GNN-based pipeline for track finding in the Velo at LHCb

Anthony Correia, on behalf of the LHCb collaboration

10th October 2023

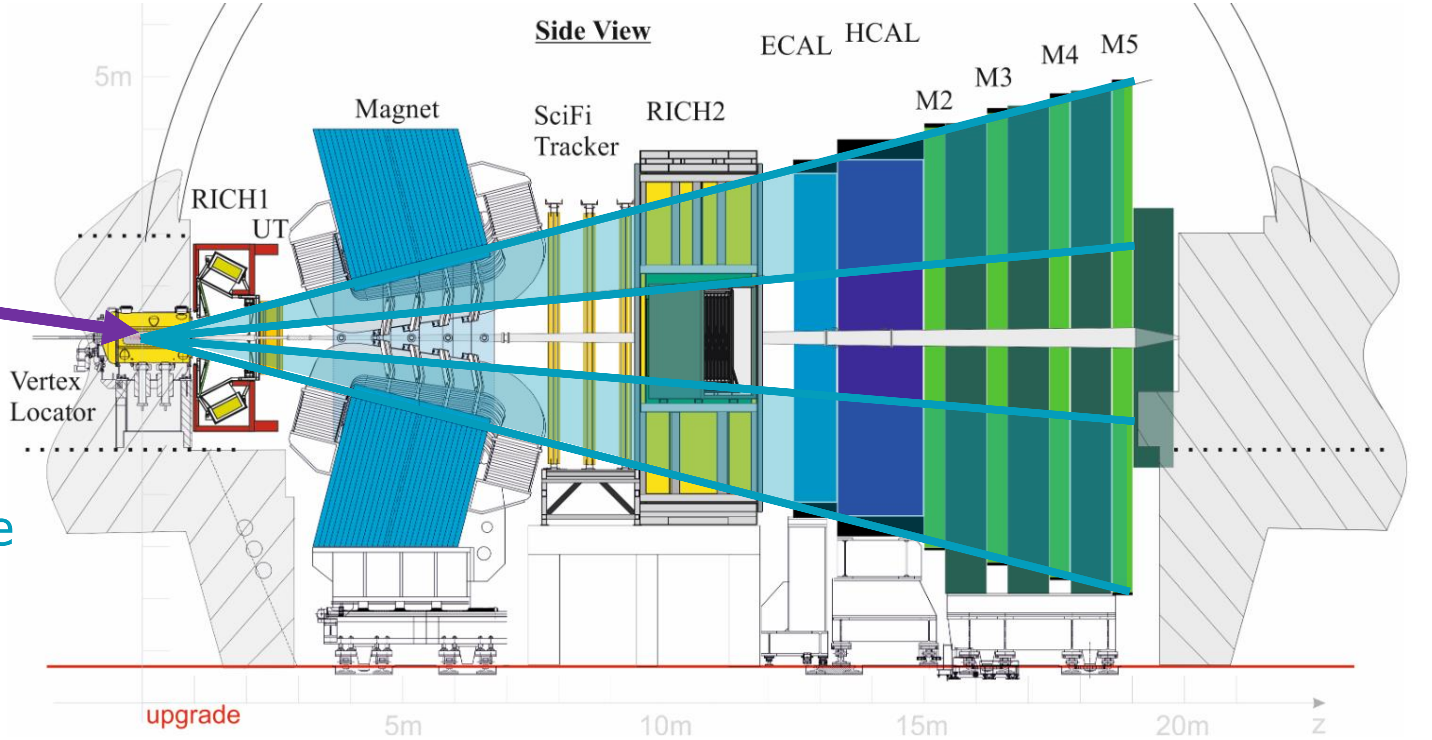
In collaboration with

0. LHCb Detector in Run 3

Collisions and Trigger

Collisions (Run 3)

- $20\ \mathrm{MHz}$ non-empty bunch crossing rate
- p-p collision at $\sqrt{s} = 13{,}6\ \mathrm{TeV}$
- $\sim$ 5 collisions / bunch crossing



[J. Phys.: Conf. Ser., vol. 878, p. 012012, 2017](#)

LHCb Subdetectors

Acceptance $2 < \eta < 5$

5 TB / s

Allen (High-Level Trigger 1)
fully GPU-based online partial reconstruction and selection

Better trigger efficiency than previous HLT1 FPGA-based trigger

70-200 GB / s

Storage buffer

High-Level Trigger 2
CPU-based full reconstruction and selection

10 GB/s

Numbers taken from [LHCB-FIGURE-2020-016](#)

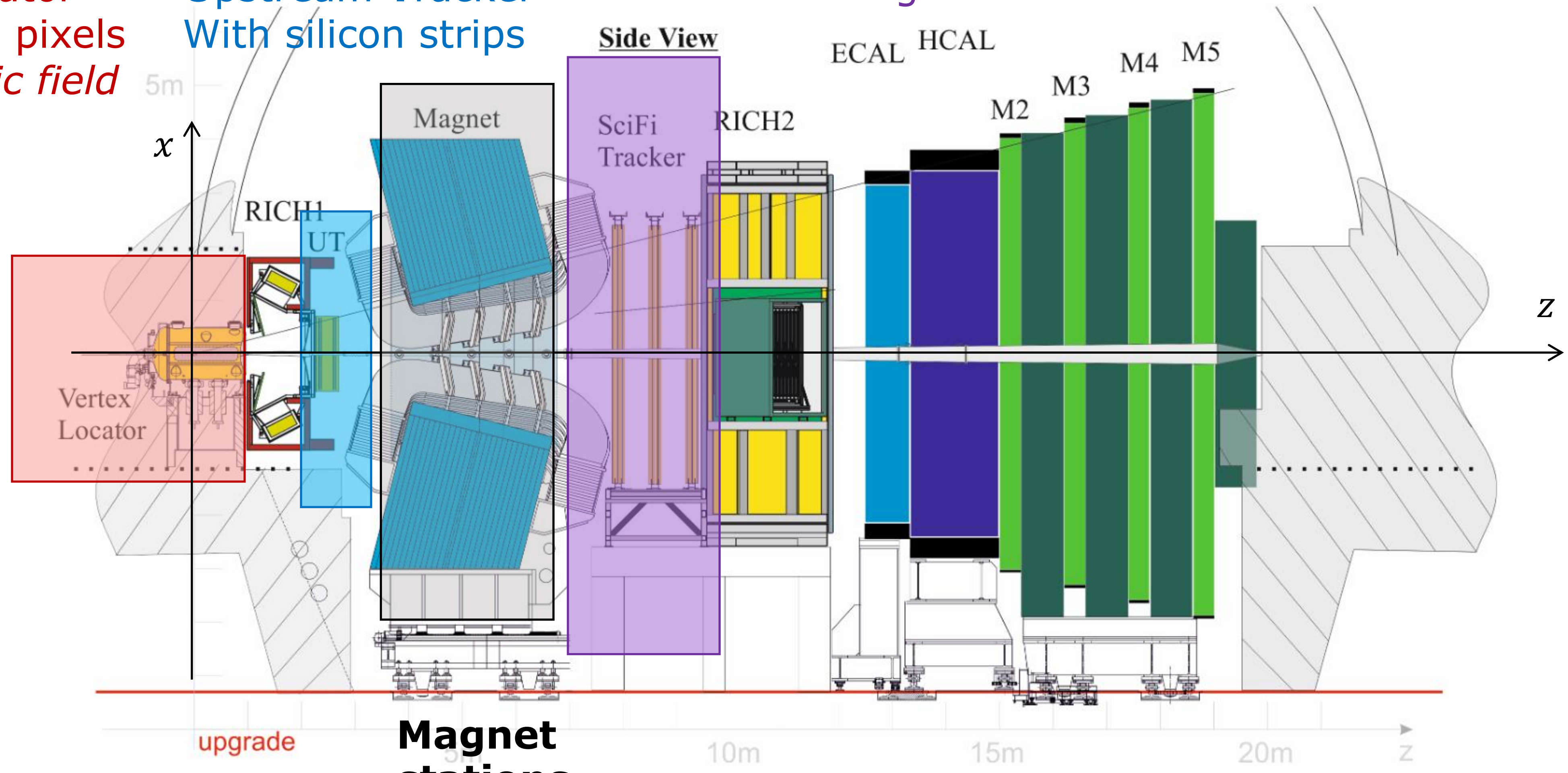
0. LHCb Detector in Run 3

3 Tracking detectors

Velo
Vertex **Lo**cator
With silicon pixels
No magnetic field

UT
Upstream **T**racker
With silicon strips

SciFi
With **Sci**ntillating **Fi**bres

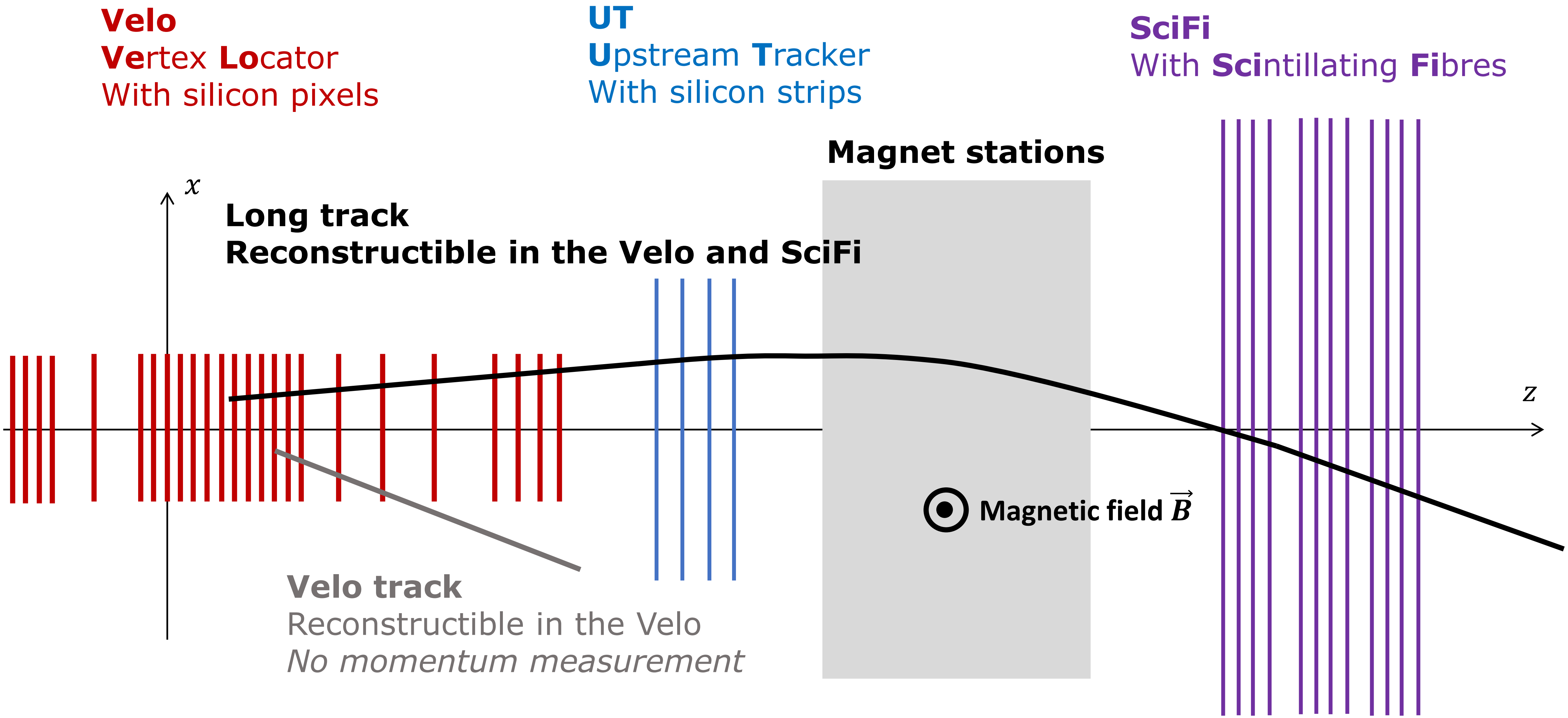


Magnet stations

J. Phys.: Conf. Ser., vol. 878, p. 012012, 2017

0. LHCb Detector in Run 3

Tracks



1. Graph Neural Network Track Finding

Motivations

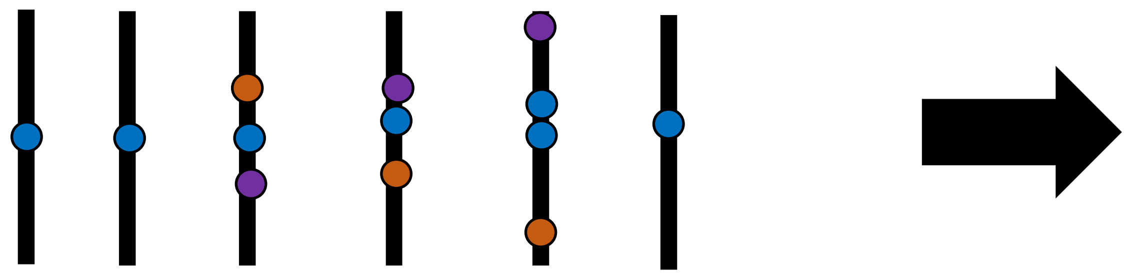
Graph Neural Network (GNN)-based track-finding pipeline based on the work of **Exa.Trkx** (*Eur. Phys. J. C* **81**, 876 (2021))

- Demonstrated **near-linear** inference time w.r.t. # hits
  - *Conventional* algorithms are worse-than-quadratic
  - Increase in instaneneous luminosity in future upgrades over the next decade
    → need for **even more high-throughput** track-finding algorithms

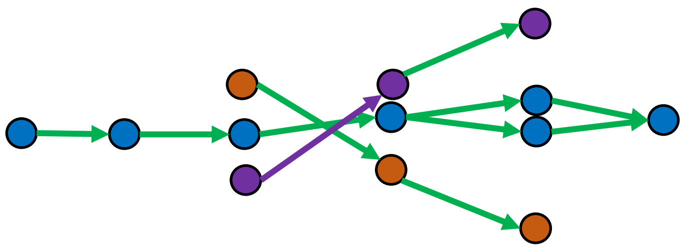
- **High-parallelisation** potential → compatible with current **GPU-based Allen** trigger

- Future implementation in Allen ⇒ allow **like-for-like comparison** with conventional algorithms

- Representation of tracks with a graph quite *natural*



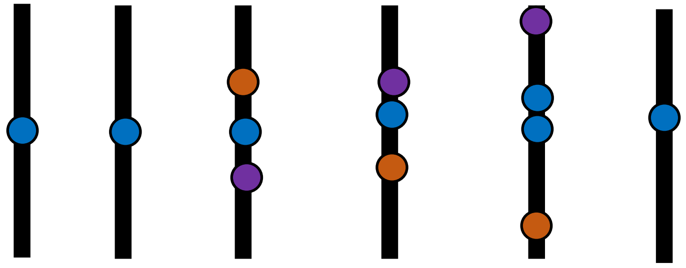
Pure graph representation



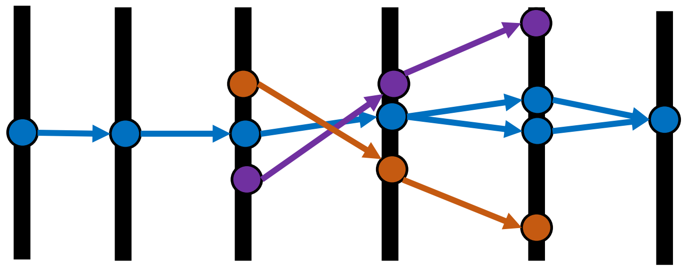
1. Graph Neural Network Track Finding

Goal

Input: Velo Hits

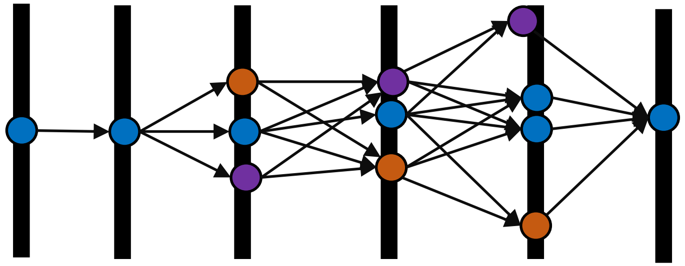


Output: Velo tracks

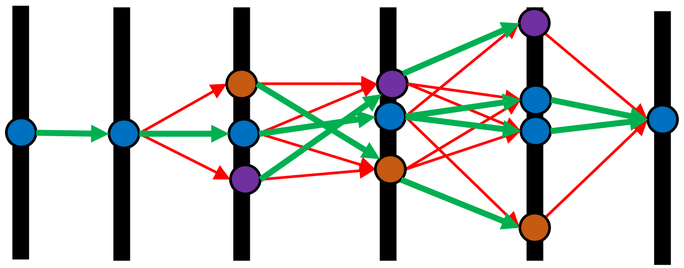


Strategy

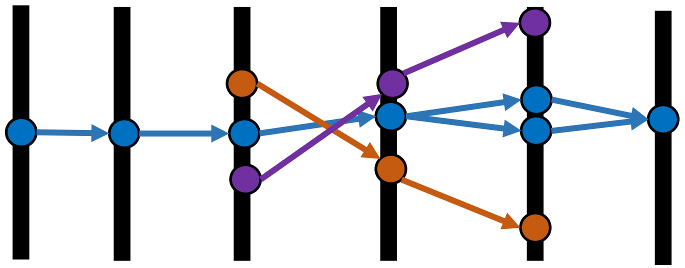
1. Build a **"rough" graph**
   - Embedding Network + Nearest-Neighbour Network



2. **Classify the edges** as genuine or fake
   - Graph Neural Network



3. Identify **connected hits**
   - Weakly Connected Component Algorithm



1. Graph Neural Network Track Finding

Graph Building → GNN: filter edges → Build tracks from graph

(as developed by the Exa.TrkX collaboration)

1. Graph Neural Network Track Finding

Graph Building → GNN: filter edges → Build tracks from graph

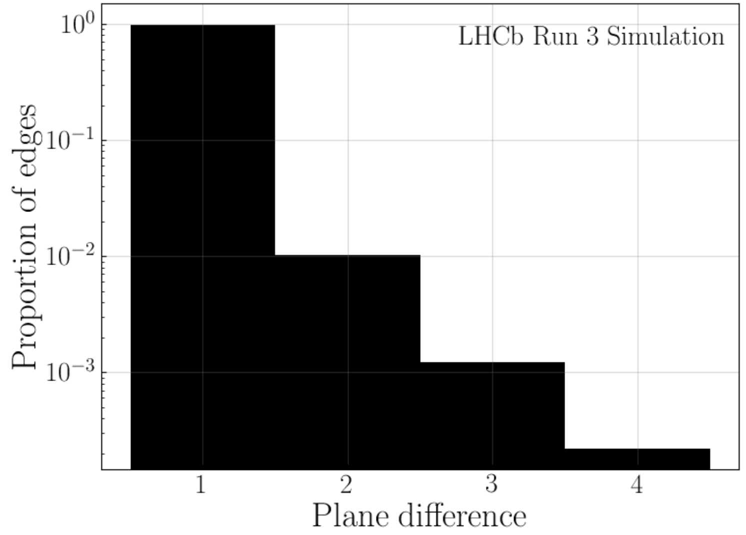
1. Graph Neural Network Track Finding

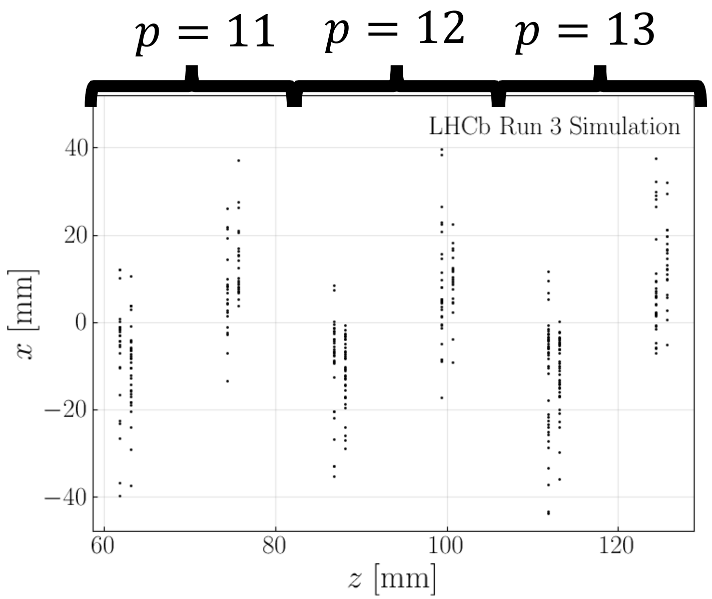
Graph Building → GNN: filter edges → Build tracks from graph

- **Goal**: **minimise # edges** while **maximising edge efficiency**

- **Hypothesis**
  - 99% of edges are 1-plane apart, 1% are 2-plane apart
    ⇒ **allow for only 1 skipped plane** (~1%)
  - Only build edges **from left to right**

Change w.r.t. Exa.Trkx





- *For every hit in plane p, how to connect it to hits belonging to the next 2 planes $p+1$ and $p+2$?*

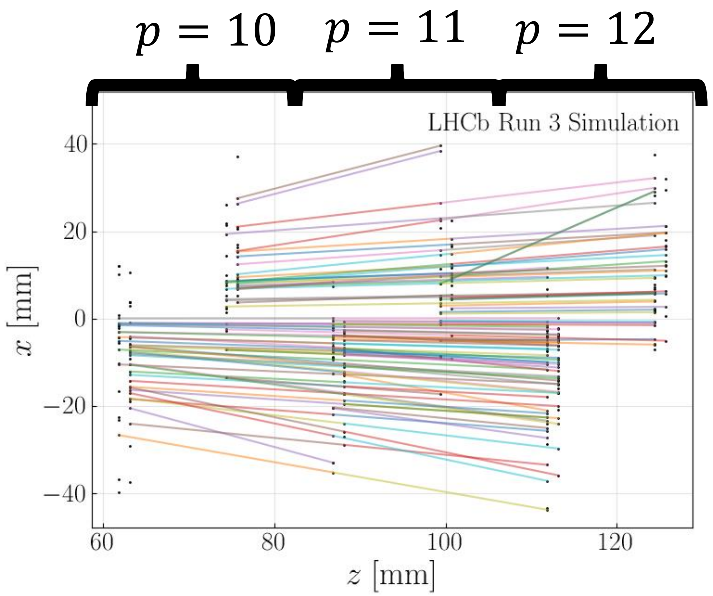
1. Graph Neural Network Track Finding

Graph Building → GNN: filter edges → Build tracks from graph

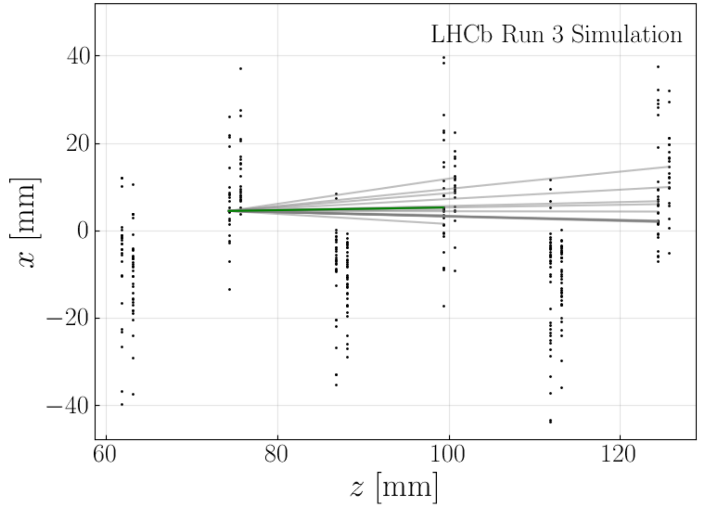
Edges are not random

- Forward
- Away from $z \leftrightarrow$ more tilted

$\Rightarrow$ this features could be learnt by a Neural Network



True edges



Example of edges drawn in the rough graph

1. Graph Neural Network Track Finding

Graph Building → GNN: filter edges → Build tracks from graph

1 **Embed every point in an embedded space**

Parallelise over hits

Cylindrical coordinates

$(r, \phi, z, \text{plane})$ → Dense Neural Network (DNN) **35K** parameters → $\vec{e} = (e_1, e_2, e_3, e_4)$

DNN trained so that in the embedding space

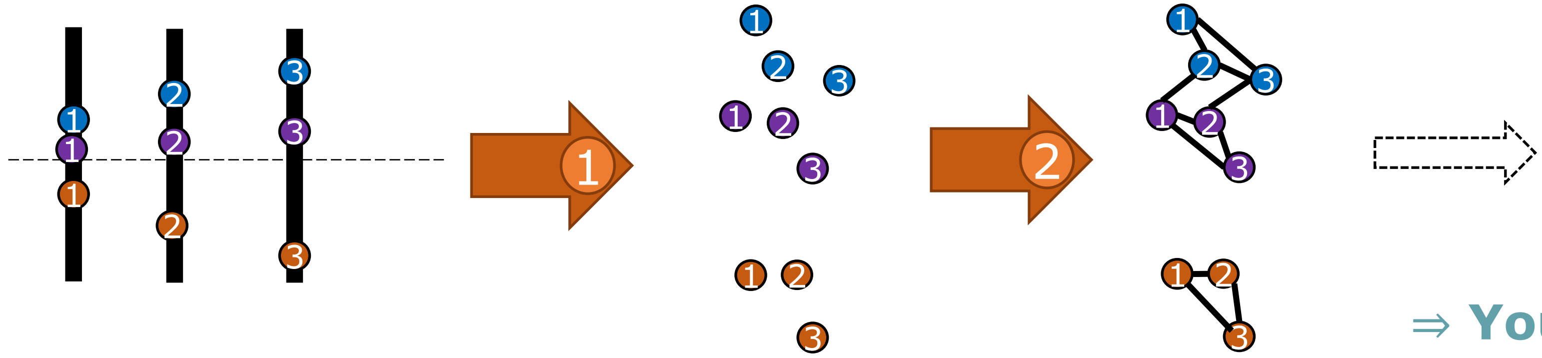
- If hit A and hit B are likely to be connected by an edge $\boldsymbol{d}(\boldsymbol{A}, \boldsymbol{B})^2 = \|\vec{e_A} - \vec{e_B}\|^2 < 0.01$
- Otherwise, $\boldsymbol{d}(\boldsymbol{A}, \boldsymbol{B})^2 > 0.01$

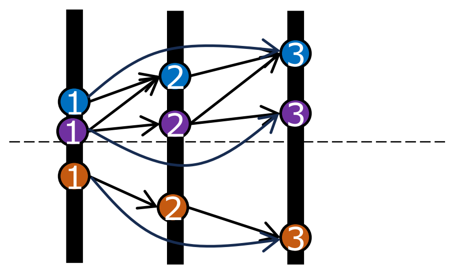
2 Loop over plane $p \in \{0, \ldots, 24\}$

Change w.r.t. Exa.Trkx

Parallelise over hits

- Apply $\boldsymbol{k}$-**Nearest Neighbour (kNN)** algorithm between plane p and planes $\{p+1, p+2\} \Rightarrow k$ edges / hit
- Discard edges for which $d^2 > \boldsymbol{d}^2_{\text{max}} = \mathbf{0.01}$ ← hyperparameter





⇒ **You've got your rough graph**

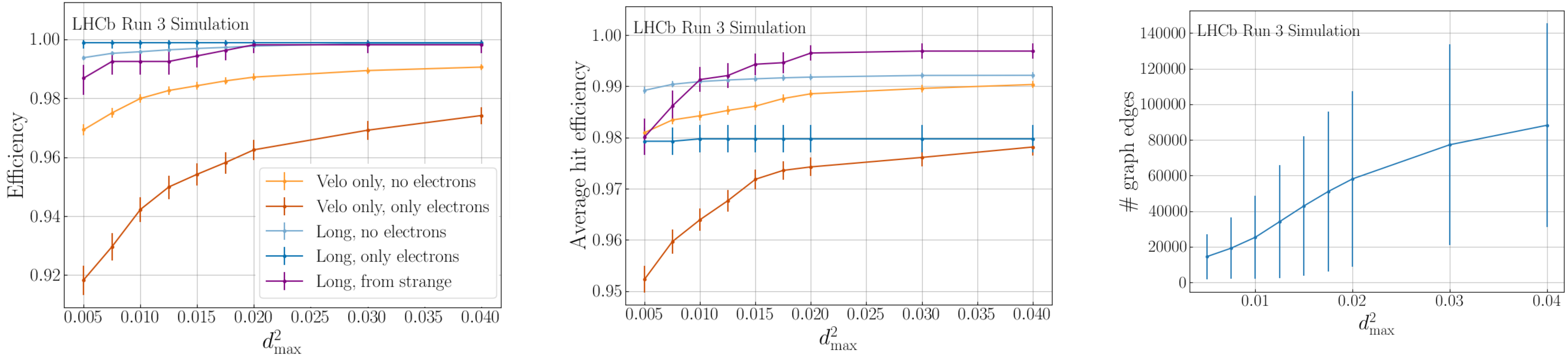
1. Graph Neural Network Track Finding

Graph Building → GNN: filter edges → Build tracks from graph

- Overall training strategy in back-up (*essentially* same as Exa.TrkX)
- *After training*, we choose maximal number of neighbours $\boldsymbol{k_{\max}} = \mathbf{50}$ (not optimised)

- To choose maximal squared distance $d^2_{\max}$, for various values for $d^2_{\max}$:
  1. Build the rough graph using $d^2_{\max}$
  2. **Remove all fake edges** in the rough graph and build the tracks from this purified graph
  3. Compute track-finding performance ⇒ correspond to the **best performance given $\boldsymbol{d^2_{\max}}$**

Performance if all the fake edges are discarded(≡ best performance)



⇒ We will try $\boldsymbol{d^2_{\max} = 0.010}$ and $\boldsymbol{d^2_{\max} = 0.020}$

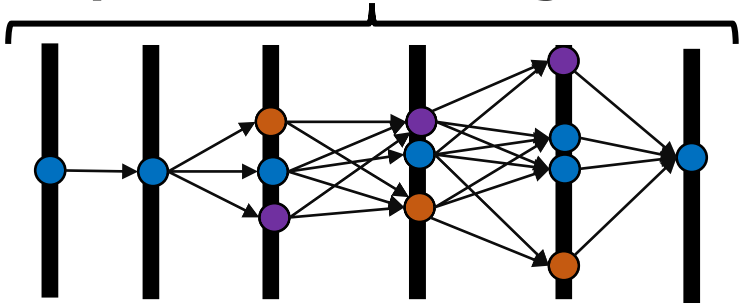
(evaluated on 200 events)

1. Graph Neural Network Track Finding

Graph Building | **GNN: filter edges** | Build tracks from graph

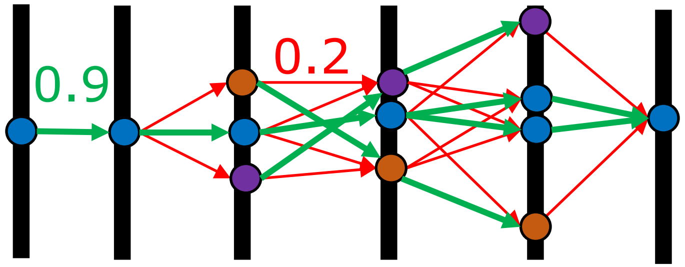
Reminder of the strategy

Output of Embedding + kNN



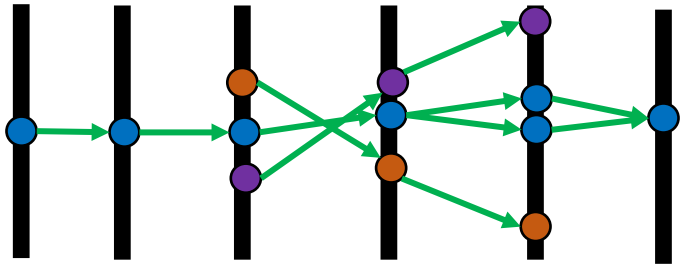
GNN edge classifier

⇒ score $s \in [0, 1]$ for every edge



Edge score cut

$s > s_{\text{min}}$



1. Graph Neural Network Track Finding

Graph Building → **GNN: filter edges** → Build tracks from graph

1. **Encode every hit and edge** in a high-dimensional space

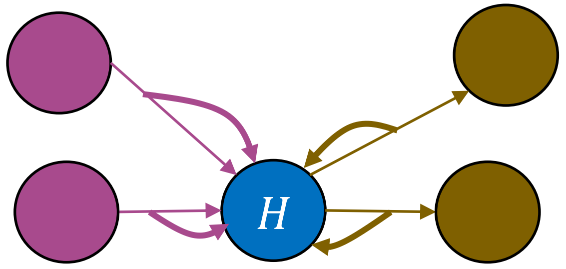
$\vec{r} = (r, \phi, z) \longrightarrow$ Node Encoder $\longrightarrow \vec{n} \in \mathbb{R}^{256}$

$(r_{\text{in}}, \phi_{\text{in}}, z_{\text{in}}, r_{\text{out}}, \phi_{\text{out}}, z_{\text{out}}) \longrightarrow$ Edge Encoder $\longrightarrow \vec{e} \in \mathbb{R}^{256}$

2. **Message passing**: repeat 6 times

hyperparameter

a. Build "**message**" by aggregating neighbour **edge encodings**



Change w.r.t. Exa.Trkx

Incoming and outgoing neighbours are **aggregated separately**

$$\textbf{Message} = [\ \overrightarrow{\max}(\{\vec{e}_{\text{input}}\}),\ \overrightarrow{\text{sum}}(\{\vec{e}_{\text{input}}\}),\ \overrightarrow{\max}(\{\vec{e}_{\text{output}}\}),\ \overrightarrow{\text{sum}}(\{\vec{e}_{\text{output}}\})\ ]$$

b. Update edge and node encodings

$[\vec{n}, \text{message}] \longrightarrow$ Node Network $\longrightarrow \oplus \longrightarrow \overrightarrow{n_{\text{updated}}}$

$[\vec{e}, \vec{n}^{in}_{\text{updated}}, \vec{n}^{\text{out}}_{\text{updated}}] \longrightarrow$ Edge Network $\longrightarrow \oplus \longrightarrow \overrightarrow{e_{\text{updated}}}$

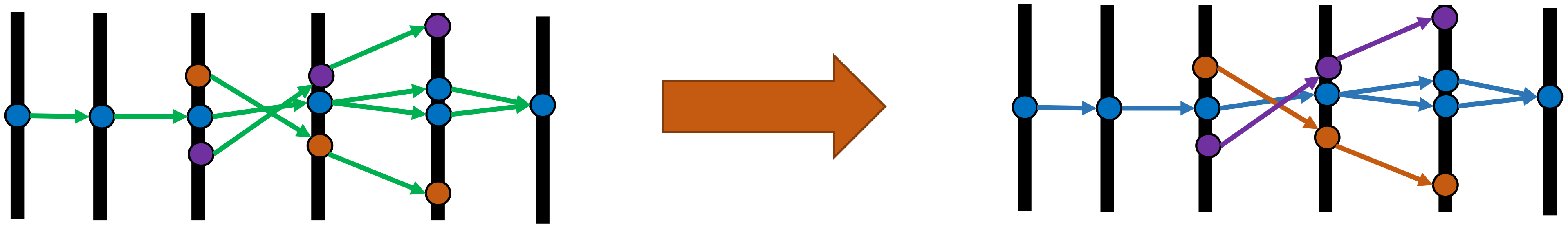
3. **Compute edge scores**

$[\vec{n}_{\text{in}}, \vec{n}_{\text{out}}, \vec{e}] \longrightarrow$ Edge Classifier $\longrightarrow$ Edge score $s \in [0, 1]$

Trained with a [sigmoid focal loss]()

1. Graph Neural Network Track Finding

Graph Building → GNN: filter edges → **Build tracks from graph**



Tracks obtained by identifying connected hits

But if you do this… **track efficiency on long electrons** is **terrible!**

Metric	**Allen**	**etx4velo**
Efficiency	**98.17%**	**46.23%**
Clone rate	3.07%	0.47%
Hit efficiency	95.35%	98.89%
Hit purity	99,67%	93.89%

(evaluated on 1000 events)

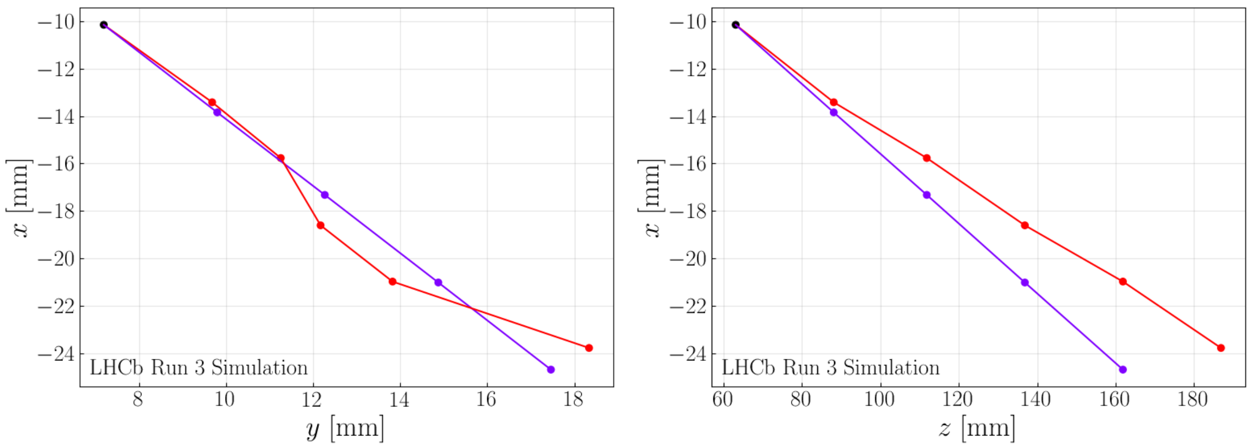
2. Issue of Shared Hits

The Case of Electrons

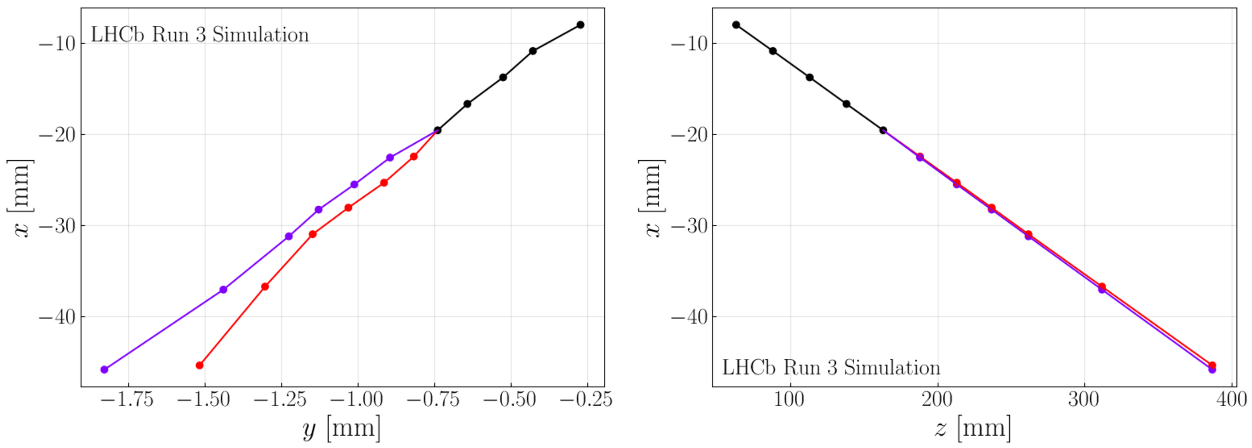
Observations

- ~ 55 % electrons share hits with another electron
- The 2 electrons share ≥ 1 hit(s) before splitting up

Example 1: share the first hit only



Example 2: share several hits before splitting up

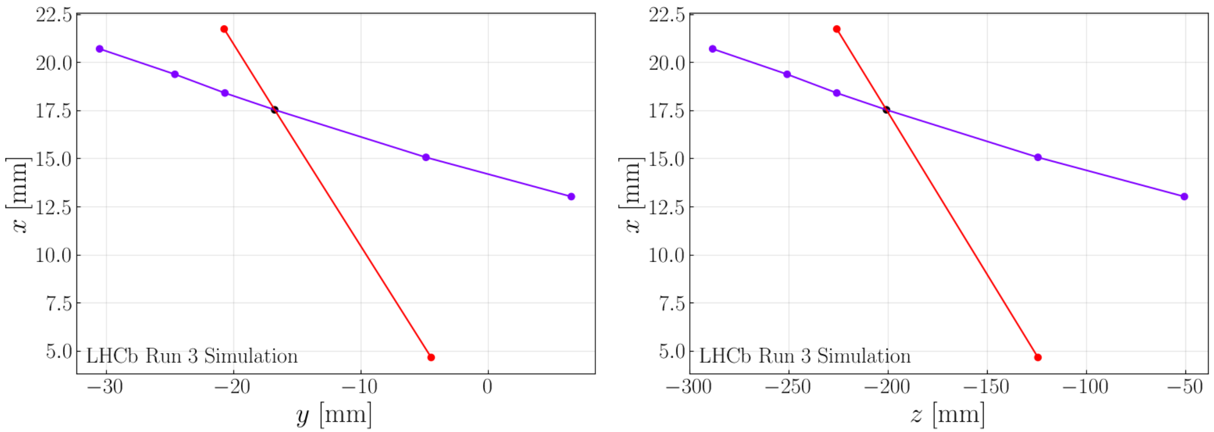


The connected component algorithm consider the 2 electron tracks as a single track

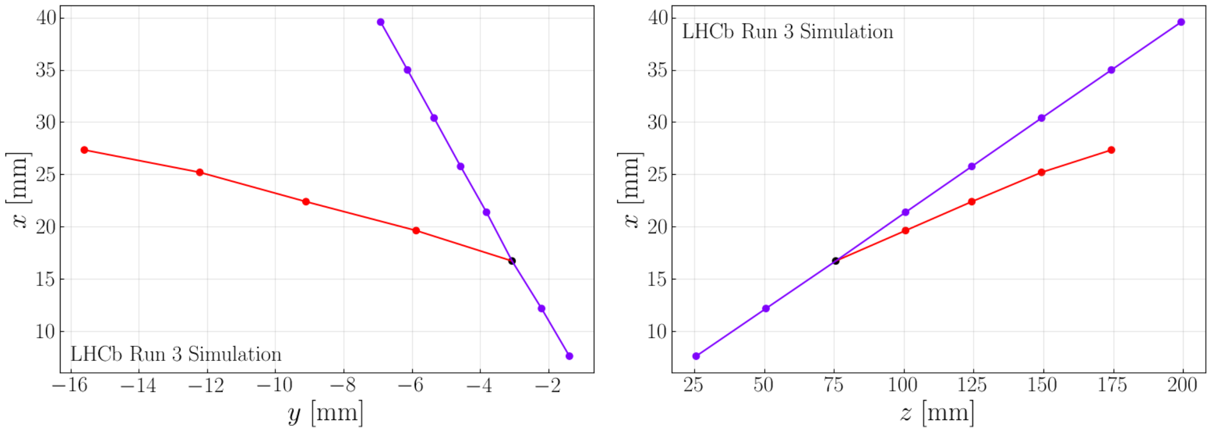
2. Issue of Shared Hits

Other Tracks With Shared Hits

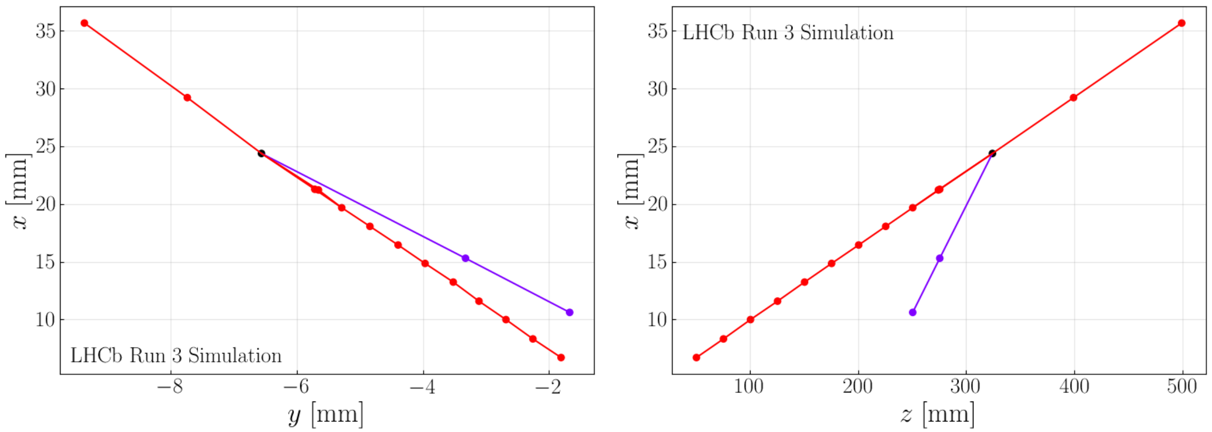
- **Tracks crossing** (> 524 in 1000 events)



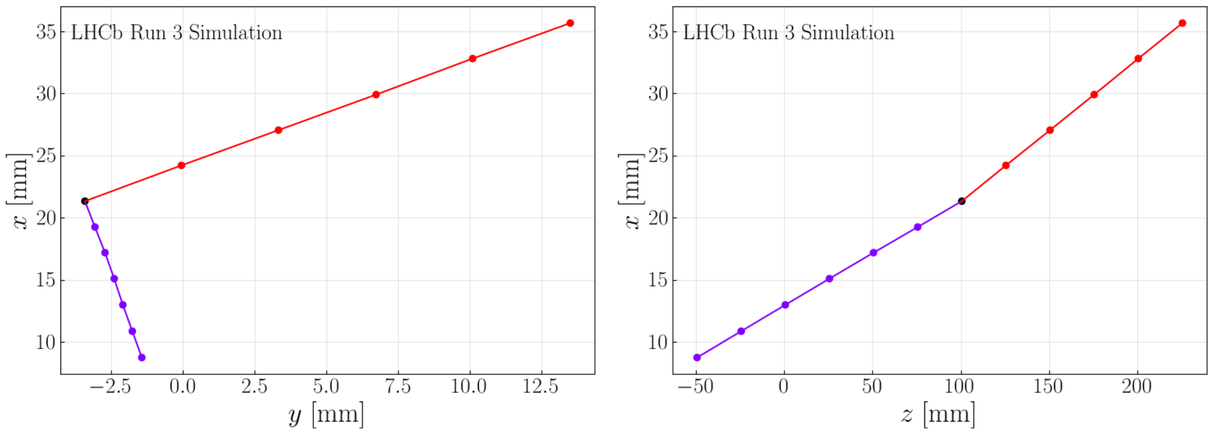
- **Track starts on a shared hit**



- **Tracks ends on a shared hit**

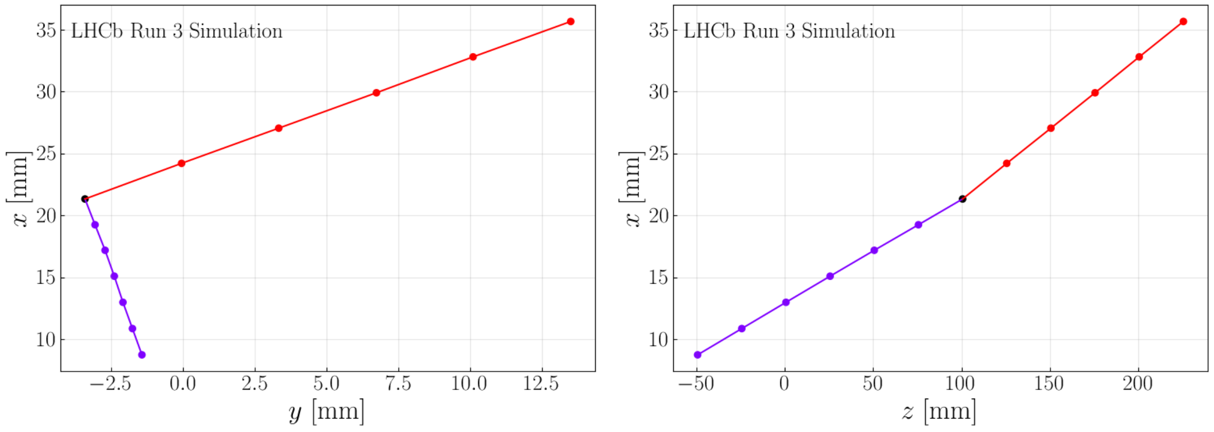


- **The last hit of a track is the first hit of another track**
(>141 in 1000 events)

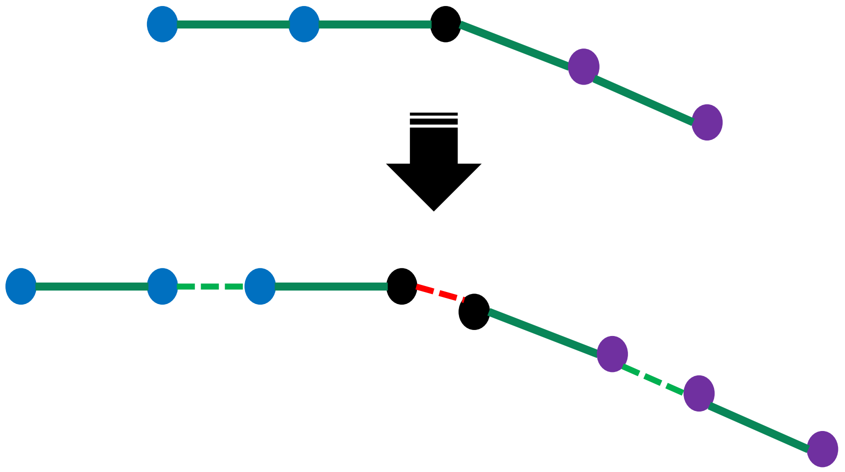


2. Issue of Shared Hits

Edge-Edge Connections



In this case, one cannot even guess that there are *possibly* 2 tracks!



Hit-hit connection is not enough
⇒ need **edge-edge connections**

2. Issue of Shared Hits

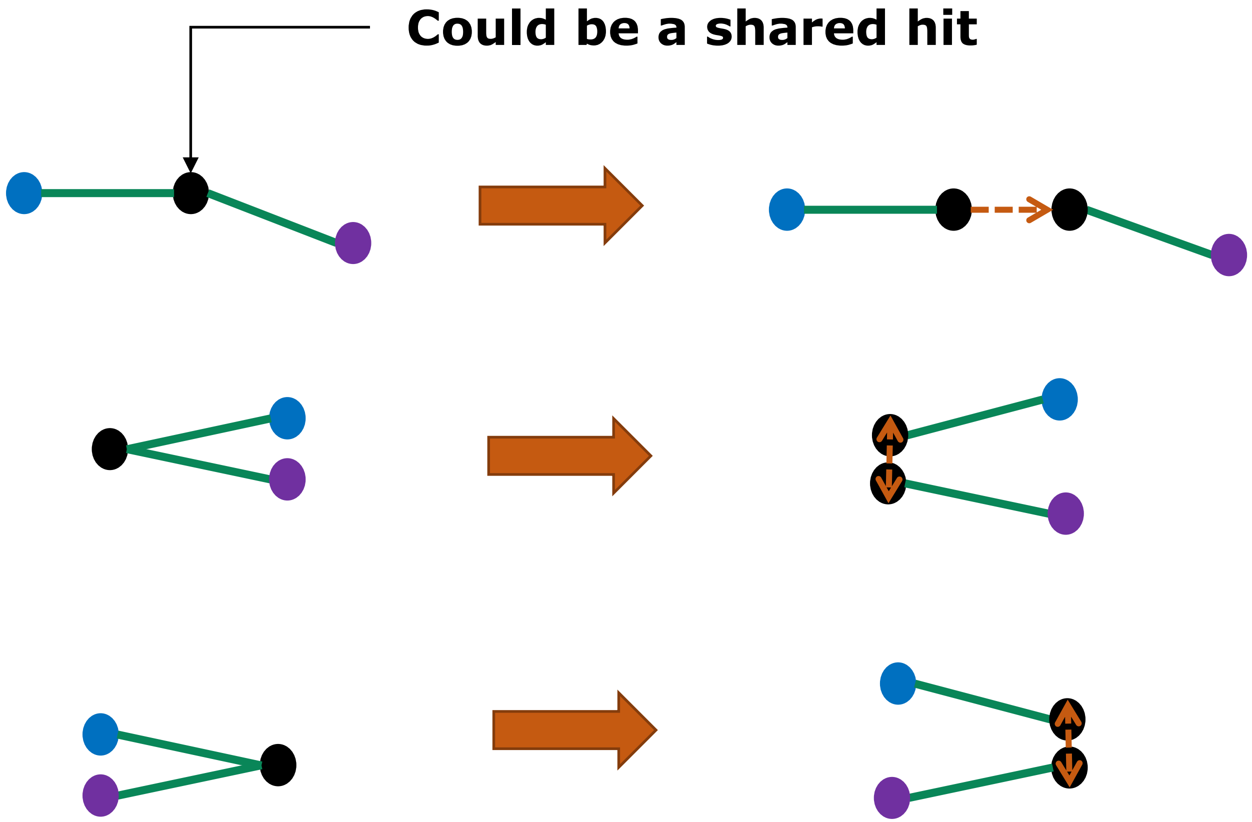
Edge-Edge Connections

3 kind of **edge connections** (or *triplets*)

Articulation

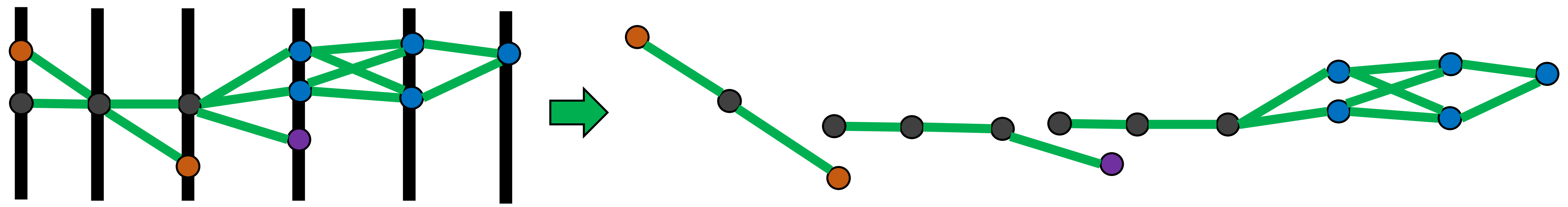
Left elbow

Right elbow



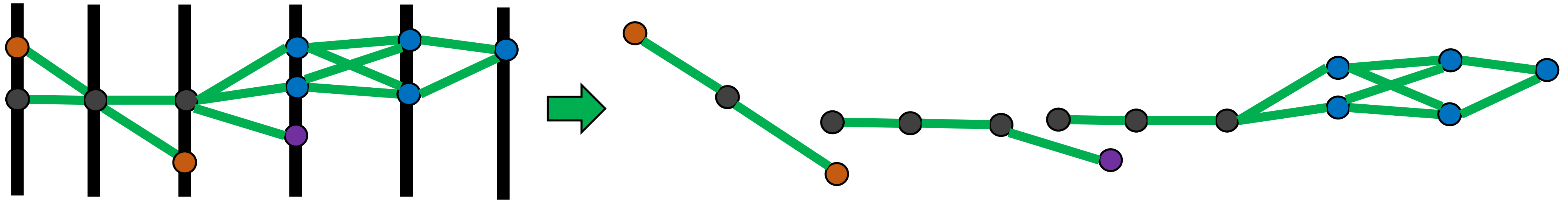
2. Issue of Shared Hits

Goal

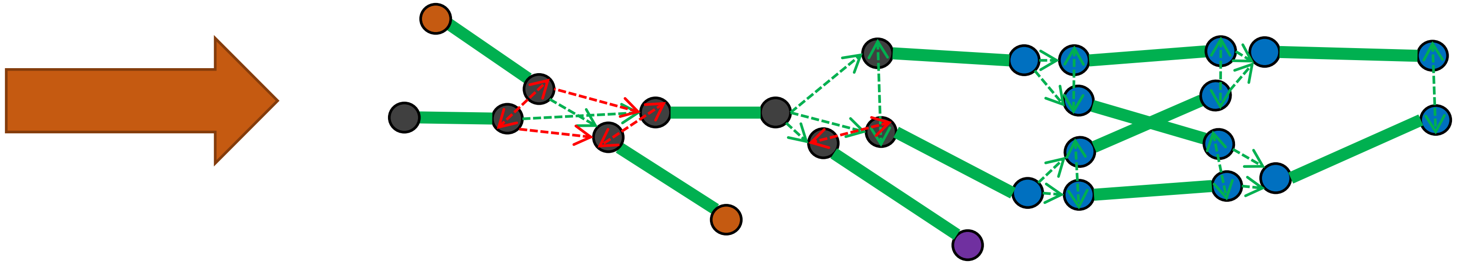


2. Issue of Shared Hits

Goal

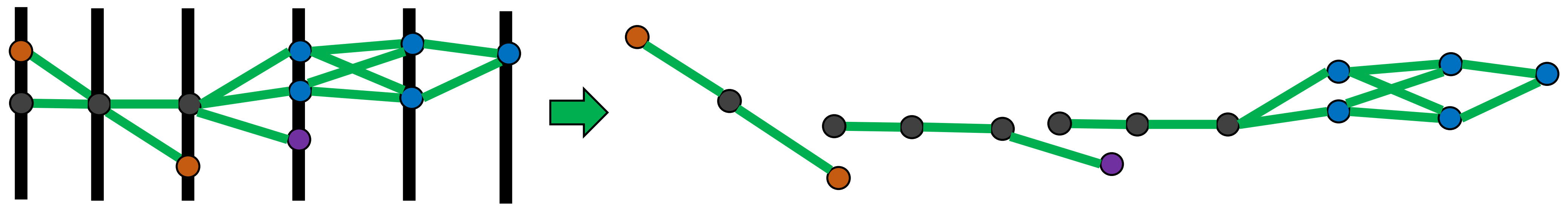


1 Build edge-edge connections

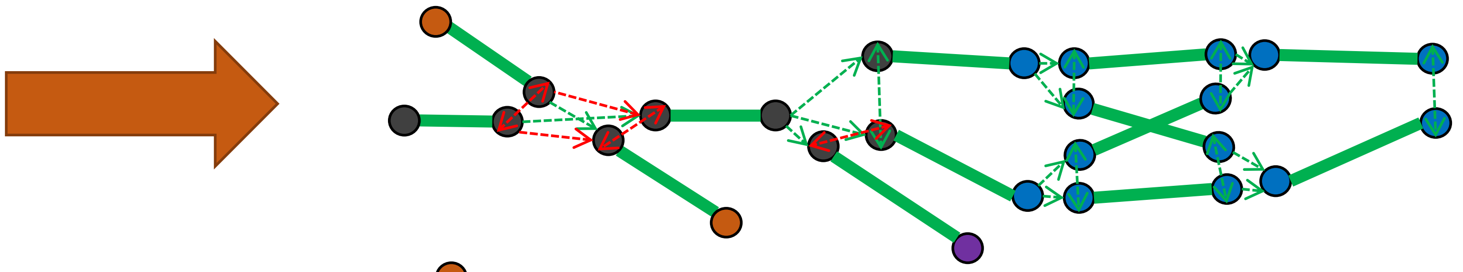


2. Issue of Shared Hits

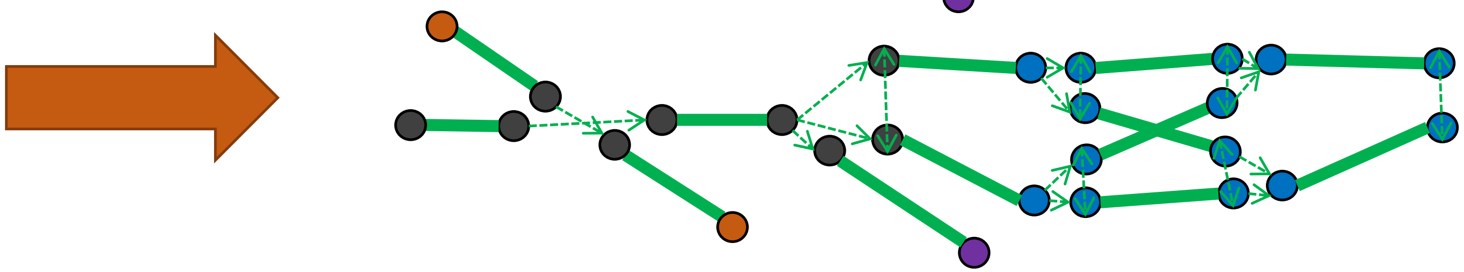
Goal



1 Build edge-edge connections

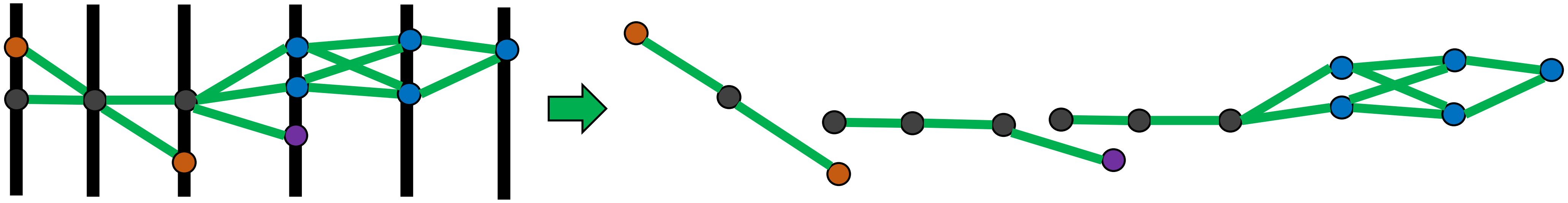


2 Classify the edge-edge connections
Filter out the fake edge-edge connections

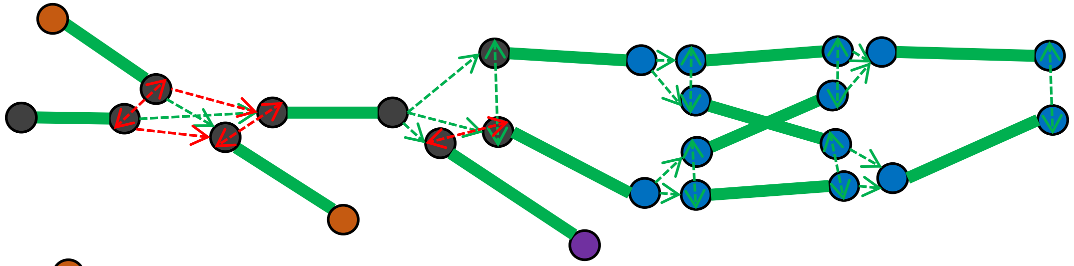


2. Issue of Shared Hits

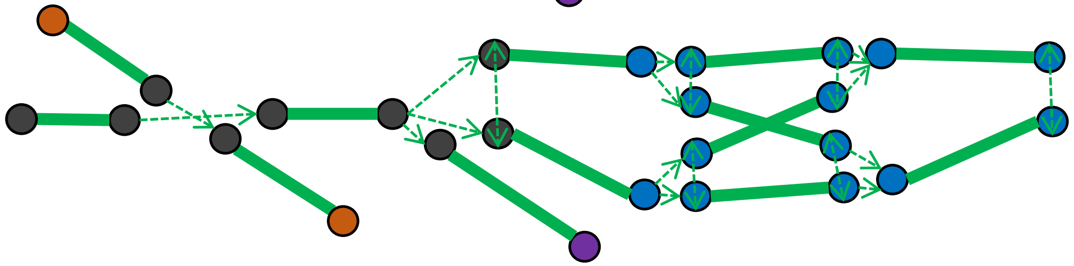
Goal



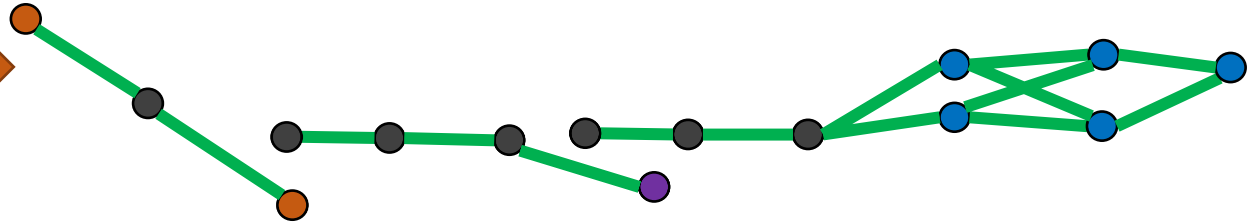
1. Build edge-edge connections



2. Classify the edge-edge connections
Filter out the fake edge-edge connections



3. Algorithm to build tracks from edge-edge connections



2. Issue of Shared Hits

Embedding Network → kNN $k_{\max}, d^2_{\max}$ → GNN on edges → Filter edges $s_{\text{edge,min}}$ → Build triplets → GNN on triplets → Filter triplets $s_{\text{triplet,min}}$ → Build tracks

2. Issue of Shared Hits

Embedding Network → kNN $k_{\mathrm{max}}, d^2_{\mathrm{max}}$ → GNN on edges → Filter edges $s_{\mathrm{edge,min}}$ → Build triplets → **GNN on triplets** → Filter triplets $s_{\mathrm{triplet,min}}$ → Build tracks

Don't repeat the 6-step message passing: **start from the previous GNN**

3 **Compute edge scores**

$[\vec{n}_{\mathrm{in}}, \vec{n}_{\mathrm{out}}, \vec{e}]$ → Edge Classifier → Edge score $s_{\mathrm{edge}} \in [0, 1]$

4 Filter out the fake edges by requiring $s_{\mathrm{edge}} > s_{\mathrm{edge,min}}$ to reduce # edge-edge connections

5 Build **edge-edge connections** (≡ **triplets**)

6 **Directly compute triplet scores from the edge and nodes encoding of the triplet**

$[\vec{n}_{\mathrm{shared}}, \vec{n}_{\mathrm{first}}, \vec{n}_{\mathrm{last}}, \vec{e}_{\mathrm{in}}, \vec{e}_{\mathrm{out}}]$ → Triplet Classifier → Triplet score $s_{\mathrm{triplet}} \in [0, 1]$

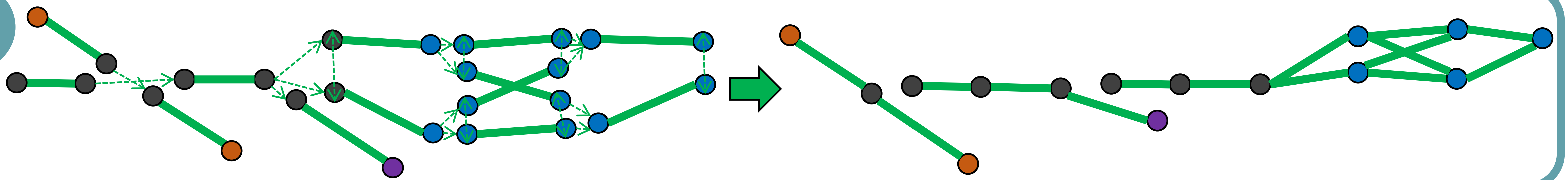
Filter out the fake edge-edge connections by requiring $s_{\mathrm{triplet}} > s_{\mathrm{triplet,min}}$

GNN trained with the overall loss

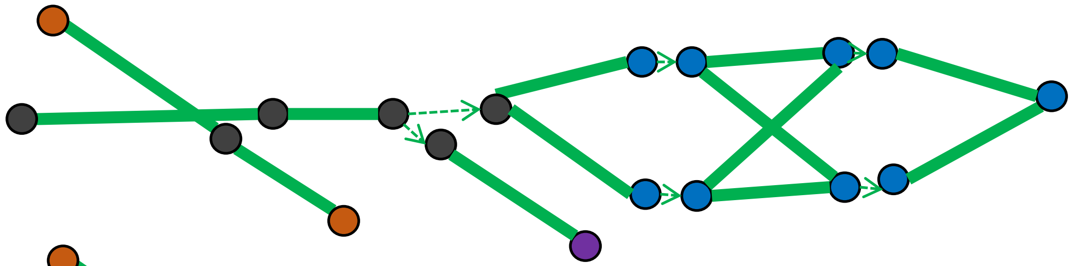
$$\mathcal{L}_{\mathrm{tot}} = \mathcal{L}_{\mathrm{edge}} + \mathcal{L}_{\mathrm{triplet}}$$

2. Issue of Shared Hits

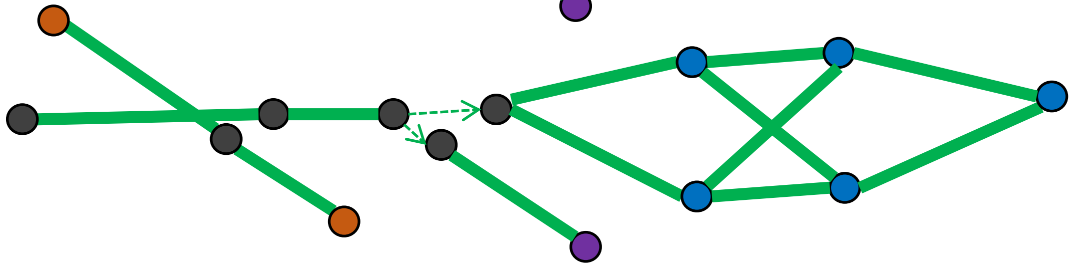
Goal



1 **Connect left and right elbows** and remove duplicate edge-edge connections



2 **Apply connected components**, excluding splitting edge-edge connections

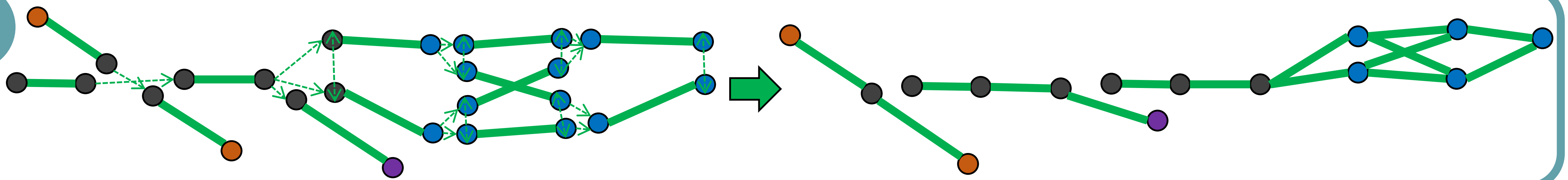


New Hypothesis: a track may split into 2 tracks **only one time**
→ Allow to keep *locality*

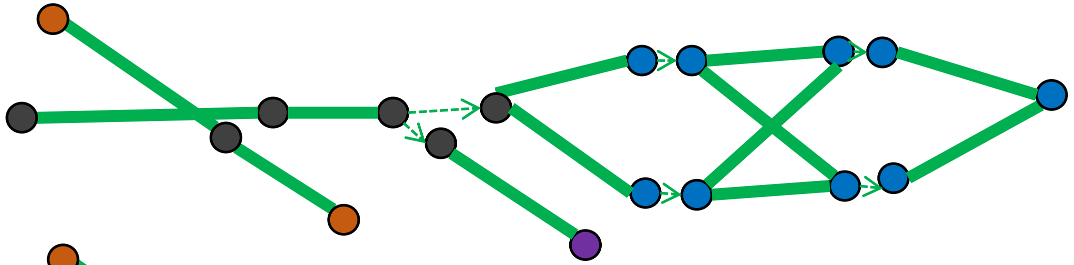
2. Issue of Shared Hits

Embedding Network → kNN $k_{\max}, d^2_{\max}$ → GNN on edges → Filter edges $s_{\text{edge,min}}$ → Build triplets → GNN on triplets → Filter triplets $s_{\text{triplet,min}}$ → **Build tracks**

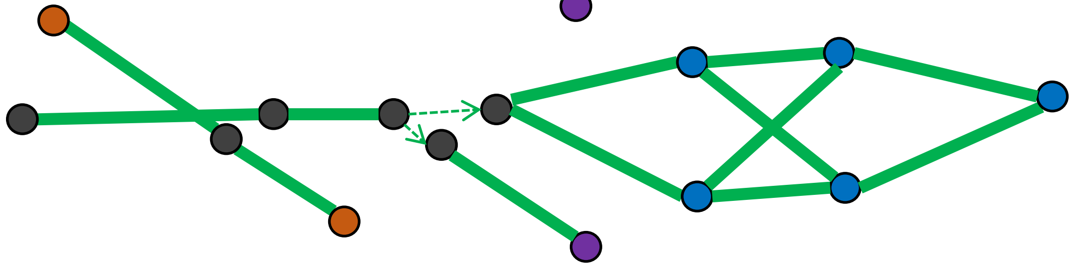
Goal



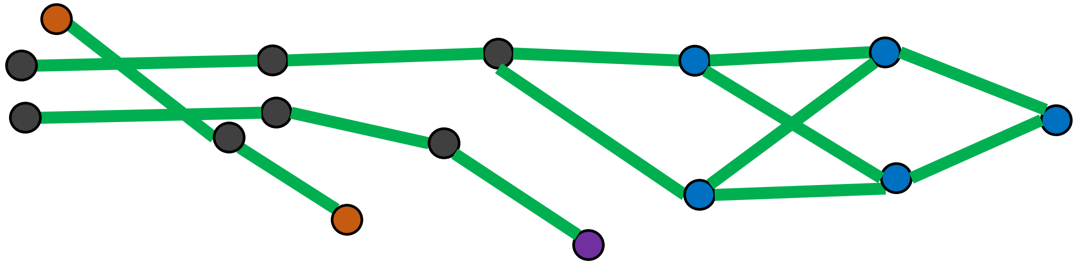
1 **Connect left and right elbows** and remove duplicate edge-edge connections



2 **Apply connected components**, excluding splitting edge-edge connections



3 **Each remaining link** correspond to a **new track**



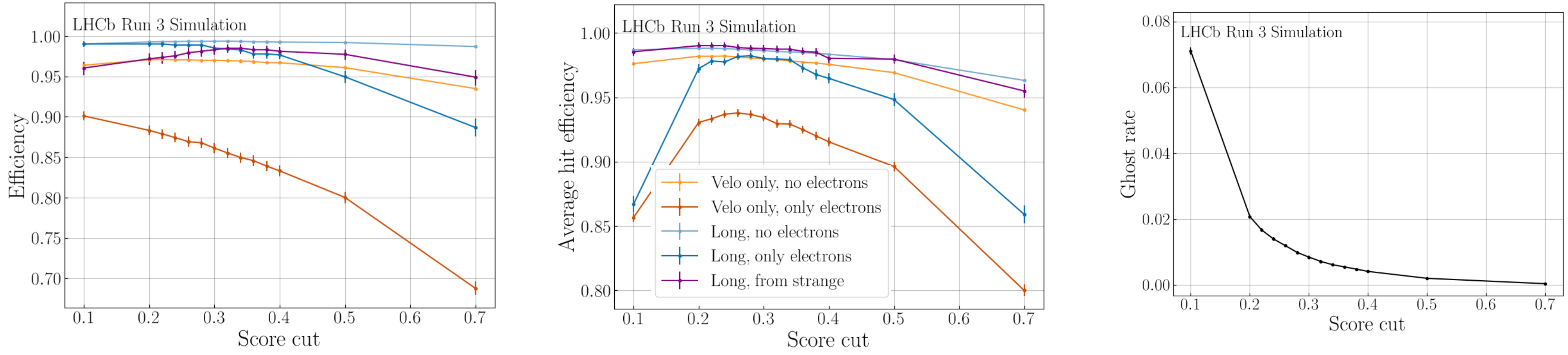
3. Track-Finding Performance

Embedding Network → kNN $k_{\max}, d^2_{\max}$ → GNN on edges → Filter edges $s_{\text{edge,min}}$ → Build triplets → GNN on triplets → **Filter triplets** $s_{\text{triplet,min}}$ → Build tracks

Before building the tracks from the graph of triplets
- Choose $s_{\text{edge,min}} = 0.4$ to optimise performance *(could be increased to optimise throughput)*
- Choose $s_{\text{triplet,min}}$ by evaluating track-finding performance as a function of $s_{\text{triplet,min}}$
    - High efficiency
    - Ghost rate $< 1\%$

(evaluated on 200 events)

Performance as a function of the triplet score cut $s_{\text{triplet,min}}$



⇒ choose $s_{\text{triplet,min}} = 0.32$

3. Performance of ETX4VELO

Category	Metric	Allen	$s_{\text{triplet}} > 0.32$ Etx4velo $d^2_{\max} = 0.10$	$s_{\text{triplet}} > 0.36$ Etx4velo $d^2_{\max} = 0.20$
Long, no electrons ✓ In acceptance ✓ Reconstructible in the velo ✓ Reconstructible in the SciFi ✓ Not an electron	Efficiency	99.26%	99.28%	99.51%
	Clone rate	2.54%	0.96%	0.89%
	Hit efficiency	96.46%	98.73%	98.90%
	Hit Purity	99.78%	99.94%	99.94%
Long electrons ✓ In acceptance ✓ Reconstructible in the velo ✓ Reconstructible in the SciFi ✓ Electron	Efficiency	97.11%	98.80%	99.22%
	Clone rate	4,25%	7.42%	7.31%
	Hit efficiency	95.24%	96.54%	96.79%
	Hit purity	97.11%	98.46%	98.46%
Long, from strange ✓ In acceptance ✓ Reconstructible in the velo ✓ Decays from a strange *Good proxy for displaced tracks*	Efficiency	97.69%	97.50%	98.06%
	Clone rate	2.50%	0.92%	0.81%
	Hit efficiency	97.69%	98.22%	98.77%
	Hit purity	99.34%	99.68%	99.68%
X	Ghost rate	2.18%	0.76%	0.81%

- Evaluation with 5,000 events
- 2 different GNN trainings for $d^2_{\max} = 0.10$ and $d^2_{\max} = 0.20$

Long categories

Worse — Better

3. Performance of ETX4VELO

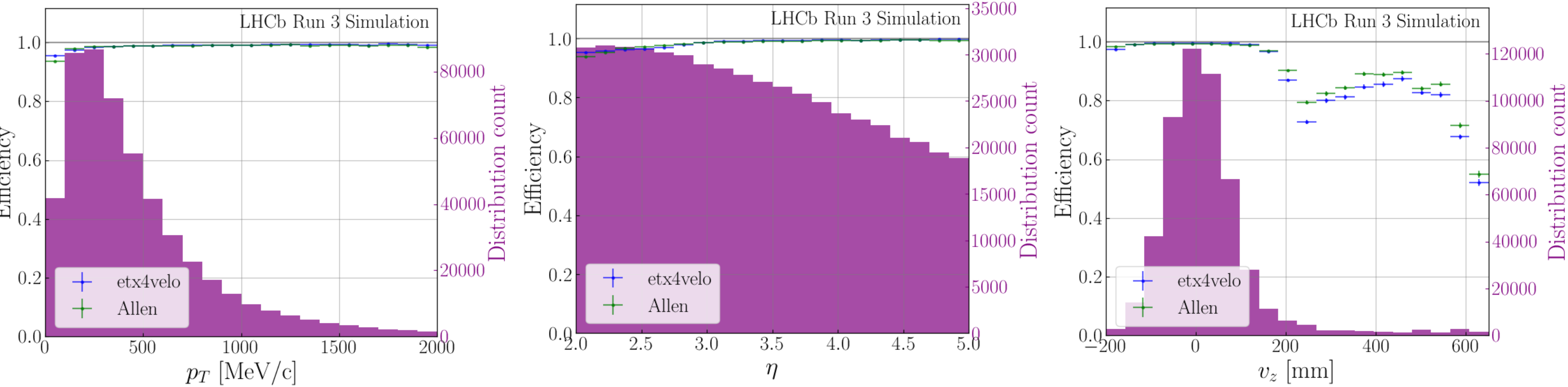
Category	Metric	Allen	$s_{\text{triplet}} > 0.32$ Etx4velo $d^2_{\text{max}} = 0.10$	$s_{\text{triplet}} > 0.36$ Etx4velo $d^2_{\text{max}} = 0.20$
Velo-only, no electrons ✓ In acceptance ✓ Reconstructible in the velo ✓ Not reconstructible in the SciFi ✓ Not an electron	Efficiency	96.84%	97.03%	97.86%
	Clone rate	3.84%	1.08%	1.02%
	Hit efficiency	93.89%	97.93%	98.32%
	Hit Purity	99.50%	99.84%	99.82%
Velo-only electrons ✓ In acceptance ✓ Reconstructible in the velo ✓ Not reconstructible in the SciFi ✓ Electron	Efficiency	67.81%	85.10%	86.69%
	Clone rate	10.27%	5.02%	4.97%
	Hit efficiency	79.21%	93.33%	93.88%
	Hit purity	97.35%	99.07%	98.99%
Velo-only, from strange ✓ In acceptance ✓ Not reconstructible in the velo ✓ Decays from a strange *Good proxy for displaced tracks*	Efficiency	93.53%	93.07%	96.05%
	Clone rate	5.60%	1.97%	1.77%
	Hit efficiency	90.05%	93.92%	96.05%
	Hit purity	99.36%	99.67%	99.64%
X	Ghost rate	2.18%	0.76%	0.81%

- Evaluation with 5,000 events
- 2 different GNN trainings for $d^2_{\text{max}} = 0.10$ and $d^2_{\text{max}} = 0.20$

Velo-only categories

Worse → Better

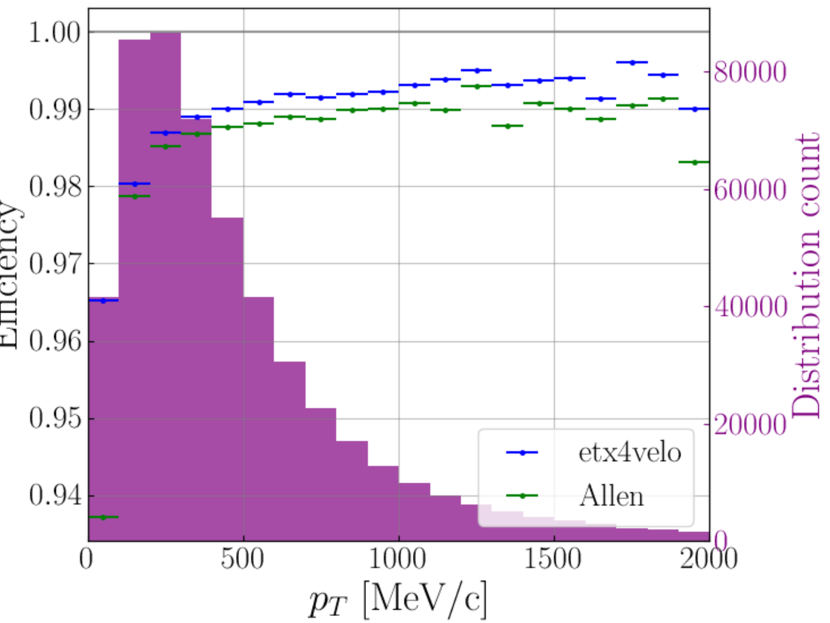
3. Performance of ETX4VELO

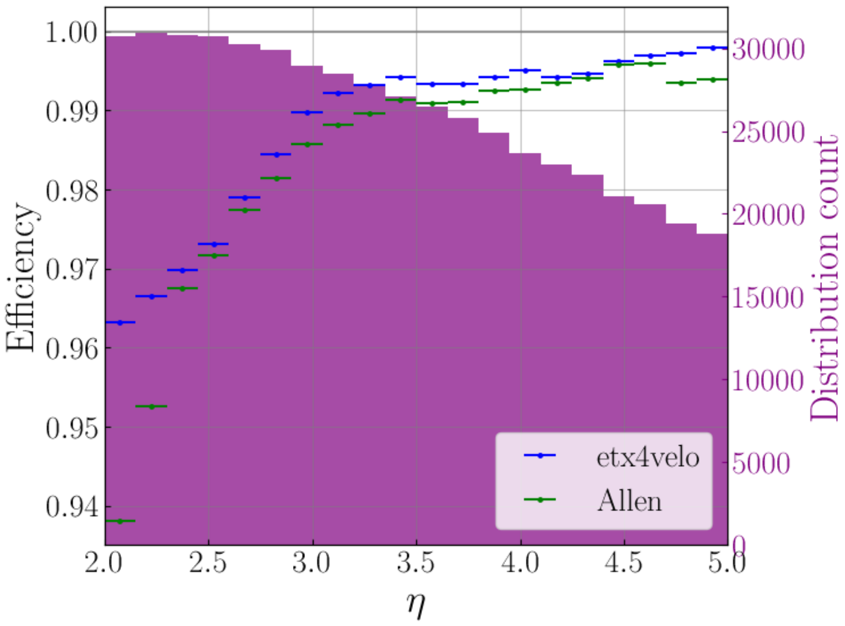


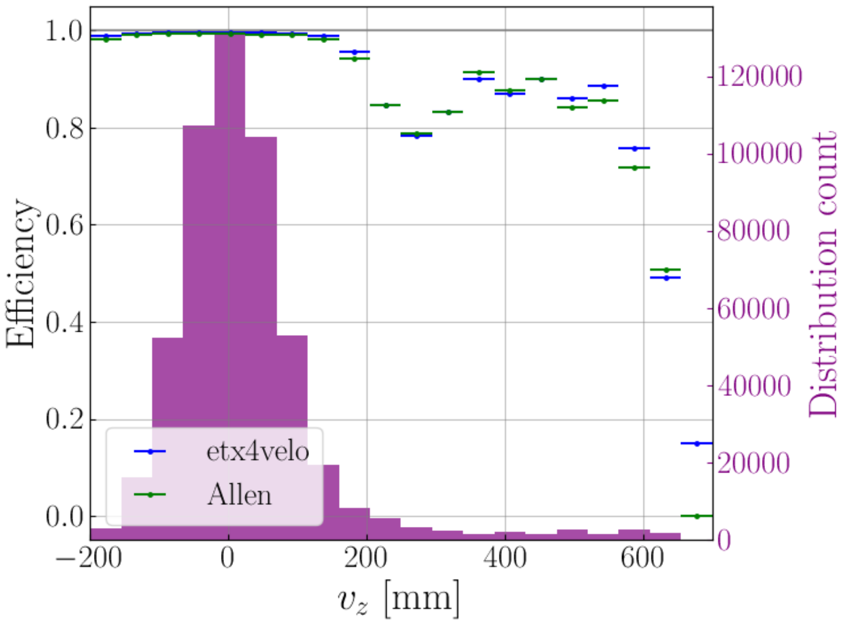
Velo,
no electrons

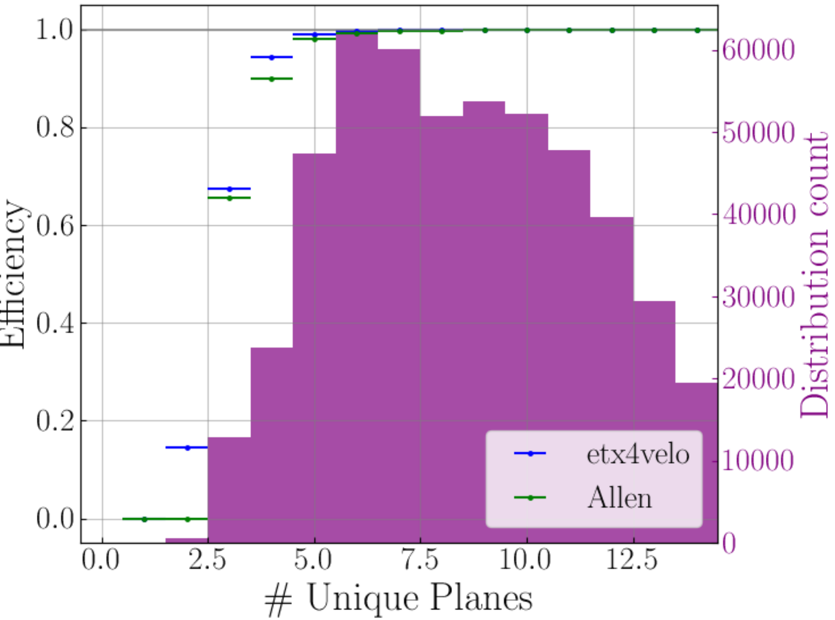
$$d^2_{\max} = 0.10$$

3. Performance of ETX4VELO





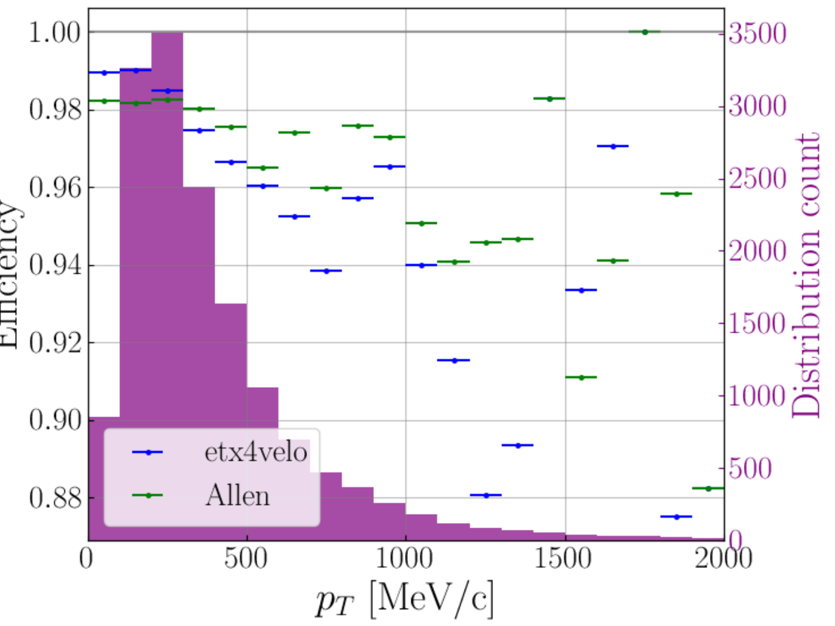


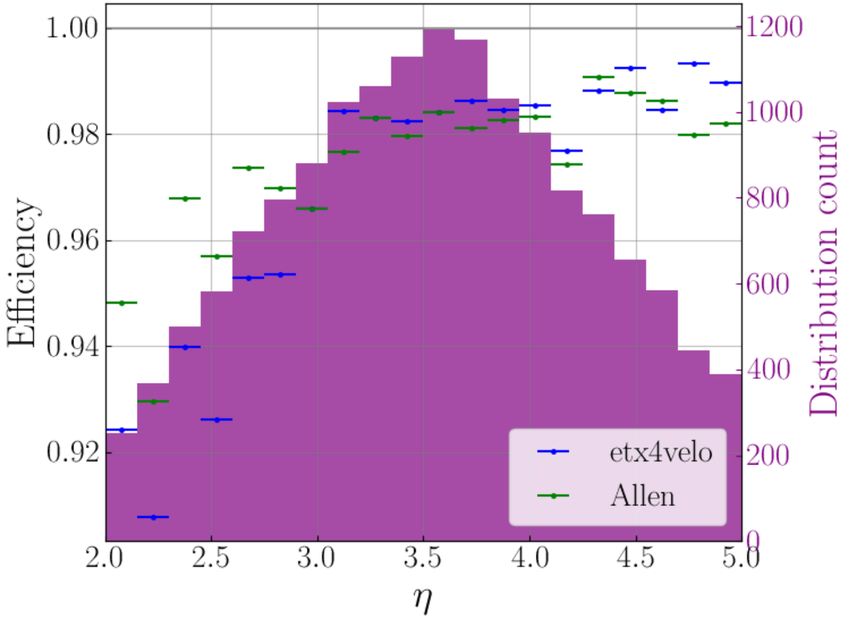


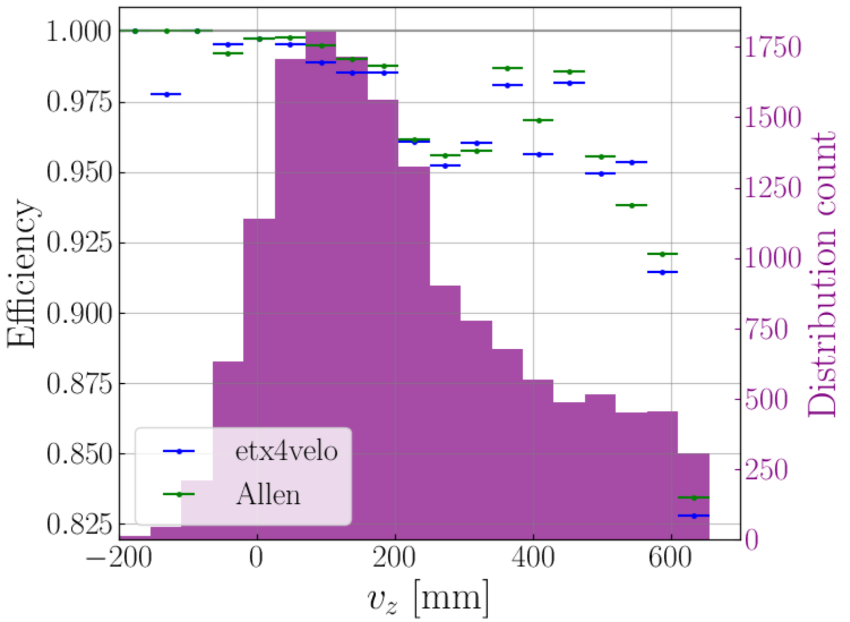
Velo,
no electrons

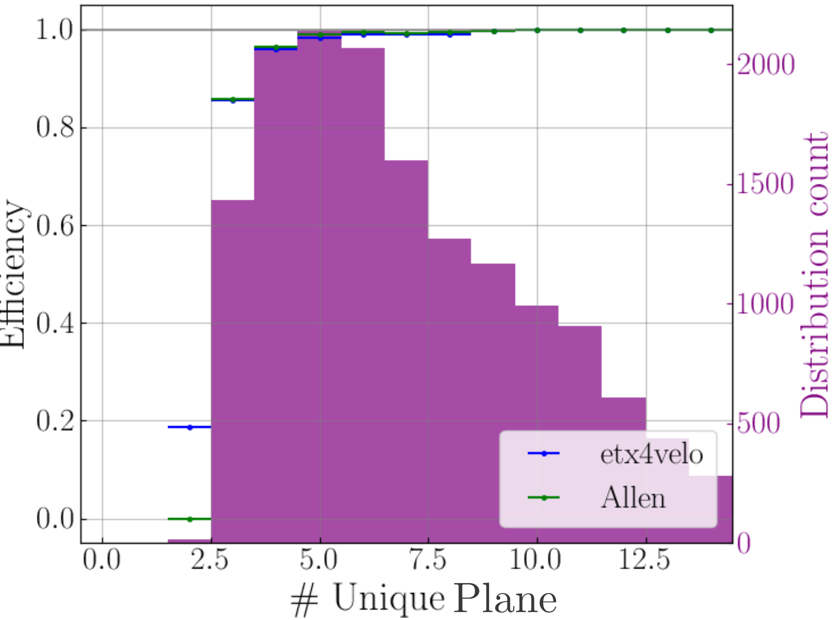
$$d^2_{\max} = 0.20$$

3. Performance of ETX4VELO





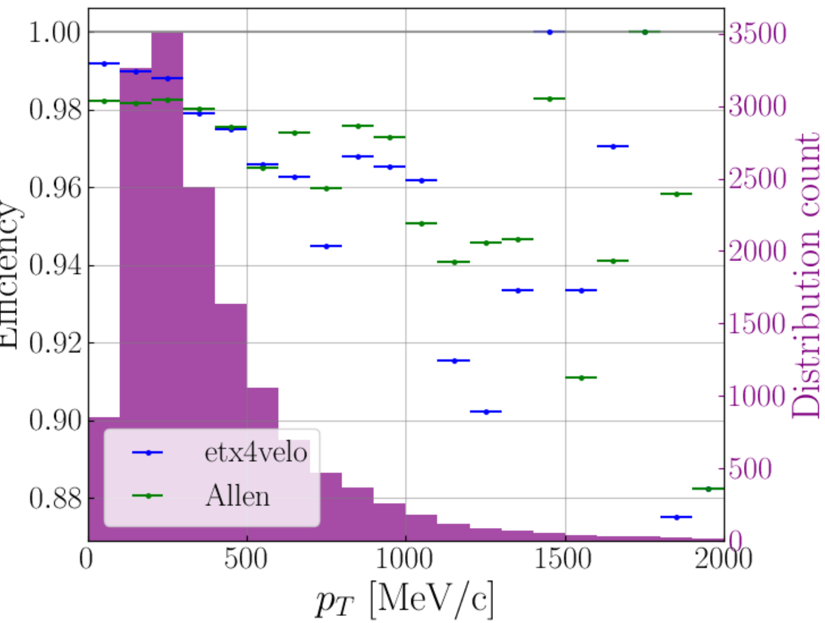


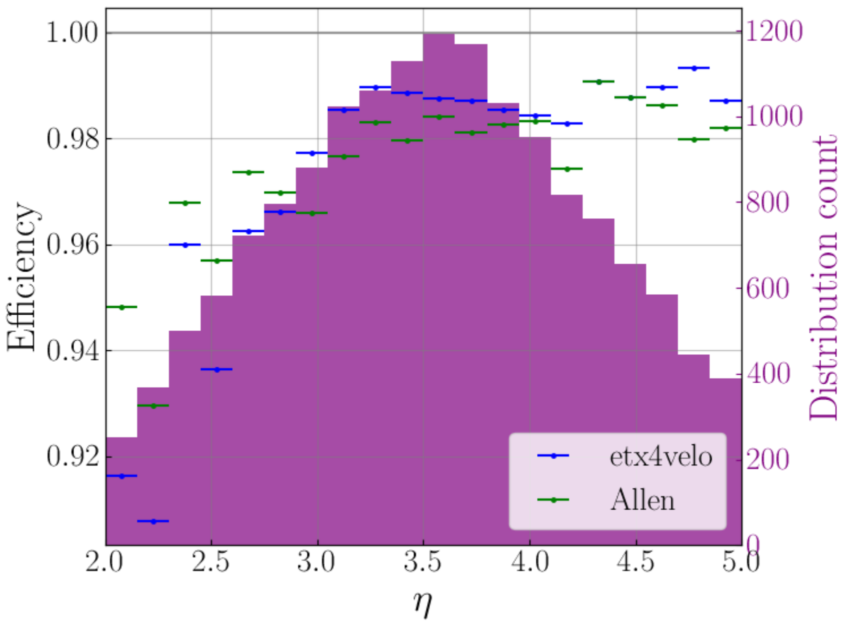


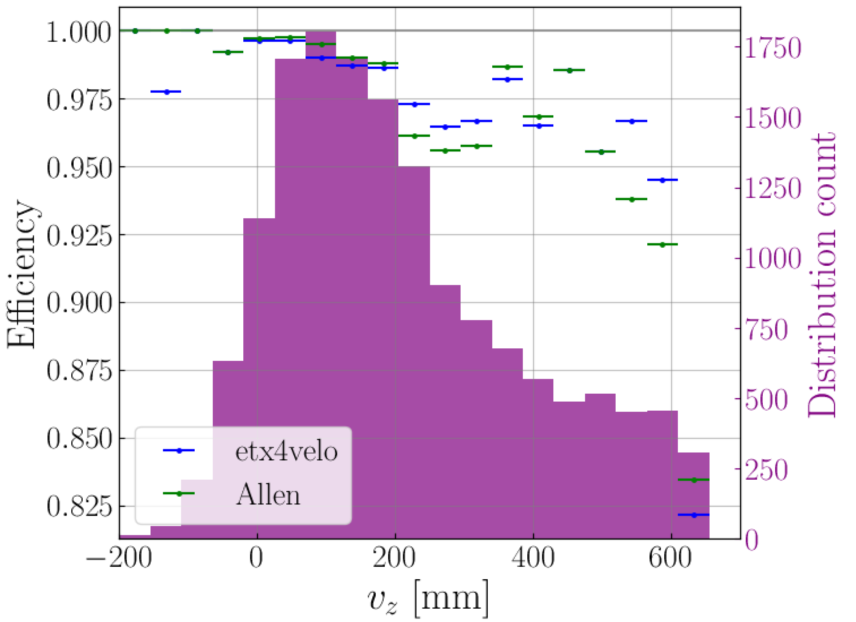
Long, from strange

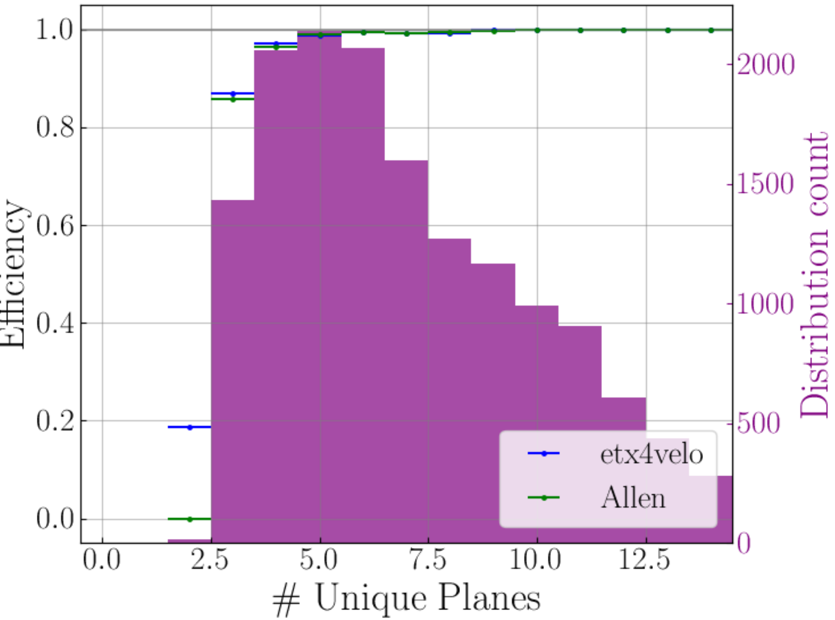
$d^2_{\max} = 0.10$

3. Performance of ETX4VELO









Long,
from strange

$\boldsymbol{d^2_{\max} = 0.20}$

Conclusion

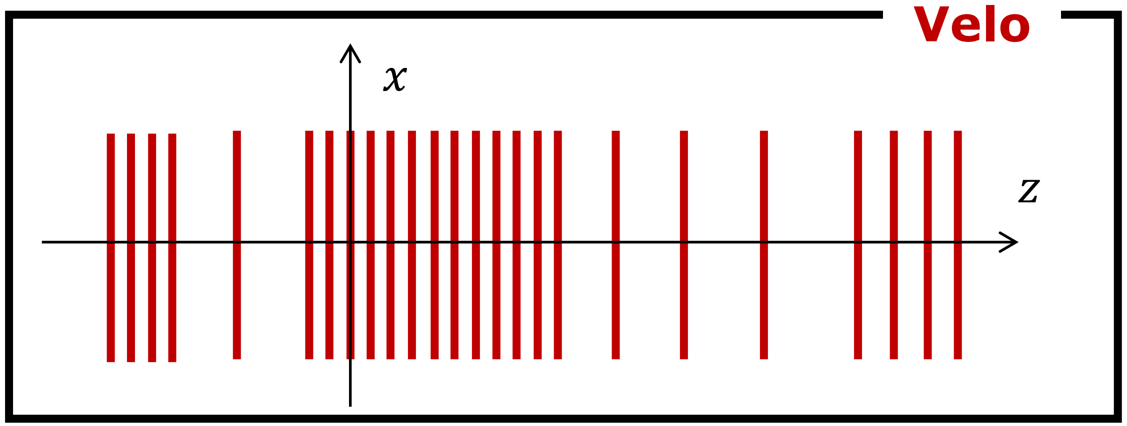
- Overall good physics reconstruction performance, still room for improvement
- Fotis Giasemis working on C++ inference in Allen
- Next step: moving to the **SciFi detector**

Git repositories

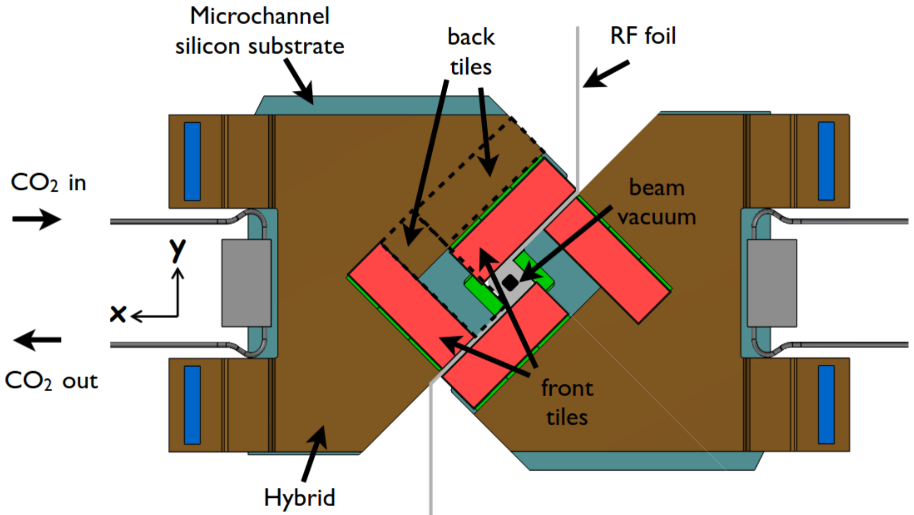
xdigi2csv	Convert XDIGI simulated files on the grid to CSV-like files
montetracko	Perform track matching and evaluation in Python
etx4velo (dev branch)	Perform track finding in the velo with a GNN-based approach

Backup Slides

Velo geometry

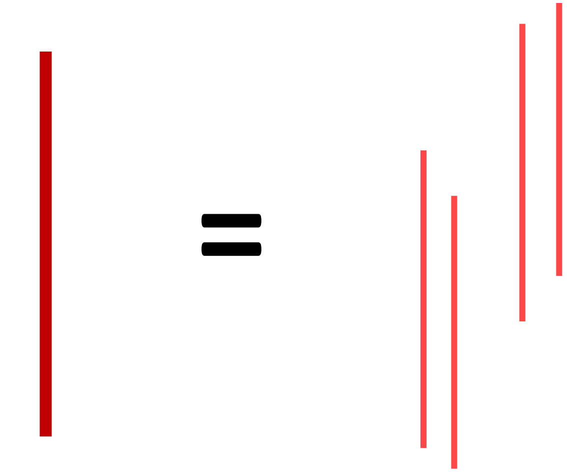


1 plane = 4 sensor planes



P. C. Tsopelas, 'A Silicon Pixel Detector for LHCb', PhD Thesis, Vrije U., Amsterdam, 2016.
https://inspirehep.net/literature/1645999

1 plane **4 sensor planes**



1. GNN-based Track Finding Approach

Graph Building | GNN: filter edges | Build tracks from graph

Training step

1. Embed all the hits using the network $(r, \phi, z, \mathrm{plane}) \xrightarrow{\text{DNN}} \vec{e} = (e_1, e_2, e_3, e_4)$

2. For a random given set of hits, build a **dataset of genuine edges and fake edges**.
Compute the distances between their hits in the embedding space:
$\{d^2_{\mathrm{genuine},i}, \forall i\}$ and $\{d^2_{\mathrm{fake},j}, \forall j\}$

3. Minimise hinge loss $\mathcal{L}_{\mathrm{total}} = \underbrace{3}_{\text{hyperparameter}}\mathcal{L}_{\mathrm{genuine}} + \mathcal{L}_{\mathrm{fake}}$ where

$$\mathcal{L}_{\mathrm{genuine}} = \frac{1}{n_{\mathrm{genuine}}} \sum_i d^2_{\mathrm{genuine},i}$$

Minimise $d_{\mathrm{genuine},i}$

$$\mathcal{L}_{\mathrm{fake}} = \frac{1}{n_{\mathrm{fake}}} \sum_j \max\left(\underbrace{0.01}_{\text{hyperparameter}} - d^2_{\mathrm{fake},j}, 0\right)$$

Maximise $d_{\mathrm{fake},j}$

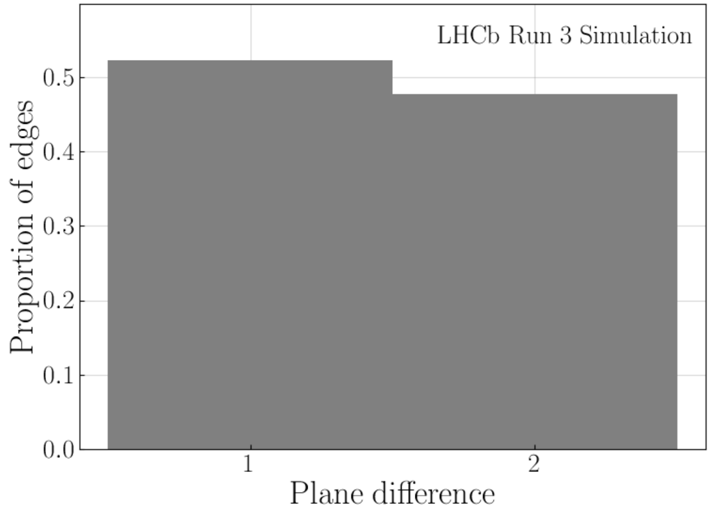
Training dataset

- **Hard Negative Mining**: edges built by a kNN *(→ "hard" negatives)*
- **True** edges
- **Random** edges

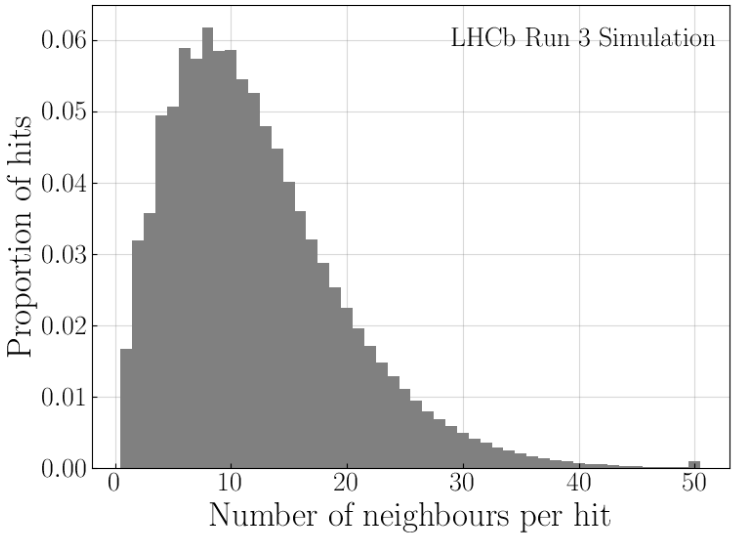
1. GNN-based Track Finding Approach

Graph Building → GNN: filter edges → Build tracks from graph

Rough graph with $k_{\max} = 50$ and $d^2_{\max} = 0.010$

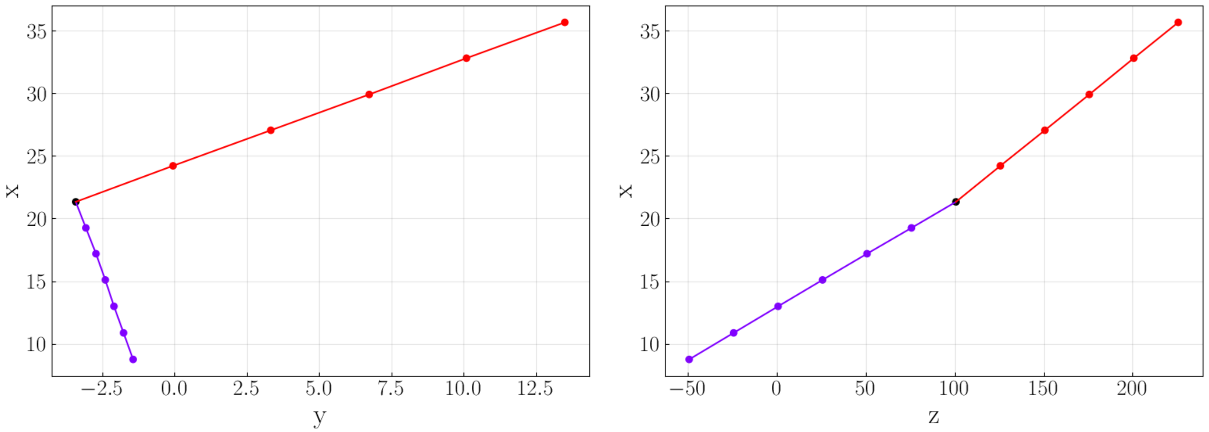


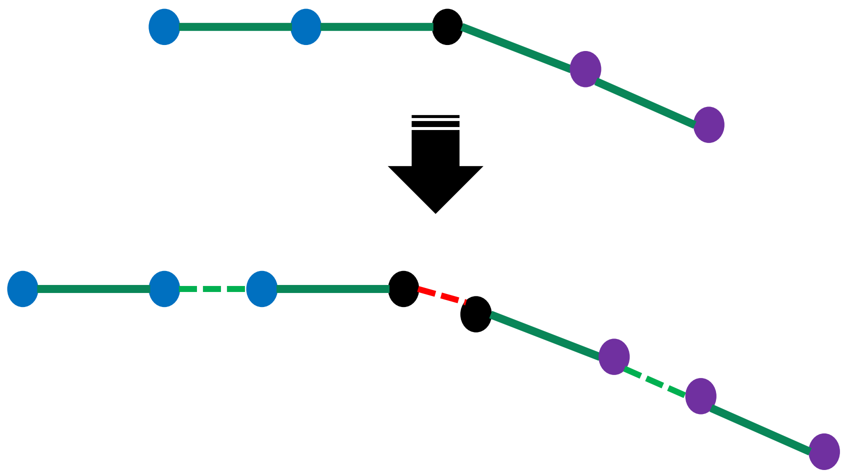
Even though 1% **of genuine edges are 2-plane apart**, the rough graph needs to contain **almost 50% of such edges**



$\Rightarrow$ $\boldsymbol{k_{\max}}$ **could probably be reduced** to increase throughput

2. Issue of Shared Hits

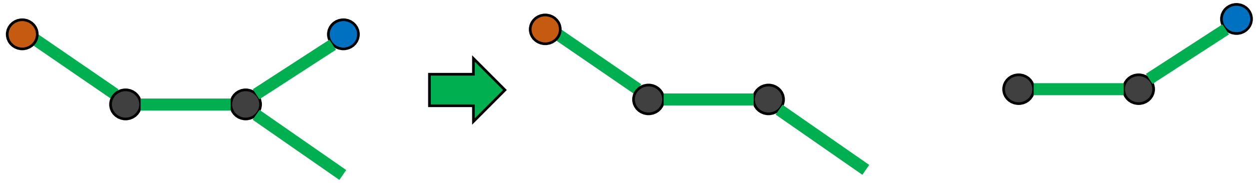




Hit-hit connection is not enough
⇒ need **edge-edge connections**

- Solve the ambiguity of shared hits under the following hypothesis:
  "All hits that precede a splitting point can be attributed to all the newly identified tracks"

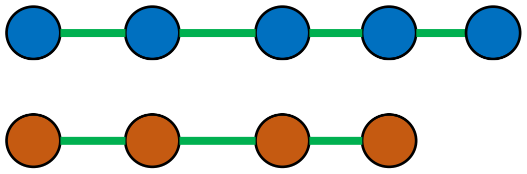
- ⇒ Assume that this does not happen



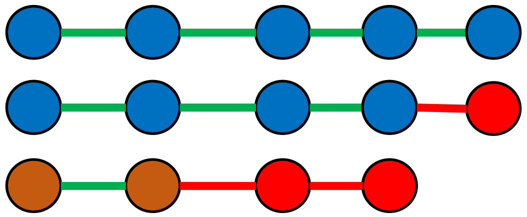
0. LHCb Detector in Run 3

Track-Finding Evaluation

True particles



Tracks found by the track-finding algorithm



- To perform track-finding, need to match found tracks to true particles
  - At LHCb: track matched to particle if **at least 70% of its hits** belong to this particle