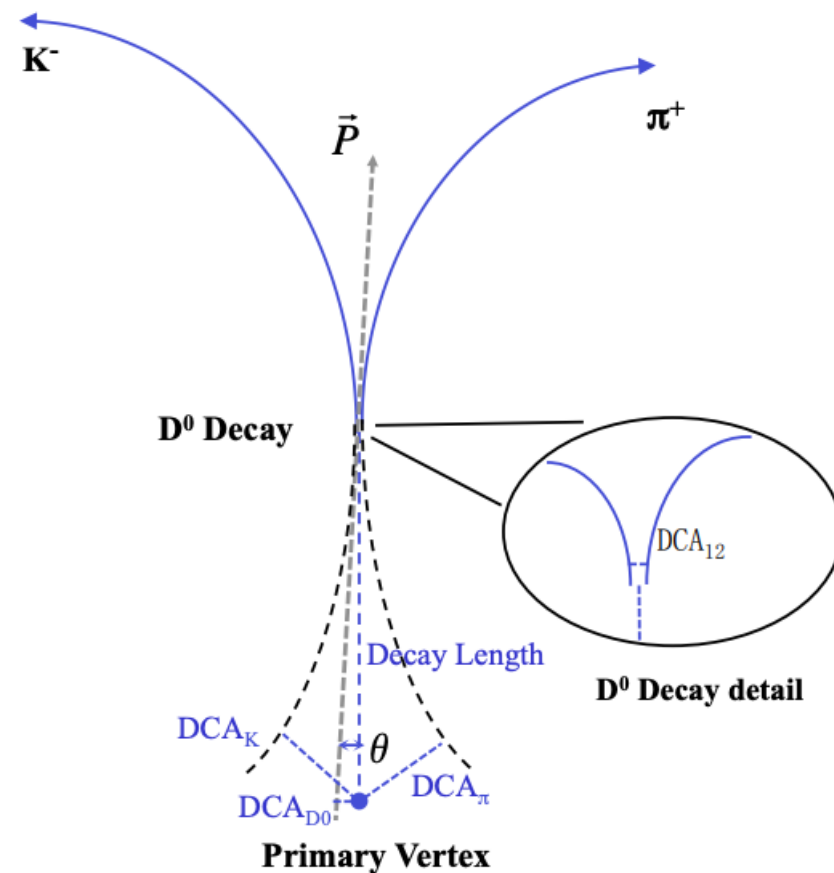


Helix and Secondary Vertices Factory in ElCrecon

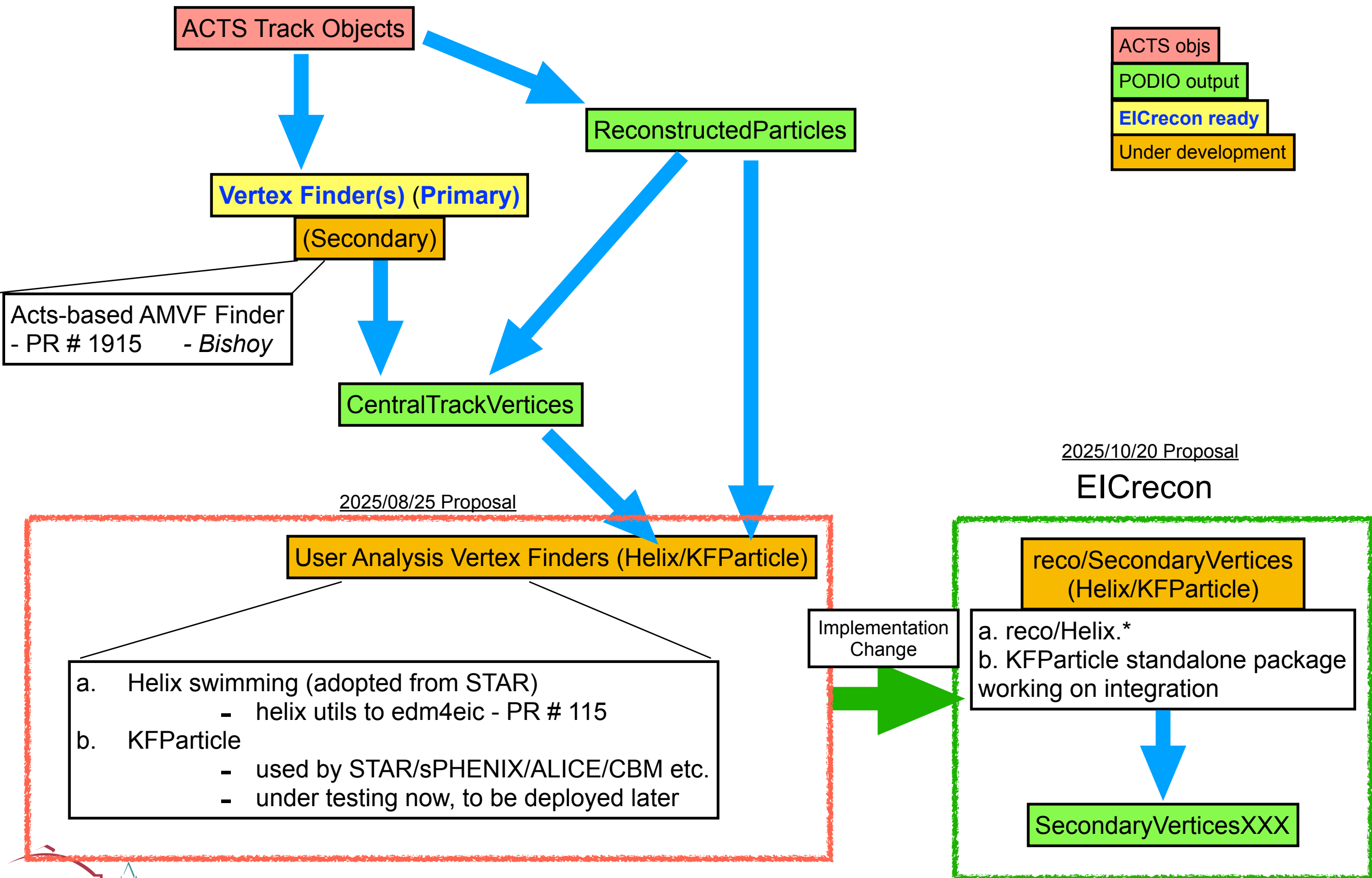
Xin Dong



Thanks to:

Bishoy Dongwi, Lokesh Kumar, Shyam Kumar, Rongrong Ma, Joe Osborn, Ashish Pandav, *Harsimran Singh*, *Khushi Singla*, Deepa Thomas, *Connie Yang* etc.

Vertex Finders



ElCrecon: PR #2144

+ [src/algorithms/reco/Helix.cc](#)

+ [src/algorithms/reco/Helix.h](#)

+ [src/algorithms/reco/SecondaryVerticesHelix.cc](#)

+ [src/algorithms/reco/SecondaryVerticesHelix.h](#)

+ [src/algorithms/reco/SecondaryVerticesHelixConfig.h](#)

+ [src/factories/reco/SecondaryVerticesHelix_factory.h](#)

• [src/global/reco/reco.cc](#)

• [src/services/io/podio/JEventProcessorPODIO.cc](#)

adopted from STAR code, adjusted for ElCrecon

```
/// ReconstructParticle, b field  
Helix(const edm4eic::ReconstructedParticle& p, const double b_field);
```

1. new SecondaryVertices factory, now based on Helix, can be expanded for others (e.g. KFParticle)
2. take ReconstructedParticle as input, run secondary finder (Ks included so far, can be expanded to include others)
3. output -> SecondaryVerticesHelix (edm4eic::Vertex) in PODIO
- *proposed to add edm4eic::SecondaryVertex container (next slide)*
4. geometric cut variables controlled by SecondaryVerticesHelixConfig.h

SecondaryVerticesHelix Factory

```
if ((*rcvtx).size() == 0) {
    info(" No primary vertex in this event! Skip secondary vertex finder!");
    return;
}
const auto pVtxPos4f = (*rcvtx)[0].getPosition();
// convert to cm
edm4hep::Vector3f pVtxPos(pVtxPos4f.x * edm4eic::unit::mm / edm4eic::unit::cm,
                          pVtxPos4f.y * edm4eic::unit::mm / edm4eic::unit::cm,
                          pVtxPos4f.z * edm4eic::unit::mm / edm4eic::unit::cm);

auto fieldObj = m_det->field();
auto field = fieldObj.magneticField(
    {pVtxPos4f.x / edm4eic::unit::mm * dd4hep::
      pVtxPos4f.z / edm4eic::unit::mm * dd4h
float b_field = field.z();

info("\t Primary vertex = ({},{},{})cm \t b fi
      b_field / dd4hep::tesla);

std::vector<Helix> hVec;
hVec.clear();
std::vector<unsigned int> indexVec;
indexVec.clear();
for (unsigned int i = 0; const auto& p : *rcpa
    if (p.getCharge() == 0)
        continue;
    Helix h(p, b_field);
    double dca = h.distance(pVtxPos) * edm4eic::
    if (dca < m_cfg.minDca)
        continue;

hVec.push_back(h);
indexVec.push_back(i);
```

```
for (unsigned int i1 = 0; i1 < hVec.size(); ++i1) {
    for (unsigned int i2 = i1 + 1; i2 < hVec.size(); ++i2) {
        const auto& p1 = (*rcparts)[indexVec[i1]];
        const auto& p2 = (*rcparts)[indexVec[i2]];

        if (!(m_cfg.unlikesign && p1.getCharge() + p2.getCharge() == 0))
            continue;

        const auto& h1 = hVec[i1];
        const auto& h2 = hVec[i2];

        // Helix function uses cm unit
        double dca1 = h1.distance(pVtxPos) * edm4eic::unit::cm;
        double dca2 = h2.distance(pVtxPos) * edm4eic::unit::cm;
        if (dca1 < m_cfg.minDca || dca2 < m_cfg.minDca)
            continue;

        std::pair<double, double> const ss = h1.pathLengths(h2);
        edm4hep::Vector3f h1AtDcaTo2 = h1.at(ss.first);
        edm4hep::Vector3f h2AtDcaTo1 = h2.at(ss.second);

        double dca12 = edm4hep::utils::magnitude(h1AtDcaTo2 - h2AtDcaTo1) * edm4eic::unit::cm;
        if (std::isnan(dca12))
            continue;
        if (dca12 > m_cfg.maxDca12)
            continue;

        edm4hep::Vector3f pairPos = 0.5 * (h1AtDcaTo2 + h2AtDcaTo1);

        edm4hep::Vector3f h1MomAtDca = h1.momentumAt(ss.first, m_cfg.b_field);
        edm4hep::Vector3f h2MomAtDca = h2.momentumAt(ss.second, m_cfg.b_field);
        edm4hep::Vector3f pairMom = h1MomAtDca + h2MomAtDca;
```

```
struct SecondaryVerticesHelixConfig {
    bool unlikesign = true;
    float minDca = 0.03 * edm4eic::unit::mm; // mm, daughter to pVtx
    float maxDca12 = 1. * edm4eic::unit::mm; // mm, dca between daughter 1 and 2
    float maxDca = 1. * edm4eic::unit::mm; // mm, dca of V0 to pVtx
    float minCostheta = 0.8; // costheta, theta: angle of V0 decay direction and momentum
};
```

initial config file

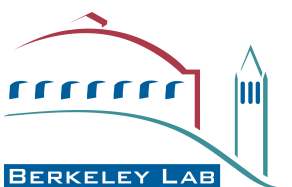
```
double e1 =
    std::hypot(edm4hep::utils::magnitude(h1MomAtDca), particleSvc.particle(p1.getPDG()).mass);
double e2 =
    std::hypot(edm4hep::utils::magnitude(h2MomAtDca), particleSvc.particle(p2.getPDG()).mass);
double pairE = e1 + e2;
// double pairP = edm4hep::utils::magnitude(pairMom);

// double m_inv2 = pairE * pairE - pairP * pairP;
// double m_inv = (m_inv2 > 0) ? sqrt(m_inv2) : 0.;
double angle = edm4hep::utils::angleBetween(pairMom, pairPos - pVtxPos);
if (cos(angle) < m_cfg.minCostheta)
    continue;

double beta = edm4hep::utils::magnitude(pairMom) / pairE;
double time = edm4hep::utils::magnitude(pairPos - pVtxPos) / (beta * dd4hep::c_light);
edm4hep::Vector3f dL = pairPos - pVtxPos; // in cm
edm4hep::Vector3f decayL(dL.x * edm4eic::unit::cm, dL.y * edm4eic::unit::cm,
                        dL.z * edm4eic::unit::cm);
double dca2pv = edm4hep::utils::magnitude(decayL) * sin(angle);
if (dca2pv > m_cfg.maxDca)
    continue;

auto v0 = out_secondary_vertices->create();
v0.setType(2); // 2 for secondary
v0.setPosition({(float)(pairPos.x * edm4eic::unit::cm / edm4eic::unit::mm),
                (float)(pairPos.y * edm4eic::unit::cm / edm4eic::unit::mm),
                (float)(pairPos.z * edm4eic::unit::cm / edm4eic::unit::mm), (float)time});
v0.addToAssociatedParticles(p1);
v0.addToAssociatedParticles(p2);

info("One secondary vertex found at (x,y,z) = ({}, {}, {}) mm.",
     pairPos.x * edm4eic::unit::cm / edm4eic::unit::mm,
     pairPos.y * edm4eic::unit::cm / edm4eic::unit::mm,
     pairPos.z * edm4eic::unit::cm / edm4eic::unit::mm);
```



SecondaryVertex to edm4eic: add-secondaryvertex branch

1. Current edm4eic::vertex object doesn't contain many topological variables for secondary vertices.
2. Though one can in principle re-calculate all these based on the associated ReconstructedParticle, it will be much more convenient to save them during reconstruction and the down-stream analysis can directly use them for physics analysis and also this will avoid repeated calculations.

Propose to add edm4eic::SecondaryVertex container

<https://github.com/eic/EDM4eic/issues/127>

```
edm4eic::Vertex:
Description: "EIC vertex"
Author: "J. Osborn"
Members:
- int32_t          type          // Type flag, to identify what type of vertex it is (e.g. primary or secondary)
- float            chi2          // Chi-squared of the vertex fit
- int              ndf           // NDF of the vertex fit
- edm4hep::Vector4f position     // position [mm] + time t0 [ns] of the vertex
## this is named "covMatrix" in EDM4hep, renamed for consistency with the edm4eic::Cov4f
- edm4eic::Cov4f   positionError // Covariance matrix of the position+time
OneToManyRelations:
- edm4eic::ReconstructedParticle associatedParticles // particles associated to this vertex.
```

```
edm4hep::Vertex:
Description: "Vertex"
Author: "EDM4hep authors"
Members:
- uint32_t          type          // flagword that defines the type of the vertex,
- float            chi2          // chi-squared of the vertex fit
- int32_t          ndf           // number of degrees of freedom of the vertex fit
- edm4hep::Vector3f position [mm] // position of the vertex
- edm4hep::CovMatrix3f covMatrix [mm^2] // covariance matrix of the position
- int32_t          algorithmType // type code for the algorithm that has been used
VectorMembers:
- float            parameters     // additional parameters related to this vertex
OneToManyRelations:
- edm4hep::ReconstructedParticle particles // particles that have been used to form this vertex.
```

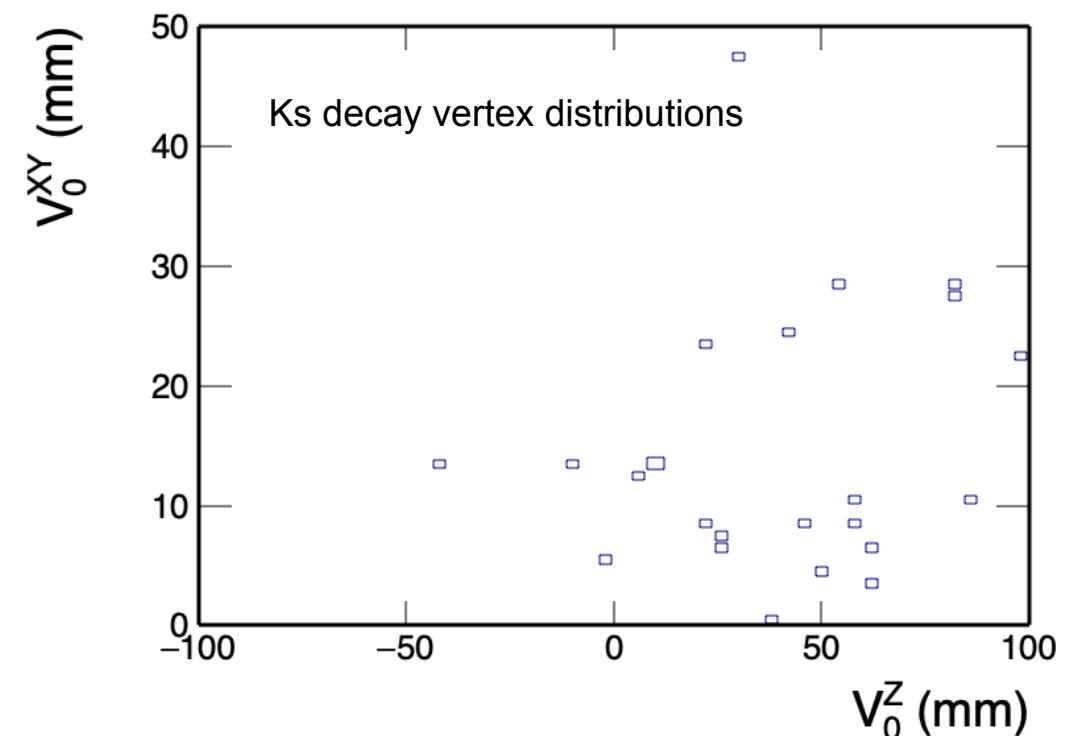
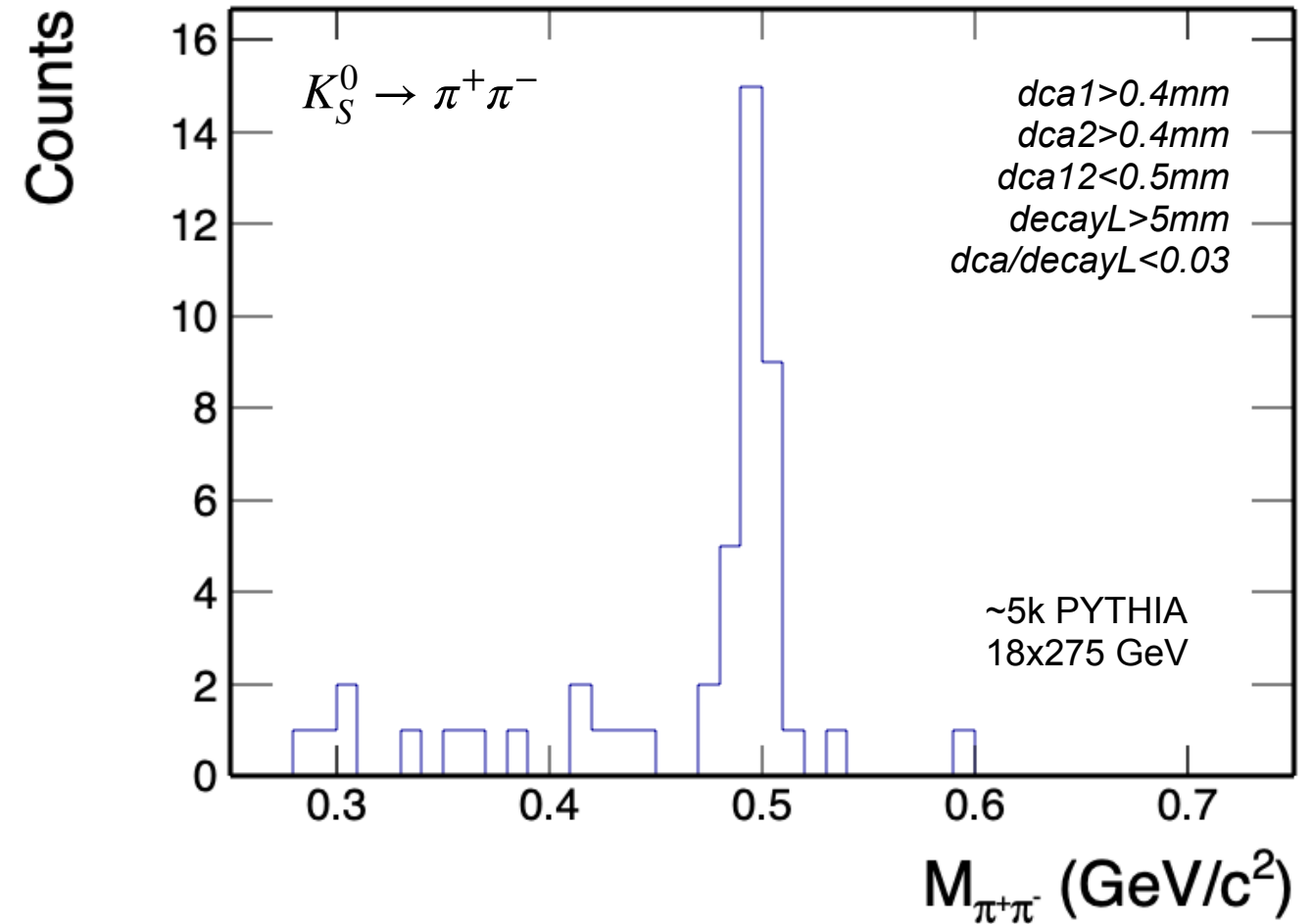
```
edm4eic::SecondaryVertex:
Description: "EIC secondary vertex"
Author: "X. Dong"
Members:
- int32_t          type          // Type flag, to identify what type of vertex it is (e.g. primary or secondary)
- float            chi2          // Chi-squared of the vertex fit
- int              ndf           // NDF of the vertex fit
- edm4hep::Vector4f position     // position [mm] + time t0 [ns] of the vertex. Time is 4th component
- edm4eic::Cov4f   positionError // Covariance matrix of the position+time. Time is 4th component
- edm4hep::Vector3f parentMomentum // parent momentum
- float            parentInvariantMass // parent invariant mass
- float            parentInvariantMassError // parent invariant mass error
- float            parentDecayLength // parent decay length
- float            parentDecayLengthError // parent decay length error
- float            parentDca2PV // parent dca to primary vertex
- float            parentDca2PVErr // parent dca_error to primary vertex
VectorMembers:
- edm4hep::Vector3f daughterMomentum // daughter track momentum at the decay vertex
- int              daughterPDG // daughter PDG
- float            daughterDca2PV // daughter dca to primary vertex
- float            daughterDca2PVErr // daughter dca_error to primary vertex
- int              daughterPairIndices // track indices for any pair
- float            daughterPairDca // dca between any pair of tracks
- float            daughterPairDcaError // dca_error between any pair of tracks
OneToOneRelations:
- edm4eic::Vertex primaryVertex // associated primary vertex
OneToManyRelations:
- edm4eic::ReconstructedParticle associatedParticles // particles associated to this vertex.
```

New structure has been integrated to EICrecon locally and tested to work well!
A new brunch add-secondaryvertex on edm4eic repo.

Test Output

PODIO output

	ScatteredElectronsTruth_objIdx
	SecondaryVerticesHelix
	SecondaryVerticesHelix.type
	SecondaryVerticesHelix.chi2
	SecondaryVerticesHelix.ndf
	SecondaryVerticesHelix.position.x
	SecondaryVerticesHelix.position.y
	SecondaryVerticesHelix.position.z
	SecondaryVerticesHelix.position.t
	SecondaryVerticesHelix.positionError.xx
	SecondaryVerticesHelix.positionError.yy
	SecondaryVerticesHelix.positionError.zz
	SecondaryVerticesHelix.positionError.tt
	SecondaryVerticesHelix.positionError.xy
	SecondaryVerticesHelix.positionError.xz
	SecondaryVerticesHelix.positionError.xt
	SecondaryVerticesHelix.positionError.yz
	SecondaryVerticesHelix.positionError.yt
	SecondaryVerticesHelix.positionError.zt
	SecondaryVerticesHelix.parentMomentum.x
	SecondaryVerticesHelix.parentMomentum.y
	SecondaryVerticesHelix.parentMomentum.z
	SecondaryVerticesHelix.parentDecayLength.x
	SecondaryVerticesHelix.parentDecayLength.y
	SecondaryVerticesHelix.parentDecayLength.z
	SecondaryVerticesHelix.parentInvariantMass
	SecondaryVerticesHelix.parentDecayLengthChi2
	SecondaryVerticesHelix.parentDca2PV
	SecondaryVerticesHelix.parentDca2PVChi2
	SecondaryVerticesHelix.daughterMomentum_begin
	SecondaryVerticesHelix.daughterMomentum_end
	SecondaryVerticesHelix.daughterMass_begin
	SecondaryVerticesHelix.daughterMass_end
	SecondaryVerticesHelix.daughterDca2PV_begin
	SecondaryVerticesHelix.daughterDca2PV_end
	SecondaryVerticesHelix.daughterDca2PVChi2_begin
	SecondaryVerticesHelix.daughterDca2PVChi2_end
	SecondaryVerticesHelix.daughterPairIndices_begin
	SecondaryVerticesHelix.daughterPairIndices_end



Summary

1. The proposal was presented and endorsed at the Reconstruction WG meeting 10/20.
 - current version is working with edm4eic::Vertex container
 - PR #2144 submitted under review
2. We will propose a new edm4eic container: SecondaryVertex (add-secondaryvertex brunch)
 - once this is added to edm4eic, will update SecondaryVertexHelix_factory then.
3. Continue the development of secondary VFs (Helix with more particles included, KFParticle etc.)

Backup

Adding Helix Functions in EDM4eic (obsolete!)

added helix functions (adopted from STAR) #115

Edit <> Code

Open starsdong wants to merge 1 commit into main from pr/helix_utils

Conversation 0

Commits 1

Checks 4

Files changed 3

+820 -0



starsdong commented 4 days ago

Member

Briefly, what does this PR introduce?

What kind of change does this PR introduce?

- ☐ Bug fix (issue #__)
- ☒ New feature (issue #__)
- ☐ Documentation update
- ☐ Other: __

Please check if this PR fulfills the following:

Reviewers

Suggestions

wdconinc

Request

At least 1 approving review is required to merge this pull request.

Still in progress? [Convert to draft](#)

Assignees

No one—[assign yourself](#)

Obsolete

- 1) Helix afterburner reconstruction is used in D^0 reconstruction (later part), targeted to be used for updated physics projection plots.
- 2) Constructor includes using EICrecon TrackParameters as input.
- 3) Handling constant z-magnetic field (or zero field - straight-line)
 - can be extended to handle varying B-field for track projection
- 4) Iterative varying-step-scan to find DCA positions between helices numerically.

Usage of Helix Method

Helix method has been used in many heavy flavor hadron analysis (Rongrong/Shyam/Connie etc.)

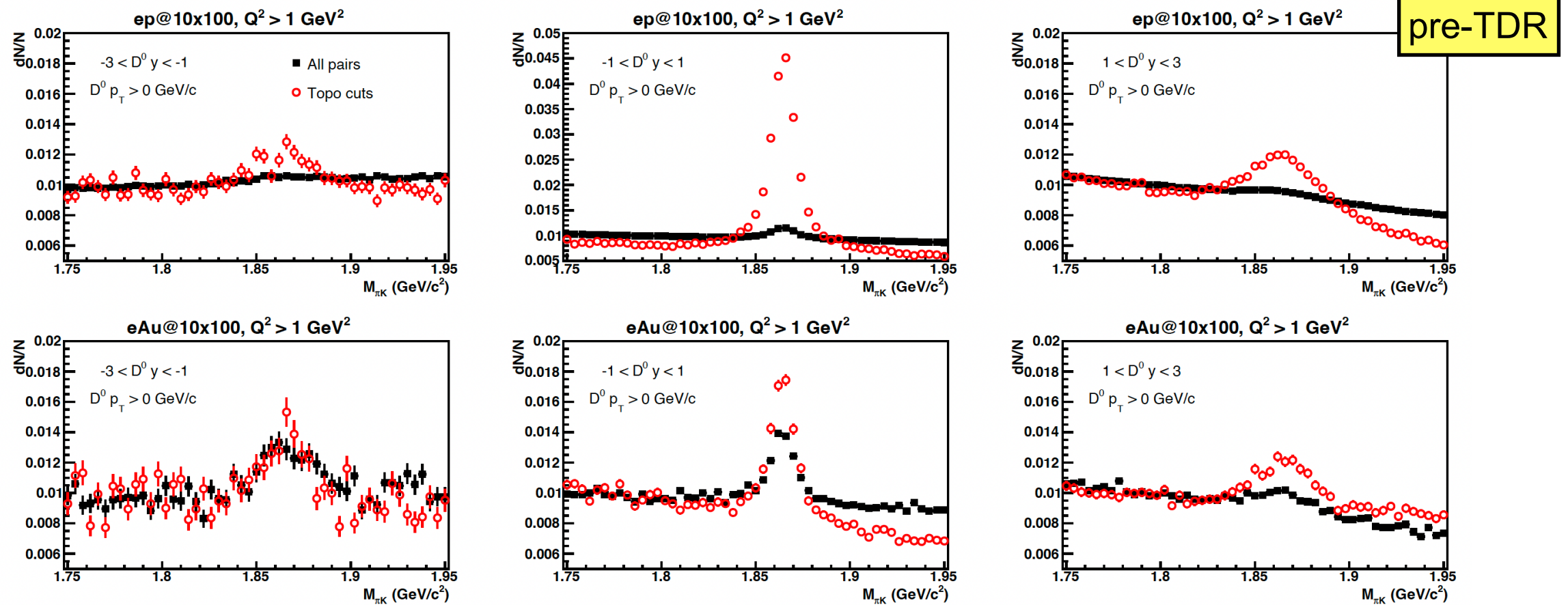


Figure 2.22: Invariant mass distributions of $\pi + K$ pairs with (red circles) and without (black squares) topological selections in 10×100 GeV $e+p$ (top) and $e+Au$ (bottom) collisions with a minimum Q^2 of 1 GeV^2 . Different panels from left to right correspond to different D^0 rapidity intervals: $-3 < y < -1$ (left), $-1 < y < 1$ (middle) and $1 < y < 3$ (right).

Example of using Helix method:
https://github.com/marrbnl/ePIC/tree/main/HF_reco/helix