Corryvreckan Tutorial

THE MAELSTROM FOR YOUR TEST BEAM DATA

BTTB 7, Tuesday 15th January 2019

Morag Williams, University of Glasgow and CERN
On behalf of the CLICdp collaboration
morag.williams@cern.ch

WELCOME

- Corryvreckan reconstruction software hands-on tutorial
- What you've signed up for:
  - brief introduction to Corryvreckan: *https://gitlab.cern.ch/corryvreckan/corryvreckan*
  - installation and compilation
  - reconstruction: all together, start with defaults and build up to complexity
- By the end you'll be able to reconstruct raw data into tracks, configure your reconstruction to your needs, create DUT analyses plots required, and get multiple different outputs from Corryvreckan.
- 10 minute break part way through
- Encourage questions and interaction!

HOW TO INSTALL CORRYVRECKAN

There are four options:

1. Compile and install Corryvreckan locally, with local ROOT version - please follow the installation instructions in the user manual.
2. Use the Docker images - please refer to the user manual
3. Use Corryvreckan on LXPLUS using the centrally provided version on CVMFS. For this, you only need to source the appropriate script and you are ready to go:

   For CERN CentOS7:

   source /cvmfs/clicdp.cern.ch/software/corryvreckan/<version>/x86_64-centos7-gcc7-opt/setup.sh

   For CERN Scientific Linux 6:

   source /cvmfs/clicdp.cern.ch/software/corryvreckan/<version>/x86_64-slc6-gcc7-opt/setup.sh

4. Compile and install Corryvreckan locally or on LXPLUS, while using CVMFS version of ROOT - this works only for SLC6 and CentOS7 systems - install the CERN CVMFS daemon and source appropriate ROOT version using its .sh-script. Then compile Corryvreckan.

- **For all options including dependencies from CVMFS:** It might take a while until the CVMFS cache is populated with the necessary libraries when starting the program for the first time.
- More detailed instructions can be found in the "Installation" chapter of the Corryvreckan: *https://gitlab.cern.ch/corryvreckan/corryvreckan*.

INSTALLATION AND COMPILATION

- We will use lxplus installation for this tutorial, for those of you who don't have a working copy of Corryvreckan already.
- Step 1: check out the Corryvreckan repository into a local directory "corryvreckan"

```
git clone https://gitlab.cern.ch/corryvreckan/corryvreckan.git Corryvreckan
```

INSTALLATION AND COMPILATION

- We will use lxplus installation for this tutorial, for those of you who don't have a working copy of Corryvreckan already.
- Step 1: check out the Corryvreckan repository into a local directory "corryvreckan"

```
git clone https://gitlab.cern.ch/corryvreckan/corryvreckan.git Corryvreckan
```

- Step 2: Move to this directory and source the lxplus setup script:

```
cd corryvreckan/
source etc/setup_lxplus.sh
```

INSTALLATION AND COMPILATION

- We will use lxplus installation for this tutorial, for those of you who don't have a working copy of Corryvreckan already.
- Step 1: check out the Corryvreckan repository into a local directory "corryvreckan"

```
git clone https://gitlab.cern.ch/corryvreckan/corryvreckan.git Corryvreckan
```

- Step 2: Move to this directory and source the lxplus setup script:

```
cd corryvreckan/
source etc/setup_lxplus.sh
```

- Step 3: Compile the code with cmake:

```
mkdir build
cmake ..
make install –j8
```

DATA AND FILES FOR TUTORIAL

- Up to you if you want to run this in the Corryvreckan directory, or in a separate directory for analysis only.
- To download the data:

  wget https://cern.ch/corryvreckan/data/tutorial_data.tar.gz

  tar -xvf tutorial_data.tar.gz
- To get the required configuration files etc. needed, they can be downloaded from the indico page for this tutorial:

  https://indico.cern.ch/event/731649/contributions/3237289/

  3 x configuration files

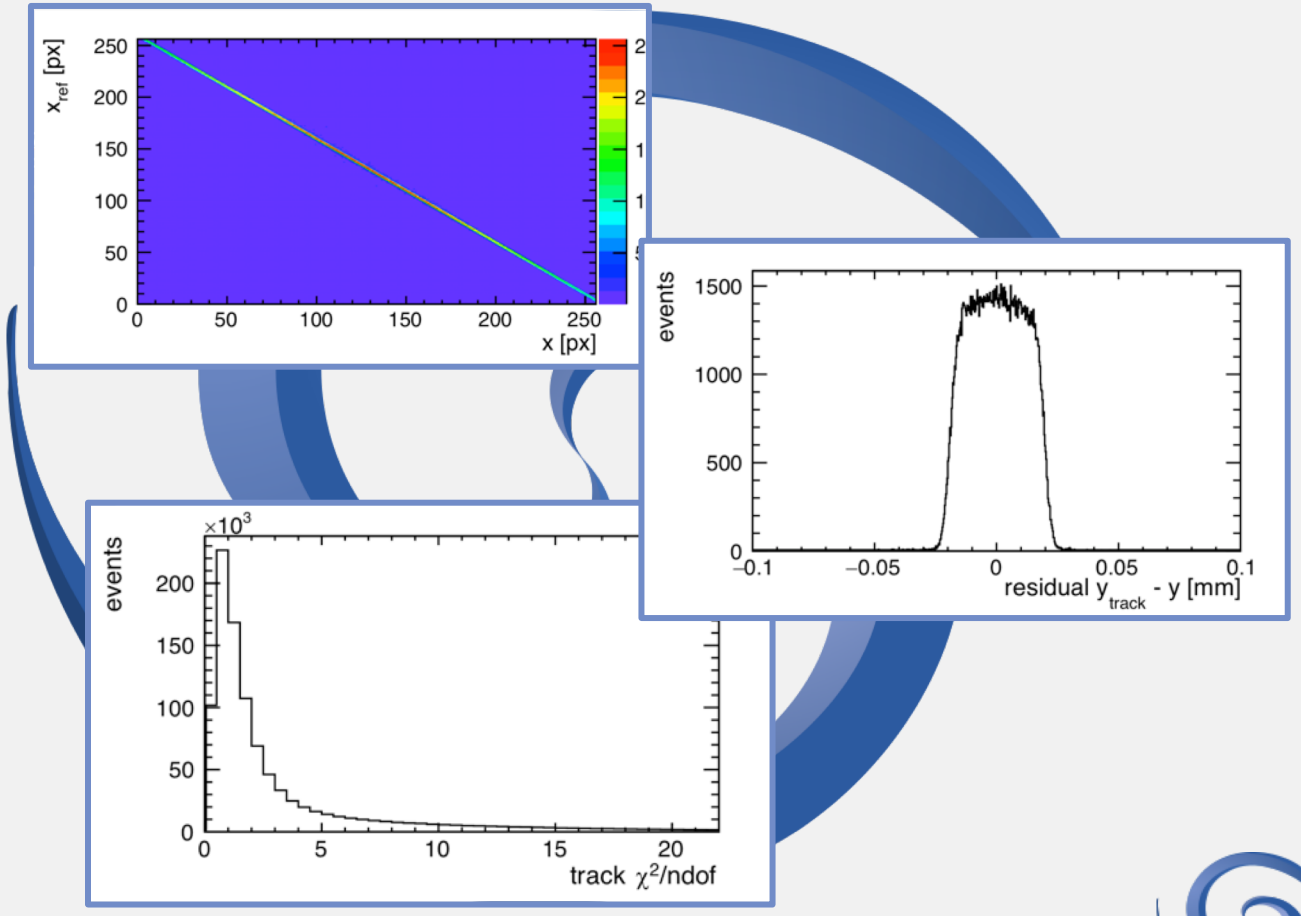
  2 x geometry files

  1 x mask file (for masking noisy pixels)

CORRYVRECKAN

BRIEF RE-INTRODUCTION TO CORRYVRECKAN

- **Reconstruct and analyse data from pixel assemblies**
- Modular structure - similar to Allpix-squared
- Highly flexible and configurable
- Easy to understand - written in modern C++, well documented



COOL FEATURES!

- 4D-tracking – Use both spatial and timing cuts to associate clusters to your track, improving track quality
- Millepede alignment – option to use millipede to align your telescope planes to high precision
- Online data monitoring – check data quality while reconstructing, very useful in test-beam environments
- Frame-based and data-driven readouts – can easily use combinations of triggered and/or trigger-less devices in the same reconstruction using the "Metronome" module
- Modular approach - plug-and-play algorithms for specific tasks, allows quick set-up, and quick configuration

WHAT'S IN CORRYVRECKAN?

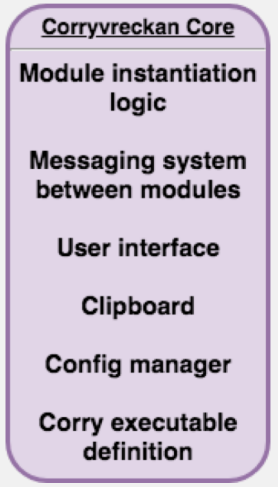
Lets have a look at what's in the main Corryvreckan directory:

- src/ : contains all source code
- src/modules/ : code and readmes for all available modules
- Readme.md : short introduction, getting started info
- Contributing.md : guide to contributing to Corryvreckan

- doc/ : contains documentation, users manual, …
- etc/ : lxplus set-up script and more
- jobsub/ : all info needed to use jobsub with corry
- src/core/ : source code of the core framework

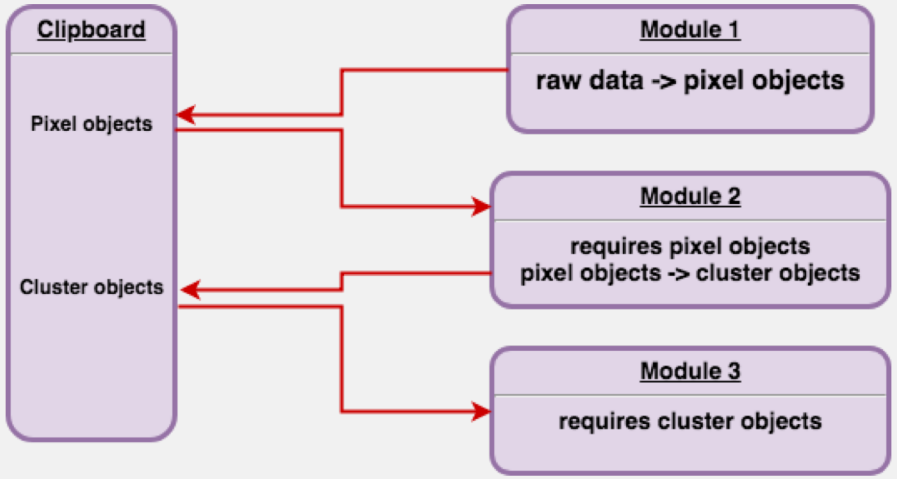
CORRYVRECKAN FRAMEWORK

- Written in modern C++
- Clipboard = framework's infrastructure for temporary storing information during the event processing.
- Corryvreckan storage objects: pixel hits, clusters, tracks, Spidr signals, mc particles, etc.
- Modules = 'plug-and-play' concept for algorithms for specific tasks, using objects from the clipboard.



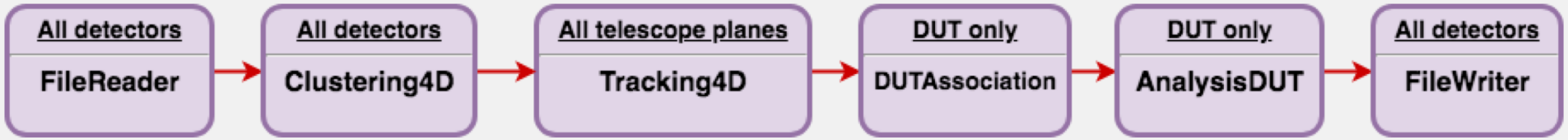
CORRYVRECKAN FRAMEWORK

- Written in modern C++
- Clipboard = framework's infrastructure for temporary storing information during the event processing.
- Corryvreckan storage objects: pixel hits, clusters, tracks, Spidr signals, mc particles, etc.
- Modules = 'plug-and-play' concept for algorithms for specific tasks, using objects from the clipboard.



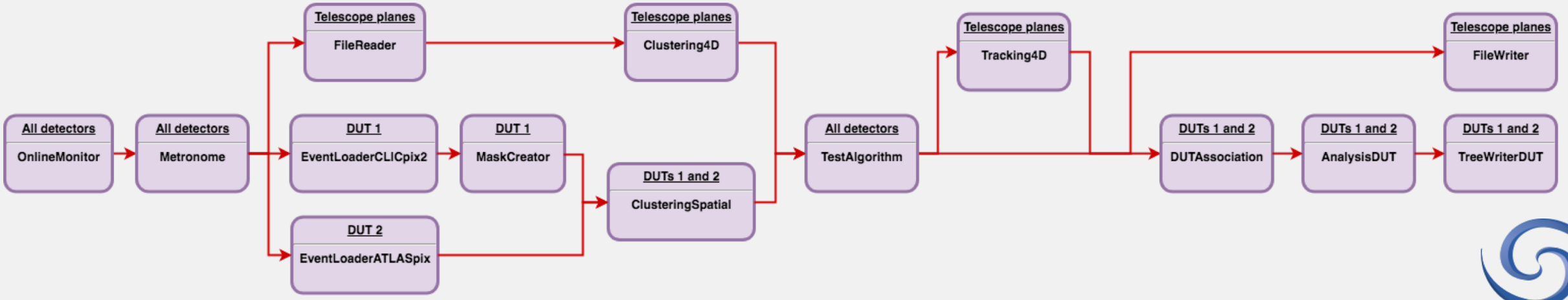
MODULAR APPROACH

- Select suitable clustering, tracking, ... modules
- Modular approach allows quick set-up, and quick configuration

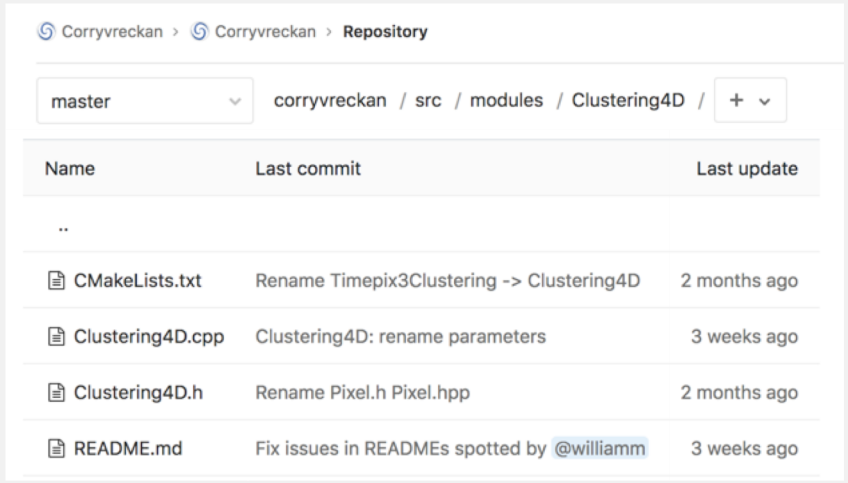


MODULAR APPROACH

- Can create more involved reconstruction chains
- Apply different modules to different devices in the same reconstruction (shown later in tutorial)
- Modules are either type 'DETECTOR', 'DUT' or type 'GLOBAL': for the 1st type, the module is made once for each detector it is to work on; for the 2nd the module is made once for each DUT; for the 3rd type, the module is only created once.



FIND ALL MODULE INFO IN SRC/MODULES/



Clustering4D

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

Description

This module performs clustering on data from a Timepix3 device. The clustering method is a charge-weighted centre of gravity calculation, using a positional cut and a timing cut on proximity.

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

Parameters

- `timing_cut`: The maximum value of the time difference between two pixels for them to be associated in a cluster. Default value is `100ns`.
- `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)

Plots produced

For each detector the following plots are produced:

- Cluster size histogram
- Cluster width (rows, in X) histogram
- Cluster width (columns, in Y) histogram
- Cluster ToT histogram
- 2D cluster positions in global coordinates

Usage

```
[Timepix3Clustering]
timing_cut = 200ns
```

FIRST ANALYSIS

TUTORIAL DATA

- Data: test-beam data from June 2018, SPS North Area (120GeV pion/proton beam), CLICdp telescope set-up
- Telescope planes: Timepix3 hybrid pixel detector (pitch 55um, 256x256 pixels, 6 planes)
- Data-driven readout, operated in ToT + ToA mode
- Look only at the telescope for now

LETS DO SOME ANALYSIS!

- 1. Set-up of configuration and geometry
- 2. Running Corryvreckan reconstruction
- 3. Looking at the analysis output

1. SET-UP FOR YOUR ANALYSIS

- 2 files needed for reconstruction:
  1. configuration file
  2. geometry file

config_firstanalysis.cfg

```
[Corryvreckan]
detectors_file=alignment_firstanalysis.geo
histogram_file=histograms_firstanalysis.root

[Metronome]
#default event length is 10us

[EventLoaderTimepix3]
input_directory=tutorial_data/

[Clustering4D]

[TestAlgorithm]

[Tracking4D]
```

alignment_firstanalysis.geo

```
[Detector_1]
number_of_pixels = 256,256
orientation = 11.0662deg,186.374deg,-1.21358deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = 922.323um,285.641um,0
resolution = 4um,4um
type = "Timepix3"

[Detector_2]
number_of_pixels = 256,256
orientation = 11.292deg,186.662deg,-0.962569deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = -275.557um,397.894um,21.5mm
resolution = 4um,4um
type = "Timepix3"

[Detector_3]
number_of_pixels = 256,256
```

1. SET-UP FOR YOUR ANALYSIS

- Open our first config file "config_firstanalysis.conf".

- Configuration files define:
  - modules to be run
  - local module and global framework parameters
  - input data and analysis outputs

- For first analysis, we'll use default settings

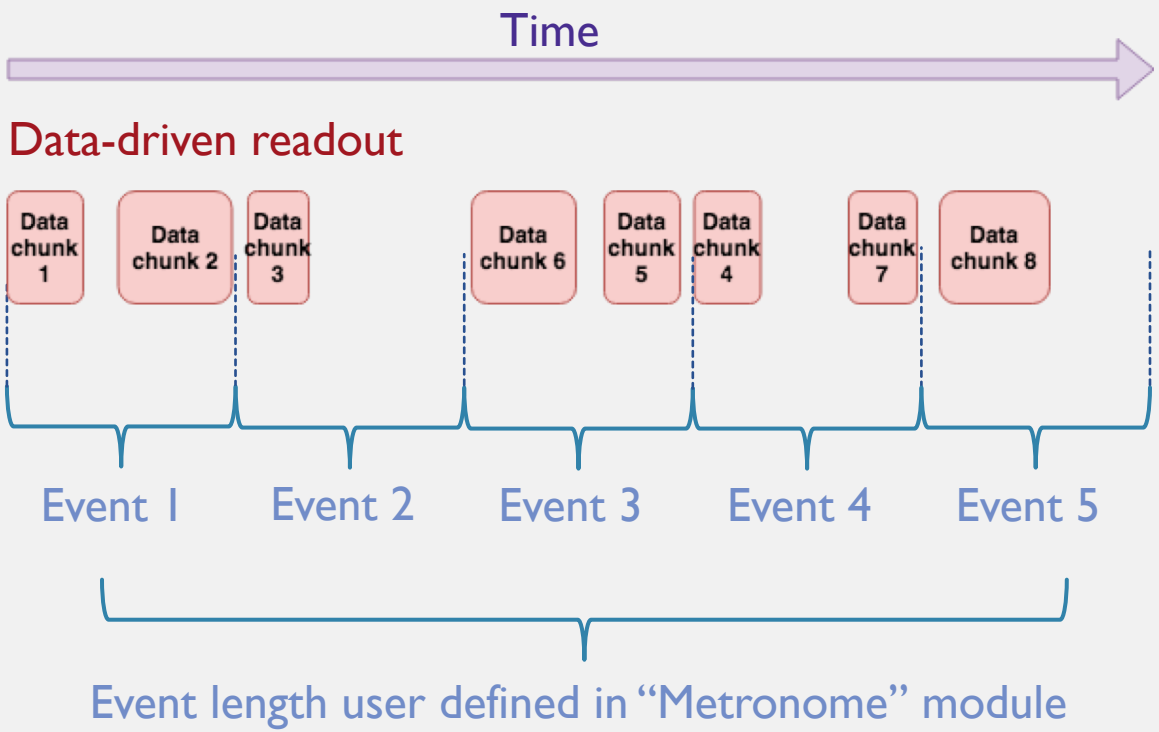
config_firstanalysis.cfg

```
1  [Corryvreckan]
2  detectors_file=alignment_firstanalysis.geo
3  histogram_file=histograms_firstanalysis.root
4
5  [Metronome]
6  #default event length is 10us
7
8  [EventLoaderTimepix3]
9  input_directory=tutorial_data/
10
11 [Clustering4D]
12
13 [TestAlgorithm]
14
15 [Tracking4D]
16
```

NOTE: METRONOME MODULE

- Can easily use combinations of devices with frame-based and/or data-driven readouts in the same reconstruction
- Need arbitrary chunks of data to process ('events')
- Frame-based: Corryvreckan uses the start and end points of the frame to define an event
- Data-driven: either use the "Metronome" module to define the length of an event…

  or use the definition of an event from a frame-based readout device



1. SET-UP FOR YOUR ANALYSIS

- Open our first geometry file "alignment_firstanalysis.geo"

- Geometry files define:

  - number and types of devices

  - position and orientation of each device

  - role of reference plane and role(s) of DUT

- For first analysis, we'll use an already aligned geometry

alignment_firstanalysis.geo

```
[Detector_1]
number_of_pixels = 256,256
orientation = 11.0662deg,186.374deg,-1.21358deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = 922.323um,285.641um,0
resolution = 4um,4um
type = "Timepix3"

[Detector_2]
number_of_pixels = 256,256
orientation = 11.292deg,186.662deg,-0.962569deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = -275.557um,397.894um,21.5mm
resolution = 4um,4um
type = "Timepix3"

[Detector_3]
number_of_pixels = 256,256
orientation = 10.5845deg,187.075deg,-1.54544deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = 5.292um,378.368um,43.5mm
resolution = 4um,4um
type = "Timepix3"
role="reference"
```

alignment_firstanalysis.geo

```
[Detector_4]
number_of_pixels = 256,256
orientation = 8.99876deg,8.99412deg,-0.0178763deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = -10.202um,-11.943um,186.5mm
resolution = 4um,4um
type = "Timepix3"

[Detector_5]
number_of_pixels = 256,256
orientation = 7.79561deg,9.70442deg,1.26538deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = 421.587um,-590.797um,208.5mm
resolution = 4um,4um
type = "Timepix3"

[Detector_6]
number_of_pixels = 256,256
orientation = 8.01465deg,9.97686deg,0.0731667deg
orientation_mode = "xyz"
pixel_pitch = 55um,55um
position = -1.10371mm,-18.72um,231.5mm
resolution = 4um,4um
type = "Timepix3"
```

1. SET-UP FOR YOUR ANALYSIS

Note on units:

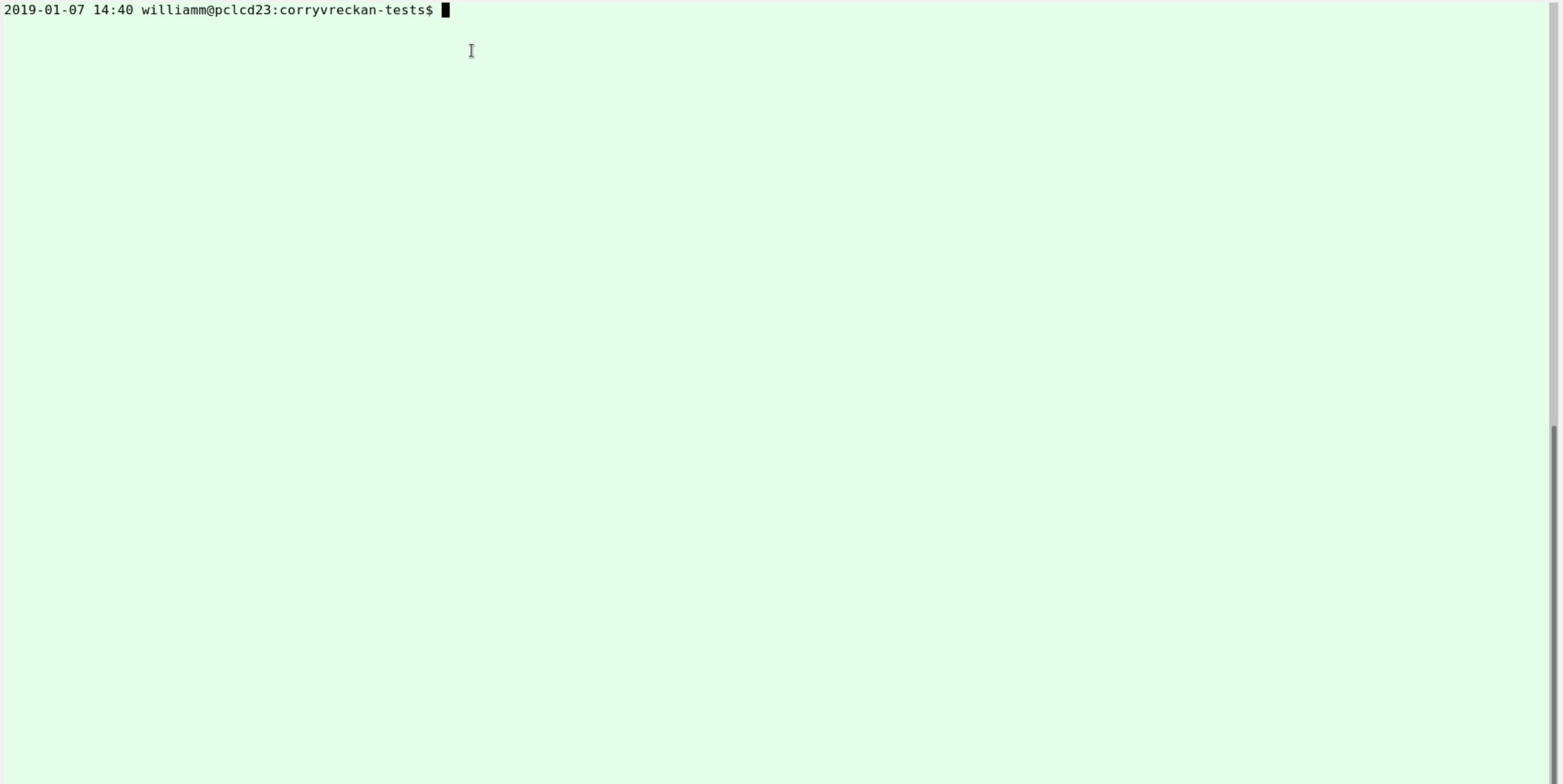
- Corryvreckan can interpret readable units (e.g. nm, ps, fF).
- If no units are specified, values will be interpreted in the base units of the framework → can lead to unexpected results!
- E.g. "bias_voltage = 50 results" in an applied voltage of **50 MV**.
- Recommended to always specify units in the configuration files.

```
5   pixel_pitch = 55um,55um
6   position = 922.323um,285.641um,0
7   resolution = 4um,4um
8   type = "Timepix3"
9
10  [Detector_2]
11  number_of_pixels = 256,256
12  orientation = 11.292deg,186.662deg,-0.962569deg
13  orientation_mode = "xyz"
14  pixel_pitch = 55um,55um
15  position = -275.557um,397.894um,21.5mm
16  resolution = 4um,4um
17  type = "Timepix3"
18
19  [Detector_3]
20  number_of_pixels = 256,256
21  orientation = 10.5845deg,187.075deg,-1.54544deg
```

2. THE CORRY EXECUTABLE

- First step, please type `corry` to see Corryvreckan version and possible options
- To start analysis: `corry -c /path/to/config/config_firstanalysis.conf`

2. GET SOMETHING LIKE THIS:

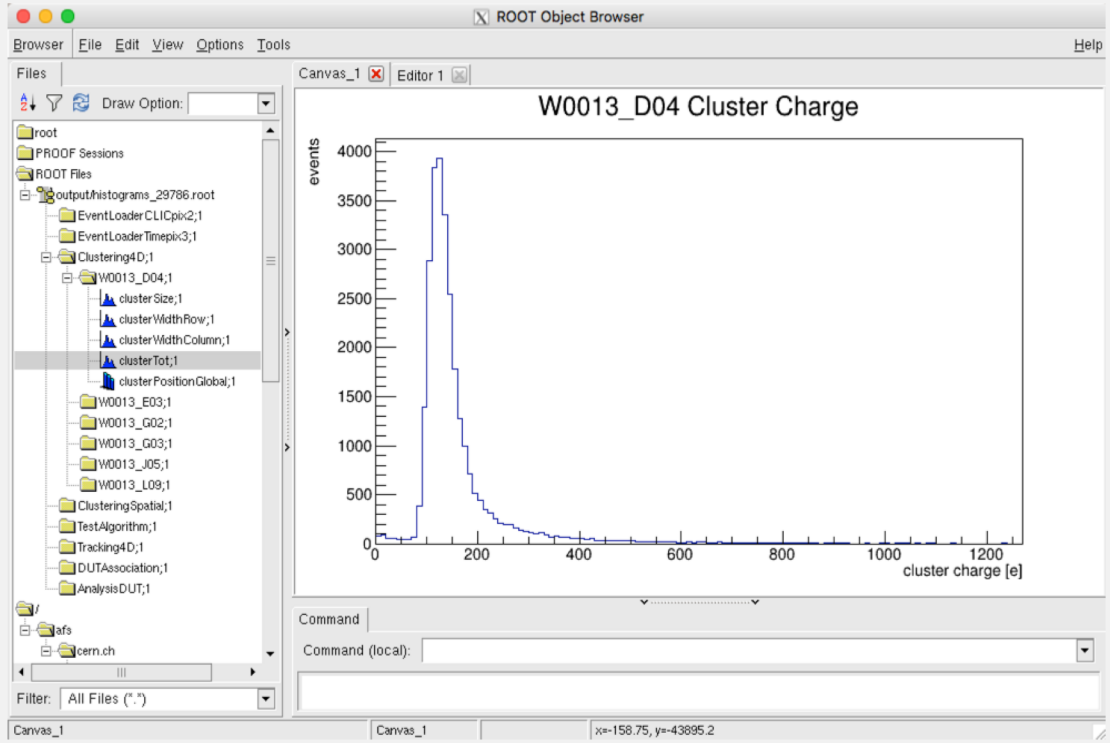


3. OUTPUTS

- On terminal output and root file output
- There are other available outputs (explored later in the tutorial)
- Lets open the root file named "histograms_firstanalysis.root":

  root -l histograms_firstanalysis.root

- Directory for each module instantiated, containing all analysis plots produced



ADDING SOME COMPLEXITY

CONFIGURING MODULE PARAMETERS AND ADDING A DUT

TYPES OF PARAMETERS

- Global framework parameters (config file) – parameters inherited by all modules and defined at the top of the configuration file, for example the path to the output histogram file.
- Module parameters (config file) – these parameter settings are only used by the module they are defined in, for example verbosity and spatial cut size.
- Detector parameters (geometry file) – detector specific parameters, such as plane orientation and pixel pitch.

telescope_cp2_commandline.conf

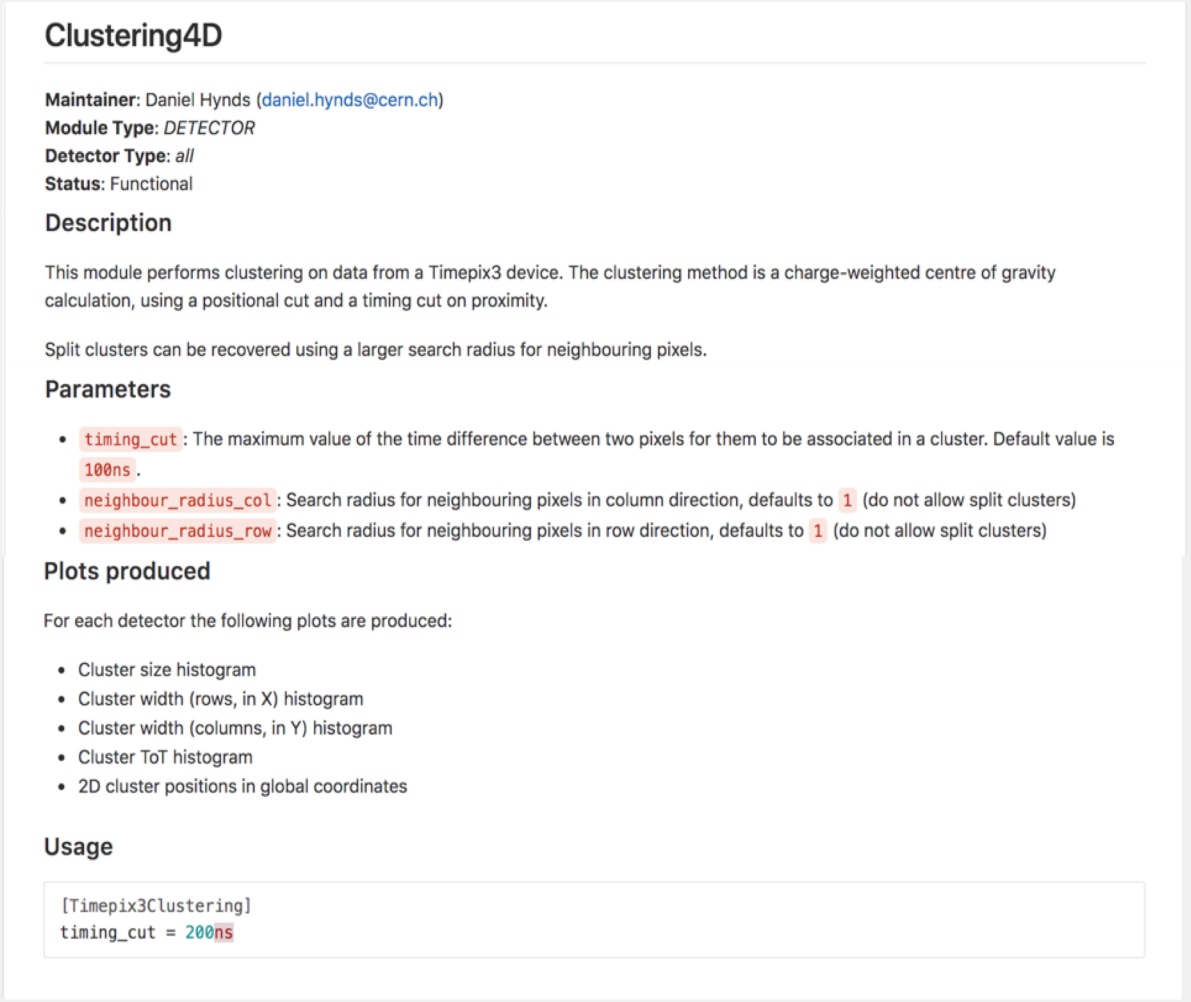
```
[Corryvreckan]
log_level = "WARNING"
log_format = "DEFAULT"
detectors_file=../geometries/Alignment_November2018_3.geo
histogram_file=histograms_29786.root
number_of_tracks=20000

[EventLoaderCLICpix2]
discard_zero_tot = true
input_directory=/eos/experiment/clicdp/grid/ilc/user/c/clictel/CERN_TB_November_2018/clicpix2/Run29786

[EventLoaderTimepix3]
input_directory=/eos/experiment/clicdp/grid/ilc/user/c/clictel/CERN_TB_November_2018/data/Run29786

[Clustering4D]
type = "Timepix3"

[ClusteringSpatial]
type = "CLICpix2"

[TestAlgorithm]
make_correlations=true
timing_cut=2ms

[Tracking4D]
min_hits_on_track=5

[DUTAssociation]
spatial_cut = 200um 200um
timing_cut = 2ms
```

MODULE CONFIGURABILITY

- All parameters that are configurable for a module are outlined in the manual, or you can find them in the "README.md" file next to the module source code.
- We shall adjust the timing cut of the "Clustering4D" module in "config_firstanalysis.conf"
- Currently using default value of 100ns → lets change that to 50ns:

  [Clustering4D]

  timing_cut=50ns



Clustering4D

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

Description

This module performs clustering on data from a Timepix3 device. The clustering method is a charge-weighted centre of gravity calculation, using a positional cut and a timing cut on proximity.

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

Parameters

- `timing_cut` : The maximum value of the time difference between two pixels for them to be associated in a cluster. Default value is `100ns`.
- `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)

Plots produced

For each detector the following plots are produced:

- Cluster size histogram
- Cluster width (rows, in X) histogram
- Cluster width (columns, in Y) histogram
- Cluster ToT histogram
- 2D cluster positions in global coordinates

Usage

```
[Timepix3Clustering]
timing_cut = 200ns
```

MODULE CONFIGURABILITY

- We shall also change the output of the "TestAlgorithm" module
- Currently using default value of "make_correlations", therefore no correlation plots are outputted in the ROOT file. Lets add them:

```
[TestAlgorithm]
make_correlations = true
```

- Now if we run our analysis again, we will be applying a more stringent timing cut during clustering, and will produce correlation plots in the ROOT output file.

TestAlgorithm

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

Description

This module collects `pixel` and `cluster` objects from the clipboard and creates correlation and timing plots with respect to the reference detector.

Parameters

- `make_correlatons` : Boolean to change if correlation plots should be outputted. Default value is `false` .
- `do_timing_cut` : Boolean to switch on/off the cut on cluster times for correlations. Defaults to `false` .
- `timing_cut` : maximum time difference between clusters to be taken into account. Only used if `do_timing_cut` is set to `true` , defaults to `100ns` .

Plots produced

For each device the following plots are produced:

- 2D hitmap
- 2D event times histogram
- Correlation in X
- Correlation in Y
- 2D correlation in X in global coordinates
- 2D correlation in Y in global coordinates
- 2D correlation in X in local coordinates
- 2D correlation in Y in local coordinates
- Correlation times (nanosecond binning) histogram, range covers 2 * event length
- Correlation times (integer values) histogram

Usage

```
[TestAlgorithm]
make_correlations = true
```

CONFIGURABILITY FROM THE COMMAND LINE

Sets the global log verbosity level

module configurations - specified by adding a dot (.) between the module and the key

```
corry  -c conf.cfg  -l log.txt  -v "INFO"  -o histogram_file="hists"  -o FileWriter.onlyDUT="true"  -g mydut.orientation=0deg,5deg,0deg
```

Specify an additional location such as a file to forward log output to.

framework parameters

detector parameters (i.e. changes to the geometry file)

ADDING A DUT!

- Same data set, but we shall also look at the DUT plane
- Telescope planes: Timepix3 hybrid pixel detector (pitch 55um, 256x256 pixels, 6 planes)
- Data-driven readout, operated in ToT + ToA mode
- DUT: CLICpix2 hybrid pixel detector (25um, 128x128 pixels)
- Frame-based readout, operated in ToT + counts mode

DEFINING A DUT IN THE GEOMETRY FILE

- Open up the geometry file called "alignment_withdut.geo"
- There are 2 roles a device can have:
  1. "reference" – plane to which the other are compared with
  2. "dut" - the device-under-test plane

We have a CLICpix2 device as our DUT, so it has been assigned this role in the geometry file

alignment_withdut.geo

```
19 [Detector_3]
20 number_of_pixels = 256,256
21 orientation = 10.5845deg,187.075deg,-1.54544deg
22 orientation_mode = "xyz"
23 pixel_pitch = 55um,55um
24 position = 5.292um,378.368um,43.5mm
25 resolution = 4um,4um
26 type = "Timepix3"
27 role="reference"
28
29 [CLICpix2_assembly]
30 mask_file = "mask_016_CP_PS.conf"
31 number_of_pixels = 128,128
32 orientation = 1.78299deg,0.247231deg,-1.31437deg
33 orientation_mode = "xyz"
34 pixel_pitch = 25um,25um
35 position = -670.567um,287.915um,106mm
36 resolution = 4um,4um
37 type = "CLICpix2"
38 role="dut"
39
40 [Detector_4]
41 number_of_pixels = 256,256
42 orientation = 8.99876deg,8.99412deg,-0.0178763deg
43 orientation_mode = "xyz"
44 pixel_pitch = 55um,55um
45 position = -10.202um -11.943um 186.5mm
```

USING FRAME-BASED AND DATA-DRIVEN DEVICES TOGETHER

- Open the configuration file "config_withdut.conf".
- Need data in manageable chunks ('events').
- Timepix3 has data driven readout → had to define what an event was with the "Metronome" module.
- CLICpix2 has frame-by-frame readout → already has a definition of the start and end points of an event.
- Therefore we can define events using the CLICpix2 frame definition from data, and load Timepix3 data using this event definition.
- Removes need for "Metronome" module.
- Requires "EventLoaderCLICpix2" to be before "EventLoaderTimepix3".

old

config_firstanalysis.cfg

```
[Corryvreckan]
detectors_file=alignment_firstanalysis.geo
histogram_file=histograms_firstanalysis.root

[Metronome]
#default event length is 10us

[EventLoaderTimepix3]
input_directory=tutorial_data/
```

new

config_withdut.cfg

```
[Corryvreckan]
detectors_file=alignment_withdut.geo
histogram_file=histograms_withdut.root
number_of_tracks=50000

[EventLoaderCLICpix2]
input_directory=tutorial_data/

[EventLoaderTimepix3]
input_directory=tutorial_data/
```

OTHER ADDITIONS TO THE CONFIG FILE FOR A DUT

- "ClusteringSpatial" – this CLICpix2 data doesn't contain timing information → use spatial information for clustering
- "DUTAssociation" – associates DUT clusters with tracks
- "AnalysisDUT" – produces analysis plots for the DUT plane
- Points to new geometry file "alignment_withdut.geo"
- Can we see a problem here?

config_withdut.cfg

```
[Corryvreckan]
detectors_file=alignment_withdut.geo
histogram_file=histograms_withdut.root
number_of_tracks=50000

[EventLoaderCLICpix2]
input_directory=tutorial_data/

[EventLoaderTimepix3]
input_directory=tutorial_data/

[Clustering4D]
#what's wrong here?

[ClusteringSpatial]
#what's wrong here?

[TestAlgorithm]
make_correlations=true
timing_cut=2ms

[Tracking4D]
min_hits_on_track=5

[DUTAssociation]
spatial_cut = 200um 200um
timing_cut = 2ms

[AnalysisDUT]
chi2ndof_cut = 20
time_cut_frameEdge = 100ns
```

OTHER ADDITIONS TO THE CONFIG FILE FOR A DUT

- We have 2 sets of clustering for each detector!
- How does a module know which detectors to use?
- Can specify what detectors a module processes by specifying a 'type' of detector to run on…

  …or the name of one or more detectors
- Level of importance: name > type > nothing

config_withdut.cfg

```
[Corryvreckan]
detectors_file=alignment_withdut.geo
histogram_file=histograms_withdut.root
number_of_tracks=50000

[EventLoaderCLICpix2]
input_directory=tutorial_data/

[EventLoaderTimepix3]
input_directory=tutorial_data/

[Clustering4D]
#what's wrong here?

[ClusteringSpatial]
#what's wrong here?

[TestAlgorithm]
make_correlations=true
timing_cut=2ms

```

OTHER ADDITIONS TO THE CONFIG FILE FOR A DUT

- We have 2 sets of clustering for each detector!
- How does a module know which detectors to use?
- Can specify what detectors a module processes by specifying a 'type' of detector to run on…

  …or the name of one or more detectors
- Level of importance: name > type > nothing
- Need to specify "ClusteringSpatial" to work only on detectors of type "CLICpix2", and "Clustering4D" to work only on detectors of type "Timepix3"

```
[EventLoaderTimepix3]
input_directory=tutorial_data/

[Clustering4D]

[ClusteringSpatial]
type="CLICpix2"

[TestAlgorithm]
make_correlations=true
timing_cut=2ms

[Tracking4D]
```

- Note: because of the parameter "exclude_dut" in "Tracking4D" module, the DUT(s) is excluded from tracking by default

- Run Corryvreckan with the new configuration file:

  corry -c path/to/config/config_withdut.conf

- Look at output plots from the new "AnalysisDUT" module

AnalysisDUT

Maintainer: Simon Spannagel (simon.spannagel@cern.ch)
Module Type: *DUT*
Detector Type: *all*
Status: Work in progress

Description

Analysis module for CLICpix2 prototypes. This module is still work in progress, changes to functionality and behaviour are to be expected.

Parameters

- `time_cut_frameedge`: Parameter to discard telescope tracks at the frame edges (start and end of the current CLICpix2 frame). Defaults to `20ns`.
- `spatial_cut`: Spatial cut for associating a track with a DUT cluster, defaults to `50um`.
- `chi2ndof_cut`: Acceptance criterion for telescope tracks, defaults to a value of `3`.

Plots produced

- 2D Map of associated cluster positions
- 2D Map of cluster sizes for associated clusters
- 2D Map of cluster ToT values from associated clusters
- 2D Map of associated hits
- 2D Map of associated hits within the defined region-of-interest
- Distribution of pixel ToT values from associated clusters
- 2D Map of pixel ToT values from associated clusters
- Track residuals in X and Y
- Track residuals for 1-pixel-clusters in X and Y
- Track residuals for 2-pixel-clusters in X and Y
- Distribution of cluster Tot values from associated clusters
- Distribution of sizes from associated clusters
- 2D Map of in-pixel efficiency
- 2D Map of the chip efficiency in local coordinates
- 2D Map of the chip efficiency on global coordinates
- 2D Map of track positions associated to a cluster
- 2D Map of track positions not associated to a cluster

Usage

```
[CLICpix2Analysis]
timeCutFrameEdge = 50ns
```

OUTPUTTING

LOGGING, OUTPUTTING OBJECTS, AND ONLINE DATA MONITORING

LOGGING

- Changes the level of output from a module(s)
- Helps identify problems as early as possible with clear indications of its source
- Can change log output format and level using "log_format" and "log_level" parameters respectively in the config file

```
|11:22:02.910|  (STATUS) ===============| Wall-clock timing (seconds) |================
|11:22:02.910|  (STATUS)  EventLoaderCLICpix2 :  014_CP_PS  --  5.27746s = 0.302226ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_D04  --  0.17145s = 0.009819ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_E03  --  0.11291s = 0.006466ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_G02  --  0.12782s = 0.007320ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_G03  --  0.12887s = 0.007380ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_J05  --  0.11337s = 0.006493ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_L09  --  0.10317s = 0.005908ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_D04  --  0.10084s = 0.005775ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_E03  --  0.13371s = 0.007657ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_G02  --  0.12005s = 0.006875ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_G03  --  0.08286s = 0.004745ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_J05  --  0.10069s = 0.005766ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_L09  --  0.07790s = 0.004461ms/evt
|11:22:02.910|  (STATUS)    ClusteringSpatial :  014_CP_PS  --  0.05172s = 0.002962ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_D04  --  0.37547s = 0.021502ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_E03  --  0.38559s = 0.022082ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_G02  --  0.37868s = 0.021686ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  014_CP_PS  --  0.04311s = 0.002469ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_G03  --  0.47093s = 0.026969ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_J05  --  0.40348s = 0.023106ms/evt
|11:22:02.911|  (STATUS)        TestAlgorithm :  W0013_L09  --  0.39962s = 0.022885ms/evt
|11:22:02.911|  (STATUS)           Tracking4D               --  0.36767s = 0.021055ms/evt
|11:22:02.911|  (STATUS)       DUTAssociation :  014_CP_PS  --  0.03725s = 0.002133ms/evt
|11:22:02.911|  (STATUS)          AnalysisDUT :  014_CP_PS  --  0.02980s = 0.001707ms/evt
|11:22:02.911|  (STATUS) ================================================================
```

```
2019-01-10 17:13 williamm@pclcd23:corryvreckan-tests$ ./analyse_cp2.sh 29786
|17:13:37.300|  (STATUS) Welcome to Corryvreckan v0.9.3
|17:13:37.302|  (STATUS) Loaded 7 detectors
|17:13:37.302| (WARNING) Main ROOT file /afs/cern.ch/work/w/williamm/corryvreckan-tests/output/histograms_29786.
root exists and will be overwritten.
|17:13:37.944|  (STATUS) Loaded 26 module instances
|17:13:37.944|  (STATUS) ================| Initialising modules |=================
|17:13:37.944|  (STATUS) [I:EventLoaderCLICpix2:014_CP_PS] Initialising "EventLoaderCLICpix2:014_CP_PS"
|17:13:37.985|    (INFO) [I:EventLoaderCLICpix2:014_CP_PS] Found matrix file: /eos/experiment/clicdp/grid/ilc/us
er/c/clictel/CERN_TB_November_2018/clicpix2/Run29786/matrix_totcnt_eq.cfg
```

LOGGING

- Useful levels of logging output (all options in manual):

  STATUS: Important information about the status of the reconstruction

  WARNING: indicates abnormal results, but the reconstruction can continue

  INFO: summary messages about the progress of reconstruction process

  DEBUG: in-depth details of the reconstruction, such as information on each pixel and cluster

- Try changing the "log_level" for one of your modules in "config_withdut.conf" to "DEBUG" and see the difference in terminal output!

```
|11:22:02.910|  (STATUS) ===============| Wall-clock timing (seconds) |================
|11:22:02.910|  (STATUS)  EventLoaderCLICpix2 :  014_CP_PS  --  5.27746s = 0.302226ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_D04  --  0.17145s = 0.009819ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_E03  --  0.11291s = 0.006466ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_G02  --  0.12782s = 0.007320ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_G03  --  0.12887s = 0.007380ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_J05  --  0.11337s = 0.006493ms/evt
|11:22:02.910|  (STATUS)  EventLoaderTimepix3 :  W0013_L09  --  0.10317s = 0.005908ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_D04  --  0.10084s = 0.005775ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_E03  --  0.13371s = 0.007657ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_G02  --  0.12005s = 0.006875ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_G03  --  0.08286s = 0.004745ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_J05  --  0.10069s = 0.005766ms/evt
|11:22:02.910|  (STATUS)         Clustering4D :  W0013_L09  --  0.07790s = 0.004461ms/evt
|11:22:02.910|  (STATUS)    ClusteringSpatial :  014_CP_PS  --  0.05172s = 0.002962ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_D04  --  0.37547s = 0.021502ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_E03  --  0.38559s = 0.022082ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_G02  --  0.37868s = 0.021686ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  014_CP_PS  --  0.04311s = 0.002469ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_G03  --  0.47093s = 0.026969ms/evt
|11:22:02.910|  (STATUS)        TestAlgorithm :  W0013_J05  --  0.40348s = 0.023106ms/evt
|11:22:02.911|  (STATUS)        TestAlgorithm :  W0013_L09  --  0.39962s = 0.022885ms/evt
|11:22:02.911|  (STATUS)           Tracking4D               --  0.36767s = 0.021055ms/evt
|11:22:02.911|  (STATUS)       DUTAssociation :  014_CP_PS  --  0.03725s = 0.002133ms/evt
|11:22:02.911|  (STATUS)          AnalysisDUT :  014_CP_PS  --  0.02980s = 0.001707ms/evt
|11:22:02.911|  (STATUS) ================================================================
```

```
2019-01-10 17:13 williamm@pclcd23:corryvreckan-tests$ ./analyse_cp2.sh 29786
|17:13:37.300|  (STATUS) Welcome to Corryvreckan v0.9.3
|17:13:37.302|  (STATUS) Loaded 7 detectors
|17:13:37.302| (WARNING) Main ROOT file /afs/cern.ch/work/w/williamm/corryvreckan-tests/output/histograms_29786.
root exists and will be overwritten.
|17:13:37.944|  (STATUS) Loaded 26 module instances
|17:13:37.944|  (STATUS) ================| Initialising modules |=================
|17:13:37.944|  (STATUS) [I:EventLoaderCLICpix2:014_CP_PS] Initialising "EventLoaderCLICpix2:014_CP_PS"
|17:13:37.985|    (INFO) [I:EventLoaderCLICpix2:014_CP_PS] Found matrix file: /eos/experiment/clicdp/grid/ilc/us
er/c/clictel/CERN_TB_November_2018/clicpix2/Run29786/matrix_totcnt_eq.cfg
```

OUTPUTTING OBJECTS

- Corryvreckan objects: pixels, clusters, tracks.
- Objects can be outputted at any point during reconstruction using the "FileWriter" module.
- Module writes an output file and fills it was trees containing the wanted objects.
- Files like these can be read into Corryvreckan using "FileReader" module.
- Done for you in "config_outputs.conf"

FileWriter

Maintainer: Daniel Hynds (daniel.hynds@cern.ch), Simon Spannagel (simon.spannagel@cern.ch)
Module Type: *GLOBAL*
Status: Functional

Description

This module writes an output file and fills it with trees containing the requested data objects. `Pixel`, `cluster`, and/or `track` objects can be written into the trees.

Parameters

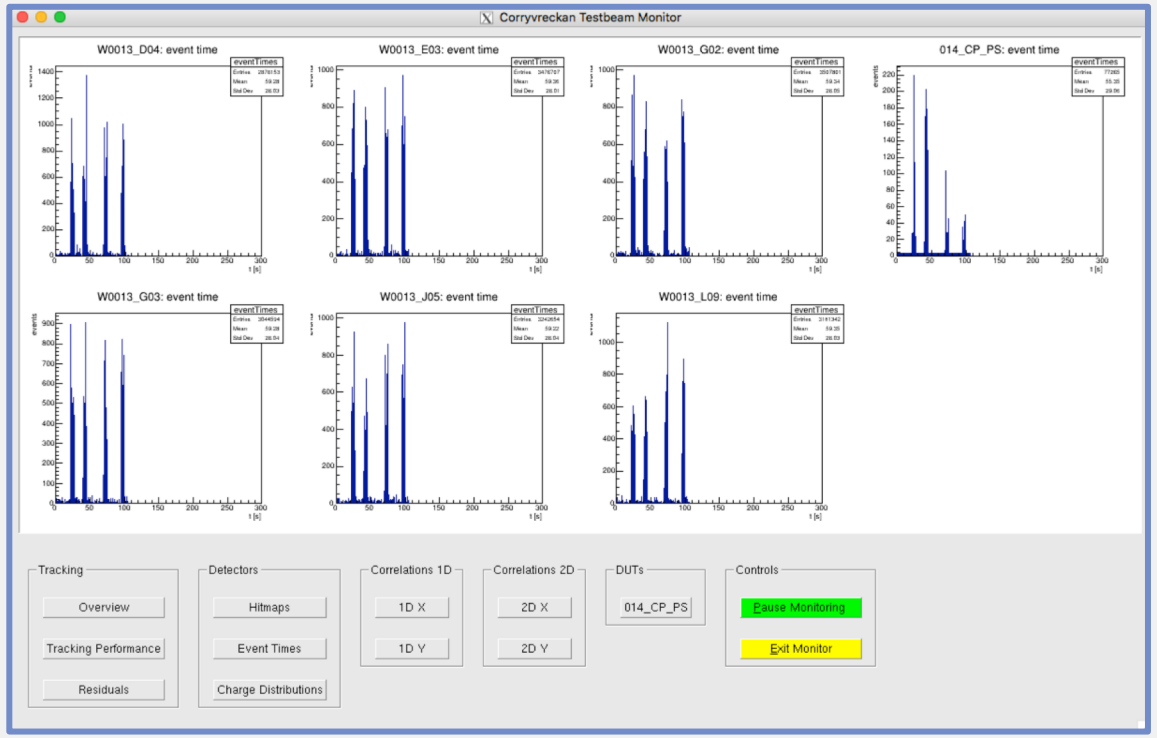
- `only_dut`: Boolean to decide if only the DUT data is to be written into the outputfile, or if all planes are to be. Default value is `true`.
- `write_pixels`: Boolean to choose if pixel objects are to be written out. Default value is `true`.
- `write_clusters`: Boolean to choose if cluster objects are to be written out. Default value is `false`.
- `write_tracks`: Boolean to choose if track objects are to be written out. Default value is `true`.
- `file_name`: Name of the data file to create, relative to the output directory of the framework. The file extension `.root` will be appended if not present. Default value is `outputTuples.root`.

Usage

```
[FileWriter]
only_dut = false
write_pixels = true
write_clusters = true
write_tracks = true
file_name = "output.root"
```

ONLINE MONITORING

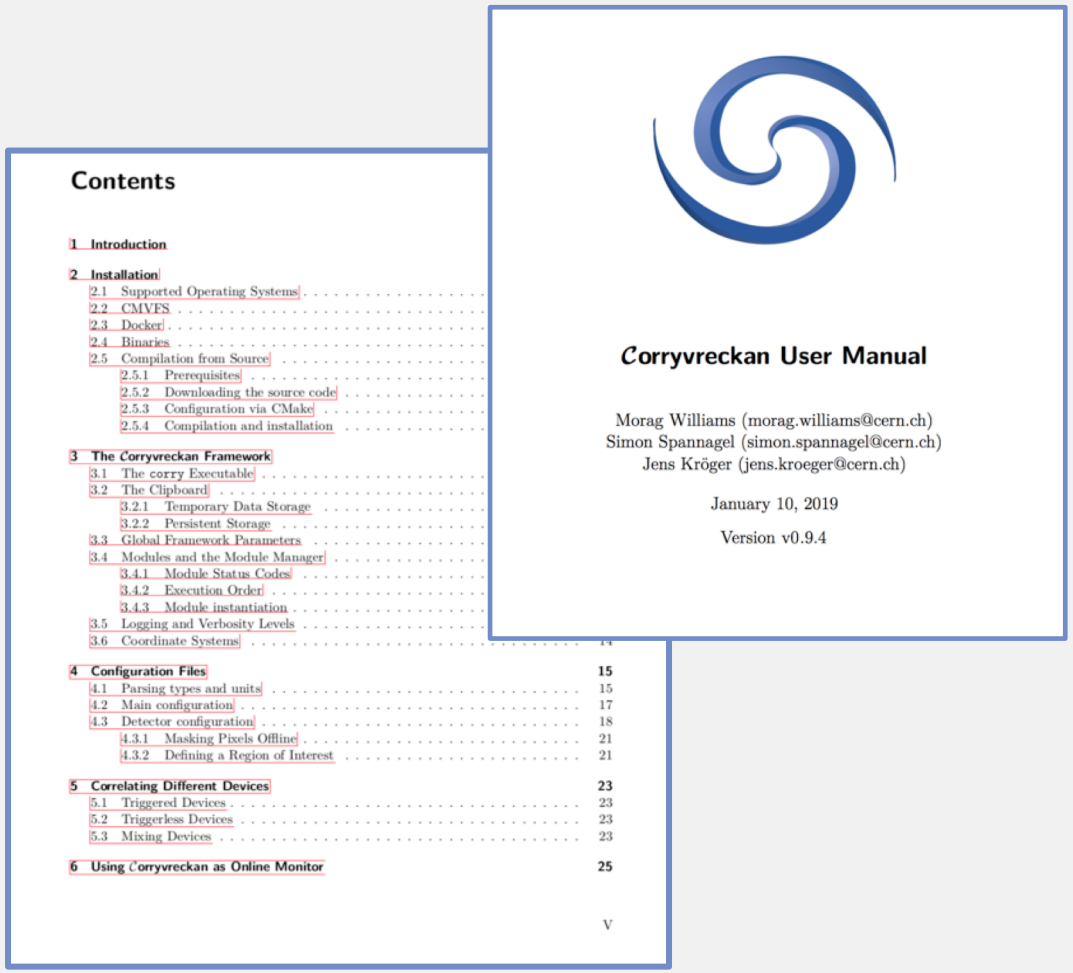
- Used to check data quality while reconstructing
- Particularly useful in test-beam environments, as you can quickly check the data quality almost immediately
- To use, add the "OnlineMonitoring" module to your configuration in any position
- Already done for you in configuration file "config_outputs.conf"
- Lets run this and see the monitoring in action!
- Also can see output of clusters in "outputTuples_tutorial.root"



AND FINALLY…

MANUAL AND CONTRIBUTION GUIDE

CORRYVRECKAN MANUAL



Contents

Corryvreckan User Manual

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

January 10, 2019

Version v0.9.4

- User manual:
  - Updated with new Corryvreckan changes
  - Autogenerated during building
  - Installation instructions, FAQ, 'getting started'
  - Full descriptions of module functionality
  - Usage examples
- Note module descriptions can also be found as readme text files in the module's src directory
- Can be downloaded here.

CONTRIBUTING TO CORRYVRECKAN

- Everyone is invited to contribute!
- Open to additional functionality and updates to existing features
- Discuss with us before starting work
    - Maybe someone is working on your feature already
    - We can help you integrate it into the existing code
- Please be open to new software tools → they'll help you in the long run!
- We are very strict with respect to code quality
    - Don't be discouraged by suggestions for change

BEST PRACTICES

- Check your code before committing
  - run make format before committing
  - remember to update readme files with your changes
- Make a separate repository for your configurations
  - software repository is for software code only
  - do not commit data files or configuration files
- Commit in a smart way
  - make small commits often
  - have descriptive commit messages
  - have development branches separate from master

RESOURCES

- Repository:

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

  Contains the source code, issue tracker, user manual.

- *User manual:*

  Can be downloaded from the gitlab page, or autogenerated during corry building

  Includes installation instructions, 'getting started' guide, FAQS, and full descriptions of the framework and all modules.

- *Email for questions:*

  corryvreckan.info@cern.ch

THE END

- Thank you for coming!
- Hope you had an enjoyable time learning and using Corryvreckan
- Thanks to all Corryvreckan authors, and CLICdp

- All and any questions are welcome

BACK-UP

KEYBOARD SHORTCUTS

- Signals can be send using keyboard shortcuts to terminate the run, either gracefully or with force.

CTRL+C (SIGINT): Request a graceful shutdown of the reconstruction. This means the currently processed event is finished, while all other events requested in the configuration file are ignored. After finishing the event, the finalization stage is executed for every module to ensure all modules finish properly.

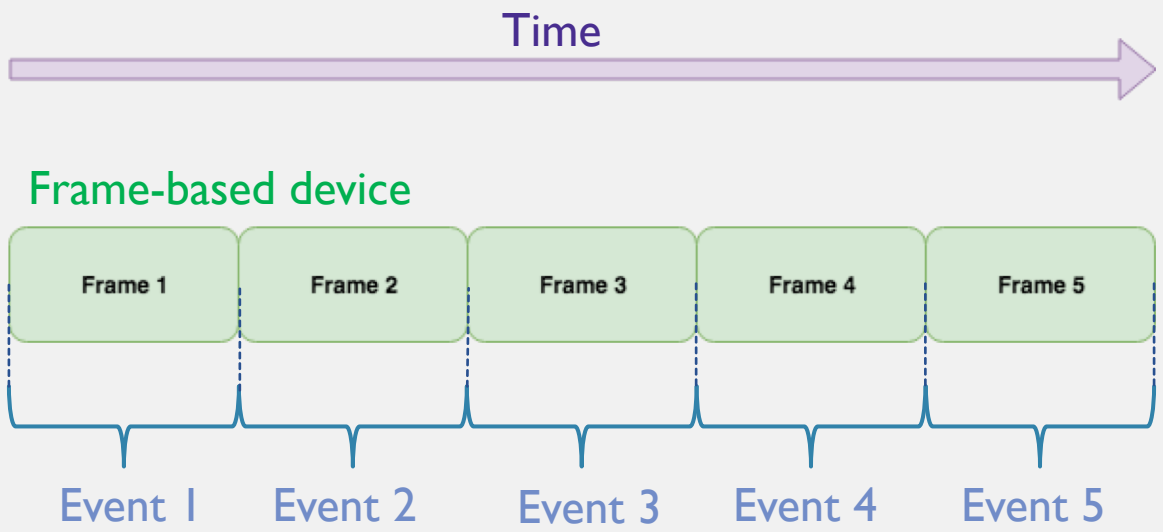
CTRL+ (SIGQUIT): Forcefully terminates the framework. It is not recommended to use this signal as it will normally lead to the loss of all generated data. This signal should only be used when graceful termination is for any reason not possible.

JOBSUB

- jobsub is a tool for the convenient run-specific modification of Corryvreckan configuration files and their execution through the corry executable. It is derived from the original jobsub written for EUTelescope by Hanno Perrey, Lund University.
- Configuration file templates are valid Corryvreckan configuration files in TOML format, where single values are replaced by variables in the form: parameter = @parametervalue@
- Variables in the configuration file template are replaced with values at run time, from command line or comma-separated list text file. From command line: *bash jobsub.py –option parametervalue=2 -c alignment.conf 1234*

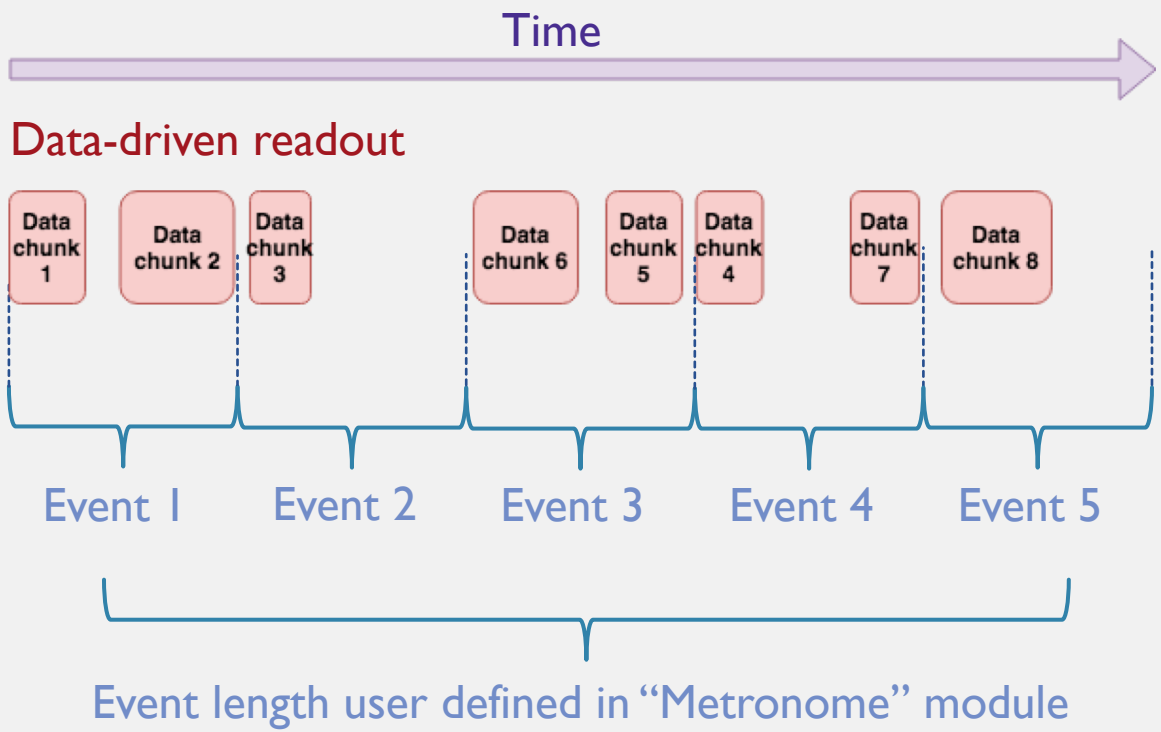
DATA-DRIVEN AND FRAME-BASED READOUT DEVICES

- Can easily use combinations of devices with frame-based and/or data-driven readouts in the same reconstruction
- Need arbitrary chunks of data to process (‘events’)
- Frame-based: Corryvreckan uses the start and end points of the frame to define an event



DATA-DRIVEN AND FRAME-BASED READOUT DEVICES

- Can easily use combinations of devices with frame-based and/or data-driven readouts in the same reconstruction
- Need arbitrary chunks of data to process ('events')
- Frame-based: Corryvreckan uses the start and end points of the frame to define an event
- Data-driven: either use the "Metronome" module to define the length of an event…



DATA-DRIVEN AND FRAME-BASED READOUT DEVICES

- Can easily use combinations of devices with frame-based and/or data-driven readouts in the same reconstruction
- Need arbitrary chunks of data to process ('events')
- Frame-based: Corryvreckan uses the start and end points of the frame to define an event
- Data-driven: either use the "Metronome" module to define the length of an event…or use the definition of an event from a frame-based readout device