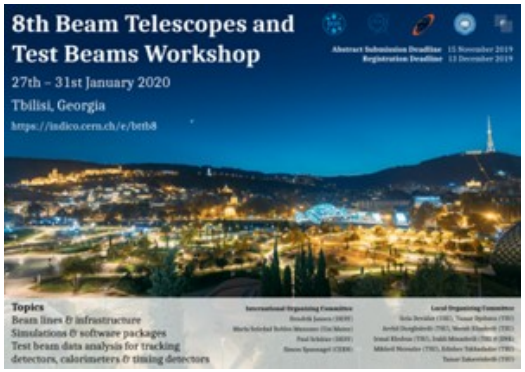
Hands-On:
Track reconstruction of testbeam data with EUTelescope.

Jan-Hendrik Arling

for the EUTelescope Developers



8th Beam Telescopes and Test Beams Workshop

What is EUTelescope?

- EUTelescope is a framework for reconstructing and analysing data taken with pixel beam telescopes.
- Implemented as part of the EUDET-project:
  1. Hardware → Mimosa26 beam telescopes.
  2. DAQ software → EUDAQ.
  3. Reconstruction software → EUTelescope.
- Integrated within the ILCsoft (ILC software) framework.

Outline.

- **Introduction to EUTelescope**

  What is the idea of EUTelescope? What is the base software structure? Who is developing?

- **Reconstruction workflow**

  Which are the parts of an EUTelescope analysis? How is the reconstruction chain?

- **Hands-On: The installation**

  What are the different ways for getting EUTelescope?

- **Hands-On: The GBL noDUT example**

  Reconstructing data from only telescope data using GBL.

- **Hands-On: The GBL SUT example**

  Measuring scattering angles for a passive DUT using GBL.

- **Hands-On: The GBL DUT example**

  Analysing data from an active DUT using GBL.

Caveats.

- This tutorial will show you the basic application of EUTelescope on three different examples with a telescope-only, a scattering-only and a DUT example.
- We will focus here only on a subset of maintained processors. As EUTelescope has grown by user contributions, there are many more processors which can be used.
- Unfortunately, there is no nice documentation, but the current status is summarized in a paper soon.
- We try to show you as best as possible how you can use EUTelescope for your own use. If we cannot answer your questions directly, please forgive us :)
- In case of any further question: please use the issue tracker on GitHub:

  https://github.com/eutelescope/eutelescope/issues

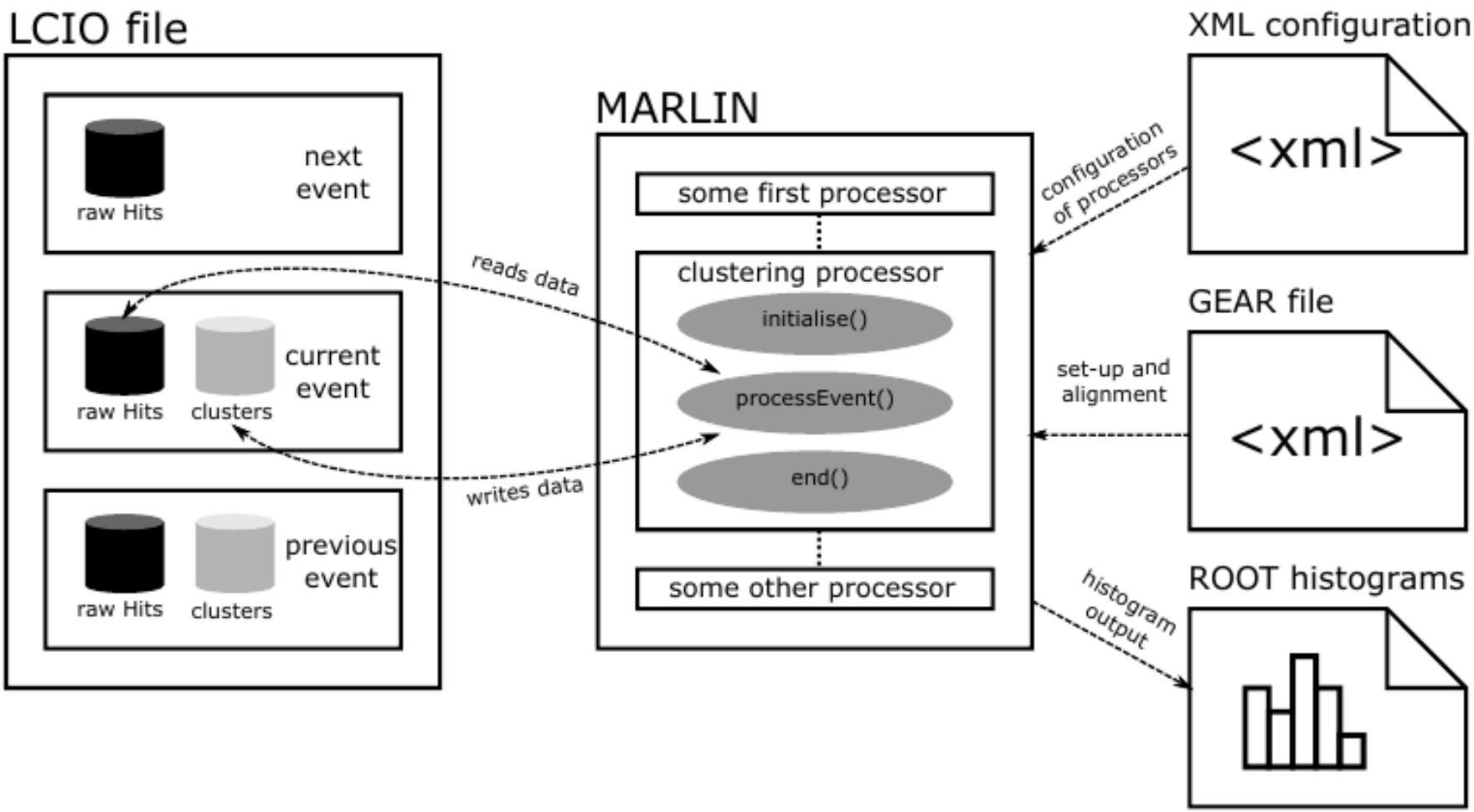
Introduction to EUTelescope.

EUTelescope overview.

EUTelescope

- has the main goal to get from raw data to high level objects like tracks crossing through the beam telescope.
- is embedded in the ILCsoft (ILC software) framework.
- is built in the Marlin (Modular Analysis & Reconstruction for the linear collider) software package.
- provides processors for different reconstruction steps, which can be daisy-chained (modular structure).
- uses the LCIO (Linear Collider I/O) data model.
- uses GEAR (Geometry API for Reconstruction) as markup language describing the telescope geometry.
- is a user-driven software project, implementation of new processors for (new) applications/devices by the user groups.

ILCsoft integration.



EUTelescope reconstruction workflow.

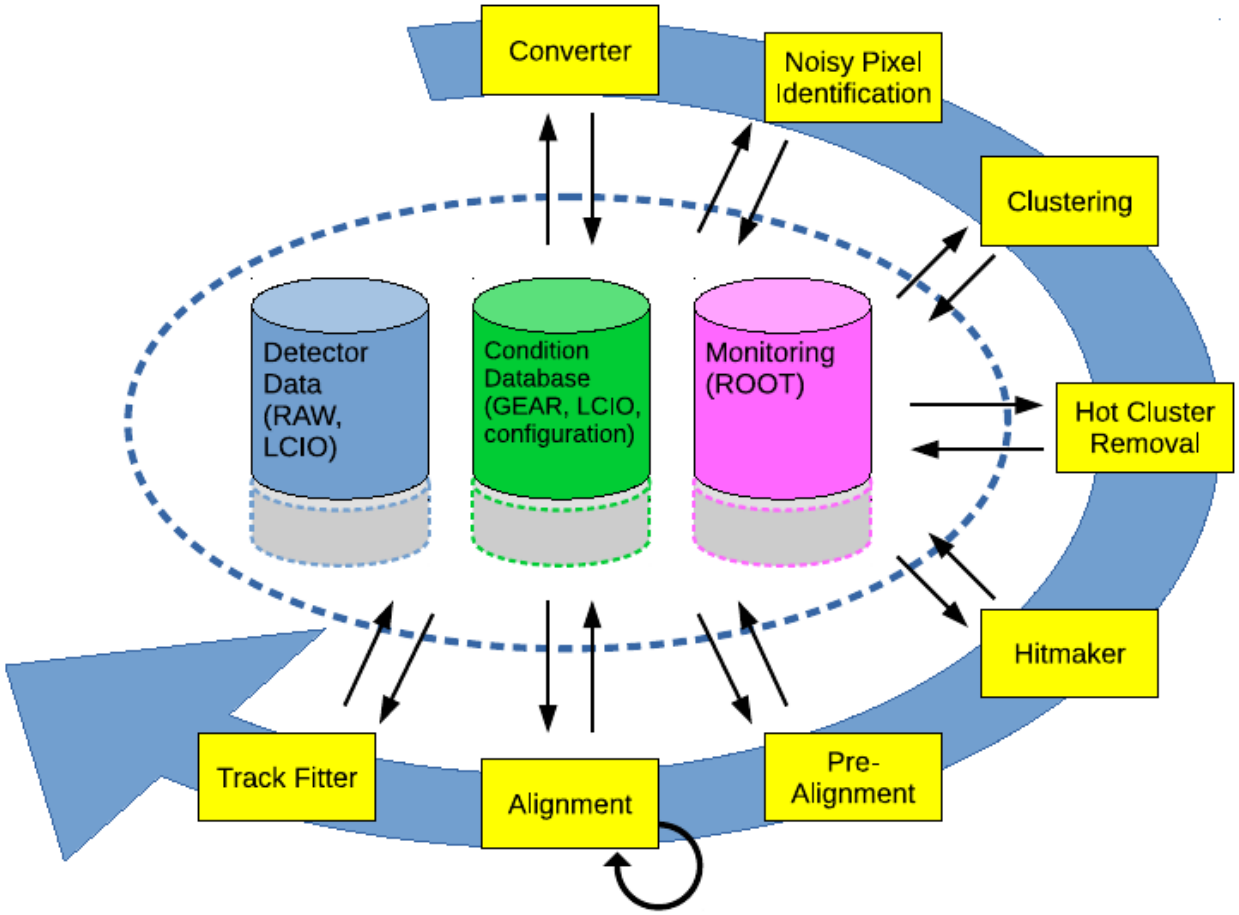
Analysis components.

- The analysis is controlled by a config file (*.cfg), a runlist csv table (*.csv), and steering file templates (*.xml).
  - The *config file* specifies the configuration for the individual processors, i.e. setting processor parameters.
  - The *csv table* contains a list of runs and associated parameters specific to that run, e.g. beam energy.
  - The *steering file* defines the reconstruction step by controlling which processors/options are executed by Marlin.
- The telescope geometry is provided by a *GEAR file* (*.xml) describing the telescope planes as well as the DUTs.
- Execution of reconstruction step by *jobsub* command.

```
Ejobsub --config config.cfg --csv runlist.csv [options] PROCESSOR RUNNR
```

Typical reconstruction workflow.

- From raw data to telescope tracks...



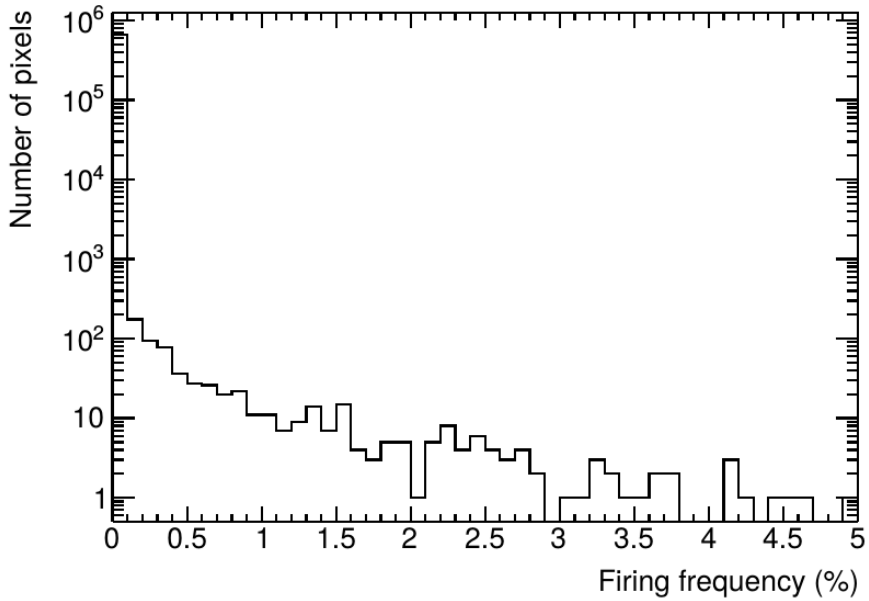
Step: CONVERTER.

- Convert data from EUDAQ raw-format to LCIO data format.
- Only needed for EUDAQ1 testbeam data.
- Use *EUTelNativeReader* as interface to EUDAQ.
- Creates LCIO file with zero-suppressed data of telescope planes.

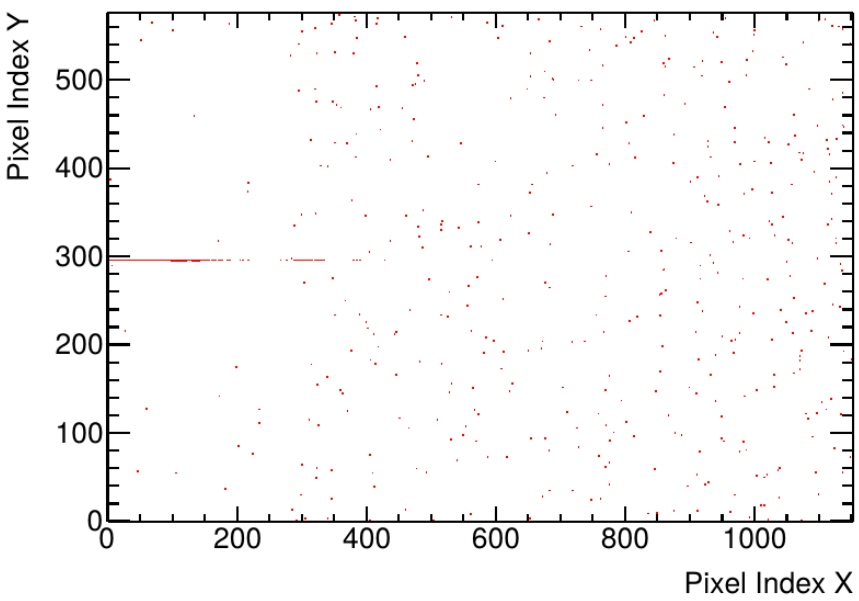
- For EUDAQ2, external converter inside EUDAQ2 package (called *euCliConverter*). Preprocessing from raw to LCIO data.

Step: NOISYPIXEL.

- Identification of noisy pixel by applying cut on firing frequency of individual pixel in the planes by the *EUTelNoisyPixelFinder* processor.
- Only masking of identified pixels by storing in a database file.



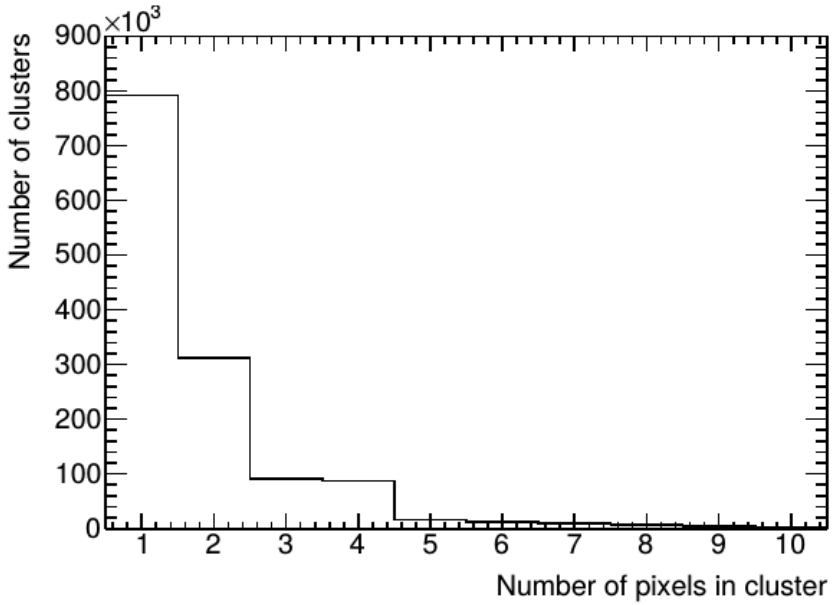
Occupancy of noisy pixels vs firing frequency.



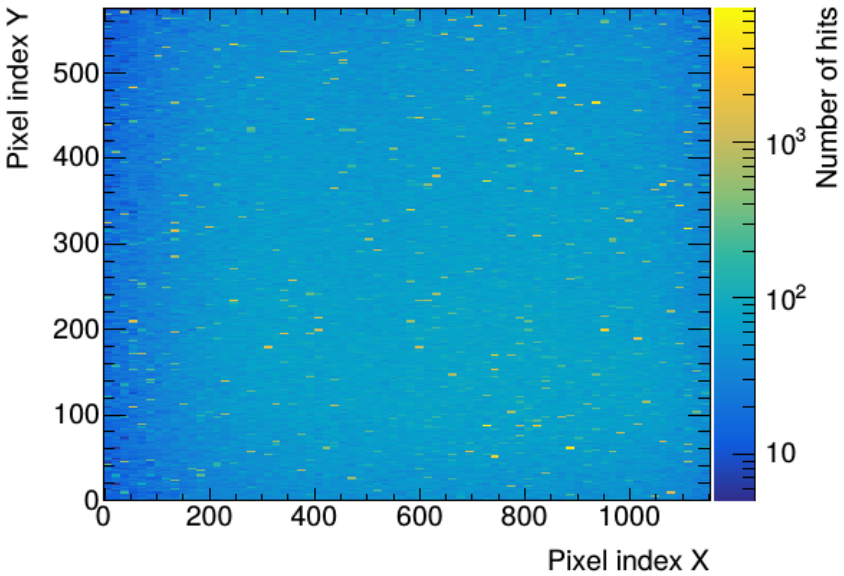
Occupancy map of noisy pixels.

Step: CLUSTERING.

- Applying sparse clustering to find clusters out of adjacent hits by the *EUTelSparseClustering* processor.
- Use the noisy pixel database to mask (*EUTelNoisyClusterMasker*) and remove noisy clusters from the data (*EUTelNoisyClusterRemover*).
- Save noise-free clusters as LCIO output file.



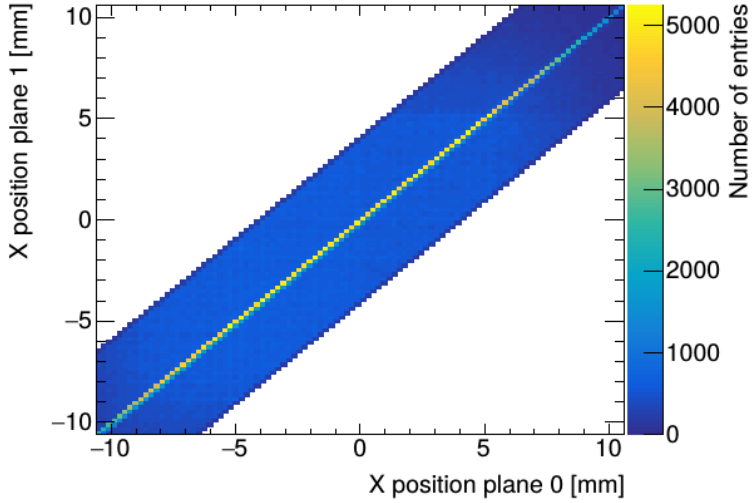
Cluster size distribution for one MIMOSA26 plane.

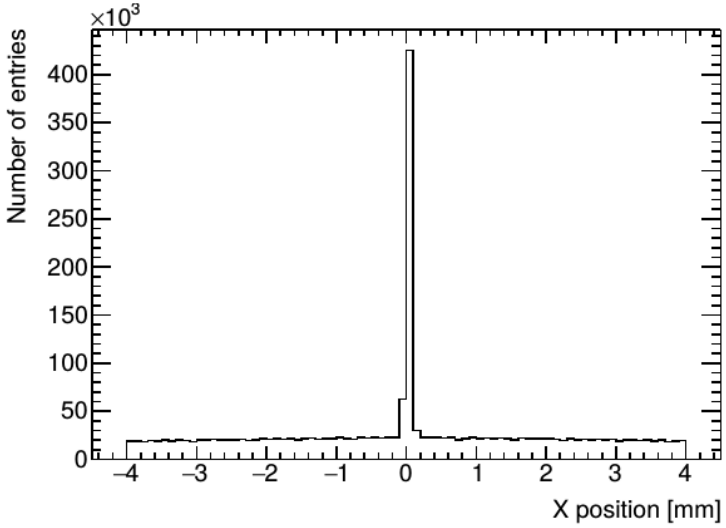


Cluster hit map for one MIMOSA26 plane.

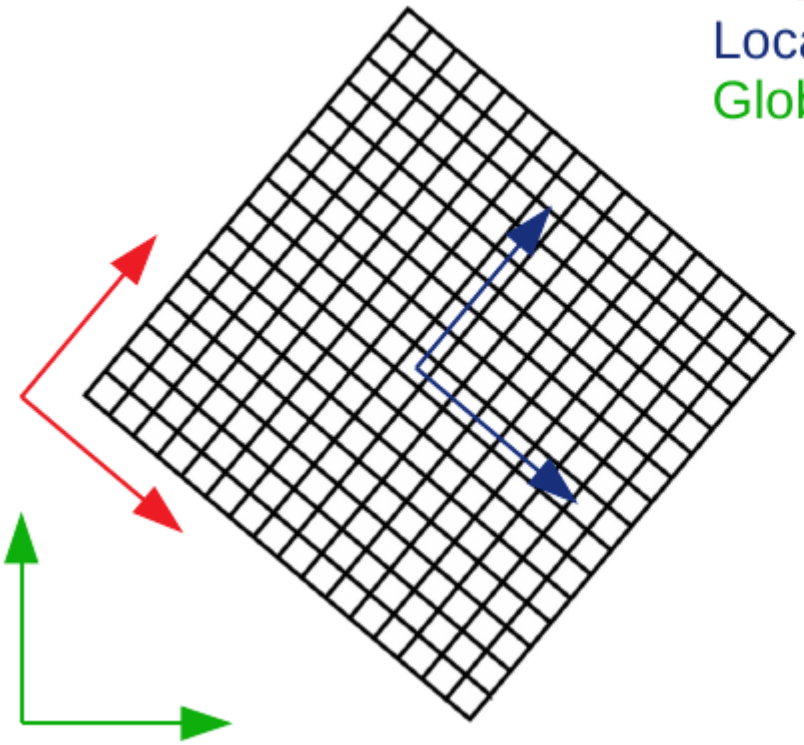
Step: HITMAKER.

- Creation of hits in local coordinate system out of clusters by the *EUTelHitMaker* processor.
- Transforming coordinates in global coordinate system using GEAR file by the *EUTelHitCoordinateTransformer* processor.
- Evaluate pre-alignment by straight line fit with one fixed plane and creating pre-aligned geometry as new GEAR file (*EUTelPreAligner*).
- Check correlation of planes by the *EUTelCorrelator* processor.
- Save collection of hits in the local coordinate system as LCIO output file.





Intermezzo: Coordinate systems.

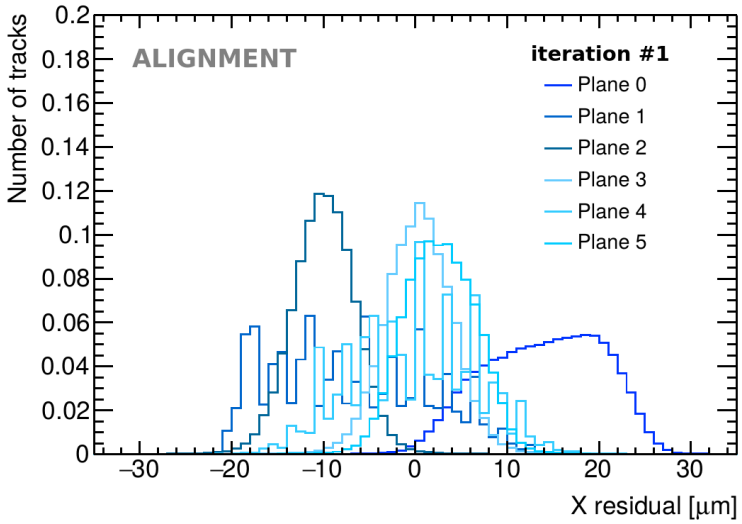


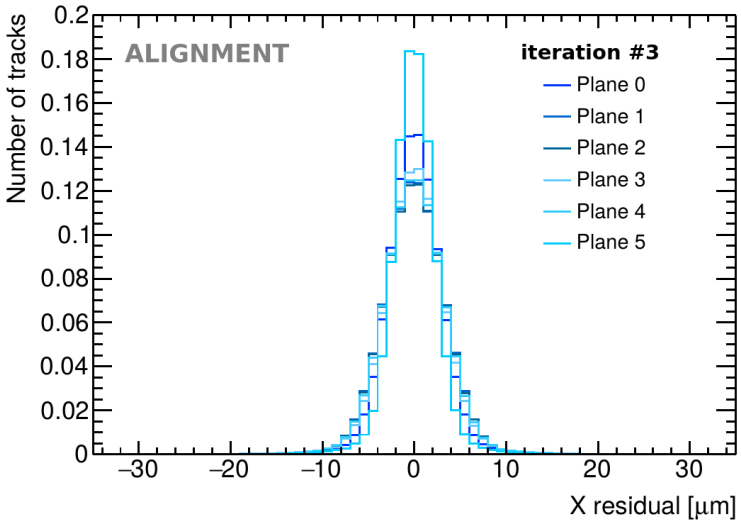
Pixel coordinate system [pixel numbers]
Local coordinate system [mm]
Global (telescope) coordinate system [mm]

Step: ALIGNGBL.

- Transform local hits to global coordinate system by loading pre-aligned GEAR file (*EUTelHitCoordinateTransformer*).
- Setting up MillePedeII for creating new GEAR file (*EUTelPedeGEAR*).
- Use GBL to perform alignment of planes by *EUTelGBL* processor:

  track finding of by upstream/downstream triplets and best match them; specify alignment mode for GBL and Millepede alignment.
- Write out new GEAR file with alignment parameters, better alignment by iterative application:





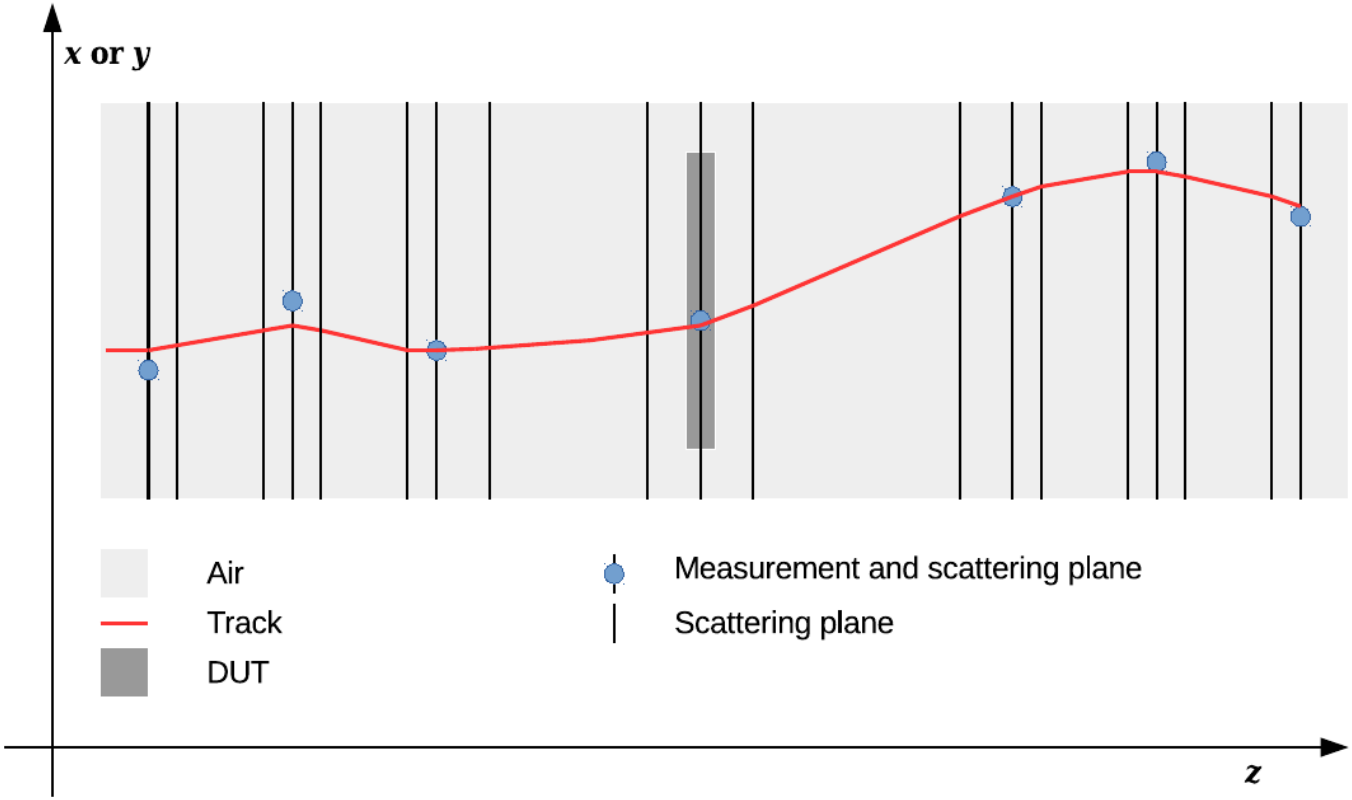
Intermezzo: Track finding.

- Track finding is performed by the Triplet method, constructing an upstream and downstream triplet and matching them to a track in the middle of the telescope.
- Track finding parameters:
  - *SlopeCut*: cut on allowed slope between straight line of first and last plane of triplet for up- and downstream arm.
  - *TripletResidualCut*: cut on the allowed residual between interpolation of hit on middle plane and the recorded hits on plane for up- and downstream arm.
  - *TripletMatchingCut*: cut on the matching radius for triplet matching at center of telescope.
- Option *suggestAlignmentCuts* provides suggestion for next iteration:

```
[ MESSAGE8 "GBLAlignment"] UpstreamTripletCut   = 0.0213309
[ MESSAGE8 "GBLAlignment"] DownstreamTripletCut = 0.022852
[ MESSAGE8 "GBLAlignment"] UpstreamSlopeCut     = 3.5678
[ MESSAGE8 "GBLAlignment"] DownstreamSlopeCut   = 3.46236
[ MESSAGE8 "GBLAlignment"] TripletMatchingCut   = 0.0511312
[ MESSAGE8 "GBLAlignment"] The DUT is propably not an active device - no hits detected
[ MESSAGE5 "GBLAlignment"] Found 532399 tracks in 99999 events
```

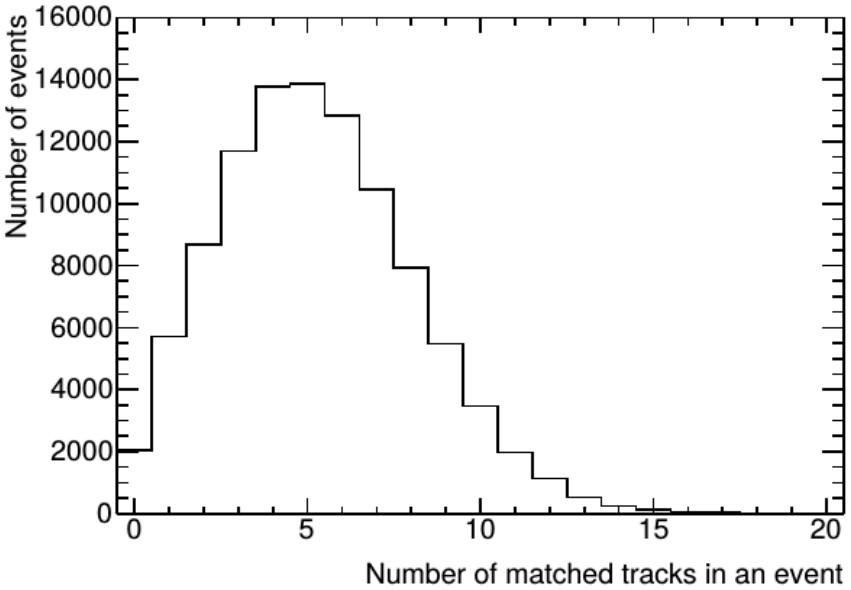
Intermezzo: General Broken Lines.

- GBL provides a re-fit of track candidates by applying a dedicated track model, taking the effect of multiple Coulomb scattering into account.
- Define GBL trajectory by adding consecutively GBL points, either providing a scattering only or also a measurement with its uncertainty.

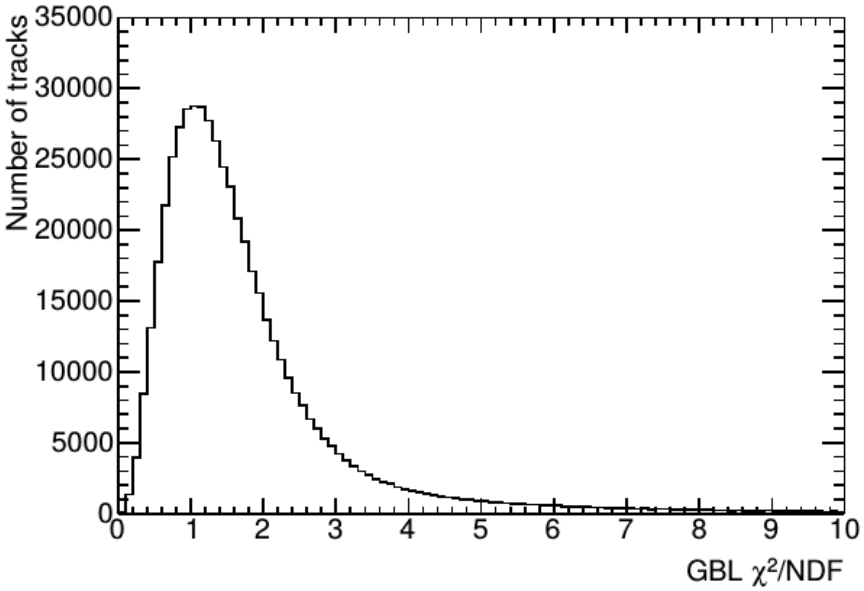


Step: FITGBL.

- Transform local hits to global coordinate system by loading aligned GEAR file (*EUTelHitCoordinateTransformer*).
- Fit telescope tracks by using GBL algorithm with *EUTelGBL* processor.
- Write out tracks and other variables to ROOT ntuple (*EUTelGBLOutput*).



Number of matched tracks per event.



Goodness-of-fit for GBL track fitting.

Hands-On:
The installation.

Installation.

- The installation of EUTelescope is performed with the ilCinstaller package (Github).
- The complete instructions are documented in the README file, with the options to install it on to your local machine or the batch cluster systems DESY NAF or CERN LXPLUS.
- The following options exist currently:
  - **Local standalone** :
    Complete installation of all required packages including ROOT. (~1,5h)
  - **Local nocmakenoroot** :
    Use local versions of cmake and ROOT. (~0,5h)
  - **DESYNAF standalone** :
    Complete installation of all required packages including ROOT. (~1h)
  - **DESYNAF cvmfs** :
    Use version of ROOT loaded from CVMFS. (~15min)
  - **LXPLUS cvmfs** :
    Use version of ROOT loaded from CVMFS. (~15min?)

Installation: *Local standalone*.

(1) Prerequisites: packet installation

Install all packages required for the compilation of the ILCsoft packages and of EUTelescope (dependent on distribution).

(2) Prepare installer:

cd /PATH/WHERE/TO/INSTALL
git clone -b master https://github.com/eutelescope/ilcinstall

(3) Running installation:

cd ilcinstall
./ilcsoft-install -i releases/release-local-standalone.cfg

(4) Source environment:

cd $ILCSOFT/eutel_v02-00-00/Eutelescope/master
source build_env.sh

(5) Remark:

For running the examples, you need AFS for accessing the data files or you can download them by hand (see instructions).

Installation: *Local nocmakenoroot*.

(1) Prerequisites: packet installation

Install all packages required for the compilation of the ILCsoft packages and of EUTelescope (dependent on distribution).

You need to have sourced your CMake and ROOT installation.

(2) Prepare installer:

cd /PATH/WHERE/TO/INSTALL
git clone -b master https://github.com/eutelescope/ilcinstall

(3) Running installation:

cd ilcinstall
./ilcsoft-install -i releases/release-local-nocmakenoroot.cfg

(4) Source environment:

cd $ILCSOFT/eutel_v02-00-00/Eutelescope/master
source build_env.sh

(5) Remark:

For running the examples, you need AFS for accessing the data files or you can download them by hand (see instructions).

Installation: DESYNAF *standalone*.

(1) Prerequisites: enable compilers, git and bash

scl enable devtoolset-4 rh-git29 bash

(2) Prepare installer:

cd /nfs/dust/GROUP/user/USER/NAME
git clone -b master https://github.com/eutelescope/ilcinstall

(3) Running installation:

cd ilcinstall
./ilcsoft-install -i releases/release-desynaf-standalone.cfg

(4) Source environment:

cd $ILCSOFT/eutel_v02-00-00/Eutelescope/master
source build_env.sh

Installation: *DESYNAF CVMFS*.

(1) Prerequisites: enable compilers, git and bash

scl enable devtoolset-4 rh-git29 bash

(2) Source ROOT-environment from CVMFS (check with *uname -a*):

i. *SLC6 machine*

source /cvmfs/sft.cern.ch/lcg/releases/LCG_94/ROOT/6.14.04/x86_64-slc6-gcc62-opt/ROOT-env.sh

ii. *Centos7 machine*

source /cvmfs/sft.cern.ch/lcg/releases/LCG_94/ROOT/6.14.04/x86_64-centos7-gcc62-opt/ROOT-env.sh

(3) Prepare installer:

cd /nfs/dust/GROUP/user/USER/NAME

git clone -b master https://github.com/eutelescope/ilcinstall

(4) Running installation:

cd ilcinstall

./ilcsoft-install -i releases/release-desynaf-cvmfs.cfg

(5) Source environment as before.

Installation: *LXPLUS CVMFS*.

(1) Prerequisites: enable compilers, git and bash

scl enable rh-git29 bash

(2) Source ROOT-environment from CVMFS (check with *uname -a*):

i. *SLC6 machine*

source /cvmfs/sft.cern.ch/lcg/releases/LCG_94/ROOT/6.14.04/x86_64-slc6-gcc62-opt/ROOT-env.sh

ii. *Centos7 machine*

source /cvmfs/sft.cern.ch/lcg/releases/LCG_94/ROOT/6.14.04/x86_64-centos7-gcc62-opt/ROOT-env.sh

(3) Prepare installer:

cd /eos/user/USERINTIAL/USERNAME

git clone -b master https://github.com/eutelescope/ilcinstall

(4) Running installation:

cd ilcinstall

./ilcsoft-install -i releases/release-lxplus-cvmfs.cfg

(5) Source environment as before.

Installation: *LXPLUS CVMFS*.

(1) Prerequisites: enable compilers, git and bash

scl enable rh-git29 bash

(2) Source ROOT-environment from CVMFS (check with *uname -a*):

i. SLC6 machine

64-

64-

(3) F

(4) F

cd ilcinstall

./ilcsoft-install -i releases/release-lxplus-cvmfs.cfg

(5) Source environment as before.

CAVEAT for CVMFS versions:

When starting from a new terminal,
you also have to source your environment again,
so for a fresh start do the following:

scl enable rh-git29 bash
source /cvmfs/sft.cern.ch/lcg/releases/LCG_94/ROOT/6.14.04/x86_64-centos7-gcc62-opt/ROOT-env.sh
cd /EUTELESCOPE/INSTALL/PATH
source build_env.sh

Usage: AFS.

- If you have AFS access (e.g. on NAF or LXPLUS), you can source my EUTelescope version for the tutorial.

AFS (NAF)

- Copy the example folder from the EUTelescope installation:

cp -r /afs/desy.de/user/a/arling/public/ilcsoft/eutel_v02-00-00/Eutelescope/master/jobsub/examples/EXAMPLE /PATH/YOU/WANT/

- Finally, you can source the EUTelescope environment:

source /afs/desy.de/user/a/arling/public/ilcsoft/eutel_v02-00-00/Eutelescope/master/build_env.sh

AFS (LXPLUS)

cp -r /afs/cern.ch/user/j/jaarling/public/BTTB8/eutel_v02-00-00/Eutelescope/master/jobsub/examples/EXAMPLE /PATH/YOU/WANT/

source /afs/desy.de/user/a/arling/public/ilcsoft/eutel_v02-00-00/Eutelescope/master/build_env.sh

source /afs/desy.de/user/a/arling/public/ilcsoft/eutel_v02-00-00/init_ilcsoft.sh

Usage: AFS.

- If you have AFS access (e.g. on NAF or LXPLUS), you can source my EUTelescope version for the tutorial.

AFS (NAF)

- Co

- Fin

AFS

c
0

source /afs/desy.de/user/a/arling/public/ilcsoft/eutel_v02-00-00/Eutelescope/master/build_env.sh

source /afs/desy.de/user/a/arling/public/ilcsoft/eutel_v02-00-00/init_ilcsoft.sh

CAVEAT:

As you have changed now the location where you want to execute the example, you have to tell EUTelescope where the reconstruction should be executed.

This is done by changing the *BasePath* in the configuration file of the example.

Usage: Docker.

Docker (interactive locally)

docker run -it
--mount type=bind,source=ABSOLUTE_OUTPUTPATH,target=/output
eutelescope/eutel-ubuntu18:latest

- Start an interactive session with the above command.
- A binding to the outside as absolute path is created as well.

Docker (Singularity e.g. LXPLUS)

- Login to LXPLUS and create storage space for image download:

  mkdir /afs/cern.ch/work/<u>/<user>/.singularity
  ln -s /afs/cern.ch/work/<u>/<user>/.singularity

- Pull the corresponding image:

  singularity pull docker://eutelescope/eutel-ubuntu18:latest

- Run and bind the Docker image:

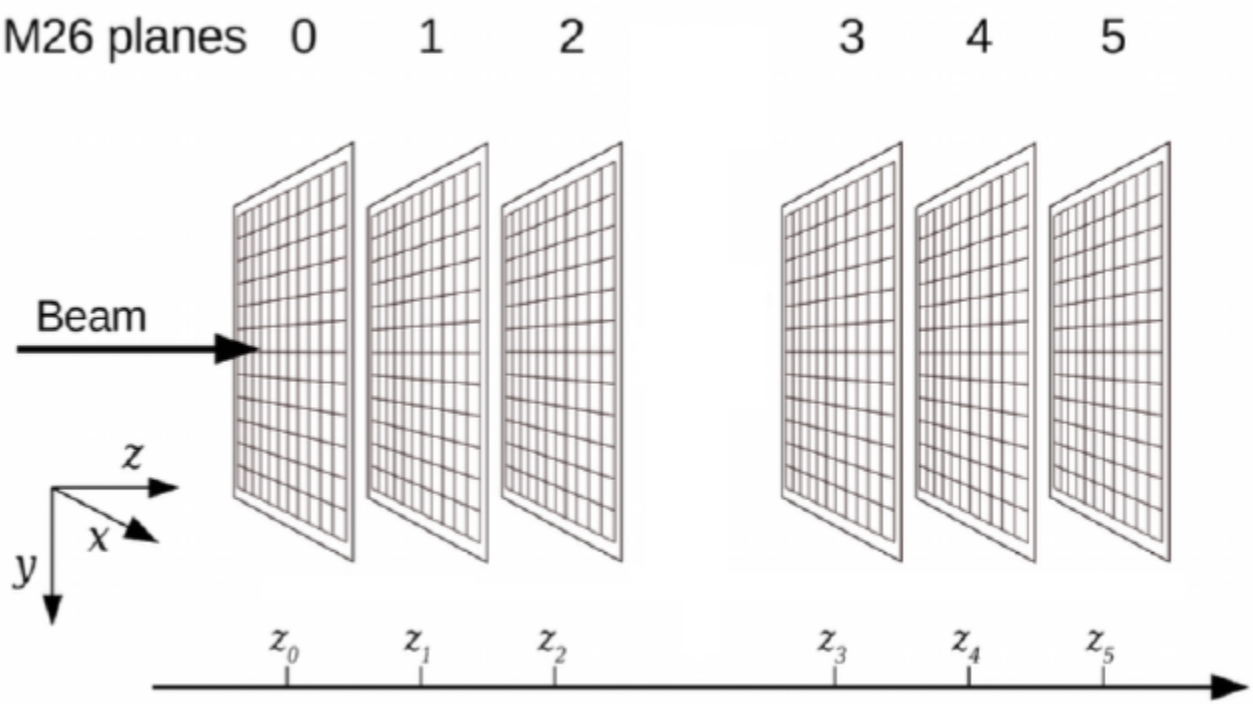
  singularity shell docker://eutelescope/eutel-ubuntu18:latest

- Remark: AFS is already bound in, otherwise non writable folders. Therefore copy examples to an AFS location and run EUTelescope there. Before: source it!

Hands-On:
The GBL noDUT example.

Empty telescope.

- Telescope consisting of six Mimosa26 planes, no DUT inserted.
- Data from DURANTA telescope at 4 GeV/c recorded with EUDAQ2.
- Use General Broken Line (GBL) algorithm for alignment and fitting.



Setting up analysis.

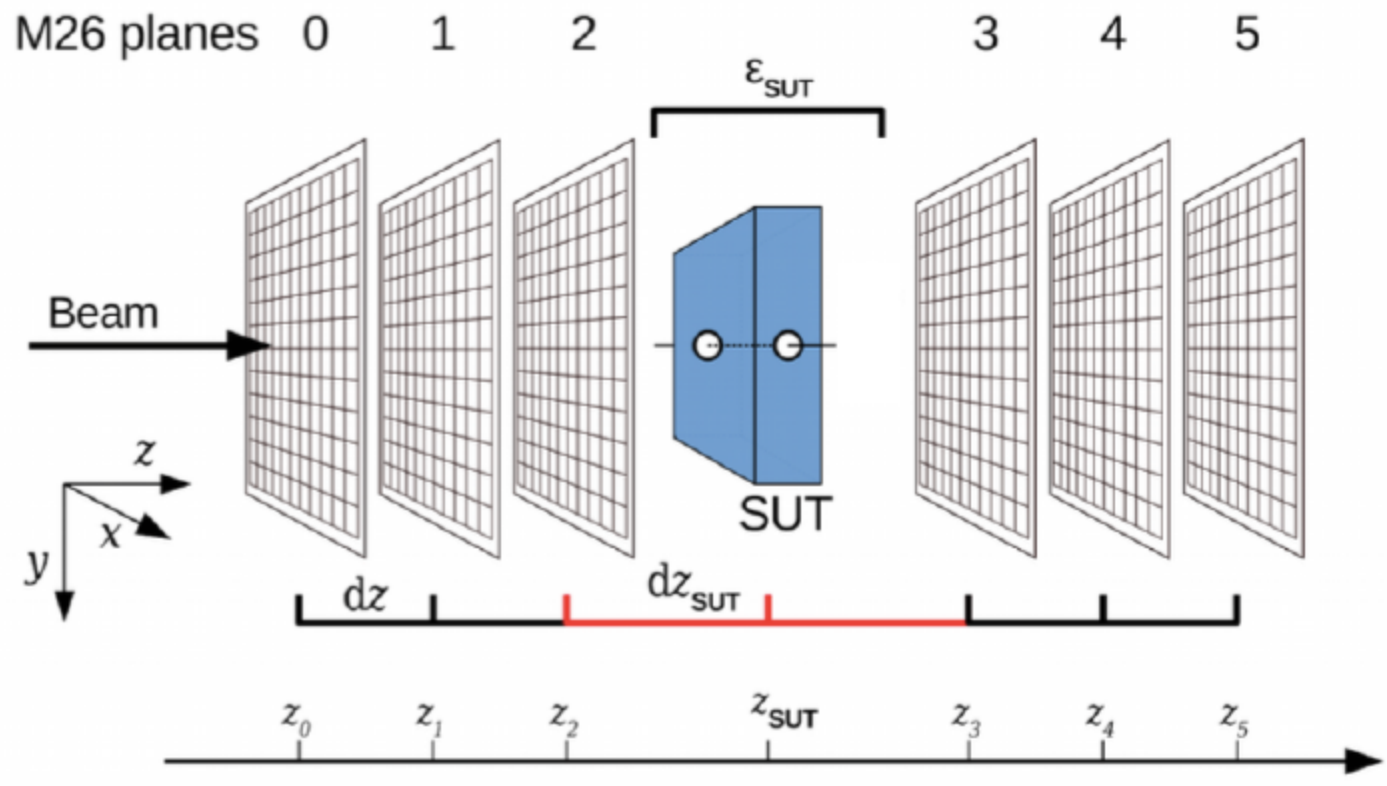
- Run step-by-step reconstruction:

```
jobsub -c config_DESY2018.cfg -csv runlist.csv -g noisypixel 701
jobsub -c config_DESY2018.cfg -csv runlist.csv -g clustering 701
jobsub -c config_DESY2018.cfg -csv runlist.csv -g hitmaker 701
jobsub -c config_DESY2018.cfg -csv runlist.csv -g alignGBL 701
jobsub -c config_DESY2018.cfg -csv runlist.csv -g alignGBL2 701
jobsub -c config_DESY2018.cfg -csv runlist.csv -g alignGBL3 701
jobsub -c config_DESY2018.cfg -csv runlist.csv -g fitGBL 701
```

Hands-On:
The GBL SUT example.

Telescope with SUT.

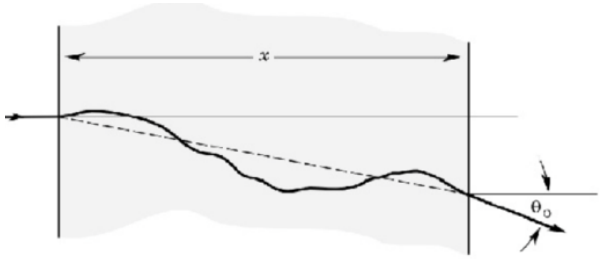
- Six Mimosa26 planes with an inserted scatterer (SUT).
- Data from DATURA telescope at 4 GeV/c recorded with EUDAQ2.
- Use General Broken Line (GBL) algorithm for alignment and fitting.

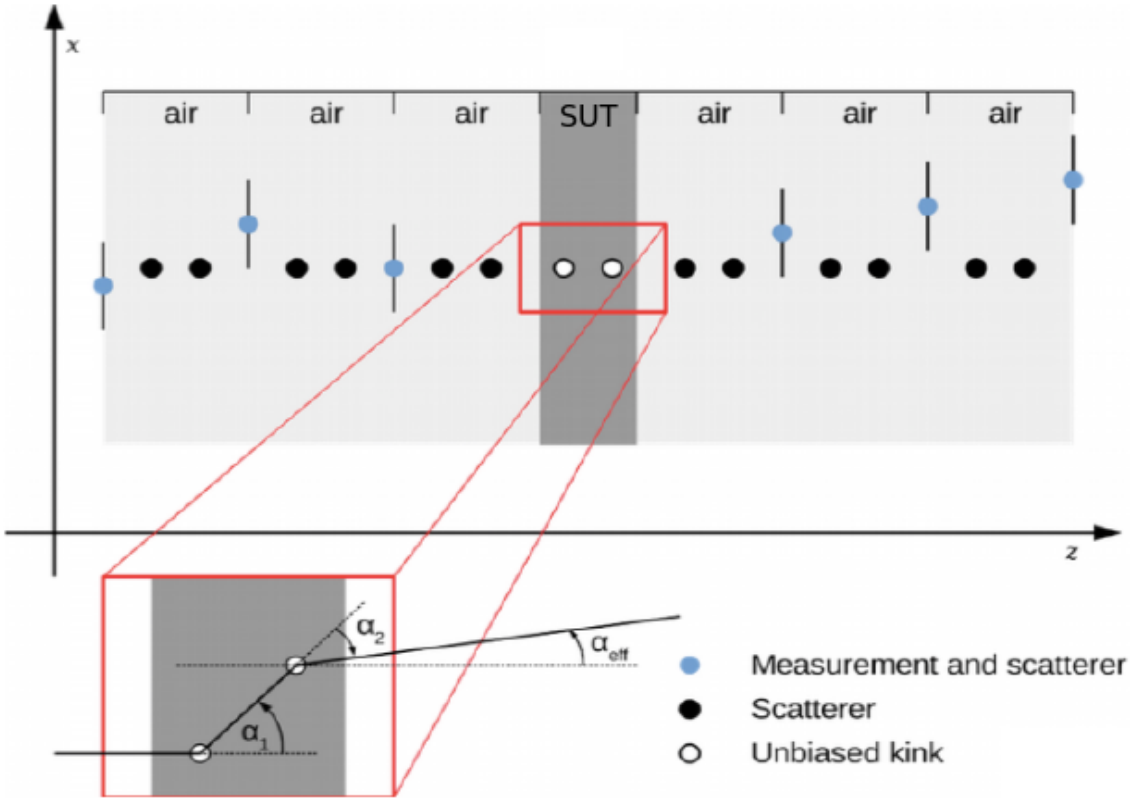


Intermezzo: Material budget imaging.

- Multiple scattering inside material dependent on material budget (X/X0) results in effective kink angle for each particle track.

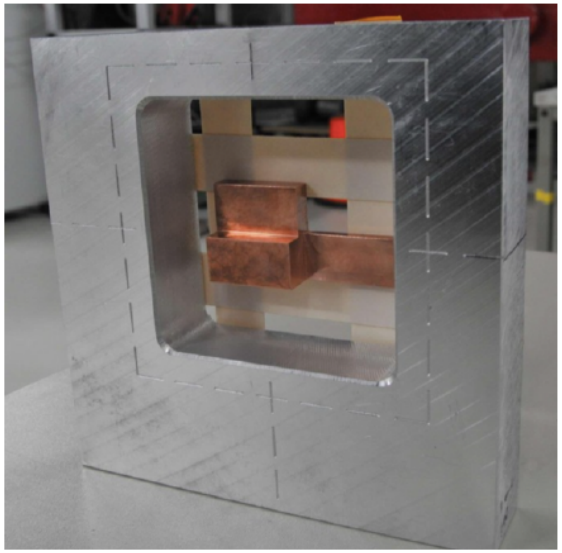
  → Reconstruct kink angles as estimator of X/X0





Setting up analysis: Cu sample.

- Data of copper sample as SUT, consisting of four different regions of X/X0= 0%, 25%, 50% and 100%.



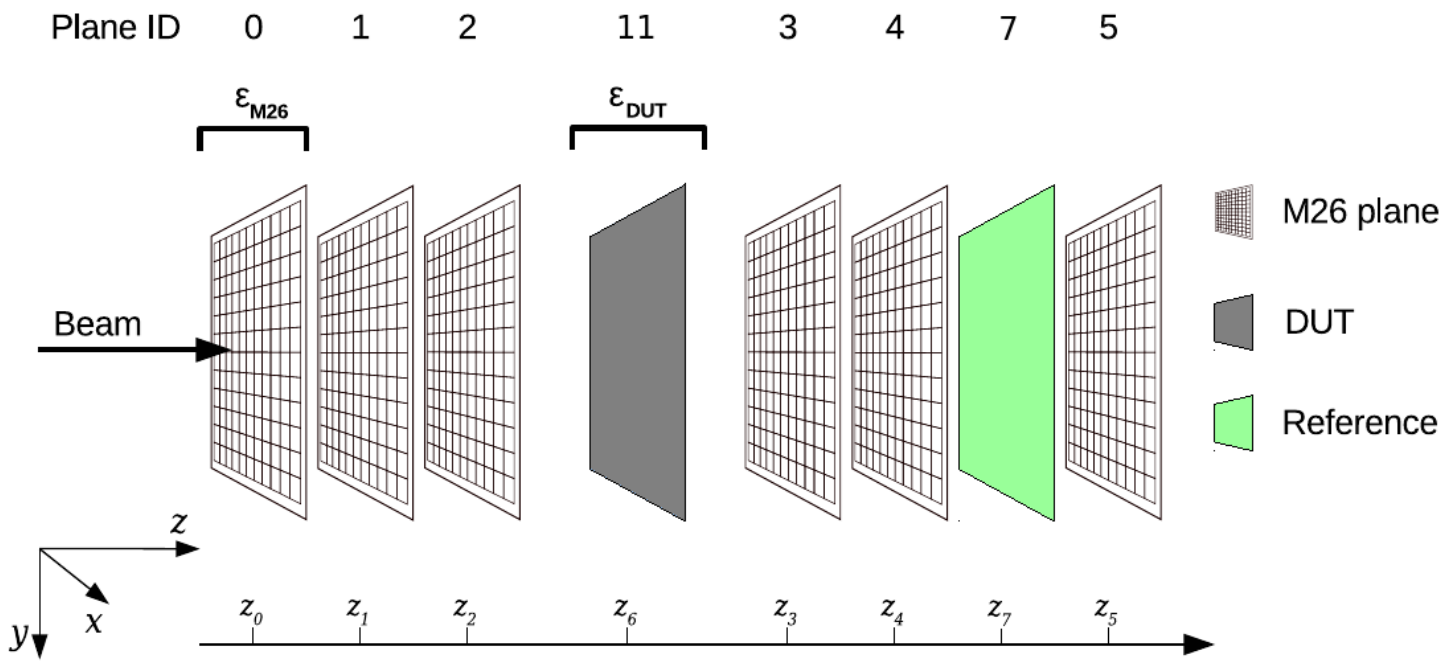
- Run step-by-step reconstruction:

```
jobsub -c config_CuSample.cfg -csv runlist.csv -g noisypixel 681
jobsub -c config_CuSample.cfg -csv runlist.csv -g clustering 681
jobsub -c config_CuSample.cfg -csv runlist.csv -g hitmaker 681
jobsub -c config_CuSample.cfg -csv runlist.csv -g alignGBL 681
jobsub -c config_CuSample.cfg -csv runlist.csv -g alignGBL2 681
jobsub -c config_CuSample.cfg -csv runlist.csv -g alignGBL3 681
jobsub -c config_CuSample.cfg -csv runlist.csv -g fitGBL 681
```

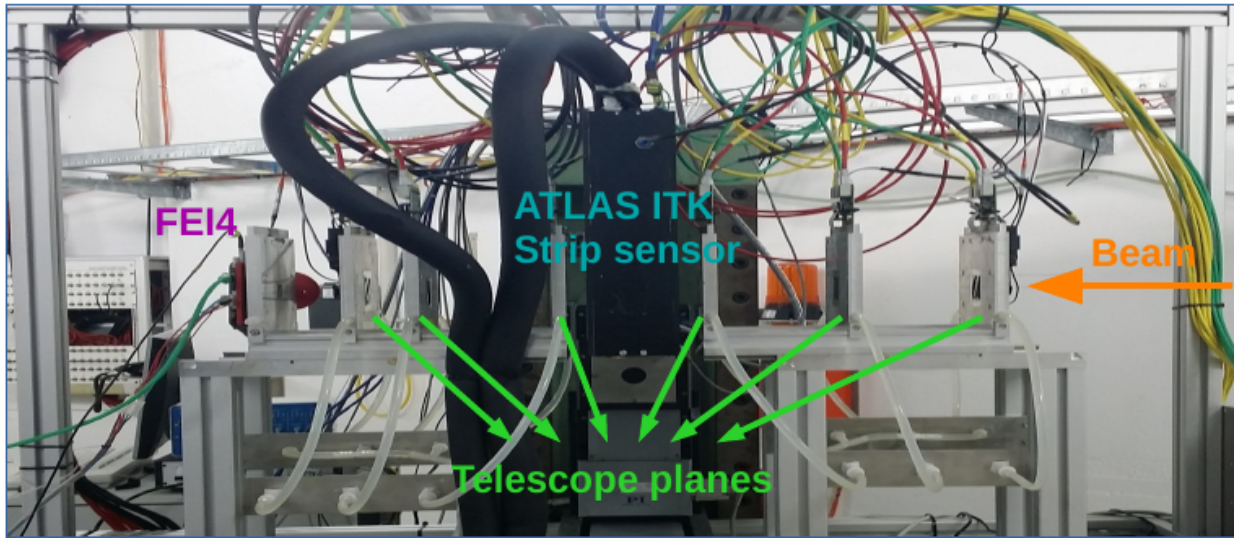
Hands-On:
The GBL DUT example.

Telescope with DUTs.

- Six Mimosa26 planes with a DUT (ATLAS ITk Strip sensor) and additional timing reference plane (ATLAS FEI4 pixel sensor).
- Data from DURANTA telescope at 4 GeV/c recorded with EUDAQ2.
- Use General Broken Line (GBL) algorithm for alignment and fitting.



Setting up analysis: Cu sample.



- Run step-by-step reconstruction:

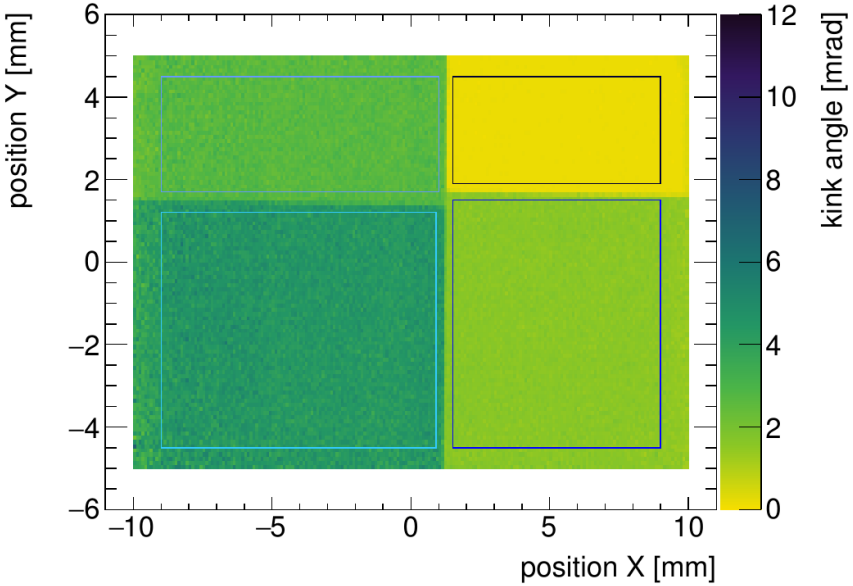
```
jobsub -c config.cfg -csv runlist.csv -g noisypixel 2365
jobsub -c config.cfg -csv runlist.csv -g clustering 2365
jobsub -c config.cfg -csv runlist.csv -g hitmaker 2365
jobsub -c config.cfg -csv runlist.csv -g alignGBL1 2365
jobsub -c config.cfg -csv runlist.csv -g alignGBL2 2365
jobsub -c config.cfg -csv runlist.csv -g alignGBL3 2365
jobsub -c config.cfg -csv runlist.csv -g fitGBL 2365
```

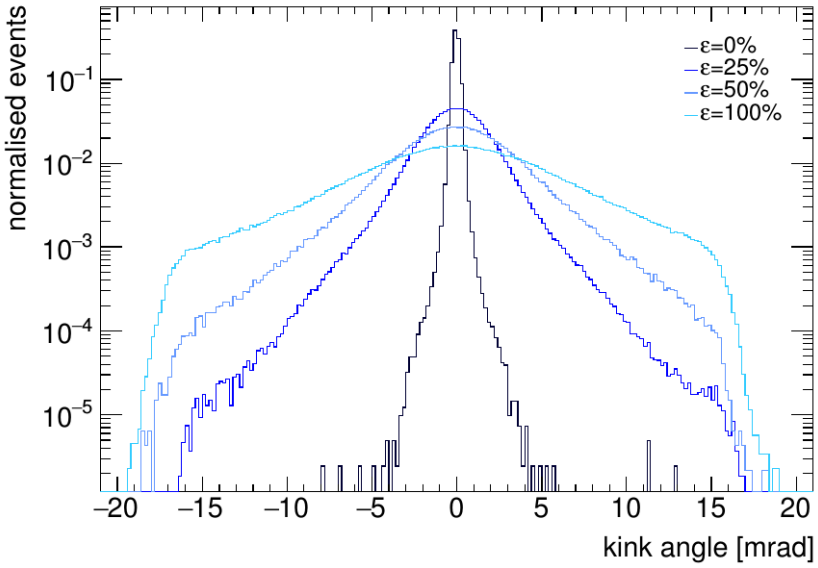
Results.

SUT analysis.

- You can perform a detailed analysis of the SUT example, using the final ROOT ntuple output.
- Example code:

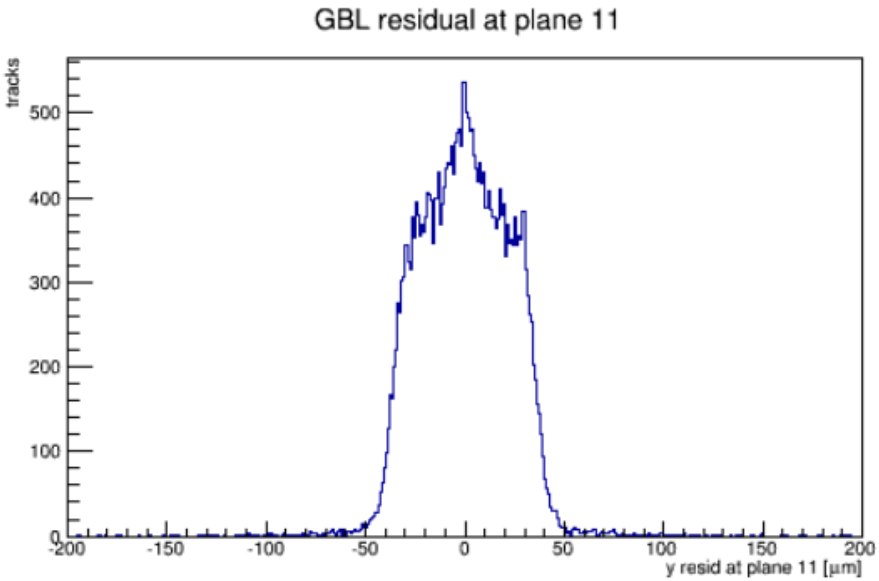
```
root -l run000681_output.root
> TTree* t = (TTree*) _file0->Get("Tracks")
> t->Draw("sqrt(kinkx**2):yPos:xPos","","PROFCOLZ")
```



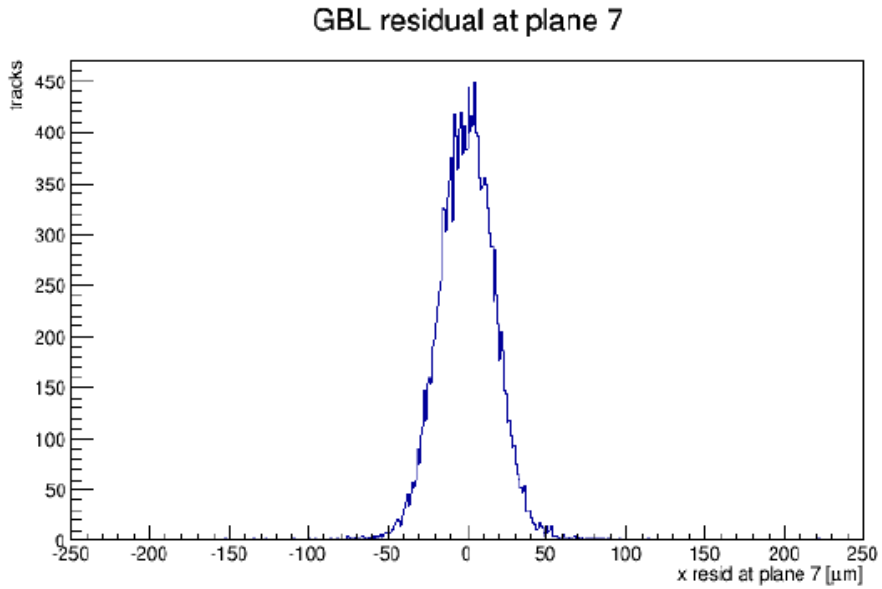


DUT analysis.

- Interesting to analyse here can be the reached spatial resolution on the DUT and the reference plane, as shown here:



Residual distribution of the ATLAS ITk strip sensor.



Residual distribution of the ATLAS FEI4 pixel sensor.

- But also more detailed analysis, such as resolution as function of cluster size, can be performed after reconstruction with the final tracks.

What else?

- **Increase of events:**

  So far, we were running only on a part of the data. You can increase the number of events, but especially the alignment takes then more time, but here it is sufficient to align with only a fraction of events.

- **Check alignment:**

  Use the cuts suggestion to optimize your alignment. Is it worth to add more iterations of the GBL alignment?

- **Create special analysis folder:**

  Now we executed the examples directly in the EUTelescope folder. Good habit is to run the analysis in external folders, but then, this has to be specified as *BasePath* in the corresponding configuration file.

- **External data:**

  The data input path is specified as the *NativePath* in the configuration file. For the examples, it is fixed to the AFS path providing the example data, but for your own data, you have to change it.

What else?

- **Advanced computing:**

  Instead of running locally, you can send your job to the batch cluster via HTCondor. This is especially helpful for parallelizing the reconstruction on more data samples. For this, check out *examples/condor_submission*.

- **Jobsub options:**

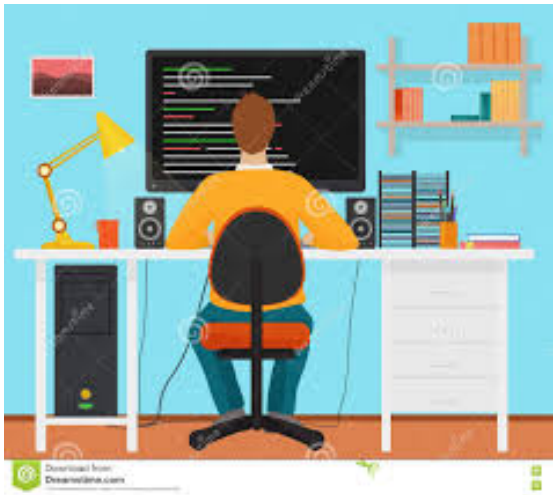
  The jobsub command provides a number of options, e.g.:

  - **-c** : configuration file (mandatory)
  - **-csv** : runlist file
  - **-o var=1** : overriding variable var from config manually
  - **-condor** : see advanced computing
  - **--dry-run** : generate filled steering file without running
  - **-g** : colorized output
  - **--subdir** : create a folder per run for run-specific output

Conclusion.

What you take home...

- You learned about the basic philosophy of reconstructing telescope data using the EUTelescope framework.
- You were able to install/run EUTelescope on your preferred system.
- You tested the analysis flow for an empty telescope using GBL.
- You were able to retrieve kink angle maps of a SUT.
- You performed the analysis on an active DUT example.
- You familiarized yourself with the processor workflow of EUTelescope.



Source: https://de.dreamstime.com/stock-abbildung-bemannen-sie-das-hintere-arbeiten-des-programmierers-seinem-pc-computer-kodierung-und-programmierung-image80997205