Study of a new algorithm for tracker alignment using Machine Learning

Pawel Bruckman De Renstrom [1], David Brunner [2], Thorsten Kuhl [3]
[1]Polish Academy of Sciences
[2]Stockholm University
[3]Deutsches Elektronen-Synchrotron

8th International Connecting The Dots Workshop
11.10.2023

Stockholm University
ATLAS EXPERIMENT

Concept of misalignment

taken from [Images for CMS Tracker alignment in Run 2]

Calibration of the tracker with offline software:

General idea of calibration algorithm

Starting geometry, e.g. design geometry, is assumed and track fit is done.

↓

Define target function $F(\mathbf{d}, \mathbf{t})$ in terms of detector parameter $\mathbf{d}$ and track parameter $\mathbf{t}$.

↓

Minimise $F(\mathbf{d}, \mathbf{t})$ and update position detector modules.

↓

With new geometry of the detector, perform the track fits again and update track parameter $\mathbf{t}$.

With MillePede:

- Define $F(\mathbf{d}, \mathbf{t})$ as χ^2
  $$\chi^2(\mathbf{d}, \mathbf{t}) = \sum_j^{\text{tracks}} \sum_i^{\text{hits}} \left(\frac{m_{ij} - f_{ij}(\mathbf{d}, \mathbf{t}_j)}{\sigma_{ij}^m} \right)^2$$
- Perform linearisation $\mathbf{d} \rightarrow \mathbf{d} + \Delta\mathbf{d}$ and $\mathbf{t} \rightarrow \mathbf{t} + \Delta\mathbf{t}$
- Problem can be reduced to solving the matrix equation $\mathbf{C} \times \Delta\mathbf{d} = \mathbf{b}$

Machine learning for minimisation procedure

Proposed algorithm using machine learning libraries (MLAligner)

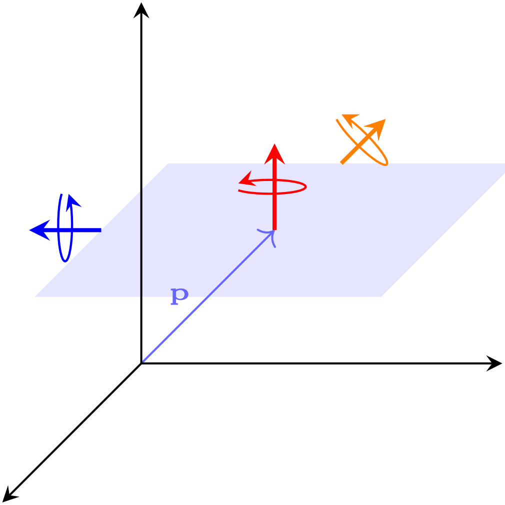
- Usage of TensorFlow/PyTorch/etc. for the minimisation procedure
- Parameterise detector and tracks as trainable tensors
- The loss function will be $F(\mathbf{d}, \mathbf{t})$, parameterised as a χ^2

  $\chi^2(\mathbf{d}, \mathbf{t}) = \sum_j^{\text{tracks in batch}} \sum_i^{\text{hits}} \left(\frac{m_{ij} - f_{ij}(\mathbf{d}, \mathbf{t}_j)}{\sigma_{ij}^m}\right)^2$
- Implement standard minimisation ML loop, update detector parameter each batch and refit tracks in each batch

Advantages of this approach

- Usage of modern ML interface everybody is familiar with
- Straightforward implementation
- Low maintenance time/cost
- Running on GPUs is a trivial task

Parametrisation of the detector in MLAligner



- For each module six free parameters
- Three parameter for the vector to the center of the module, for now in Cartesian coordinates
- Three Euler angles, with the parametrisation convention x-z'-x" intrinsic rotations
- Planned for later: Parameter for global structures

→ Each parameter will be a tensor, therefore "trainable"

Study using ACTS

Detector simulation

- Generic detector using fast simulation (Fatras)
- Constant B-Field in z direction with 2 T
- Module level misalignment with gaussian smearings
  - Displacement of module center: $\sigma_x^c = \sigma_y^c = 100\ \mu\text{m}$, $\sigma_z^c = 50\ \mu\text{m}$
  - Rotational displacement around local axes: $\sigma_x^{\text{rot}} = \sigma_y^{\text{rot}} = 20$ mrad

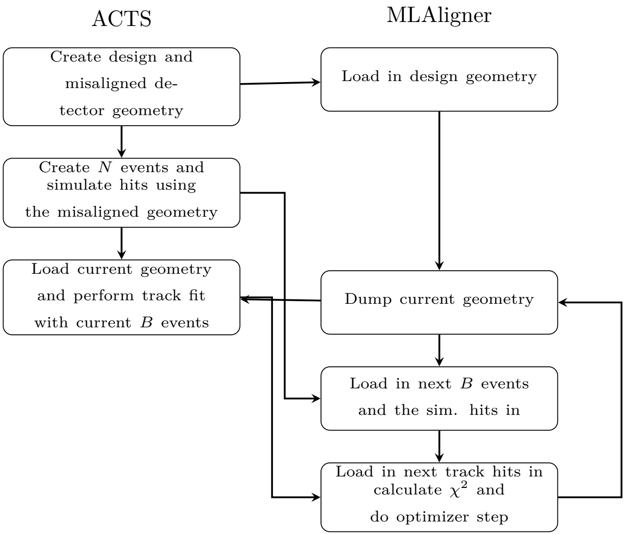
Particle simulation

- Single muon events produced with a particle gun
- $p_T \in (1\text{ GeV}, 30\text{ GeV}), \phi \in (0, 2\pi), \eta \in (-2.5, 2.5)$
- Origin in the center of the detector

Seeding and tracking

- Standard (non-ML) seeding algorithm
- Tracking with Combinatorial Kalman Filter

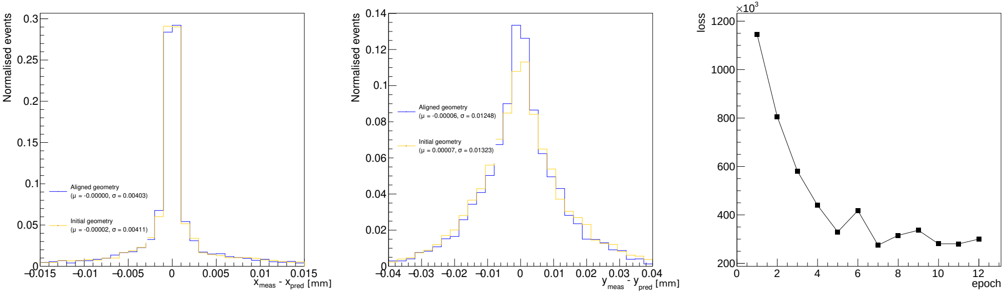
Full training algorithm of ACTS and MLAligner



Facts about the training

- 80000 single muon events simulated
- Each batch contains 200 events $\rightarrow$ 400 total batches
- Adam Optimizer with a learning rate of 10^{-4}
- PyTorch used as backend ML library

Look at first training output



More questions than answers

- No improvement in x residuals, but in y residuals?
- Network converged after six epochs, nothing more to gain in terms of better residuals?
- What exactly has been learned (in which direction moved the modules?)

Runtime breakdown

- Tested on CERN's lxplus with a single core execution
- Timed with std::chrono::high_resolution_clock from C++ standard library
- Following time stamps measured per batch

```
'Dump detector' Execution Time: 1061 ms
'Track fit' Execution Time: 1948 ms
'Load measured/track hits' Execution Time: 31 ms
'Track hits 3D to 2D' Execution Time: 154 ms
'Calc loss' Execution Time: 0 ms
'Backprop' Execution Time: 292 ms
'Update module parameter' Execution Time: 58 ms
```

Conclusion

Achieved so far

- New method for the alignment of a tracker system is presented
- Study based on simulation and tracking done by ACTS
- Build first iteration of core framework
- Complete set up is working and first training is shown

List of things to do

- Study how/in which direction modules moved (weak modes?)
- Validate performance with other quantities besides residuals
- Use more realistic simulation set up
- Compare with MillePede
- ...

Tack för er uppmärksamhet!

(Thanks for your attention!)

Contact

David Brunner
david.brunner@cern.ch
david.brunner@fysik.su.se