Tracking with Hashing

Current state

Jeremy Couthures

Generic Detector

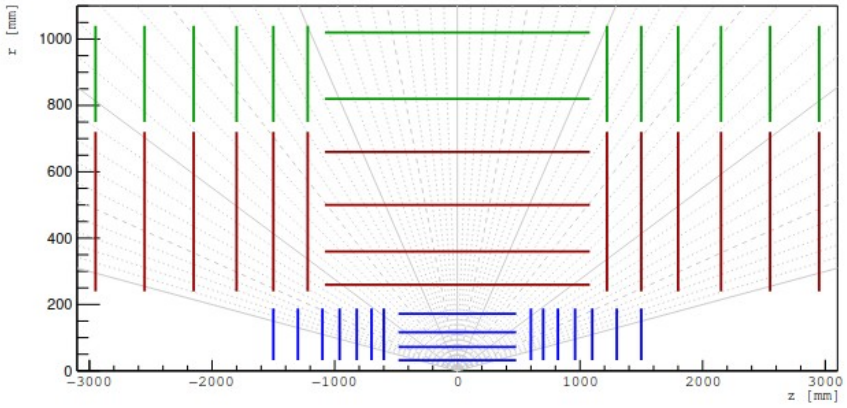
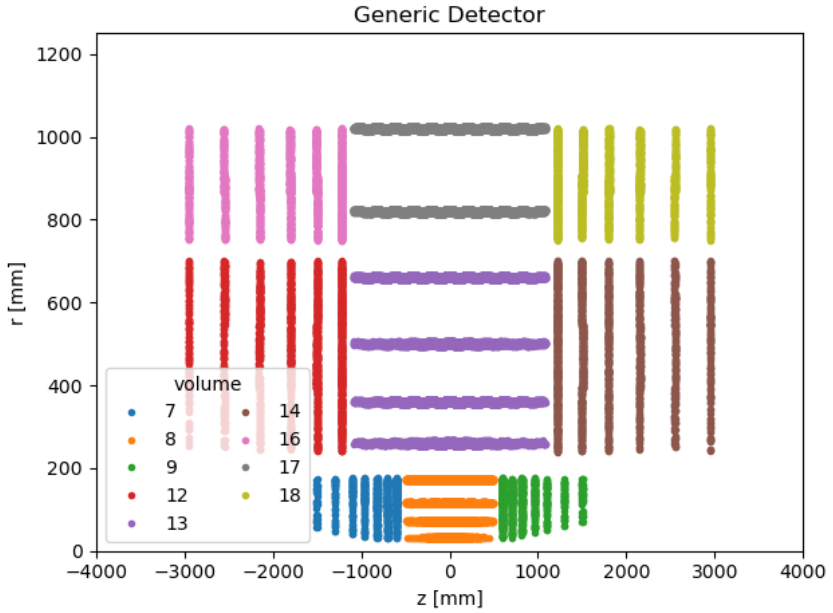


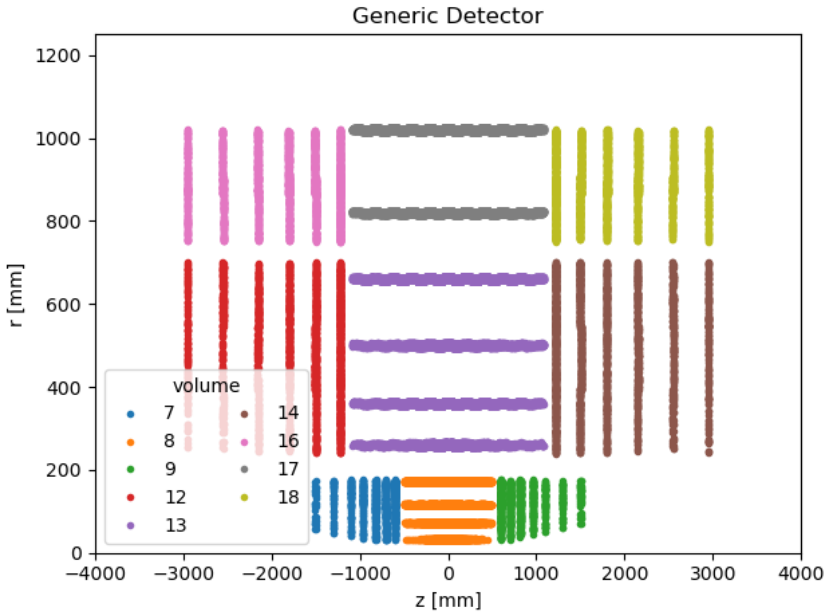
Fig. 1 Sketch of the TrackML detector as used in both the "Accuracy" and "Throughput" phase. Vertical lines indicate disks while horizontal lines indicate cylinders, all with the z axis as axis of revolution. Three different sub detectors build the overall detector setup: a central pixel system (blue), enclosed by first a short strip (red) and then a long strip detector (green).

https://arxiv.org/pdf/2105.01160.pdf



https://github.com/acts-project/acts/issues/926

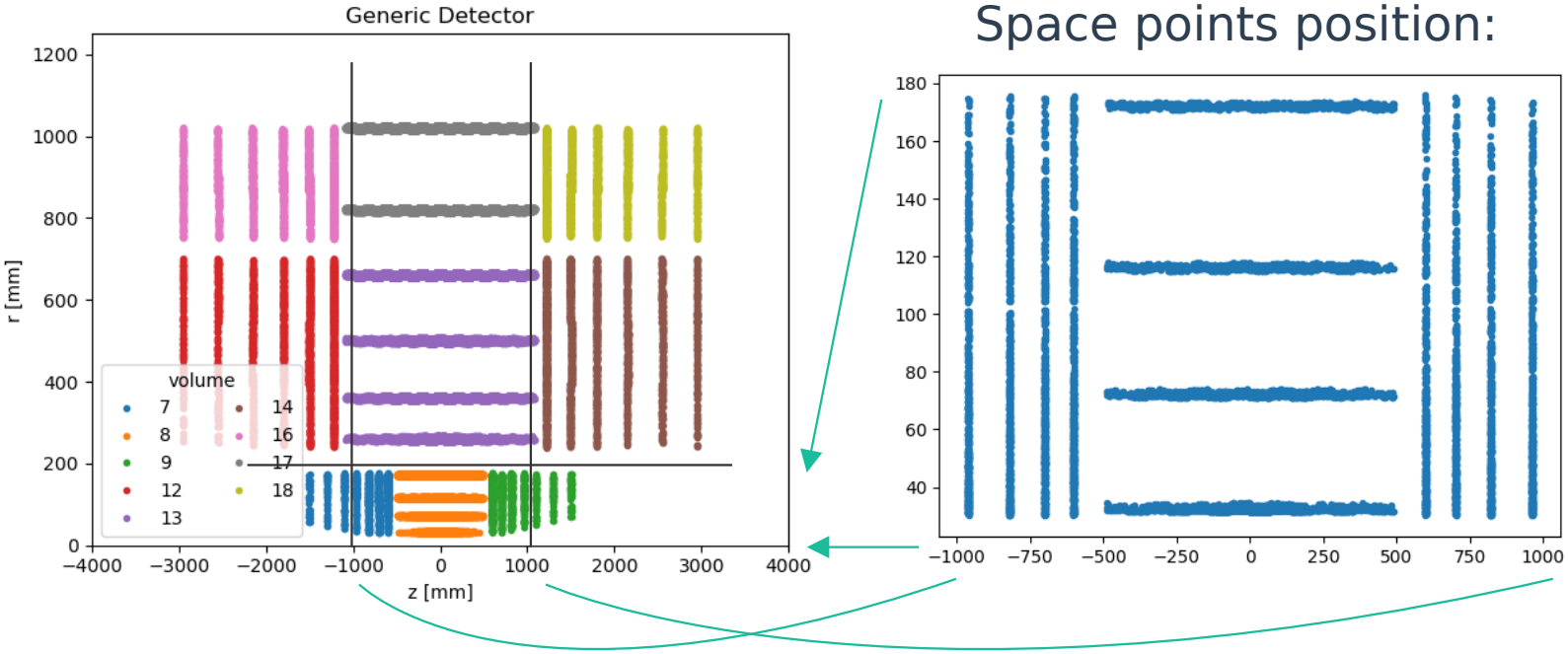
Generic Detector volumes



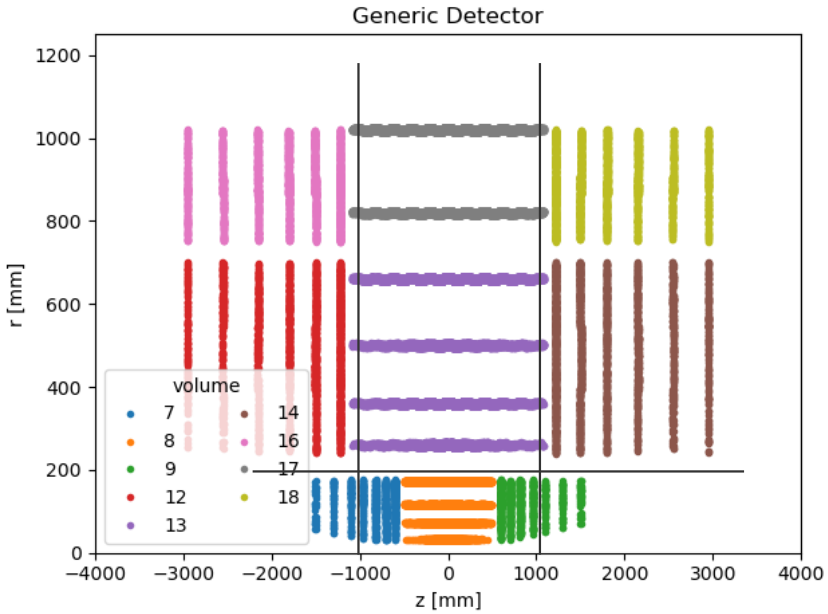
volumeID=vol=1: { { { { BeamPipe::Barrel | { Pixel::NegativeEndcap | Pixel::Barrel | Pixel::PositiveEndcap } } | PST::Barrel } | { SStrip::NegativeEndcap | SStrip::Barrel | SStrip::PositiveEndcap } } | { LStrip::NegativeEndcap | LStrip::Barrel | LStrip::PositiveEndcap } }
volumeID=vol=2: { { { BeamPipe::Barrel | { Pixel::NegativeEndcap | Pixel::Barrel | Pixel::PositiveEndcap } } | PST::Barrel } | { SStrip::NegativeEndcap | SStrip::Barrel | SStrip::PositiveEndcap } }
volumeID=vol=3: { { BeamPipe::Barrel | { Pixel::NegativeEndcap | Pixel::Barrel | Pixel::PositiveEndcap } } | PST::Barrel }
volumeID=vol=4: { BeamPipe::Barrel | { Pixel::NegativeEndcap | Pixel::Barrel | Pixel::PositiveEndcap } }
volumeID=vol=5: BeamPipe::Barrel
volumeID=vol=6: { Pixel::NegativeEndcap | Pixel::Barrel | Pixel::PositiveEndcap }
volumeID=vol=7: Pixel::NegativeEndcap
volumeID=vol=8: Pixel::Barrel
volumeID=vol=9: Pixel::PositiveEndcap
volumeID=vol=10: PST::Barrel
volumeID=vol=11: { SStrip::NegativeEndcap | SStrip::Barrel | SStrip::PositiveEndcap }
volumeID=vol=12: SStrip::NegativeEndcap
volumeID=vol=13: SStrip::Barrel
volumeID=vol=14: SStrip::PositiveEndcap
volumeID=vol=15: { LStrip::NegativeEndcap | LStrip::Barrel | LStrip::PositiveEndcap }
volumeID=vol=16: LStrip::NegativeEndcap
volumeID=vol=17: LStrip::Barrel
volumeID=vol=18: LStrip::PositiveEndcap

https://github.com/acts-project/acts/issues/946#issuecomment-900504099

Generic Detector: Space point position



Generic Detector: Space point formation



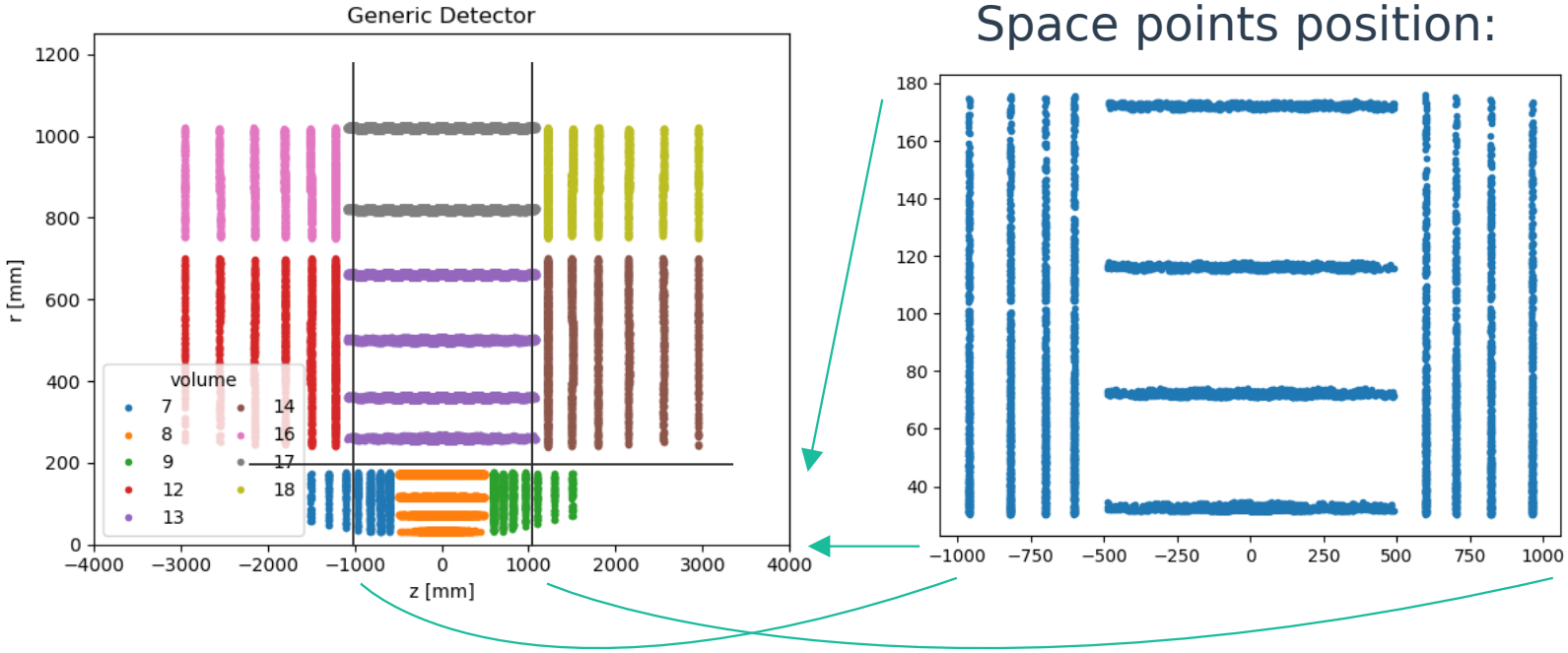
SP formation on geometry:

[{"layer": 2,"volume": 8},
{"layer": 4,"volume": 8},
{"layer": 6,"volume": 8},
{"layer": 8,"volume": 8},
{"layer": 2,"volume": 9},
{"layer": 4,"volume": 9},
{"layer": 6,"volume": 9},
{"layer": 8,"volume": 9},
{"layer": 14,"volume": 7},
{"layer": 12,"volume": 7},
{"layer": 10,"volume": 7},
{"layer": 8,"volume": 7}]

geoSelection-genericDetector.json

Layer numbering: 2 by 2, starts at 2; in a volume, increases from left to right and bottom to top

Generic Detector: SP buckets



Hashing currently uses only Pixel Space Points

SpacePointMaker: strip issue?

SpacePointMaker :

' // TODO Support strip measurements'

SpacePointMaker.cpp

Not finished yet in ACTS

Use measurements with Hashing?

ODD

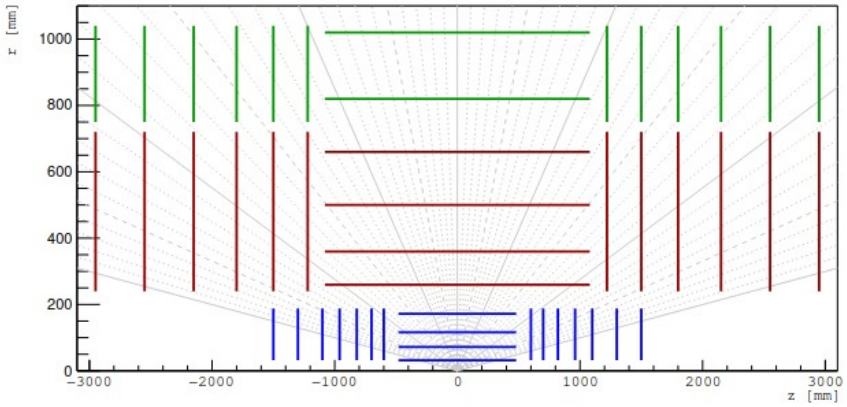
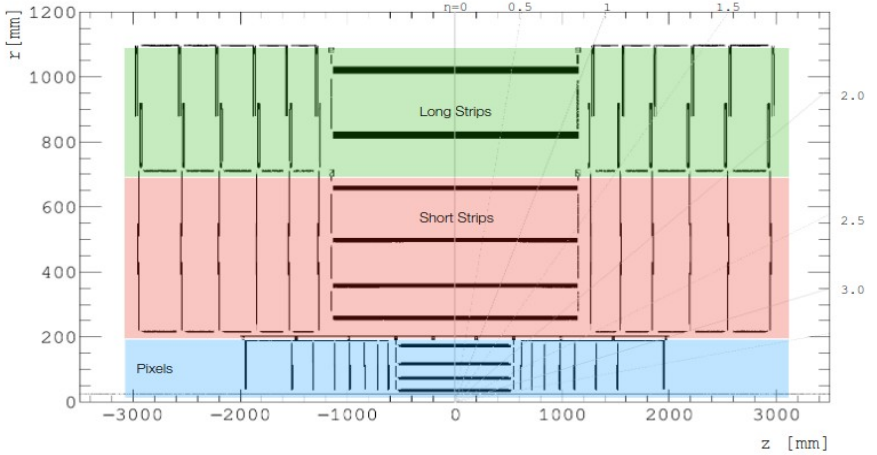


Fig. 1 Sketch of the TrackML detector as used in both the "Accuracy" and "Throughput" phase. Vertical lines indicate disks while horizontal lines indicate cylinders, all with the z axis as axis of revolution. Three different sub detectors build the overall detector setup: a central pixel system (blue), enclosed by first a short strip (red) and then a long strip detector (green).

https://arxiv.org/pdf/2105.01160.pdf



`Geant4` hit map showing the sensitive and passive material structures in the ODD in the rz-plane. The solenoid is omitted in this picture.

ODD Poster

ODD coverage

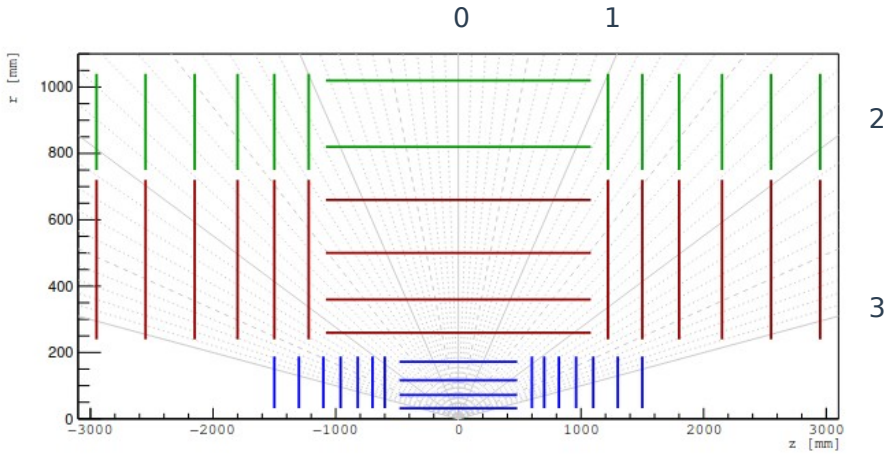
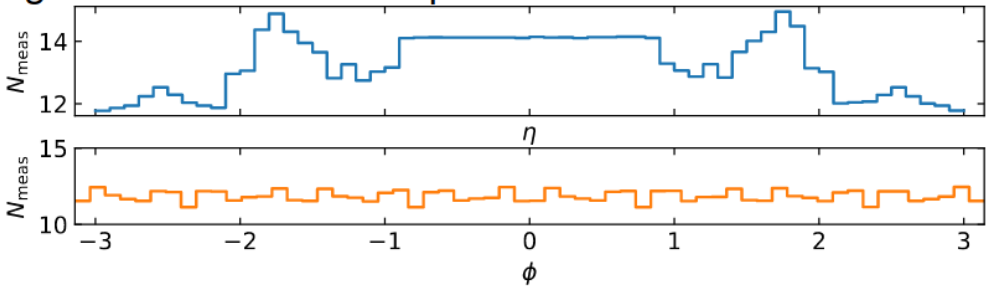


Fig. 1 Sketch of the TrackML detector as used in both the "Accuracy" and "Throughput" phase. Vertical lines indicate disks while horizontal lines indicate cylinders, all with the z axis as axis of revolution. Three different sub detectors build the overall detector setup: a central pixel system (blue), enclosed by first a short strip (red) and then a long strip detector (green).

https://arxiv.org/pdf/2105.01160.pdf

The ODD describes a fully hermetic detector with pseudo-rapidity coverage of $|\eta| < 3$. At full efficiency, it guarantees at least 12 measurements per track, which is enhanced by significant module overlap.



ODD Poster

Exa.TrkX

Model

The Exa.TrkX pipeline is a three-stage GNN-based algorithm:

1. **Graph building** (*Embedding stage*): This is done via metric learning approach. A neural network tries to learn a mapping that minimizes the distance between points of the same track in the embedded space. In this embedded space then a graph is built using a fixed nearest-neighbor search.
2. **Graph size reduction** (*Filter stage*): The resulting graph is typically to large for processing with a GNN. Therefore a binary classifier reduces the graph size further and removes easy to identify not matching edges.
3. **Edge classification** (*GNN stage*): Finally, a GNN is used to find the edges in the Graph that belong to tracks. This is done by scoring each edge with a number between 0 and 1, and a corresponding *edge cut*.

Embedding

emb_dim	8
emb_hidden	1024
knn	50
max_epochs	20
nb_layer	4
points_per_batch	130000
r_{train, test, val}	0.2
true_edges	modulewise
regime	[rp, hnm, norm]
id	292tlcta

https://acts.readthedocs.io/en/latest/plugins/exatrkx.html#exatrkxplugin

Slides Benjamin Huth exatkrkx acts