Plans for additional studies of low-momentum particles and pathlength & position resolution

Barak Schmookler

Minimum transverse momentum needed to reach barrel layers

Gyration radius

$$R = \frac{p_T}{0.3\,q\,B}$$

Max. radial reach

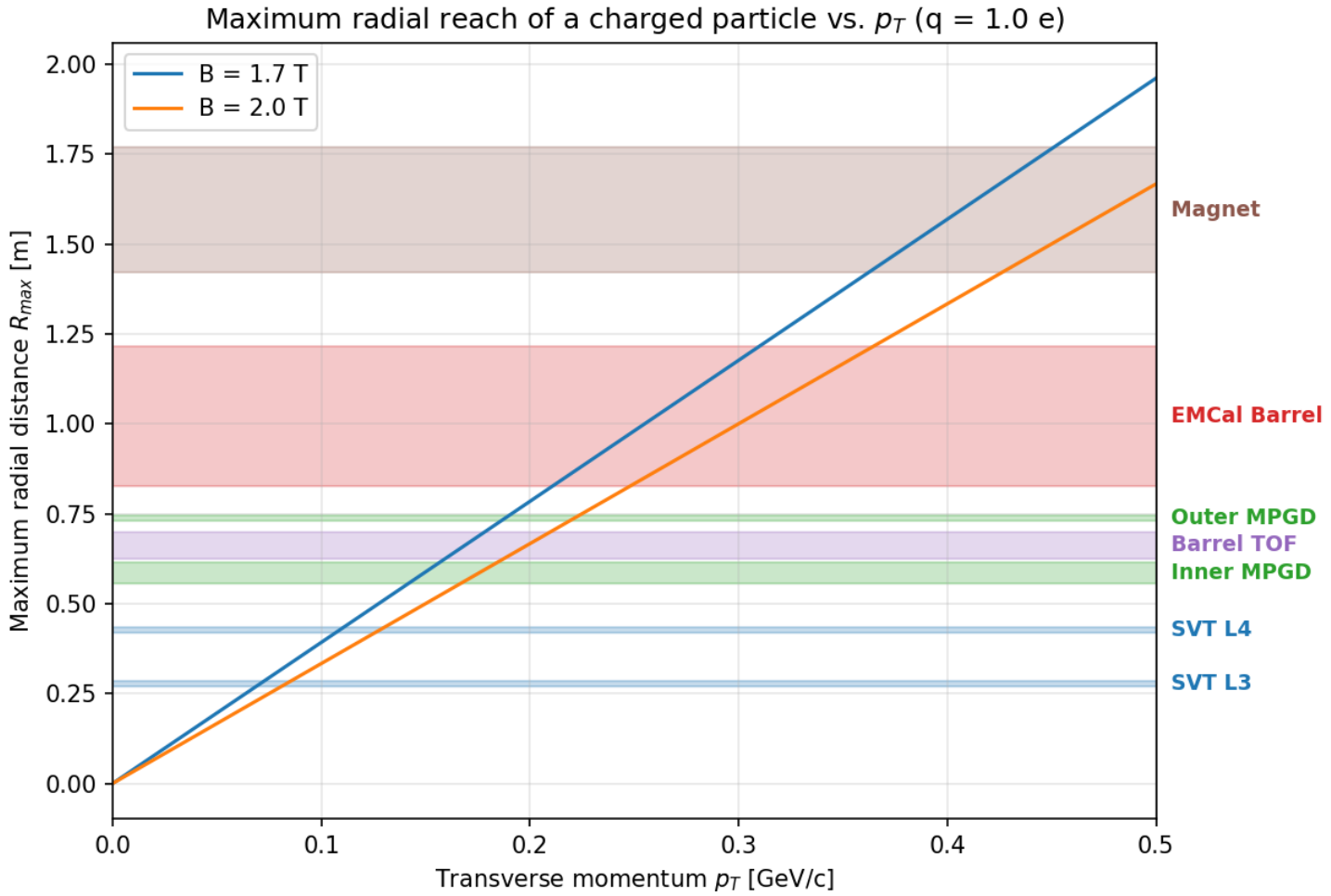
$$R_{max} = 2R = \frac{2\,p_T}{0.3\,q\,B}$$

Minimum p_T to reach radius r

$$p_T^{min} = 0.15\,q\,B\,r$$

p_T in GeV/c, B in T, r in m, q in units of e

Ignoring energy loss



Detector locations based on September 2025 CAD geometry

Minimum transverse momentum needed to reach barrel layers

Gyration radius

$$R = \frac{p_T}{0.3\,q\,B}$$

Max. radial reach

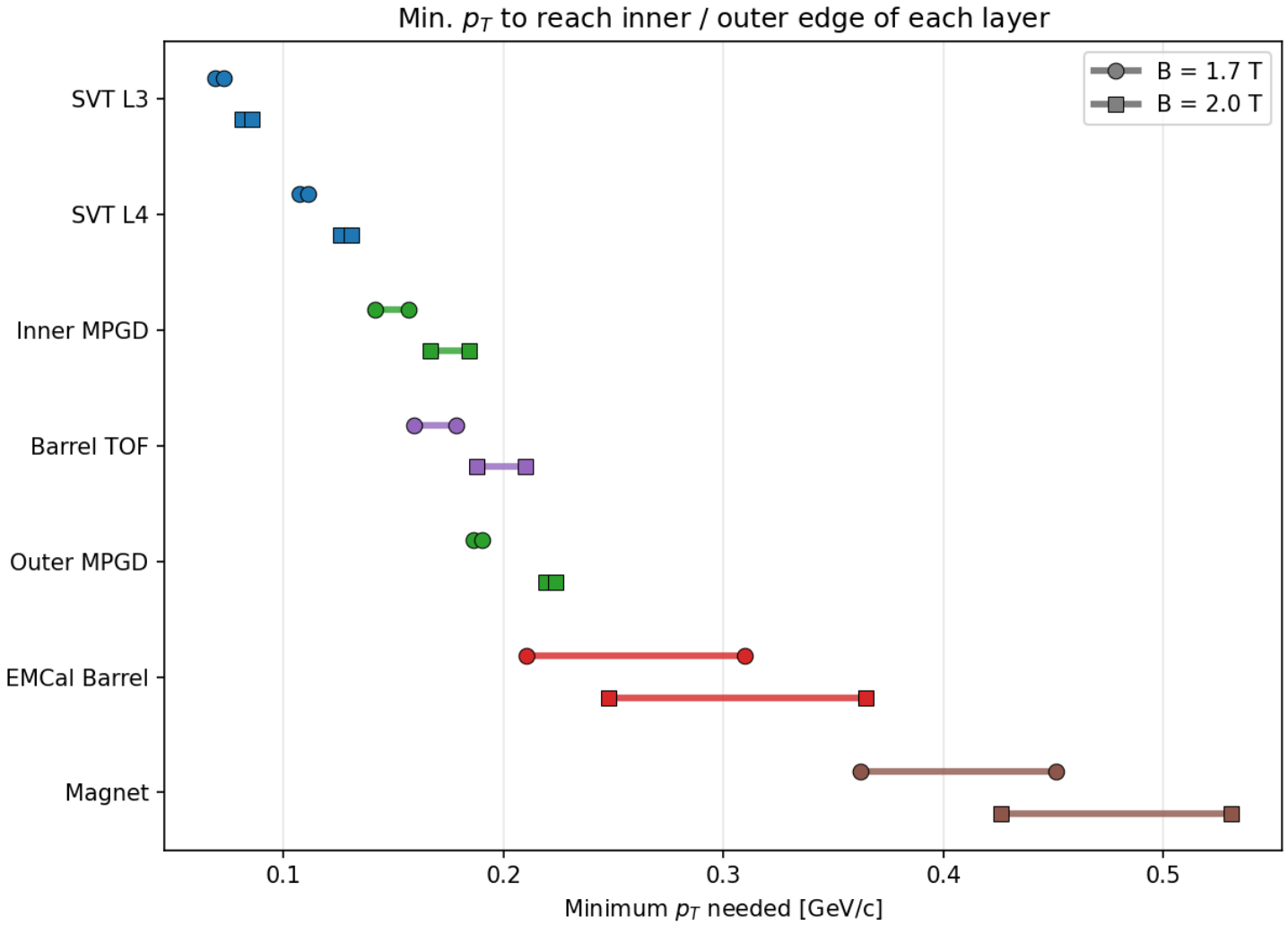
$$R_{max} = 2R = \frac{2\,p_T}{0.3\,q\,B}$$

Minimum p_T to reach radius r

$$p_T^{min} = 0.15\,q\,B\,r$$

p_T in GeV/c, B in T, r in m, q in units of e

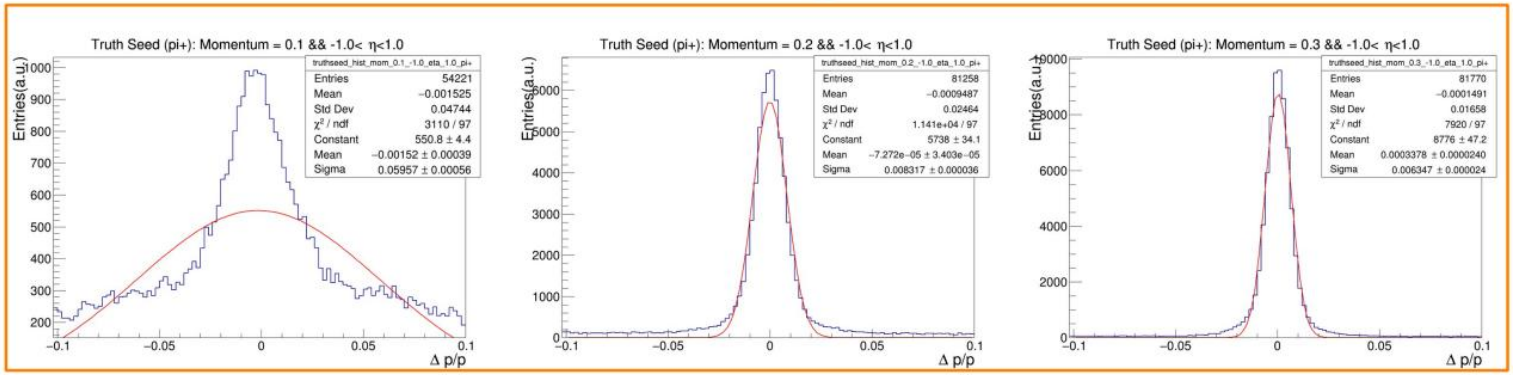
Ignoring energy loss



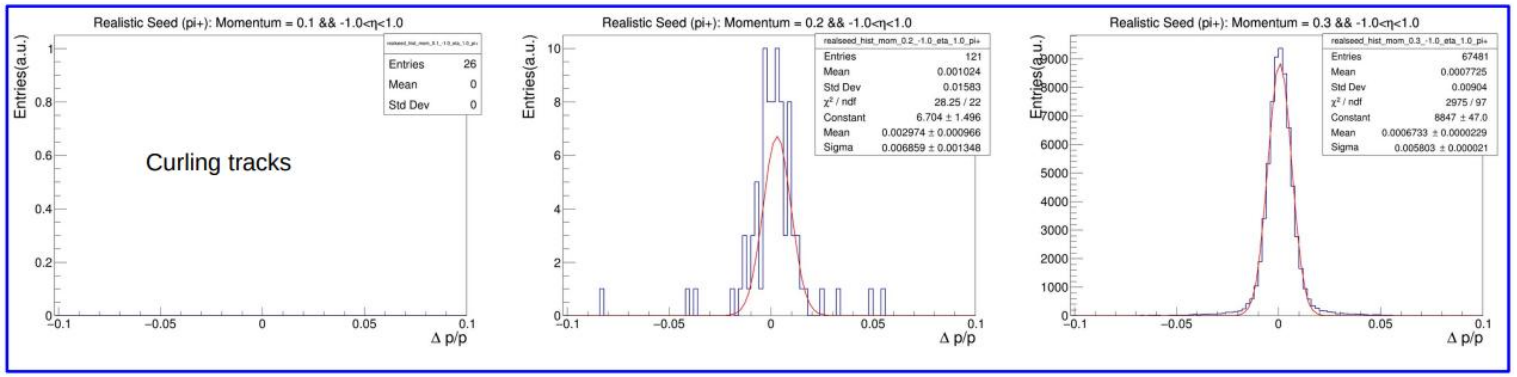
Low-momentum track reconstruction in barrel

Work by Shyam Kumar (April 2025):
https://indico.bnl.gov/event/23130/contributions/90774/attachments/54104/92571/Tracking_Performances_Shyam_25April24.pdf

Truth Seeding



Realistic Seeding



- Shyam showed that track-finding efficiency (i.e. real-seeded tracking) performance is poor for poor for tracks with 200 MeV/c momentum in the barrel.
- Other studies see similar performance (see presentation by Jeet later in this session).
- Truth-seeded tracking at 200 MeV/c momentum shows reasonable performance, suggesting lost efficiency is due to seeds not being formed.

Plan going forward

- Repeat Shyam's single-particle study with updated geometry and reconstruction (including material of barrel ECal and magnet).
- Study impact of new seeding framework that does not include previous hardcoded cut on seed angle (see details here: https://indico.bnl.gov/event/32582/contributions/123515/attachments/69619/119452/tracking_042326.pdf).
  - After this study, consider if a second pass of seeding and tracking is needed specifically for low-momentum particles.
- Hold more detailed discussions with the PWGs and the TOF group about what minimum momentum particles we need to reconstruct. For example, what is needed for exclusive phi photoproduction.

Part II: track states and pathlength error

- Three track states exist for a given track at any sensitive tracking surface (e.g. layer k):

1. Predicted track state: track state at layer k determined by previous k-1 layers
2. Filtered track state: track state at layer k including measurement at layer k
3. Smoothed track state: track state at layer k taking all measurements into account

What information we currently save to our ROOT file

- In our CentralTrackSegments collection, we save the predicted track state at every layer.
- Since the predicted track states are used for the chi-square determination, this is useful for many studies.
- However, having the final, smoothed track state information is useful for TOF PID estimates, for example.

What information we currently save to our ROOT file

- Current algorithm for CentralTrackSegments:

https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/TrackProjector.cc

```
92            // get track state bound parameters and their boundCovs
93            const auto& boundParams = trackstate.predicted();
94            const auto& boundCov    = trackstate.predictedCovariance();
95
```

- Plan is to add additional filtered and smoothed collections to the output.

Pathlength error

- Our track propagator algorithm, which is used to project the track to non-tracking layers, uses the final fitted track state:

https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/TrackPropagation.cc

```
241       // Get track state at last measurement surface
242       // For last measurement surface, filtered and smoothed results are equivalent
243       auto trackState        = trackContainer.trackStateContainer().getTrackState(tipIndex);
244       auto initSurface       = trackState.referenceSurface().getSharedPtr();
245       const auto& initParams = trackState.filtered();
246       const auto& initCov    = trackState.filteredCovariance();
```

- However, for both the CentralTrackSegments and the propagator, we don’t calculate path-length error.

```
152       const auto pathLength       = static_cast<float>(trackstate.pathLength());
153       const float pathLengthError = 0;
```

Pathlength calculation with Acts parameterization

Simple calculation of the error on the pathlength:

$$\sigma_s^2 = \vec{J}^T C \vec{J} = \frac{s^2}{(q/p)^2}\sigma_{q/p}^2 + 4s^2 \cot^2\theta\,\sigma_\theta^2 + 2 \cdot \frac{s}{q/p} \cdot (-2s\cot\theta) \cdot \mathrm{Cov}(q/p, \theta)$$

Based on some discussion / literature search, this is a simplified formula that may be missing energy-loss effects.

Pathlength calculation with Acts parameterization

Simple calculation of the error on the pathlength:

$$\sigma_s^2 = \vec{J}^T C \vec{J} = \frac{s^2}{(q/p)^2}\sigma_{q/p}^2 + 4s^2 \cot^2\theta\,\sigma_\theta^2 + 2 \cdot \frac{s}{q/p} \cdot (-2s\cot\theta) \cdot \mathrm{Cov}(q/p, \theta)$$

Based on some discussion / literature search, this is a simplified formula that may be missing energy-loss effects.

Plan going forward is not to implement this into EICRecon, but rather to speak to Acts about adding a calculation of the path-length error directly into Acts. We can work with them on the implementation.