KalmanFilter-based alignment in ACTS

Zequn Sun (IHEP), Xiaocong Ai (ZZU), Andreas Salzburger (CERN), Pawel Bruckman De Renstrom (Polish Academy of Sciences)

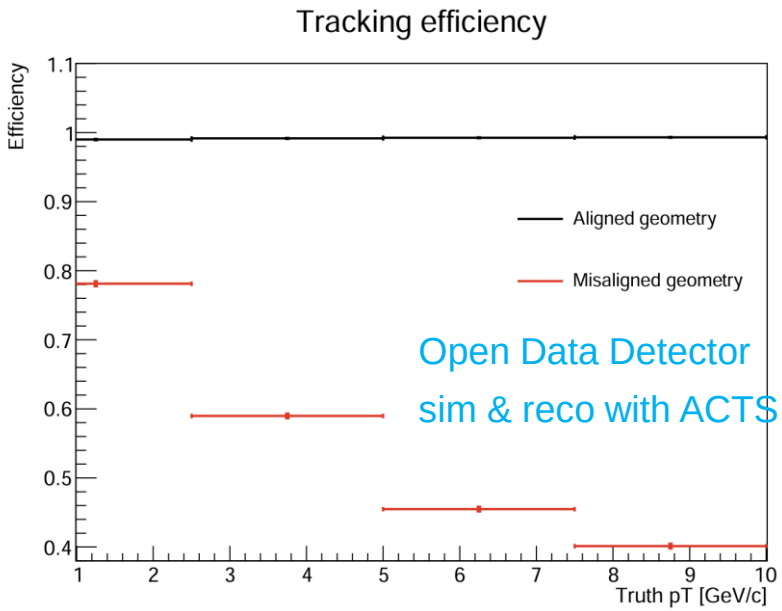
FTCF2025, Huangshan, Nov 25, 2025

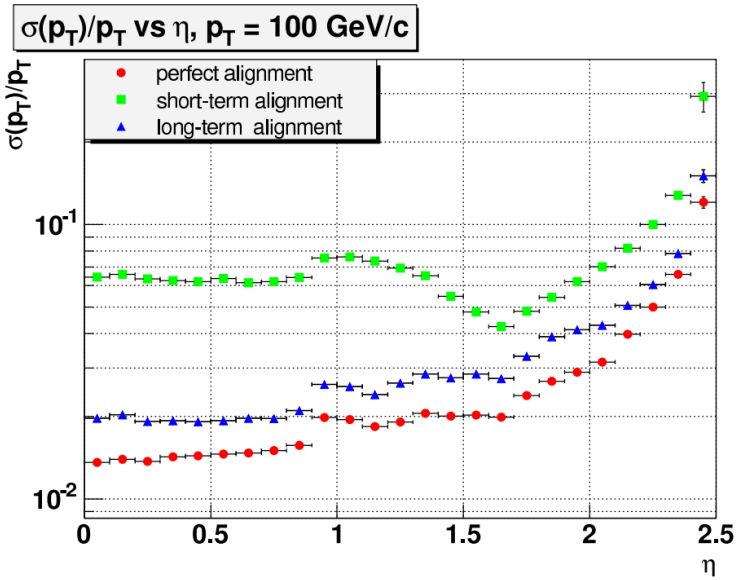
Why alignment matters?

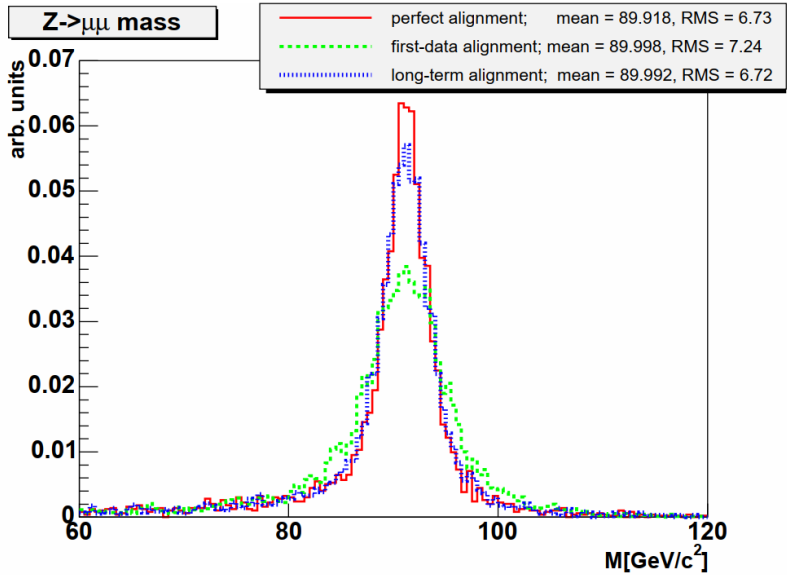
See more in Pawel Bruckman's slides

- Limited detector placement precision upon installation
- Misalignment is the dominant source of measurement resolution degradation

➡ Degradation of tracking precision and efficiency, and eventually **physics precision**!



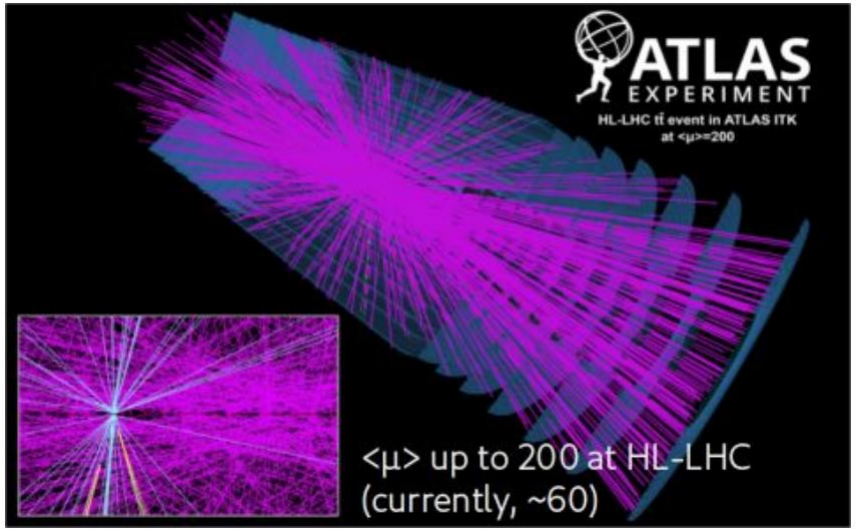




From G. Steinbruck

Why Kalman Filter based alignment?

- HL-LHC: recorded data (7-10x), pile-up <μ> = 200
  - Highly-performant tracking software
- Track-based detector alignment primarily relies on global χ^2 fitting methods:
  - High computational complexity
  - Difficulty in handling material effects and non-Gaussian noise



- Kalman Filter is widely used for track fitting in the high energy physics
  - Transition to a Kalman-Filter based alignment algorithm
  - Expected to enhance robustness and efficiency in detector alignment

ACTS recap

ACTS project

- A modern **open-source detector-independent** tracking toolkit for current&future HEP experiments based on LHC (and beyond) tracking experience
  - Data production by **ATLAS, FASER, sPHENIX**
  - Detector R&D by **CEPC, STCF, EIC, ePIC, LDMX**… *(see talk of Hao Li)*
- A R&D platform for ML-based tracking, heterogeneous computing and 4D tracking

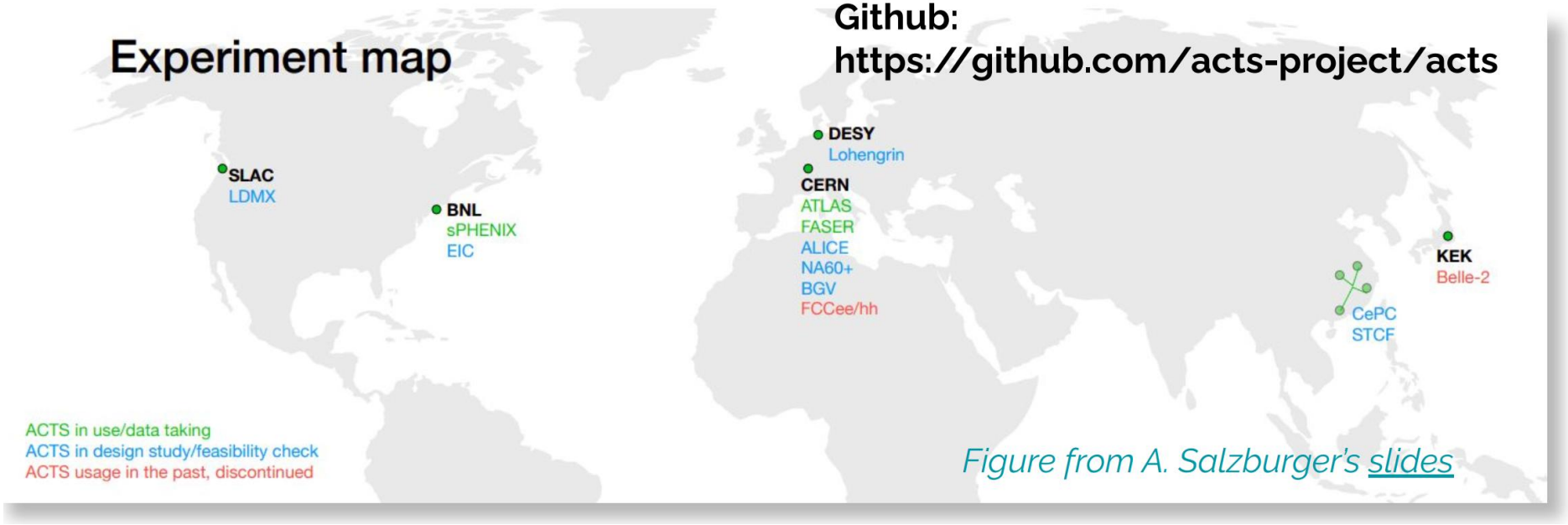
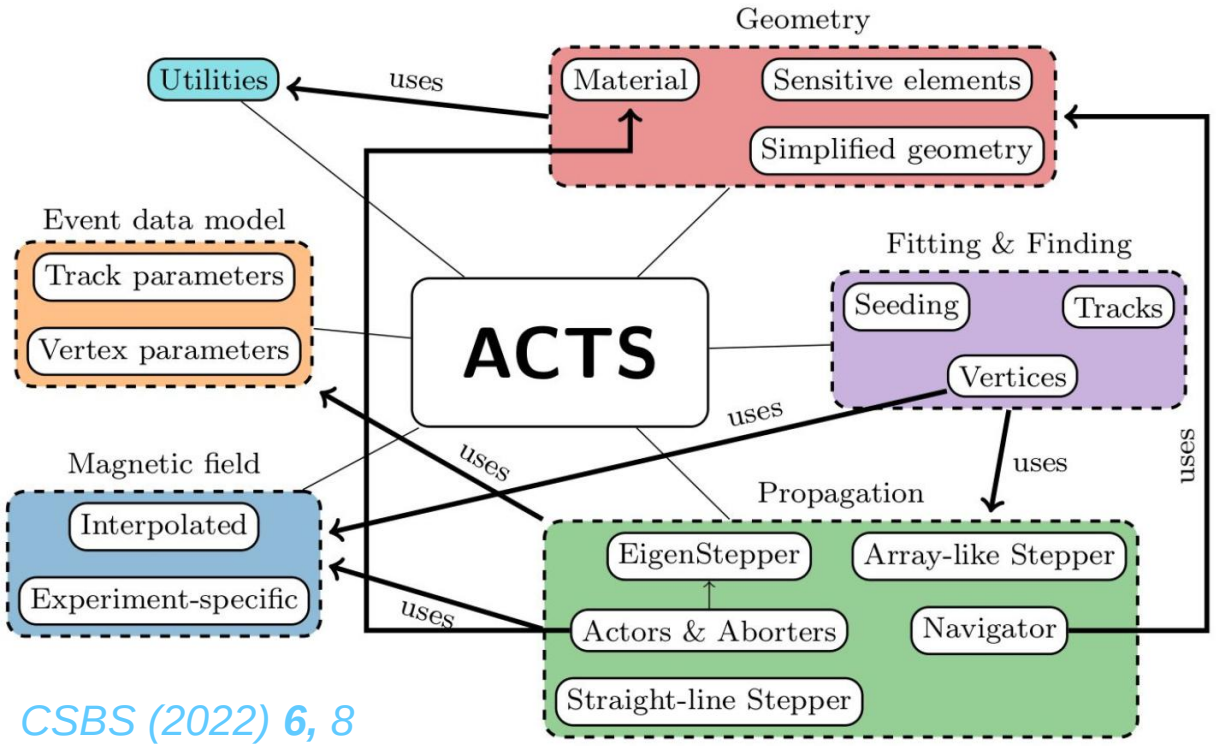


Figure from A. Salzburger's slides

ACTS design and functionalities

- Fully C++20 compliant, template design, strict thread-safety, contextual condition data …
  - More in *A. Salzburger's slides*

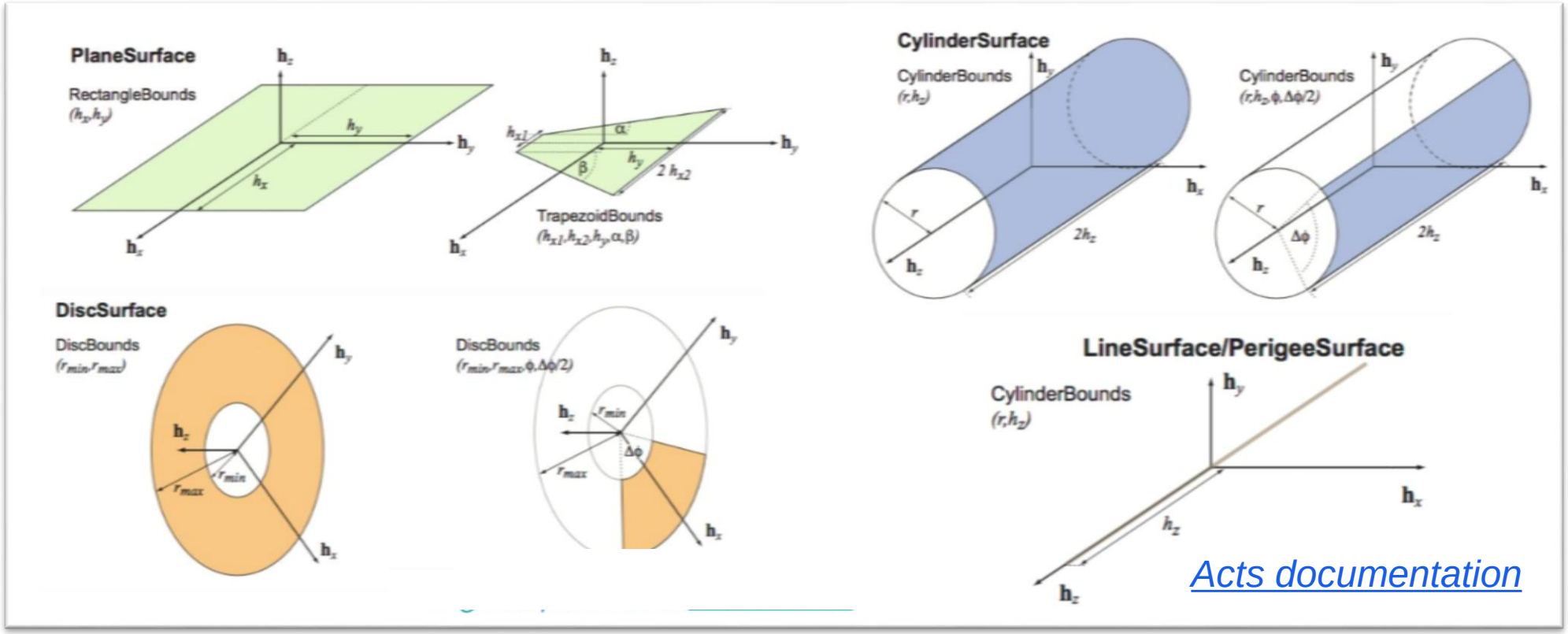


*CSBS (2022) **6**, 8*

- **Track fitting**
  - (Extended) Kalman Filter (KF), Gaussian Sum Filter, Non-linear KF
  - Global chisq fitter
- **Track finding**
  - Seeding, Combinatorial Kalman Filter (CKF), Graph Neural Networks, Hough Transform
- **Vertex finding&fitting**
  - Primary vertex: AMVF, IVF
- **KF-based Alignment** (prototype developed in 2022 and being slowly validated and optimized)

ACTS tracking geometry

- Tracking geometry is simplified from detailed full simulation geometry for fast navigation, but with material effects well taken into account
- Different concrete surfaces types for various tracking detectors
  - A surface has shape, bounds, **rotation+translation**, local coordinates and unique identifier…

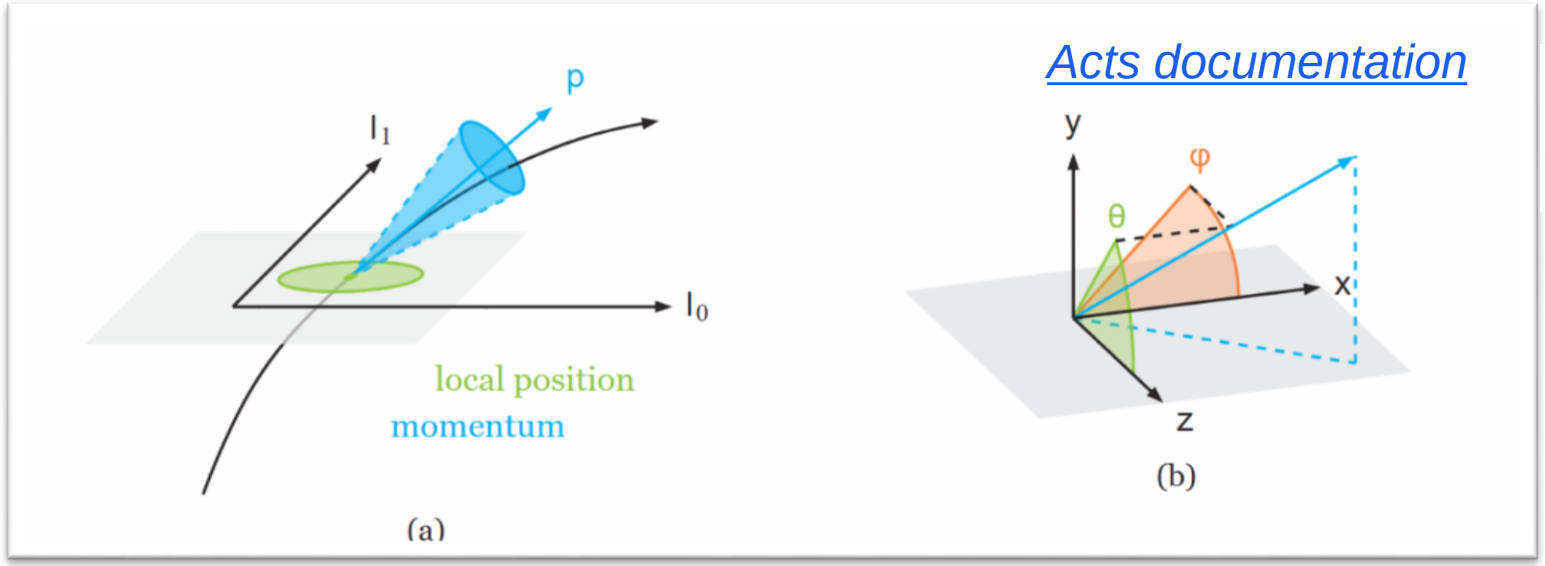


ACTS track parameterization

- 6-dimensional bound/local track parameters (integration of particle flight time in track propagation)

$$\vec{x} = (l_0, l_1, \phi, \theta, q/p, t)^T$$

- Measurement (1, 2, or 3 dimension) is a subset of the 6 parameters: $\vec{m} = H \cdot \vec{x}$



Acts documentation

e.g. $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$

Alignment in ACTS

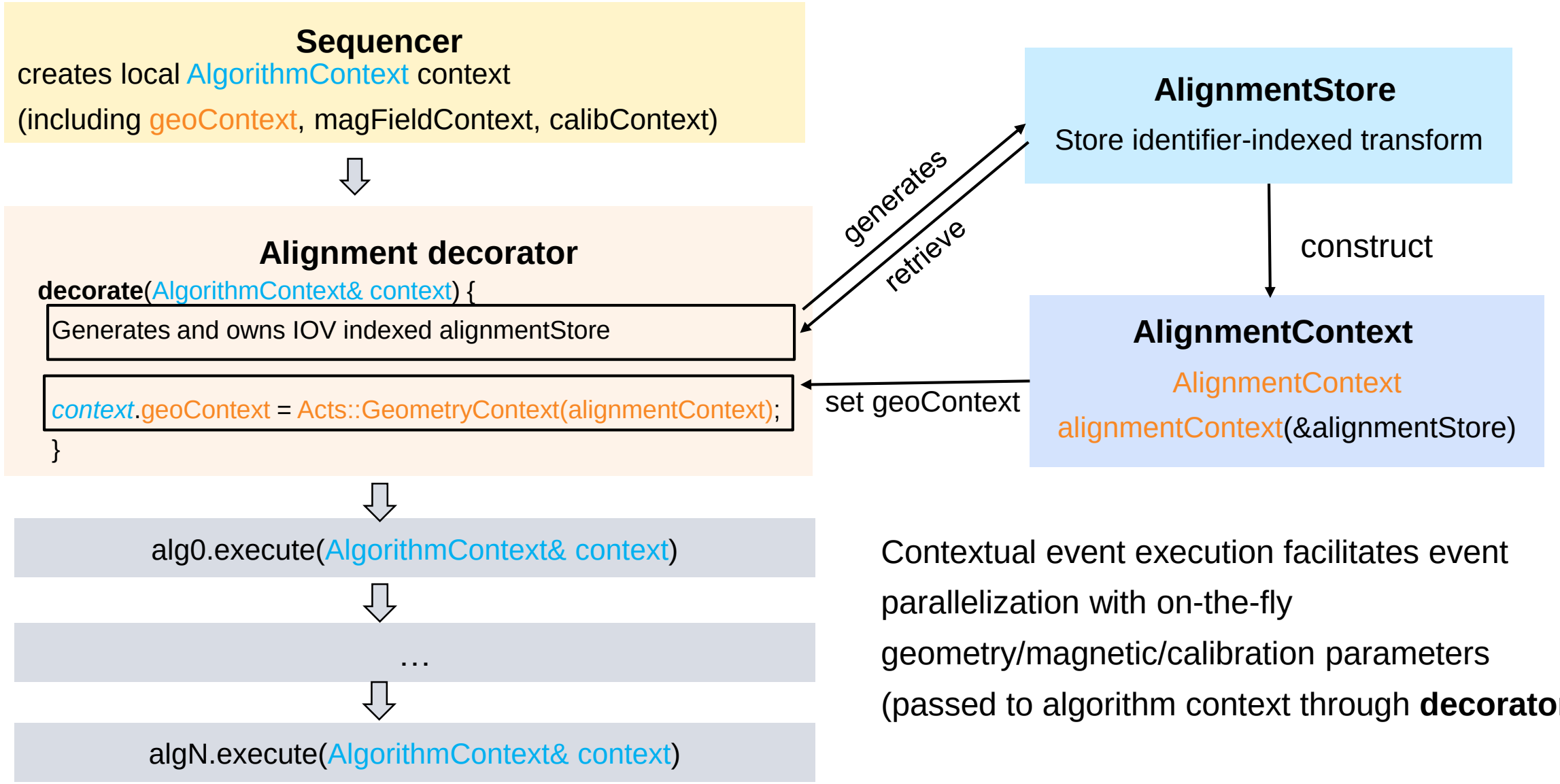
Alignment parameters

- Detector element placement description:
  - Translation (3 parameters) + Rotation (3x3 rotation matrix)
- Alignment parameters (6 parameters for now):
  - 3 parameters for translation along original local axes ($\vec{x}_L$, $\vec{y}_L$, $\vec{z}_L$)
  - 3 parameters for rotation about original local axes ($\vec{x}_L$, $\vec{y}_L$, $\vec{z}_L$) using Euler angles
    - Suppose rotation in the order: 1) around $\vec{x}_L$ about α → 2) around $\vec{y}_L$ about β → 3) around $\vec{z}_L$ about γ , then the new local axes become

$$(\vec{x}_L''', \vec{y}_L''', \vec{z}_L''') = (\vec{x}_L, \vec{y}_L, \vec{z}_L) \begin{pmatrix} cos\beta cos\gamma & sin\alpha sin\beta cos\gamma - cos\alpha sin\gamma & cos\alpha sin\beta cos\gamma + sin\alpha sin\gamma \\ cos\beta sin\gamma & sin\alpha sin\beta sin\gamma + cos\alpha cos\gamma & cos\alpha sin\beta sin\gamma - sin\alpha cos\gamma \\ -sin\beta & sin\alpha cos\beta & cos\alpha cos\beta \end{pmatrix}$$

Caveat: a rotation can be expressed in 24 equivalent sequence of Euler angles

Geometry Context



Contextual event execution facilitates event parallelization with on-the-fly geometry/magnetic/calibration parameters (passed to algorithm context through **decorators**)

Ideas of track-based alignment

- Tracks share the same detector geometry (a.k.a. **global track parameters** $\vec{\alpha}$) while they have their own track parameters (a.k.a. **local track parameters** $\vec{x_i}$)
- The global track parameters can be estimated by minimizing the χ^2 sum of a set of chosen good quality tracks:

$$\chi^2 = \sum_i {\chi_i}^2 = \sum_i [\vec{m}_i - \vec{h}_i(\vec{x}_i(\vec{\alpha}), \vec{\alpha})]^T V^{-1} [\vec{m}_i - \vec{h}_i(\vec{x}_i(\vec{\alpha}), \vec{\alpha})]$$

- This involves solving the non-linear equation iteratively, i.e. $\vec{\alpha}$ is updated iteratively to approach its optimal value:

$$\frac{d^2\chi^2}{d^2\vec{\alpha}} \Big|_{\vec{\alpha_0}} \Delta\vec{\alpha} = -\frac{d\chi^2}{d\vec{\alpha}} \Big|_{\vec{\alpha_0}}$$

The Kalman Filter fitter is used to fit each track in each iteration

Alignment ingredients

$r = m - h(x, \alpha)$ The track residual

V The measurement covariance

$H = \left.\frac{\partial h(x)}{\partial x}\right|_{x_0}$ The projection matrix from (bound) track parameters to measurement

C The covariance of track parameters at different measurements:
Straightforward with global chi2 fitter. Can provided by Kalman Filter as well

$A_{k\ell} \equiv \frac{\partial r_k}{\partial \alpha_\ell}$ The derivative of residual w.r.t. alignment parameters

The first and second derivatives for a single track and then summed over tracks

$$\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha} = 2A^T V^{-1}\left(V - HCH^T\right)V^{-1}r,$$

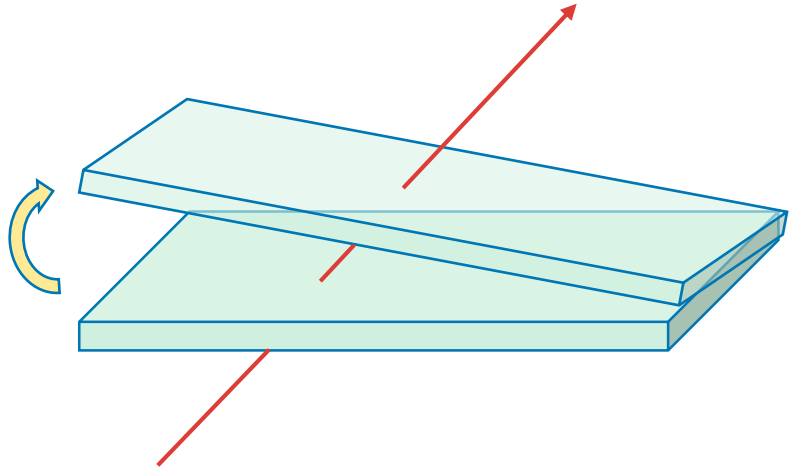
$$\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2} = 2A^T V^{-1}\left(V - HCH^T\right)V^{-1}A.$$

$$\left.\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2}\right|_{\alpha_0}\Delta\alpha = -\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\alpha}\right|_{\alpha_0}$$

Solved with Eigen LU decomposition
(claimed stable and well tested with large matrices)

Residual derivative

- Suppose the rotation (α, β, γ) around the original local axes is small, then



$$\frac{\partial \vec{x}_L'''}{\partial(\alpha,\beta,\gamma)} = \begin{pmatrix} \vec{x}_L & \vec{y}_L & \vec{z}_L \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} = R \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

$$\frac{\partial \vec{y}_L'''}{\partial(\alpha,\beta,\gamma)} = \begin{pmatrix} \vec{x}_L & \vec{y}_L & \vec{z}_L \end{pmatrix} \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = R \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\frac{\partial \vec{z}_L'''}{\partial(\alpha,\beta,\gamma)} = \begin{pmatrix} \vec{x}_L & \vec{y}_L & \vec{z}_L \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = R \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Residual $\vec{r} = (r_x, r_y)$:

$r_x = (\vec{x}_{track} - \vec{o}_{module}) \cdot \vec{x}_L - x_{hit}$

$r_y = (\vec{x}_{track} - \vec{o}_{module}) \cdot \vec{y}_L - y_{hit}$

- $\vec{x}_{track}$ is the intersection of track with detector module
  - Changed with $\vec{o}_{module}$ and $\vec{x}_L$, $\vec{y}_L$, $\vec{z}_L$
- $\vec{o}_{module}$ is the center of the detector module
- x_{hit} and y_{hit} are hit local position on module

An alignment prototype

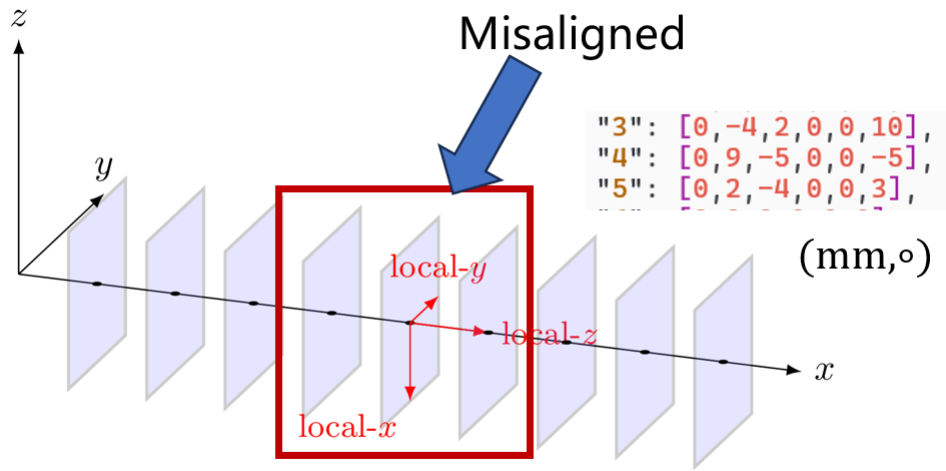
- It takes a set of tracks and sort out the sets of detector elements which can be and requested to be aligned => the initial value of $\vec{\alpha}$
- Perform the fit for each track using $\vec{\alpha}$, and estimate the χ^2 derivatives w.r.t. $\vec{\alpha}$
- Solve the equation to obtain $\Delta\vec{\alpha}$
- Update $\vec{\alpha}$ to become $\vec{\alpha}'$ using provided alignment parameter updater
- Stop iteration of the above three steps when provided converging criteria is met

```
template <typename trajectory_container_t,
          typename start_parameters_container_t, typename fit_options_t>
Acts::Result<AlignmentResult> align(
    const trajectory_container_t& trajectoryCollection,
    const start_parameters_container_t& startParametersCollection,
    const AlignmentOptions<fit_options_t>& alignOptions) const;
```

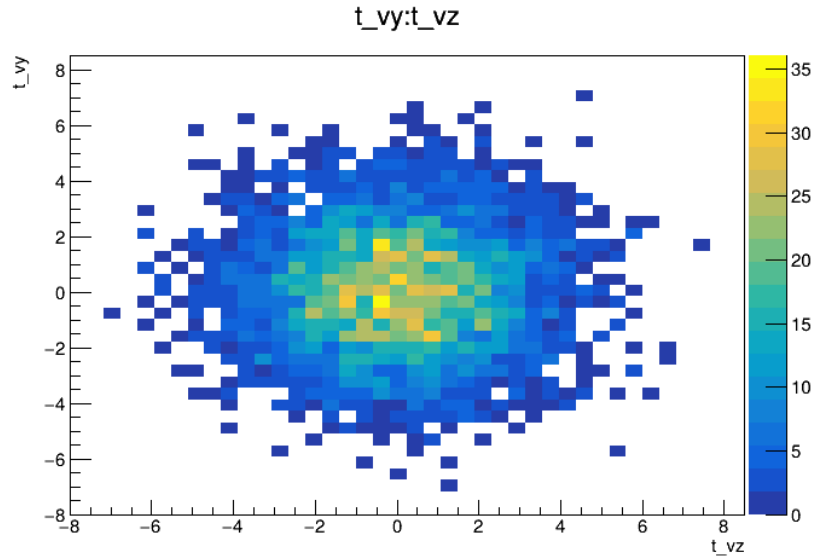
ACTS Alignment example

Sanity check with a telescope-like detector

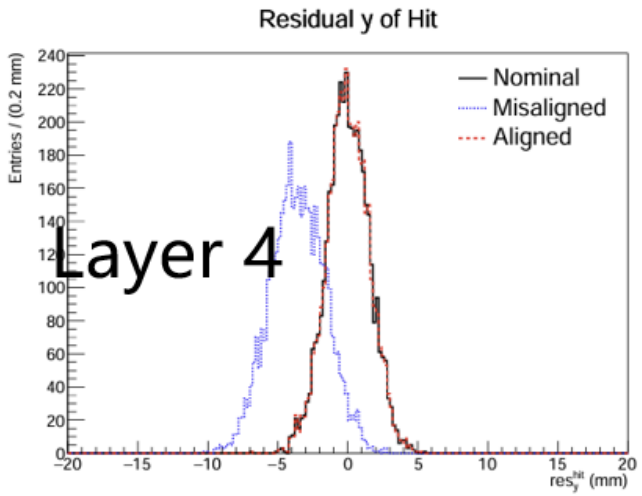
- Particle Gun:
  - 10k muon per event, $p = 10\ \mathrm{GeV}$, direction of momentum along global x-axis
  - Vertex: x, t=0, y and z~2D Gaussian with $\sigma_y = \sigma_z = 2\ \mathrm{mm}$
- Alignment degree of freedom: Translation-in-x/y+Rotation-around-z
- Only align the misaligned layers

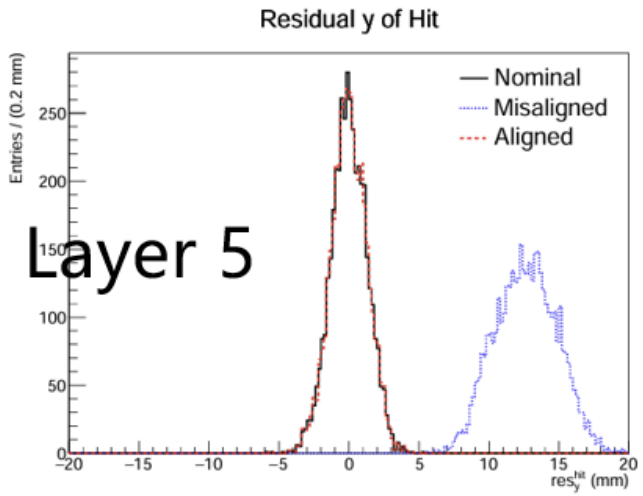


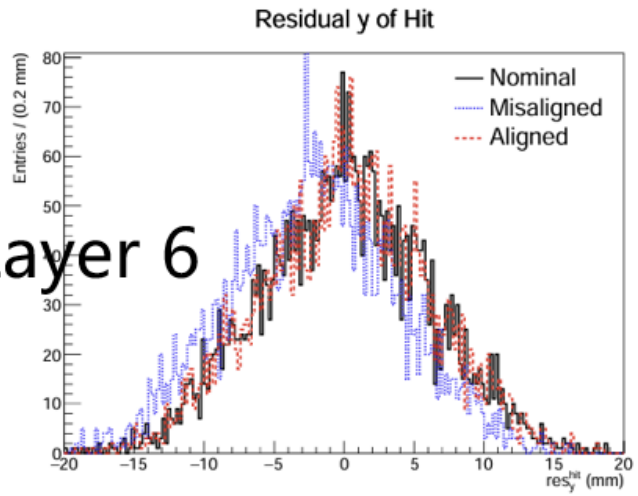
25 x 25 um2 resolution

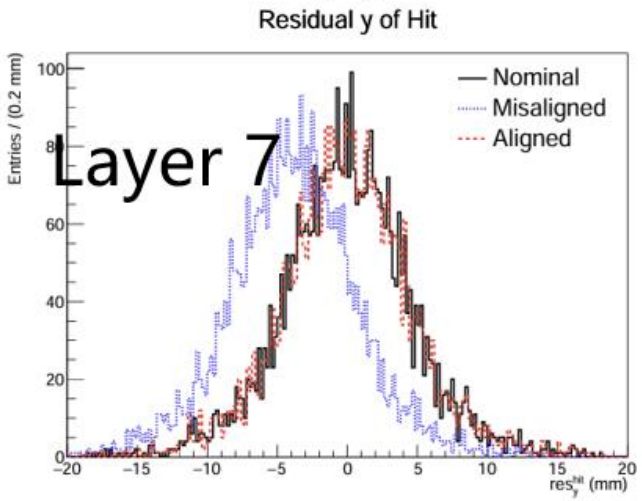


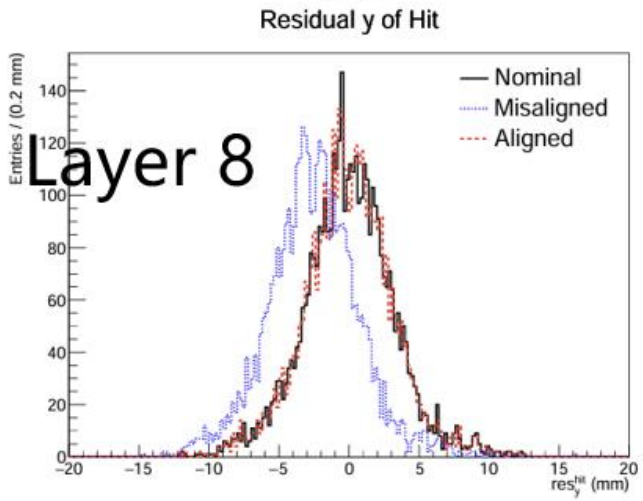
Residual at telescope layer

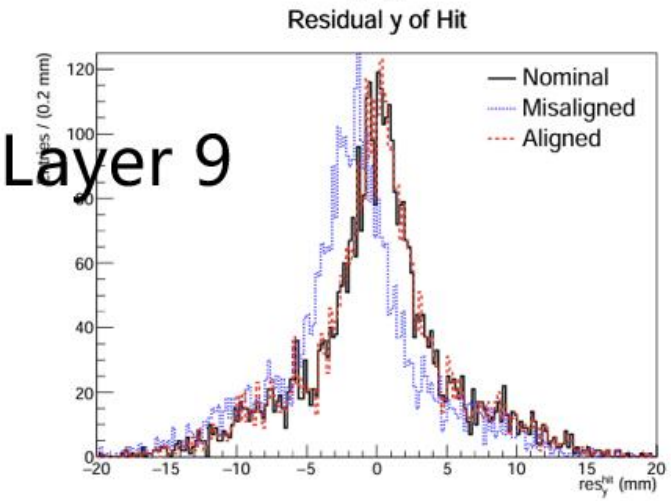








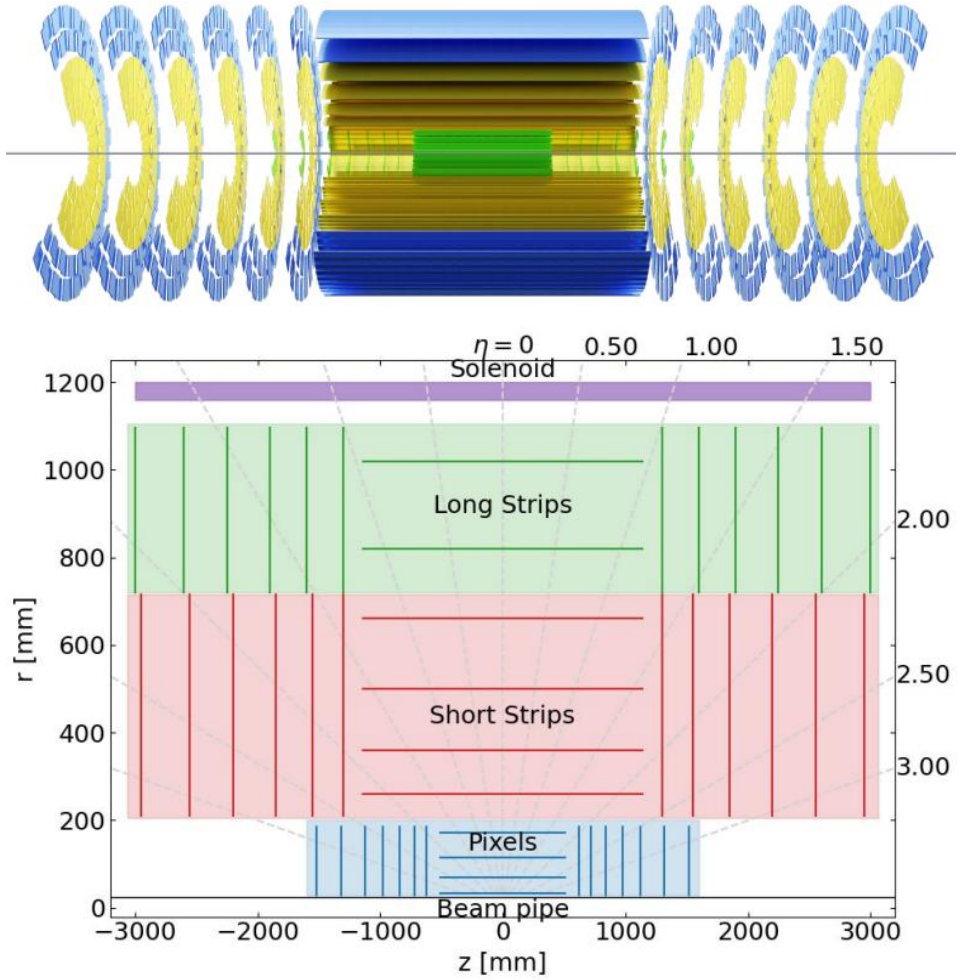




Validation with a realistic example detector

- Pixel
  - 2D Resolution: 15 μm
  - 4 barrel layers + 7 endcap disks
- Short strips
  - 2D Resolution: 43 μm / 1.2 mm
  - 4 barrel layers + 6 endcap disks
- Long strips
  - 1D, two sided with stereo angle (Resolution: 72 μm)
  - 2 barrel layers + 6 endcap disks

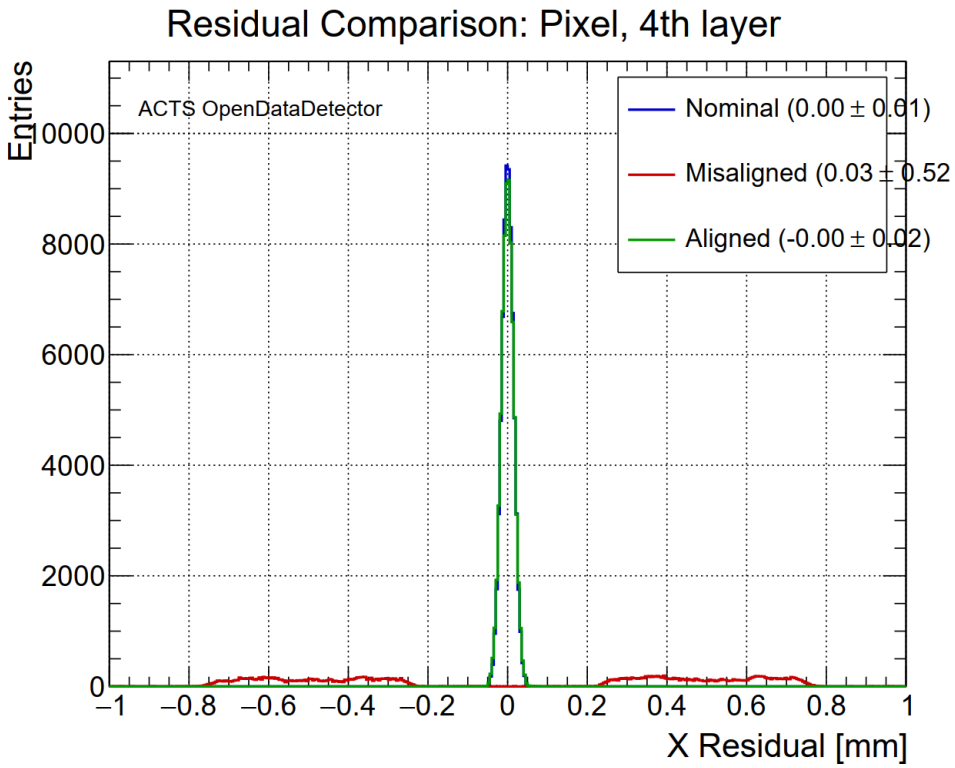
Misalignment decoration (testing two layers for now:
Pixel-4th layer and Short Strip-2nd layer):
→ Translation-in-x/y: +/- [0.25,0.75] mm
→ Rotation-about-z: +/- [0.01,0.03] rad

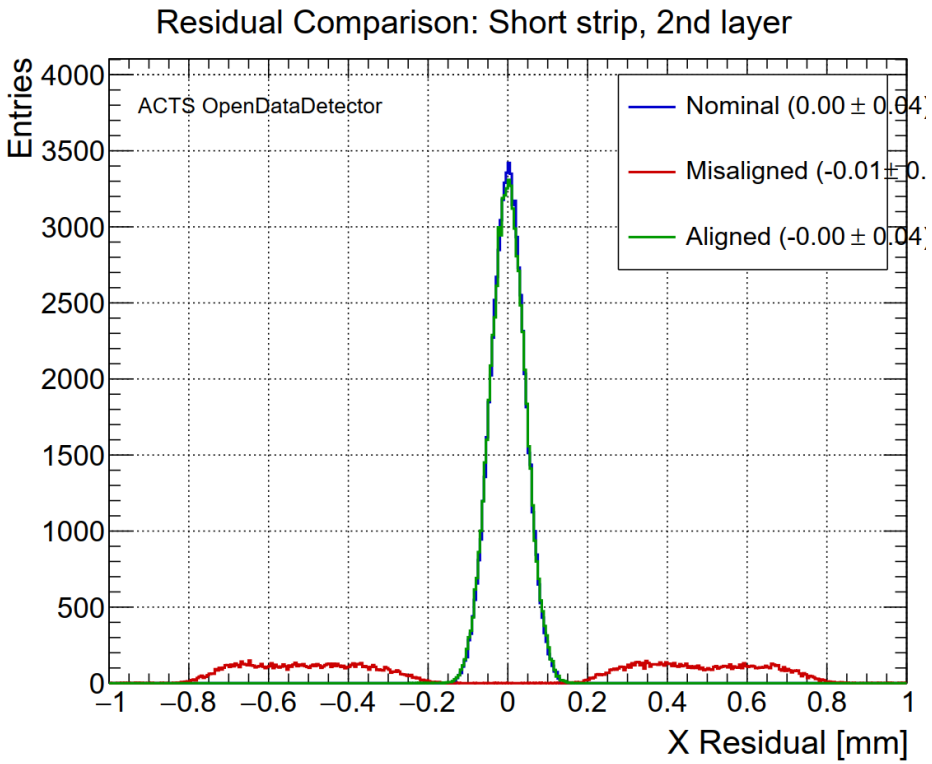


GitLab: https://gitlab.cern.ch/acts/OpenDataDetector
ACAT 2021: https://indico.cern.ch/event/855454/contributions/4596738
CHEP 2023: https://indico.jlab.org/event/459/contributions/11546

Residual

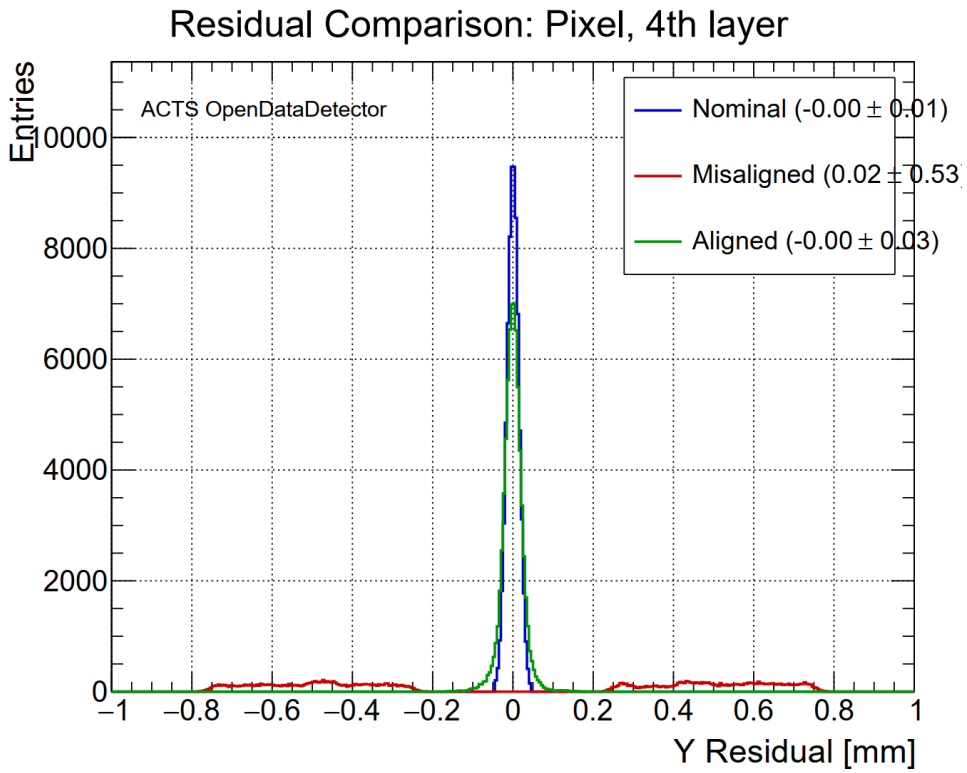
- The residual in local x direction looks good (consistent with results without ideal detector geometry)

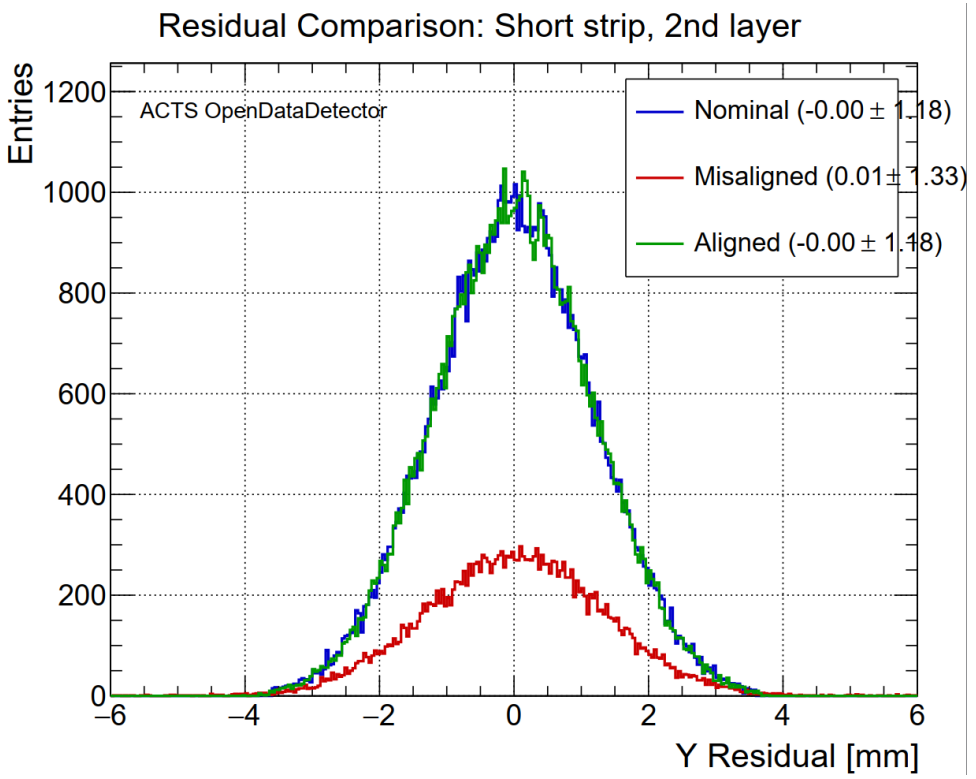




Residual

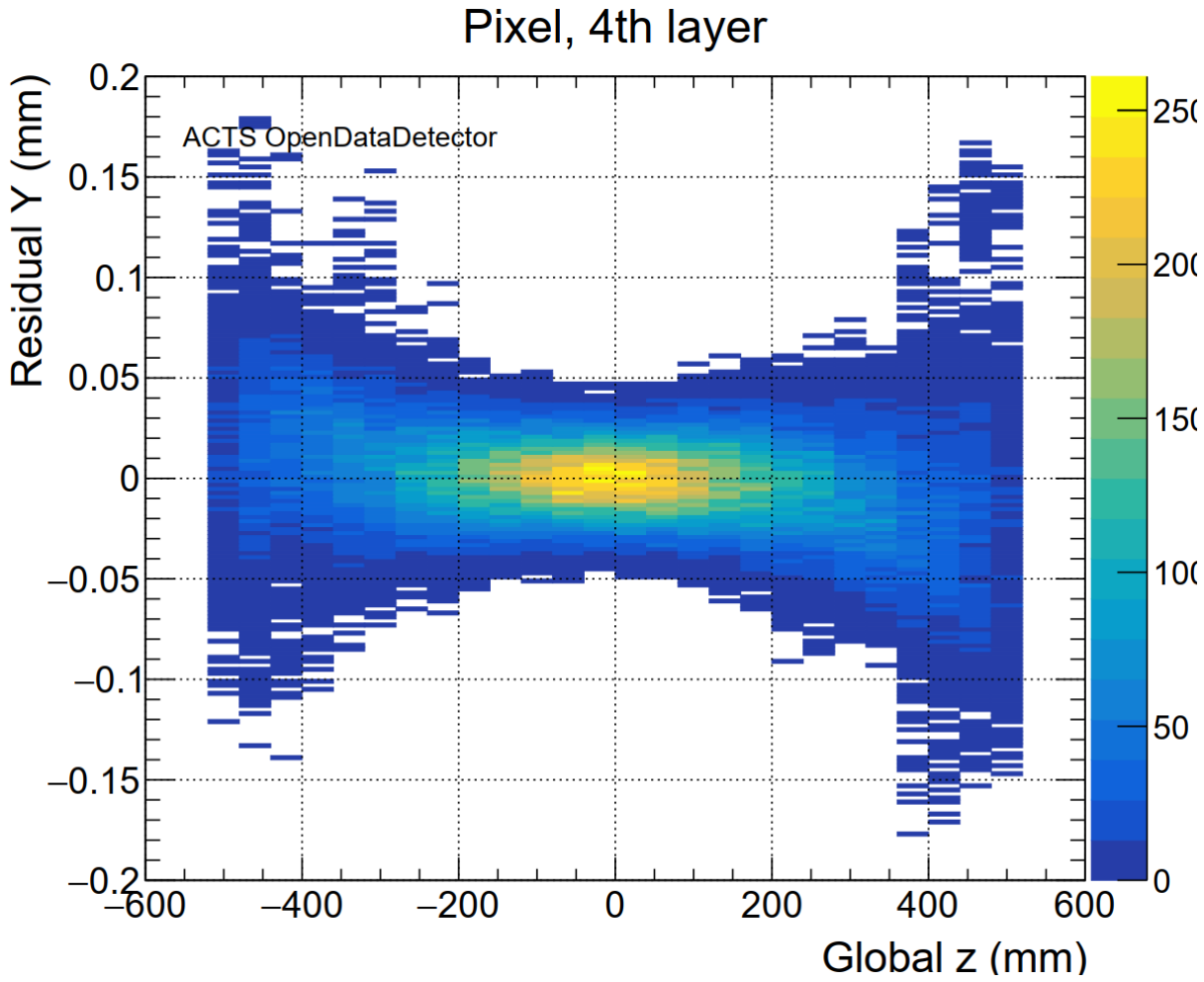
- The residual in local y direction at Pixel is sub-optimal (under investigation)





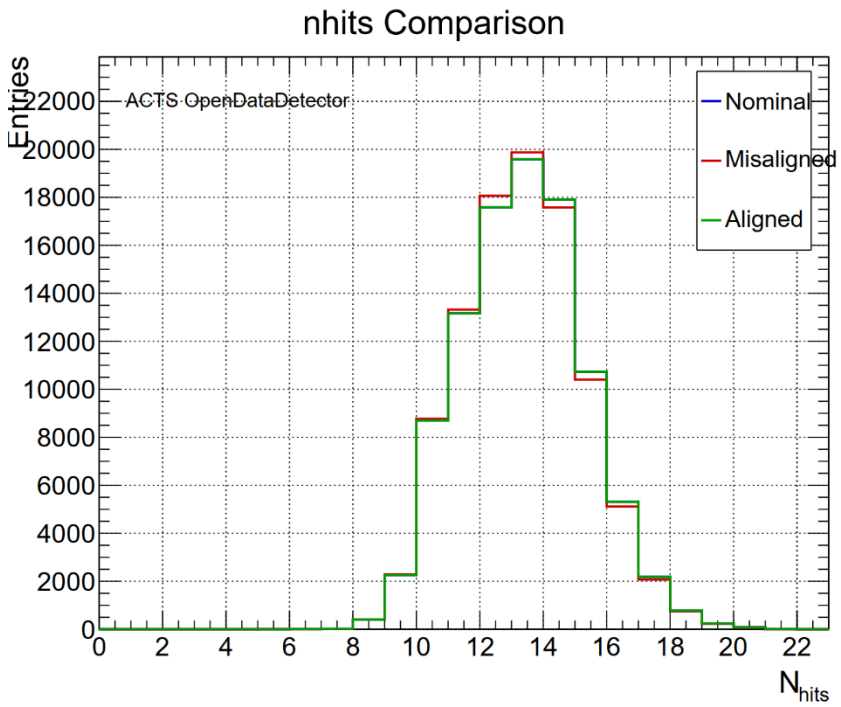
Residual

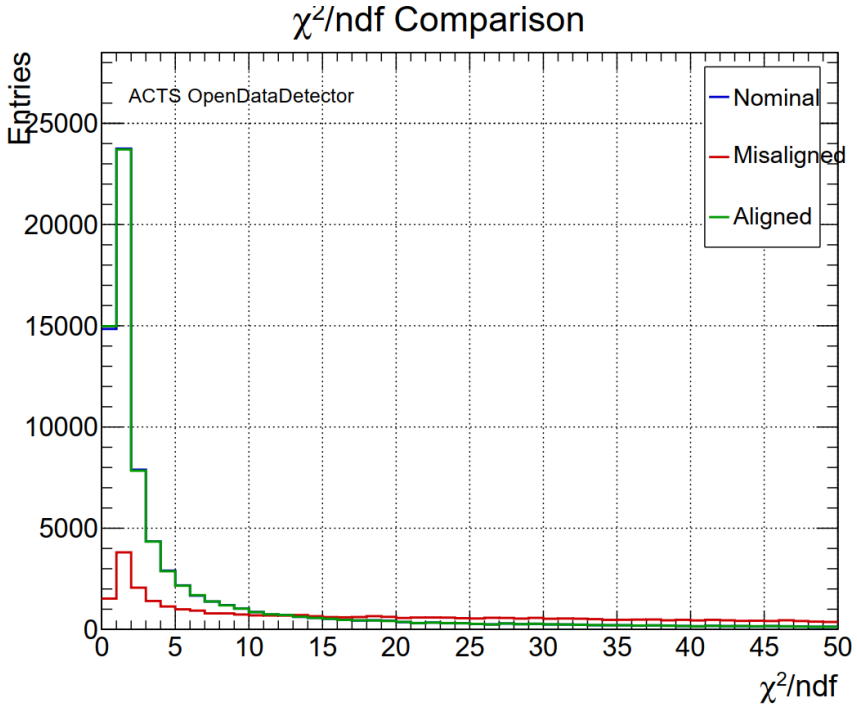
- The residual in local y direction at Pixel gets larger at larger global |z|



Track quality

- Improved nHits and χ^2/ndf for tracks with aligned geometry

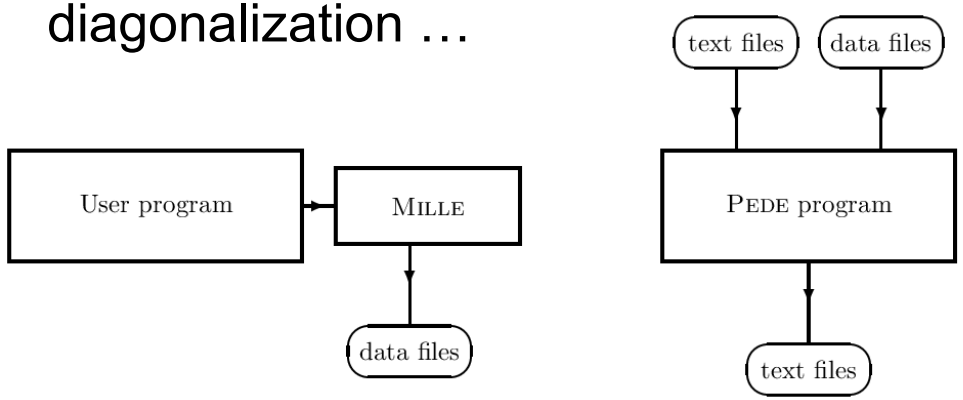




Minimization with MillePede ?

MillePede is a traditional alignment tool used and validated by BESIII, CMS …

- MillePede = Mille + Pede
  - **Mille**: write residual, derivatives of residual w.r.t. alignment parameters and track parameters into binary file
  - **Pede**: solve equation with dimension n to get solution for delta of track parameters :
    - Solver: Inversion, Cholesky decomposition, diagonalization …



Called for each dimension of a residual of each track

◆ mille()

```
void Mille::mille ( int          NLC,
                    const float * derLc,
                    int          NGL,
                    const float * derGl,
                    const int *  label,
                    float        rMeas,
                    float        sigma
                  )
```

Add measurement to buffer.

Parameters

- [in] **NLC** number of local derivatives
- [in] **derLc** local derivatives
- [in] **NGL** number of global derivatives
- [in] **derGl** global derivatives
- [in] **label** global labels
- [in] **rMeas** measurement (residuum)
- [in] **sigma** error

Summary

- ACTS is a tracking software, but also provides ingredients for alignment
  - Validated with a toy telescope-like detector and a realistic Open Data Detector
  - Only surface/module-wise for now. Superstructure alignment is in progress
- Minimization needs to be handled externally by the users, e.g. Mille-Pede.
  - A toy minimization based on Eigen solver is in place more for proof of principle
- Join us if you are interested:
  - https://mattermost.web.cern.ch/acts/channels/acts-alignment