Lancaster University

Truth matching utilities in LArSoft

Dom Brailsford
FD sim/reco workshop
18th September 2020

Truth matching utilities in LArSoft

- Truth matching is one of the most important tools in a HEP analysis
- To the best of my knowledge, there are no utilities which match multiple hits to a true particle in LArSoft
- There are (probably) a ‘bazillion’ re-inventions of truth matching in the various experiment codebases
- I’ve committed a set of backtracker-based truth matching utilities to larsim which should be experiment/detector agnostic*
    - larsim/Utils/TruthMatchUtils.h

- Important: I am —not— saying that these tools are in any way better than your/your group’s bespoke tools
- Also important: I —am— saying that we should be sharing tools between groups and experiments where possible

- This talk overviews what is in the new utility library
    - Feedback, comments, criticisms and (most importantly) contributions are welcome

***tools only work on files/events which have not removed simchannel info**

Truth matching functions

- Three distinct, freestanding functions available
- All return the g4 ID of the best-matching true particle for a vector of recob::hits
- Wrapped in *TruthMatchUtils* namespace
- All have the following form

```
G4ID TrueParticleIDFromTotalRecoHits(detinfo::DetectorClocksData const& clockData,
                                     const std::vector<art::Ptr<recob::Hit>>& pHits,
                                     const bool rollupUnsavedIDs);
```

Typedef’d to an int

Truth matching functions

- Three distinct, freestanding functions available
- All return the g4 ID of the best-matching true particle for a vector of recob::hits
- Wrapped in *TruthMatchUtils* namespace
- All have the following form

Needed for backtracker (This snuck in during other LArSoft changes*)

```
G4ID TrueParticleIDFromTotalRecoHits(detinfo::DetectorClocksData const& clockData,
                                      const std::vector<art::Ptr<recob::Hit>>& pHits,
                                      const bool rollupUnsavedIDs);
```

***I’m hopeful that the clockData dependency can be hidden from the user**

Truth matching functions

- Three distinct, freestanding functions available
- All return the g4 ID of the best-matching true particle for a vector of recob::hits
- Wrapped in *TruthMatchUtils* namespace
- All have the following form

```
G4ID TrueParticleIDFromTotalRecoHits(detinfo::DetectorClocksData const& clockData,
                                      const std::vector<art::Ptr<recob::Hit>>& pHits,
                                      const bool rollupUnsavedIDs);
```

The vector of hits that you’re truth matching to

Truth matching functions

- Three distinct, freestanding functions available
- All return the g4 ID of the best-matching true particle for a vector of recob::hits
- Wrapped in *TruthMatchUtils* namespace
- All have the following form

```
G4ID TrueParticleIDFromTotalRecoHits(detinfo::DetectorClocksData const& clockData,
                                      const std::vector<art::Ptr<recob::Hit>>& pHits,
                                      const bool rollupUnsavedIDs);
```

Counts energy depositions of non-saved child particles as coming from the primary parent (think showers)

Truth matching functions

Key

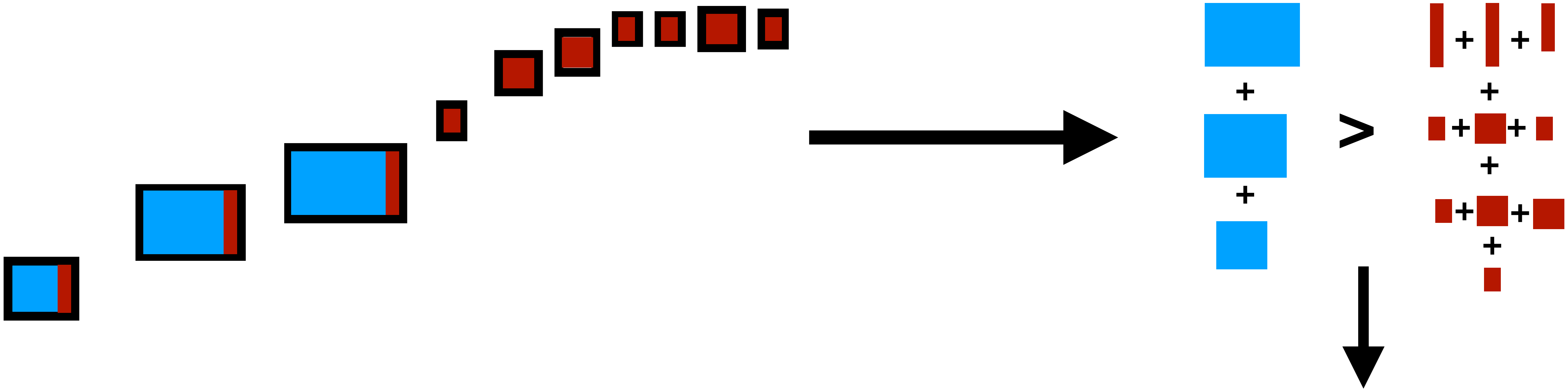
□ **Reconstructed hit**

🟦 **True proton energy dep.**

🟥 **True muon energy dep.**

Size indicates charge/energy magnitude

1) TrueParticleIDFromTotalTrueEnergy(clockData, hits, rollupIDs)



Hit vector is truth matched to proton

- Function returns g4 ID of particle which **deposits the most energy in the hit vector**
- How the energy depositions are distributed amongst the hits does not matter

Truth matching functions

Key

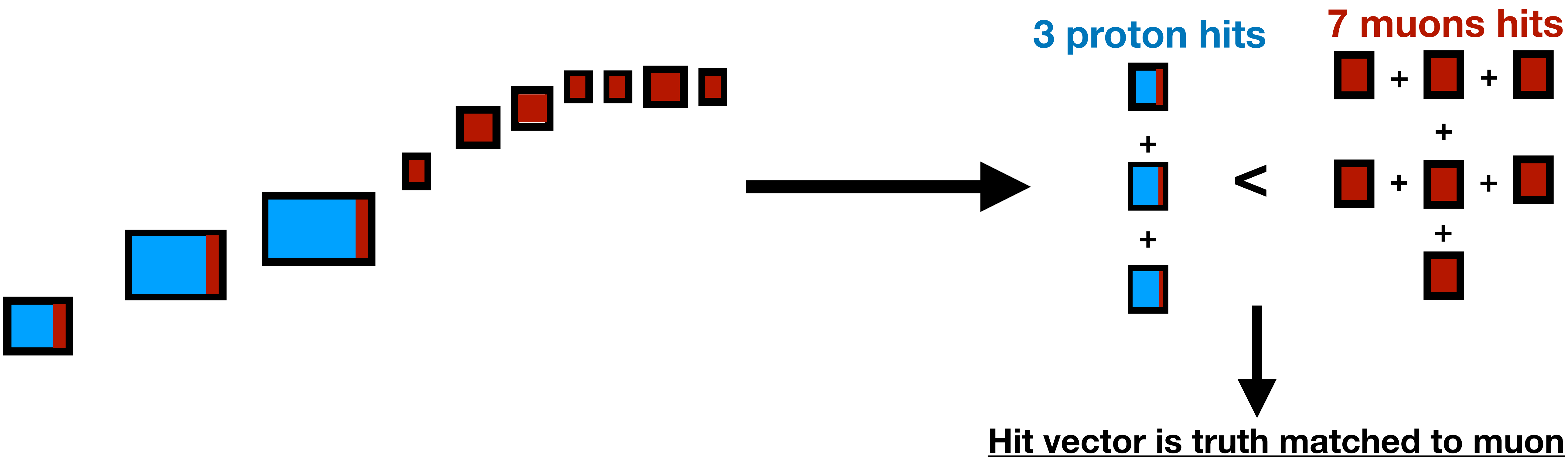
□ **Reconstructed hit**

🟦 **True proton energy dep.**

🟥 **True muon energy dep.**

Size indicates charge/energy magnitude

2) TrueParticleIDFromTotalRecoHits(clockData, hits, rollupIDs)



3 proton hits < **7 muons hits**

Hit vector is truth matched to muon

- Function returns g4 ID of particle which **is the primary contributor to the most hits**
- Sculpted by how the energy depositions are distributed amongst the hits

Truth matching functions

Key

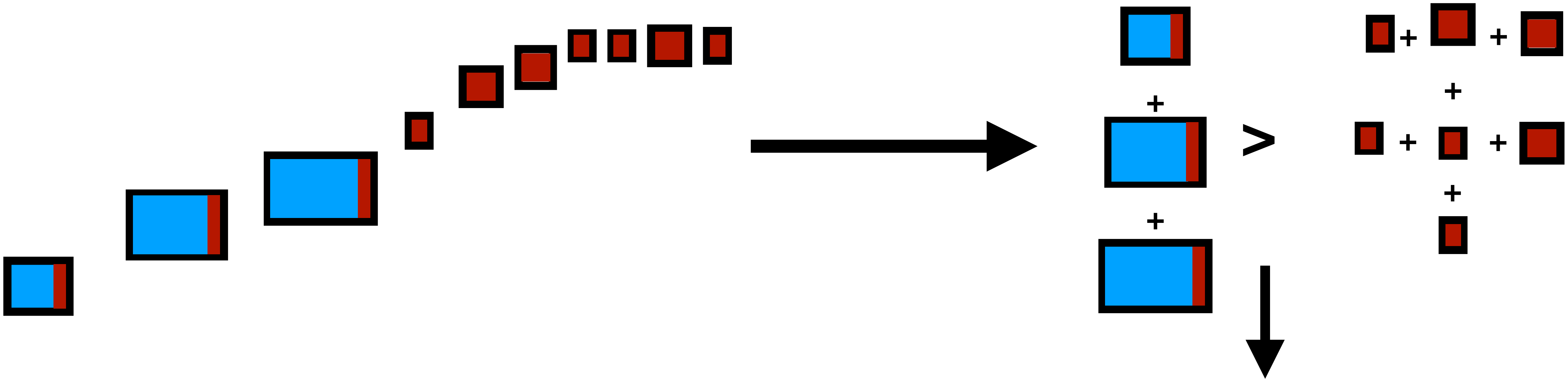
□ **Reconstructed hit**

🟦 **True proton energy dep.**

🟥 **True muon energy dep.**

Size indicates charge/energy magnitude

3) TrueParticleIDFromTotalRecoCharge(clockData, hits, rollupIDs)



Hit vector is truth matched to proton

- Function returns g4 ID of particle **whose matched hits sum to the largest reco. charge**
- This function was mostly added for completeness

What else is available?

```
bool Valid(const G4ID g4ID) noexcept;
```

Tells you whether a returned g4 ID is valid

```
G4ID TrueParticleID(detinfo::DetectorClocksData const& clockData,
                    const art::Ptr<recob::Hit>& pHit,
                    const bool rollupUnsavedIDs);
```

```
void FillG4IDToEnergyDepositMap(IDToEDepositMap& idToEDepMap,
                                detinfo::DetectorClocksData const& clockData,
                                const art::Ptr<recob::Hit>& pHit,
                                const bool rollupUnsavedIDs);
```

What else is available?

```
bool Valid(const G4ID g4ID) noexcept;
```

Tells you whether a returned g4 ID is valid

```
G4ID TrueParticleID(detinfo::DetectorClocksData const& clockData,
                    const art::Ptr<recob::Hit>& pHit,
                    const bool rollupUnsavedIDs);
```

Returns the matched G4 ID for a single recob::Hit

```
void FillG4IDToEnergyDepositMap(IDToEDepositMap& idToEDepMap,
                                detinfo::DetectorClocksData const& clockData,
                                const art::Ptr<recob::Hit>& pHit,
                                const bool rollupUnsavedIDs);
```

What else is available?

```
bool Valid(const G4ID g4ID) noexcept;
```

Tells you whether a returned g4 ID is valid

```
G4ID TrueParticleID(detinfo::DetectorClocksData const& clockData,
                    const art::Ptr<recob::Hit>& pHit,
                    const bool rollupUnsavedIDs);
```

Returns the matched G4 ID for a single recob::Hit

```
void FillG4IDToEnergyDepositMap(IDToEDepositMap& idToEDepMap,
                                detinfo::DetectorClocksData const& clockData,
                                const art::Ptr<recob::Hit>& pHit,
                                const bool rollupUnsavedIDs);
```

Fills a map that maps g4 ID for energy deposition for a single recob::Hit

Example (pseudo)code snippet

Better to store as data member and read from fcl

```
void example::analysis::analyze(const art::Event &e){
    auto const clockData(art::ServiceHandle<detinfo::DetectorClocksService const>()->DataFor(e));

    const std::vector<art::Ptr<recob::PFParticle>> pfps(dune_ana::DUNEAnaEventUtils::GetPFParticles(e, "pandora"));
    for (const art::Ptr<recob::PFParticle> pfp: pfps)
    {
        const std::vector<art::Ptr<recob::Hit>> hits(dune_ana::DUNEAnaPFParticleUtils::GetHits(pfp, e, "pandora"));
        TruthMatchUtils::G4ID g4ID(TruthMatchUtils::TrueParticleIDFromTotalRecoHits(clockData, hits, true));
        if (TruthMatchUtils::Valid(g4ID))
        {
            //....
        } //if g4ID
    } //pfps
} //analyze
```

***Code compilation not checked**

Future features

- Completeness/hit purity calculators
- Handle showers when shower children are kept
- Wrapper function to retrieve simb::MCParticles from the particle inventory service?
- Whatever you want

Summary

- I’ve committed a set of (hopefully) simple-to-use truth matching utilities to larsim
- I am hoping that the new utility library serves as a base for us to share more utilities between groups and experiments
- Feedback, comments, criticisms and **contributions** are most welcome