ePIC Electron Finder: Status and Next Steps

Tristan Protzman
Lehigh University
April 18th, 2024

Electron Identification

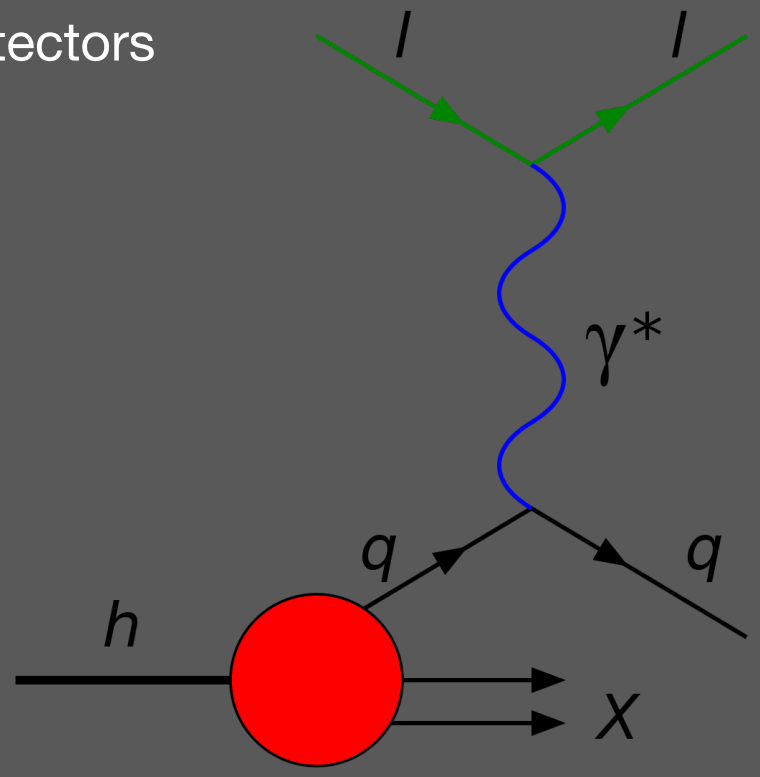
Goal: Develop an efficient and accurate algorithm for identifying electrons and identifying the scattered electron in DIS processes

Electron identification depends on the integration of multiple detectors

- Tracking -> momentum
- Calorimetry -> energy
- PID detectors -> additional e/hadron separation

Used in current algorithm:

- ~~Tracking~~ Particles
- Calorimetry
- ~~PID Detectors~~



Electron Identification

Tyler Kutz

eID currently relies on particle – cluster matching and E/p cut

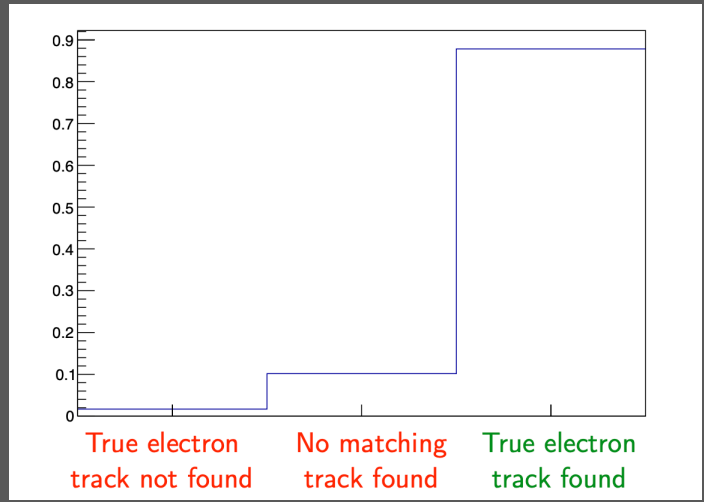
Cluster Matching

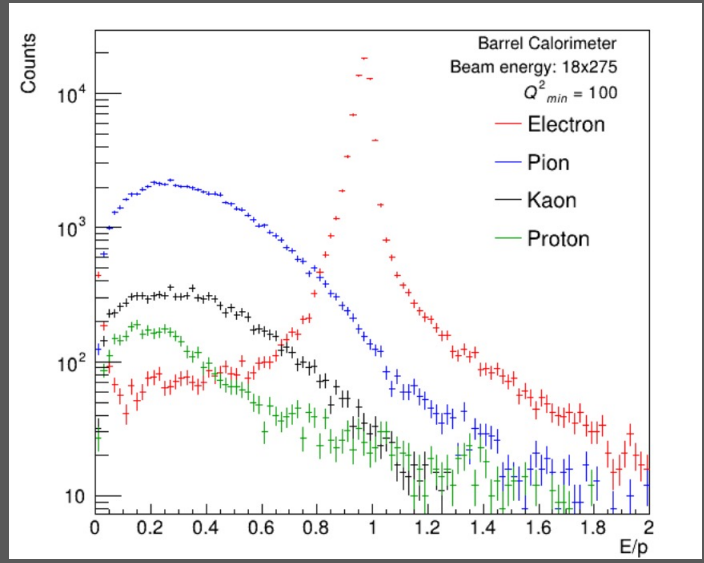
- `MatchClusters` factory
- Truth info to associate particles to clusters
- Next step: Match without truth information

E/p Cut

- Two collections produced by `ReconstructedElectrons` factory
  - `ReconstructedElectrons`: $0.9 < E/p < 1.2$
  - `ReconstructedElectronsForDIS`: $0.7 < E/p < 1.3$

- Both collections are subsets of `ReconstructedParticles`
- Associations between collections are maintained





Scattered Electron Finders

- **Goal: Identify the DIS lepton using only final state information**
  - How do select the DIS electron given multiple candidates?

- Two scattered electron finder algorithms implemented for April release
- Baseline algorithms to benchmark more advance algorithms against

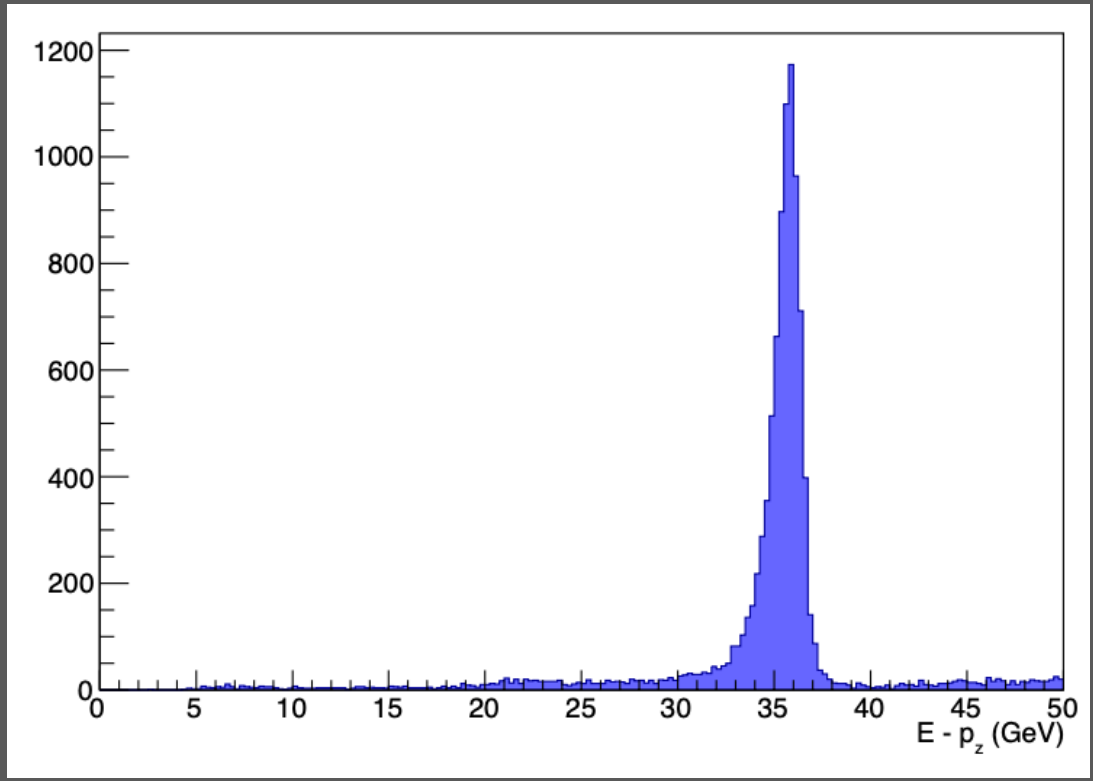
Scattered Electrons Truth

- Identifies first final state electron from MC collection as scattered electron
- Associates truth electron with reconstructed particles by object ID

Scattered Electron Finders

Scattered Electrons $E - p_z$

- Sums 4-momentum of electron candidate and hadronic final state
- Calculate `(e+X).E()-(e+X).Pz()` for each candidate electron
  - `X` is the sum of final state hadrons
- Candidate with greatest $E - p_z$ is given as the most likely scattered electron
- Even without strict electron PID, right "most" of the time



Scattered Electron Finders

The two algorithms can be used independently to check performance

```
[reco:ScatteredElectronsEMinusPz] [trace] We have 3 candidate electrons
[reco:ScatteredElectronsEMinusPz] [trace] Skipping electron in hadronic final state
[reco:ScatteredElectronsEMinusPz] [trace]       E-Pz=17.044048885959747
[reco:ScatteredElectronsEMinusPz] [trace]       ScatteredElectron has Pxyz=( 1.0658381, 2.5452766, -8.107633 )
[reco:ScatteredElectronsEMinusPz] [trace] skipping positron
[reco:ScatteredElectronsEMinusPz] [trace] skipping positron
[reco:ScatteredElectronsEMinusPz] [trace] Selecting candidates with 0 < E-Pz < 10000000
[reco:ScatteredElectronsEMinusPz] [trace] Max E-Pz Candidate:
[reco:ScatteredElectronsEMinusPz] [trace]       E-Pz=17.044048885959747
[reco:ScatteredElectronsEMinusPz] [trace]       ScatteredElectron has Pxyz=( 1.0658381, 2.5452766, -8.107633 )
[reco:ScatteredElectronsTruth] [trace] We found 1 scattered electrons using Truth assocaition
[reco:ScatteredElectronsTruth] [trace] TRUTH scattered electron has E-Pz = 17.044048885959747
[reco:ScatteredElectronsTruth] [trace] TRUTH scattered electron has Pxyz=( 1.065838098526001, 2.545276641845703, -8.107632637023926 )
 and E/p = 1.0000000507582056
```

These are the simplest algorithms – this is the setup to enable the development of more complex algorithms (Are the final state kinematics consistent, semi-hard radiation clustering, ML algorithms, …)

Using the electron finder – EICRecon

Collection names

- `ReconstructedElectrons`
- `ReconstructedElectronsForDIS`
- `ScatteredElectronsTruth`
- `ScatteredElectronsEMinusPz`

```cpp
void eID_testProcessor::ProcessSequential(const std::shared_ptr<const JEvent>& event) {
    // Get the collection of electrons output by the electron ID factory
    const auto &electrons = *(event->GetCollection<edm4eic::ReconstructedParticle>("ReconstructedElectrons"));
    for (const auto electron : electrons) {
        identified_electrons_position->Fill(edm4hep::utils::eta(electron.getMomentum()), edm4hep::utils::angleAzimuthal(electron.getMomentum()));
        identified_electrons_pt->Fill(edm4hep::utils::magnitude(electron.getMomentum()));
    }

    // Get the scattered electron collections
    const auto &scattered_electrons_truth = *(event->GetCollection<edm4eic::ReconstructedParticle>("ScatteredElectronsTruth"));
    const auto &scattered_electrons_EMinusPz = *(event->GetCollection<edm4eic::ReconstructedParticle>("ScatteredElectronsEMinusPz"));

    // Truth scattered electrons
    for (const auto electron : scattered_electrons_truth) {
        scattered_electron_truth_pt->Fill(edm4hep::utils::magnitude(electron.getMomentum()));
    }

    // Scattered electrons with E - Pz
    for (const auto electron : scattered_electrons_EMinusPz) {
        scattered_electron_EMinusPz_pt->Fill(edm4hep::utils::magnitude(electron.getMomentum()));
    }
}
```

All are subsets of `ReconstructedParticles`, so they share the same particle ID

Using the electron finder – Analysis Scripts

- Can be read with Podio reader
- Easily makes use of subcollections
- Working to develop C++ and Python examples
- Will add to presentation and snippets repository

Current limitations and next steps

Limitations

- Using truth information
- Using particles, not tracks
- No PID detector information utilized

Next Steps

- Use tracks, not particles
- Remove truth information dependency
  - Requires good track–cluster matching
- Incorporate PID information
- Feedback from analyzers – what's missing? What can be improved?

Summary

- First pass of electron finder included in April release and simulation campaign!
- Truth algorithm for benchmarking, $E - p_Z$ algorithm for more realistic ID
  - More algorithms to come
- Next steps focus on moving away from using truth information
- **Need feedback from users!**
- Thanks to Daniel, Tyler, Markus, Wouter, and all others who helped with development and review!