KalmanFilter-based alignment in ACTS

Xiaocong Ai (ZZU)

On behalf of the ACTS-alignment team

2nd Tracking Workshp in Particle Physics Experiments, Huizhou, July 23, 2025

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** *(see talks of J. Osborn and S. Zhang)*
  - Detector R&D by **CEPC, STCF, EIC, ePIC, LDMX**… *(see talks of Y. Zhang and J. Zhang)*
- A R&D platform for ML-based tracking, heterogeneous computing and 4D tracking *(see talks of A. Salzburger, V. Cairo and Y. Wang)*

Github: https://github.com/acts-project/acts

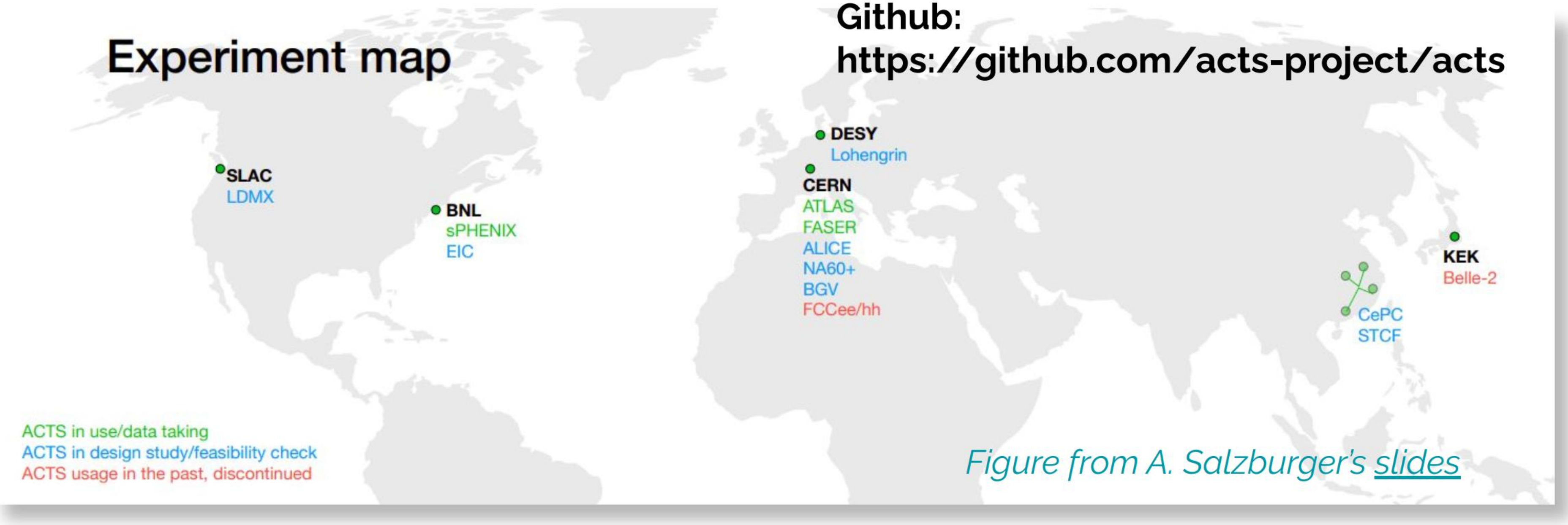
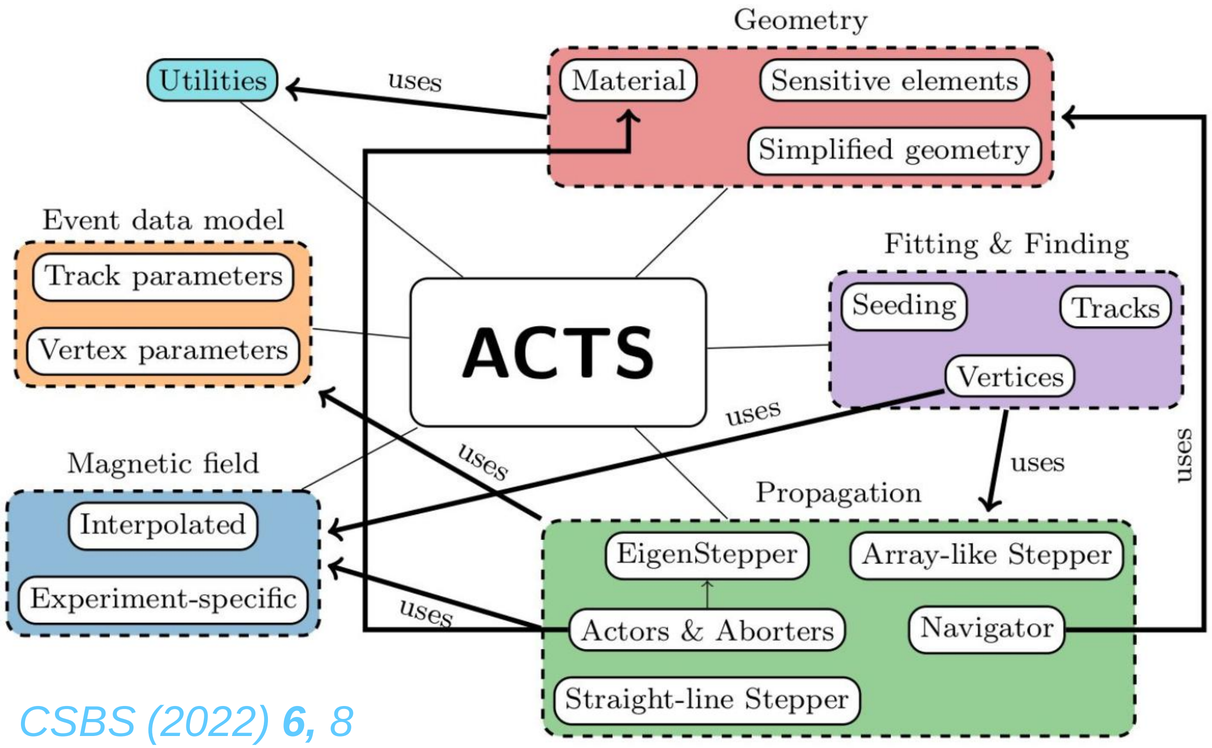


Figure from A. Salzburger's slides

ACTS design and functionalities

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

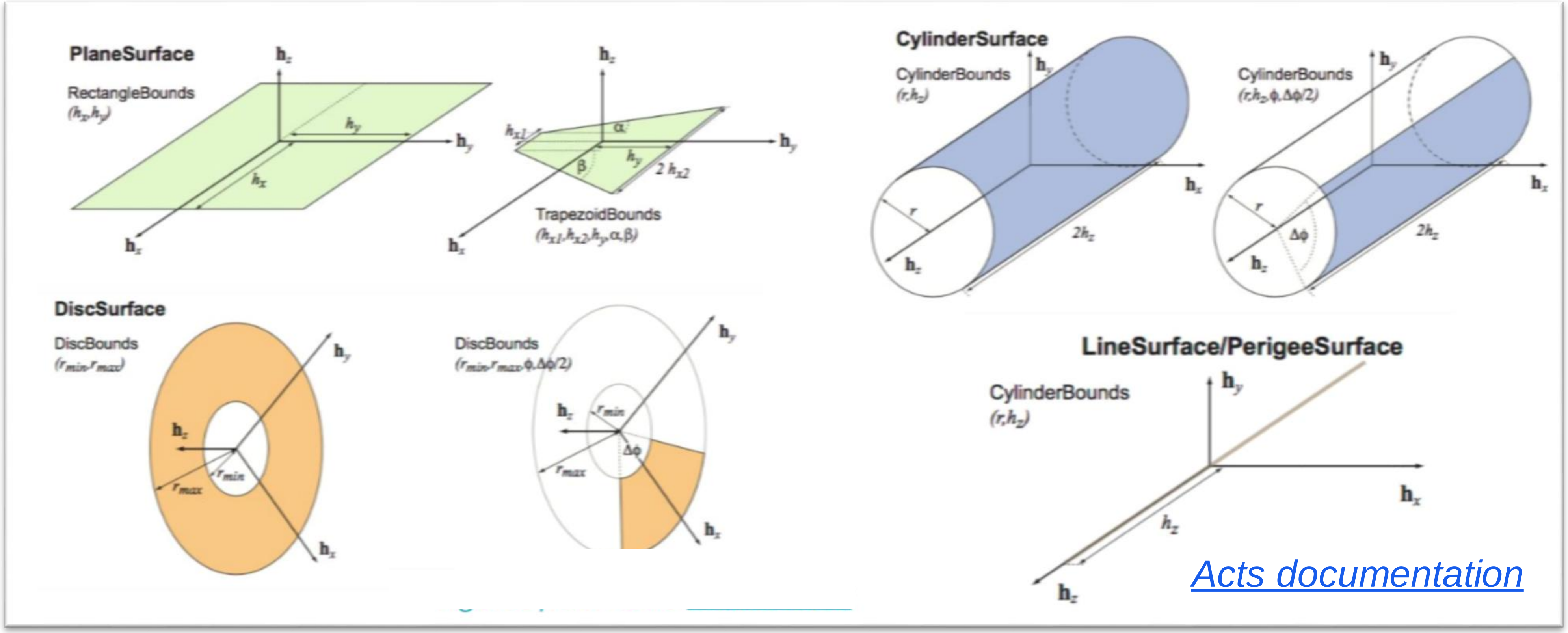


*CSBS (2022) **6**, 8*

- **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 documentation]

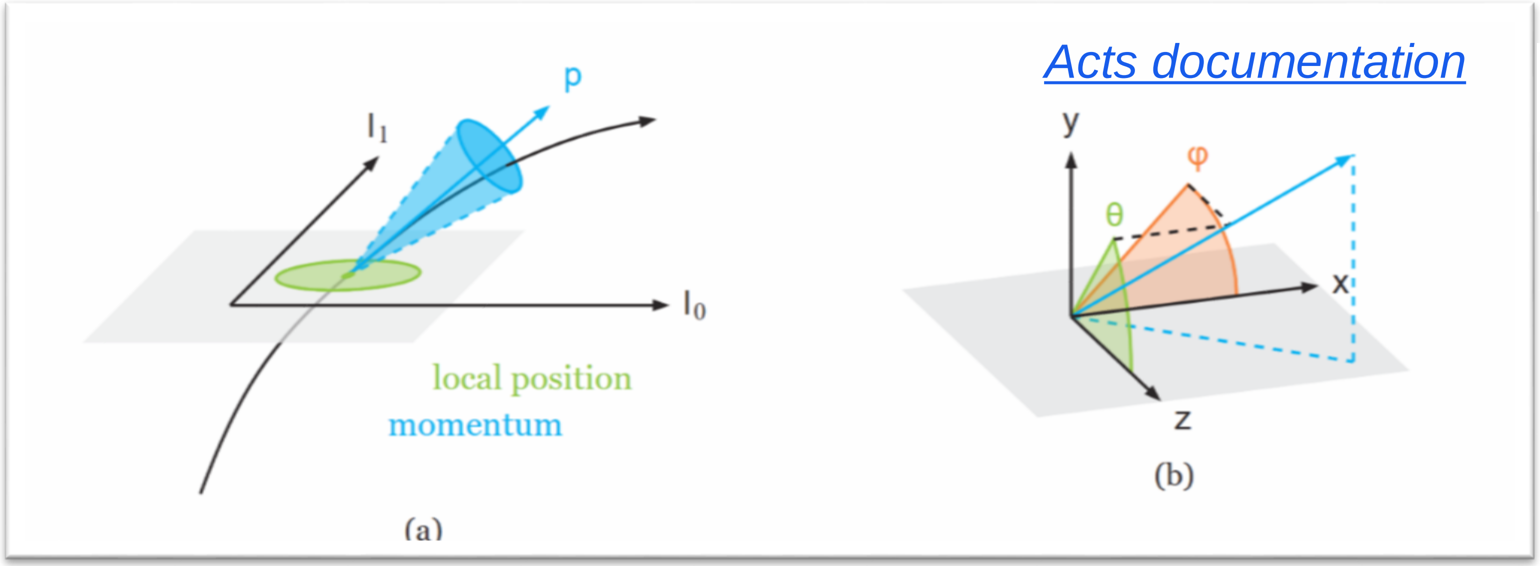
ACTS track parameterization

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

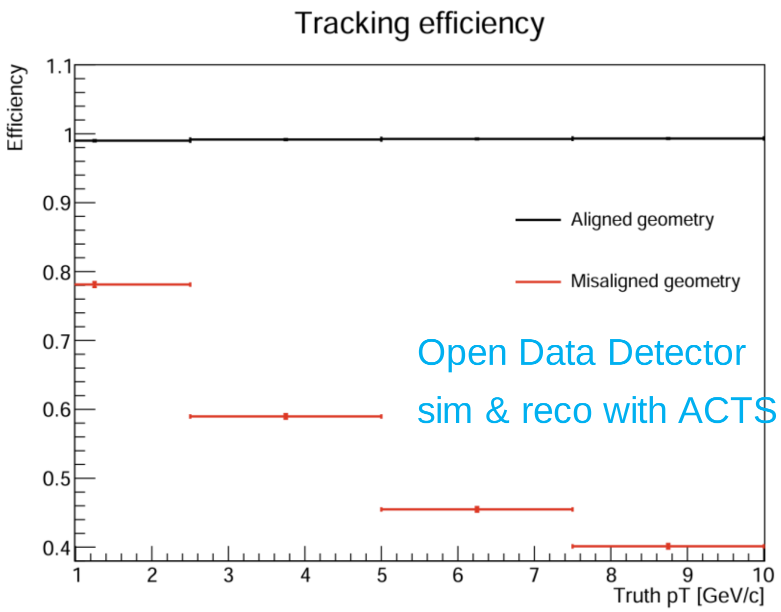
Alignment in ACTS

Why alignment matters?

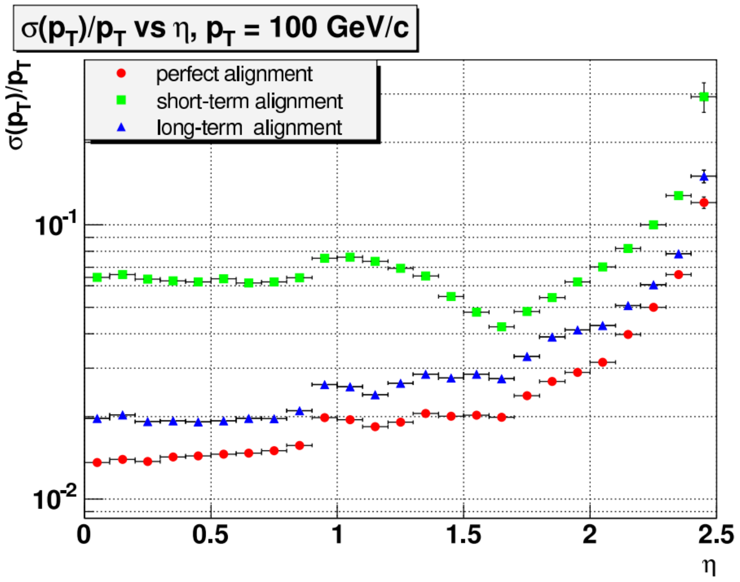
See more in P. Bruckman's talk

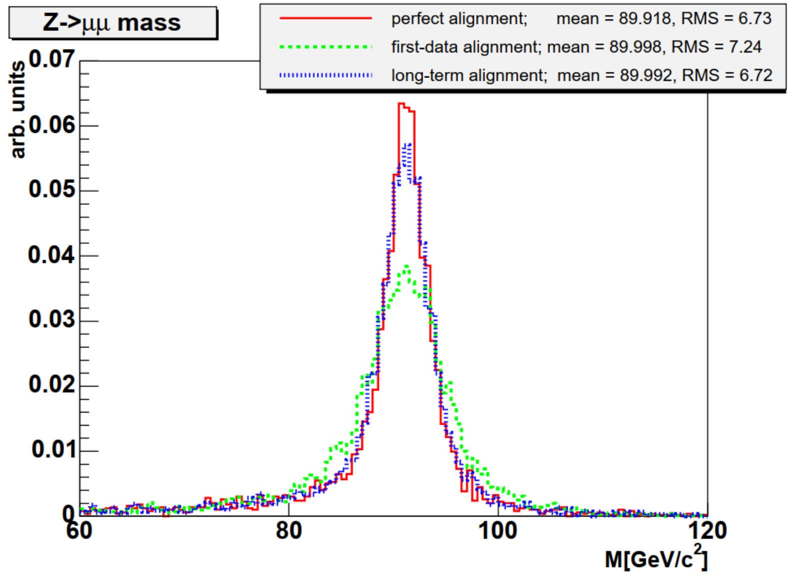
- Limited detector placement precision upon installation
- Misalignment is the dominant source of measurement resolution degradation

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



Open Data Detector sim & reco with ACTS





From G. Steinbruck

Alignment Parameters

- Detector element placement description:
  - Translation (3 parameters) + Rotation (3x3 rotation matrix)
- Alignment parameters (6 parameters for now):
  - Translation + 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

Sequencer
creates local AlgorithmContext context
(including geoContext, magFieldContext, calibContext)

⇩

Alignment decorator

decorate(AlgorithmContext& context) {

Generates and owns IOV indexed alignmentStore

context.geoContext = Acts::GeometryContext(alignmentContext);

}

⇩

alg0.execute(AlgorithmContext& context)

⇩

…

⇩

algN.execute(AlgorithmContext& context)

generates

retrieve

AlignmentStore
Store identifier indexed transform

construct

AlignmentContext
AlignmentContext
alignmentContext(&alignmentStore)

set geoContext

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 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

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

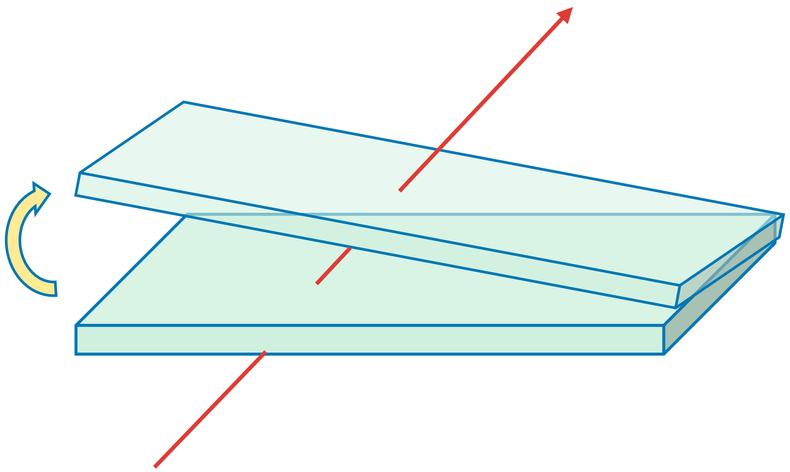
$$\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\alpha^2} \;=\; 2A^T V^{-1}\left(V - HCH^T\right)V^{-1} A.$$

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

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

Residual derivative

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

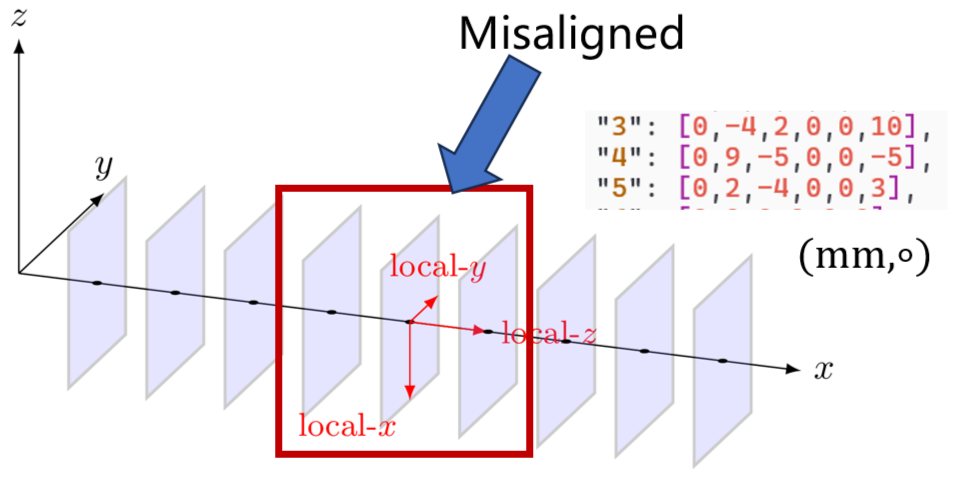
- It takes a set of tracks and sort out the sets of detector elements which can be and requested to be aligned => the initial value of $\vec{\alpha}$
- Perform the fit for each track using $\vec{\alpha}$, and estimate the χ^2 derivatives w.r.t. $\vec{\alpha}$
- Solve the equation to obtain $\Delta\vec{\alpha}$
- 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;
```

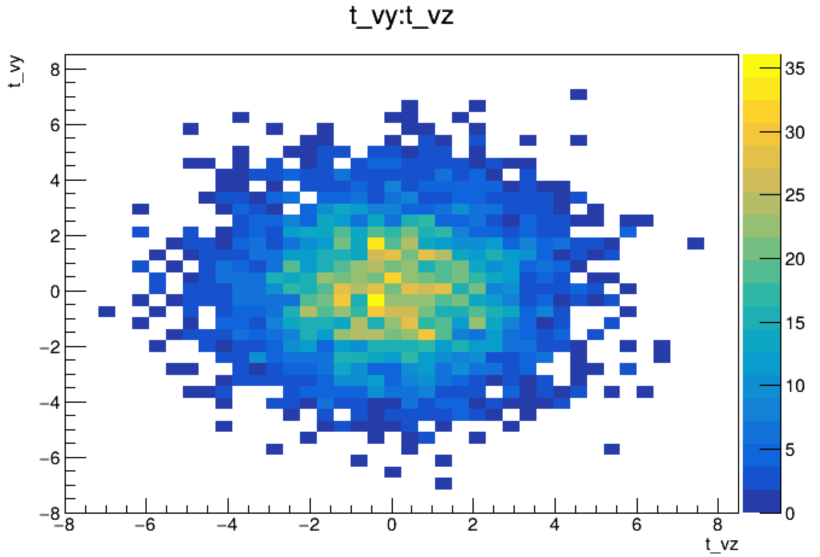
ACTS Alignment example

Sanity check with a telescope-like detector

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

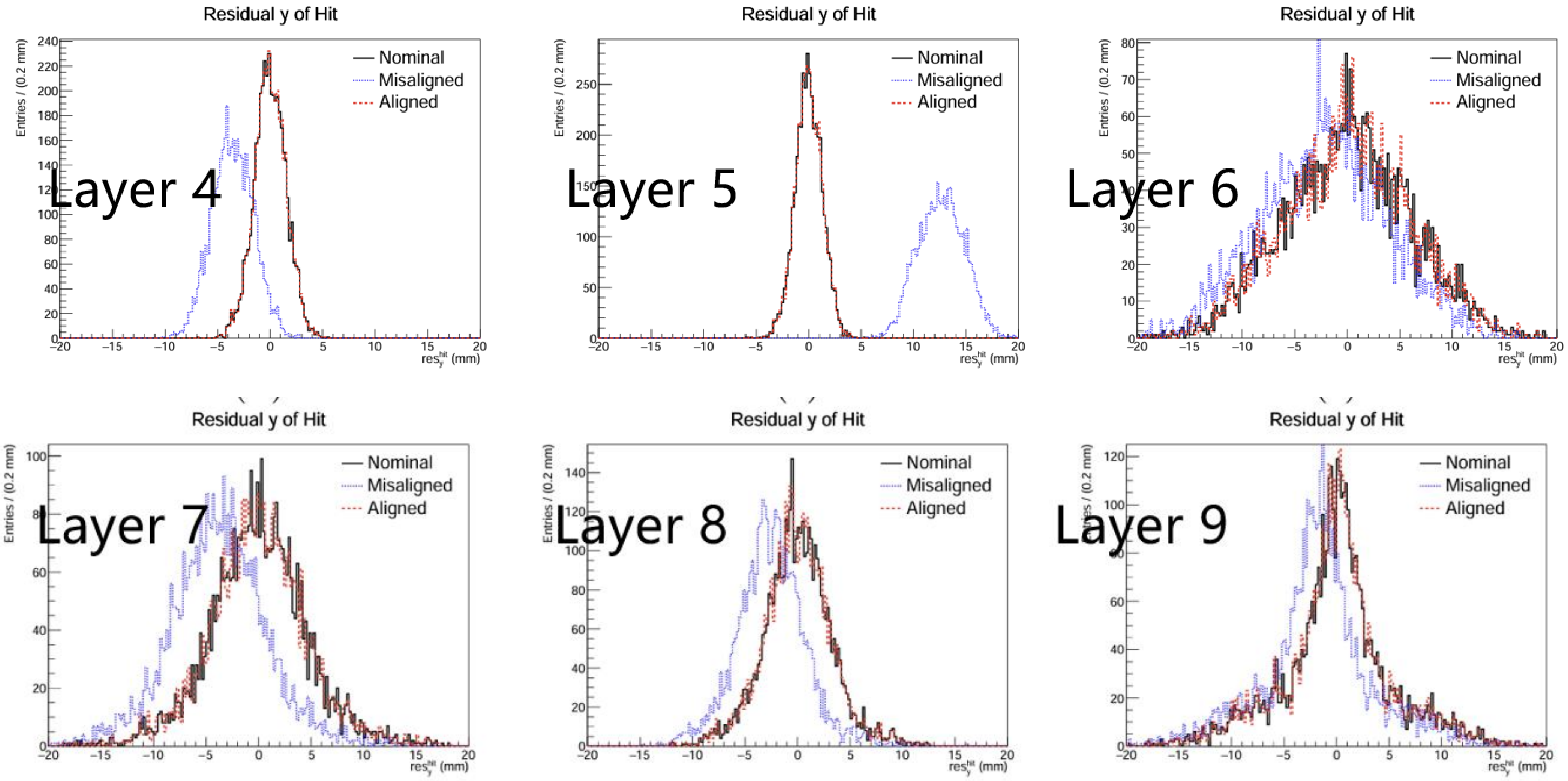


25 x 25 um2 resolution



Residual with misalignment + aligned

Credit to Z. Sun for the plots



Summary

- ACTS is a tracking software, but also provides ingredients for alignment
  - Validated with a toy telescope-like detector. Validation with more realistic Open Data Detector is in progress
  - Only surface/module-wise for now. Superstructure alignment will come soon
- 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