation, formatting, functionality (all stages explained in backup)
- pipeline runs through for every commit



GitLab Continuous Integration - all stages

- **Compilation**

- compile source code on Scientific Linux 6, CentOS7, and Mac OS X with GCC, Clang, and AppleClang

- **Testing**

- analyse test data sets and compare output to pass conditions

- **Formatting**

- check format against defined syntax rules (e.g. tabs ↔ whitespaces) to avoid changes caused e.g. by different indentation, and apply linting

- **Documentation**

- compile user manual from LaTeX sources and generate Doxygen code reference

- **Packaging**

- generate release tarballs

- **Deployment**

- publish new version of CVMFS, new docker image in registry, new user manual and code reference on the website and release tarballs

