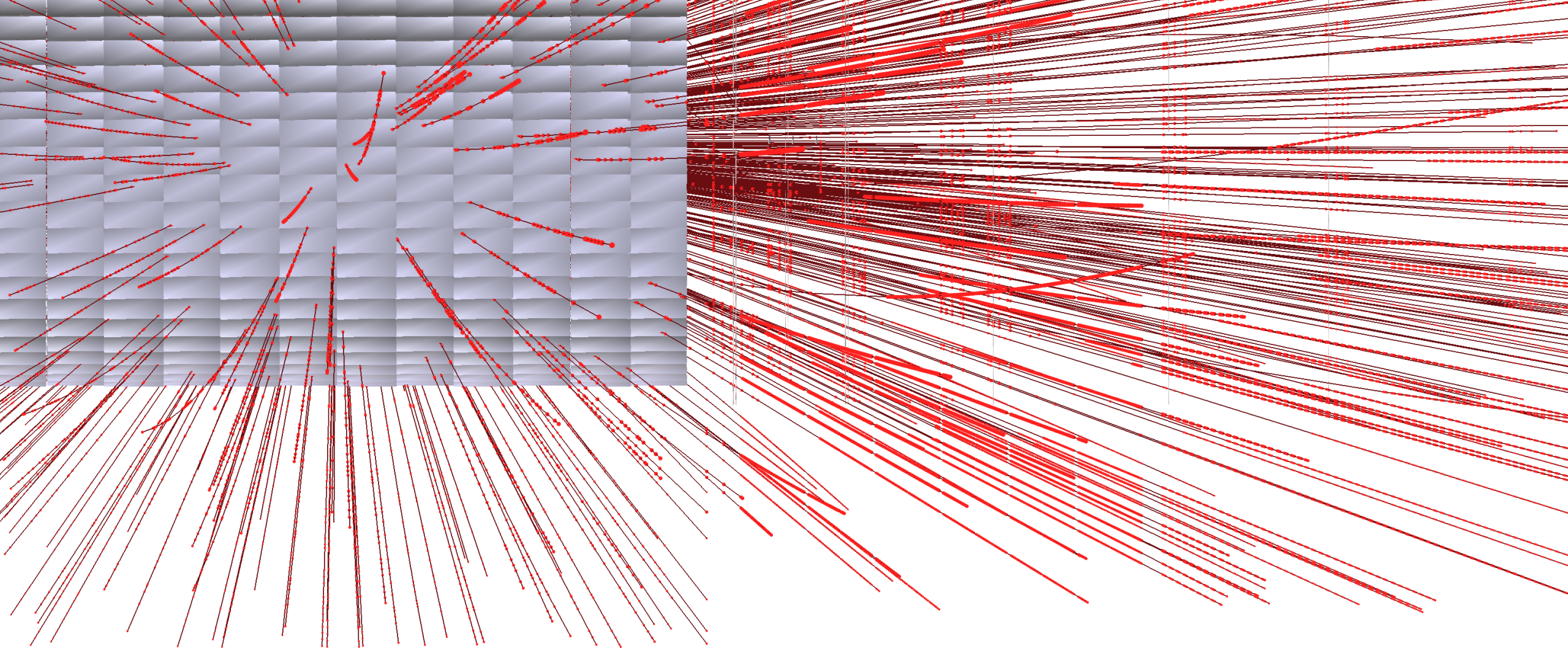


Particle tracking and the Acts project

Paul Gessinger

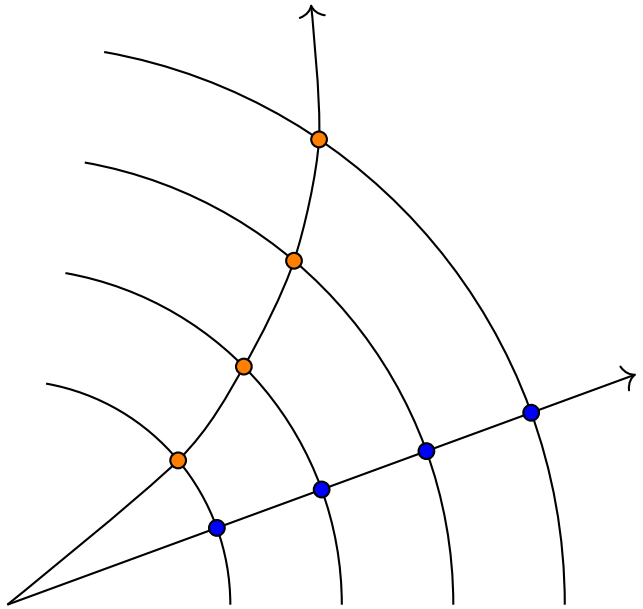
CSC 2018 – Tel Aviv - Student session

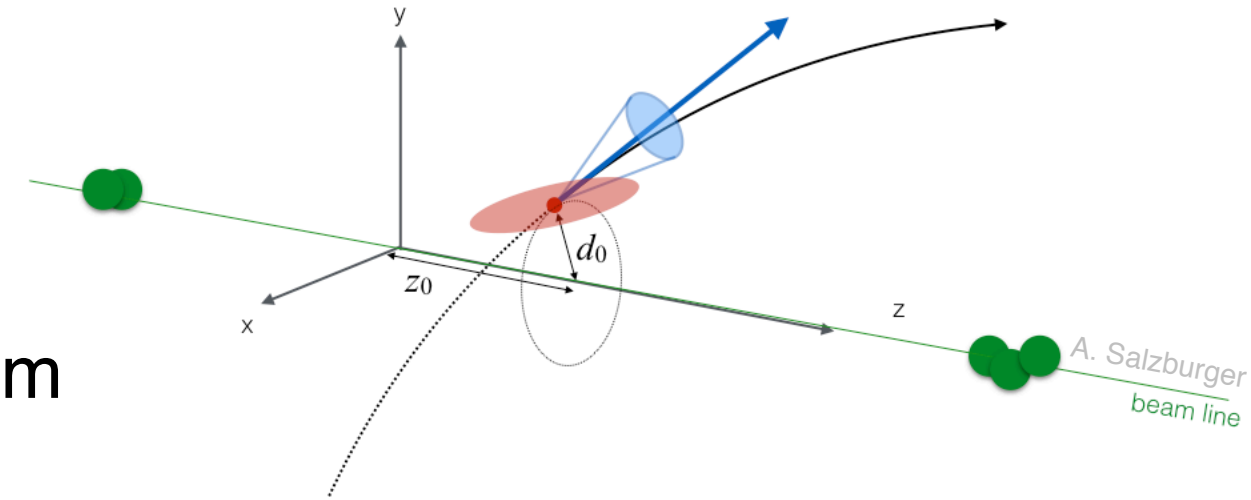
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
FSP 103 ATLAS

What is particle tracking?

What is particle tracking? (in a nutshell)

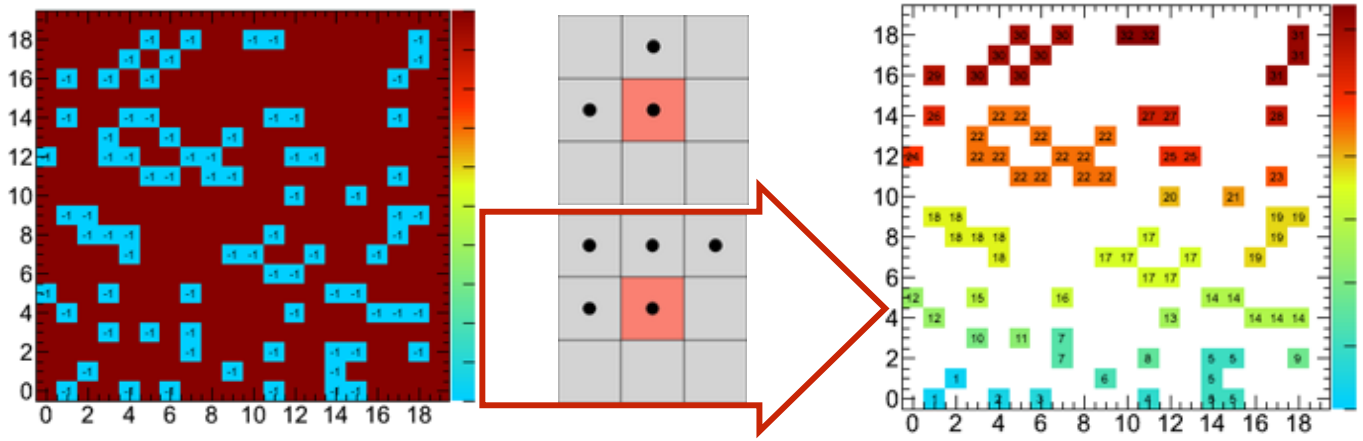
- Particle collisions in collider experiments produce particles
- We want to measure properties of the produced particles
  - Energy, momentum, vertex position, charge, mass, type…
- Charged particles: where do they "go"?
- Interaction with track detectors as charged particles pass through them (e.g. silicon)





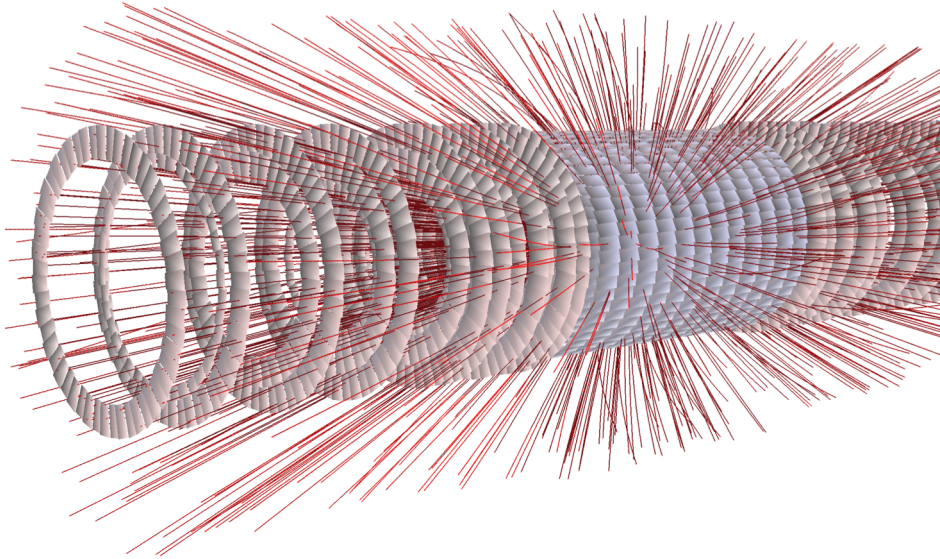
What do we measure?

- Different for different detector types
- Example: silicon detectors

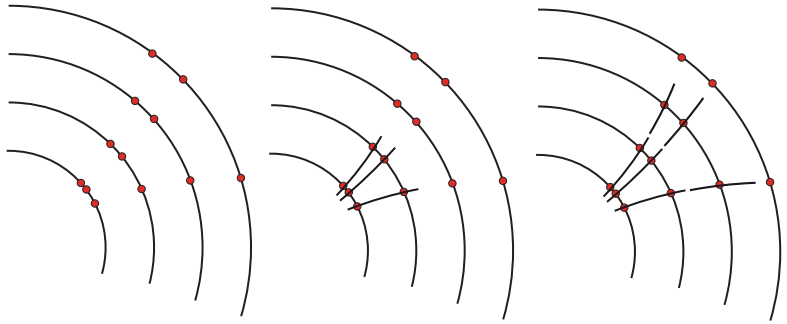


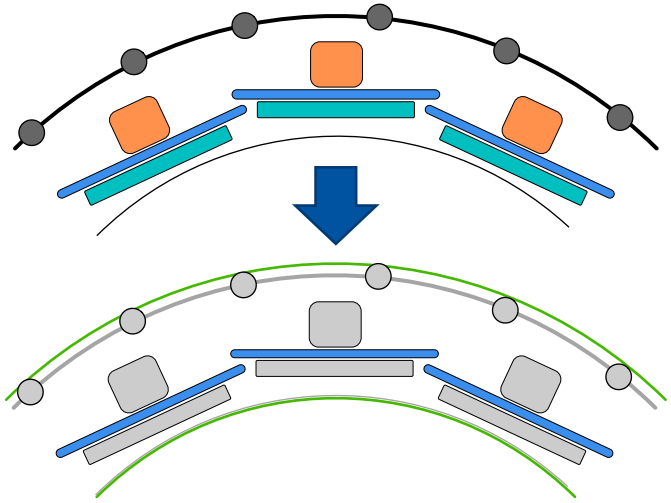
- Charge deposit in cells (1D or 2D), binary or non-binary
- Connected component analysis results in **clusters**
- Cluster has position (on sensitive surface), and an uncertainty

From measurements to tracks

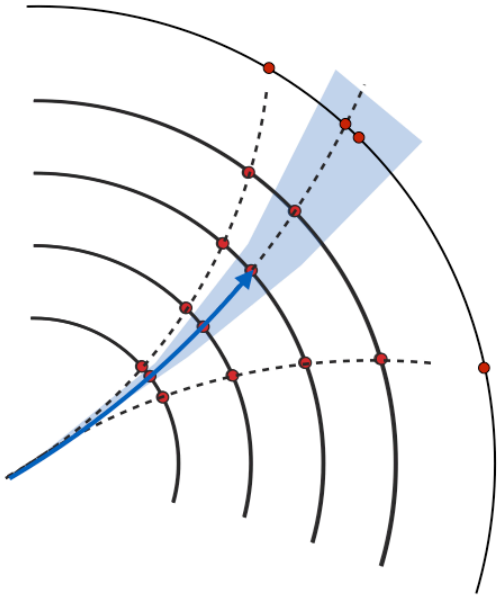


- Common approach: **seeding and following**
- First: create space points from all the measurements (clusters)
- Create “seeds” from space points





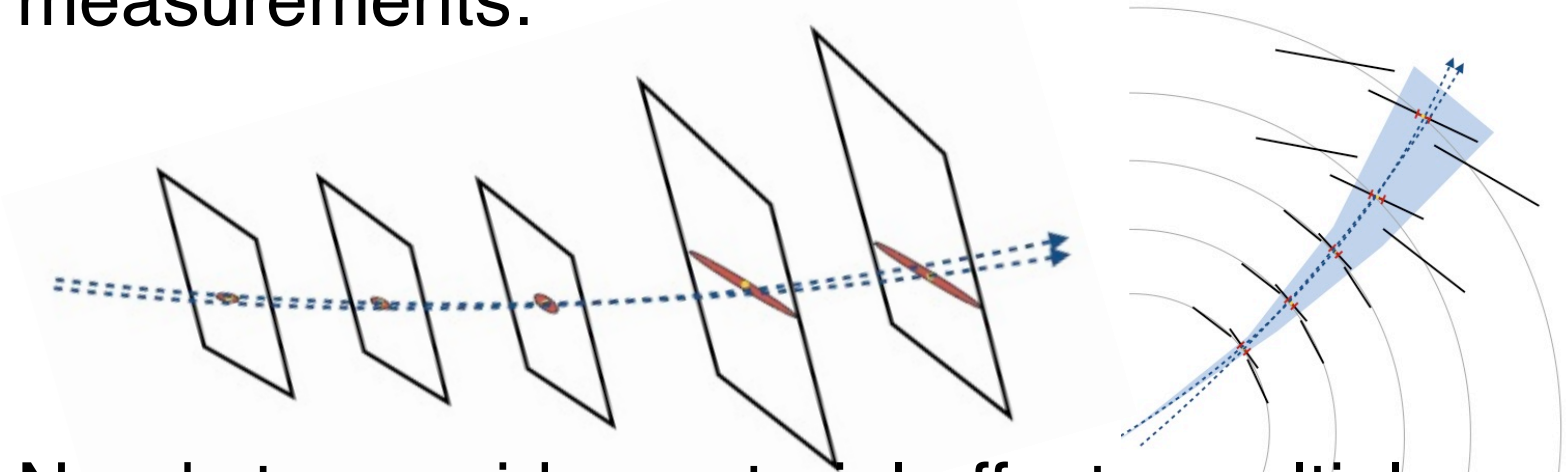
- Build track candidates (quickly becomes non-trivial), combinatorics, ambiguities



A. Salzburger

From measurements to tracks /2

- Once you have **track candidates**: proper fit of track to measurements:



- Needs to consider material effects: multiple scattering, energy loss, hadronic interactions, propagate through B-Field
- Global χ^2-fit, huge matrix inversion
- Kalman fitter: iteratively *updates* the track parameters considering measurements

The Acts project

The Acts project

- Standalone tracking library, in development
- Based on ATLAS tracking library
- Design goals
  - Experiment agnostic
  - Thread-safe
  - Modern code style (modern C++)
  - Well documented
  - Rigorously tested (unit & integration)
  - Maintainable in the long term
- Source is public (MPL): https://gitlab.cern.ch/acts/acts-core

Concurrency and thread safety

- All tools (and lots of other things) are `const`
- Mutable state is passed in function calls

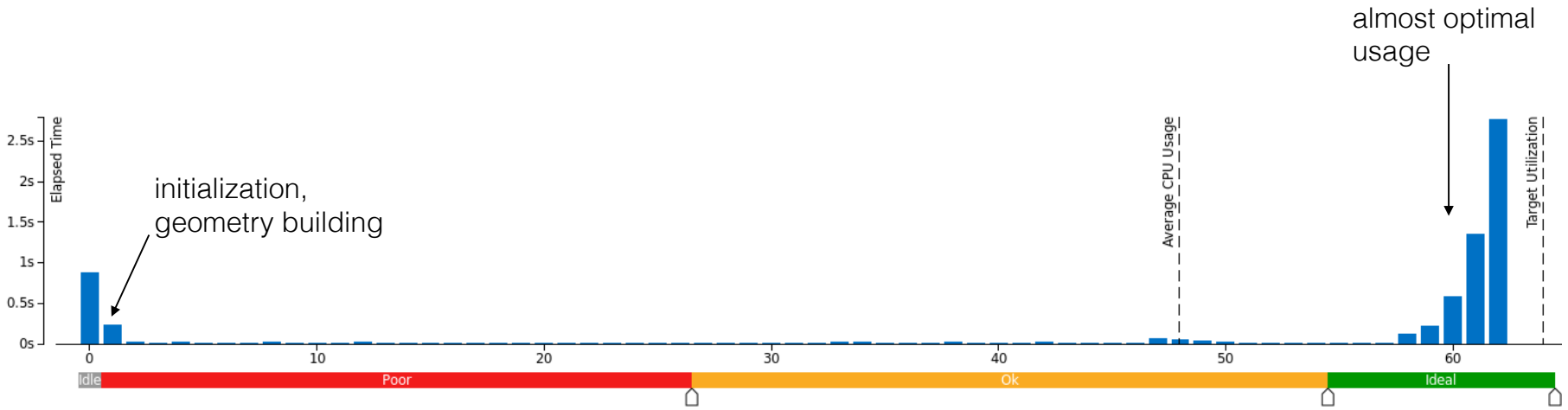
```
// create extrapolation cell with parameter type T
// and start parameters
Acts::ExtrapolationCell<T> ecc(startParameters);
// configure the cell
ecc.searchMode = -1; // only compatible surfaces
ecc.addConfigurationMode(Acts::ExtrapolationMode::StopAtBoundary);
ecc.addConfigurationMode(Acts::ExtrapolationMode::CollectSensitive);
ecc.addConfigurationMode(Acts::ExtrapolationMode::CollectPassive);
// ...
Acts::ExtrapolationCode eCode = extrapolationEngine->extrapolate(ecc);
```

```
ExtrapolationCode
extrapolate(ExCellCharged&       ecCharged, /* ... */) const final;
```

- State is thread local: saves you a lot of headaches

Aspects of the project

- Parallelization works well:



- Lots of components are designed with extensibility in mind
  - E.g. new propagator: can switch out almost all parts to adapt to different setups
  - For instance: implement Kalman Fitter as a plugin to the propagator
- Configurable, reusable, documented Seedfinder with no (= very few) magic numbers
- Integration testing in ATLAS

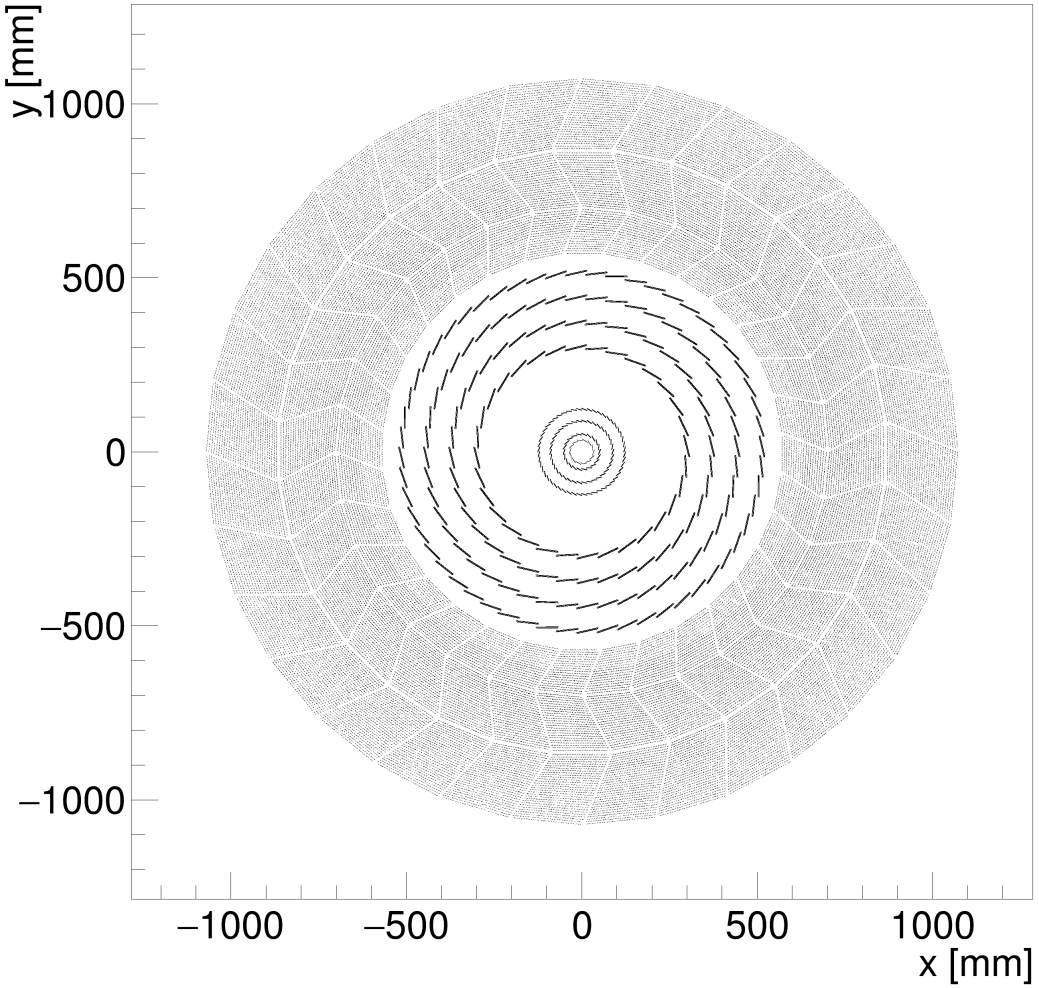
(Quick) Summary

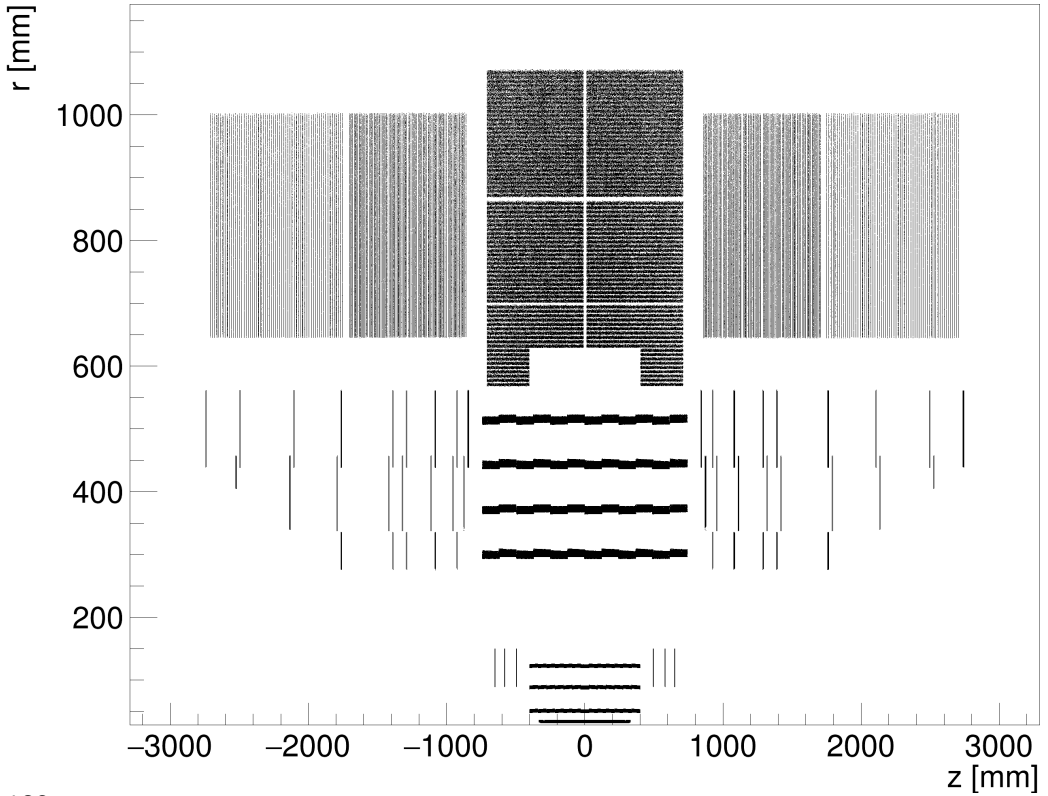
- Particle tracking is an essential part of event reconstruction
- It's composed of multiple steps, some of which have parallelization opportunities

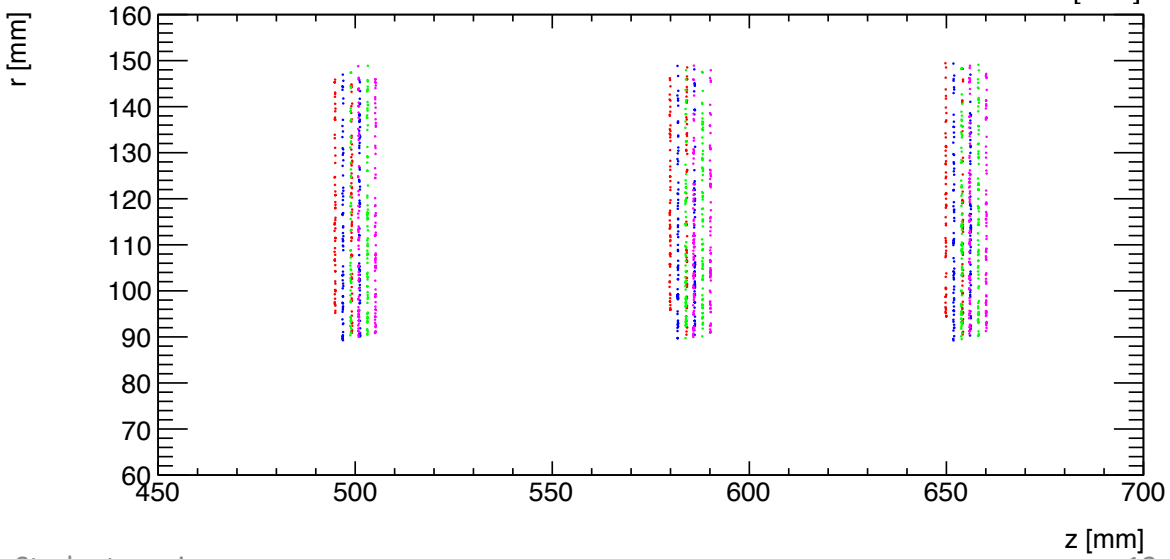
- **Acts** is an ongoing project to build a reusable, thread-safe tracking toolkit
  - Experiment agnostic, and highly configurable

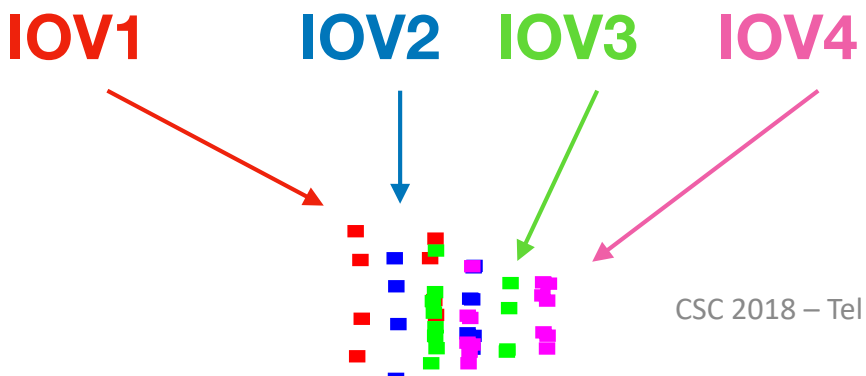
- If you're interested: http://cern.ch/acts

Backup


y [mm]
1000
500
0
−500
−1000
−1000
−500
0
500
1000
x [mm]


r [mm]
1000
800
600
400
200
−3000
−2000
−1000
0
1000
2000
3000
z [mm]


r [mm]
160
150
140
130
120
110
100
90
80
70
60
450
500
550
600
650
700
z [mm]


IOV1
IOV2
IOV3
IOV4