Displaced Vertex Collection

Bishoy H. Dongwi
CFNS Edward Bouchet Fellow

Stony Brook University, Stony Brook NY 11794

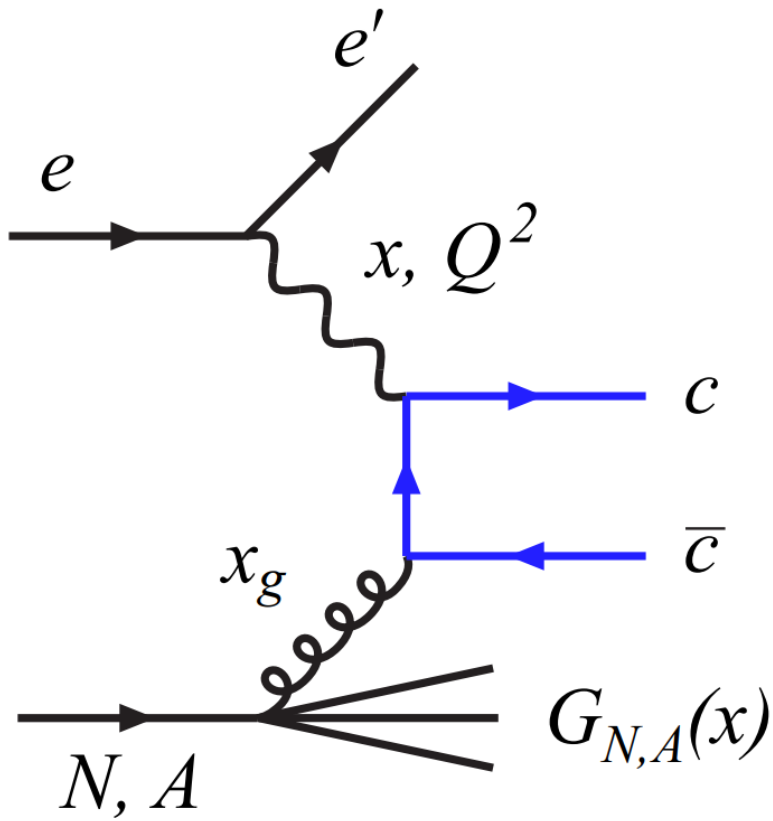
July 16, 2025

Secondary Vertex Finder

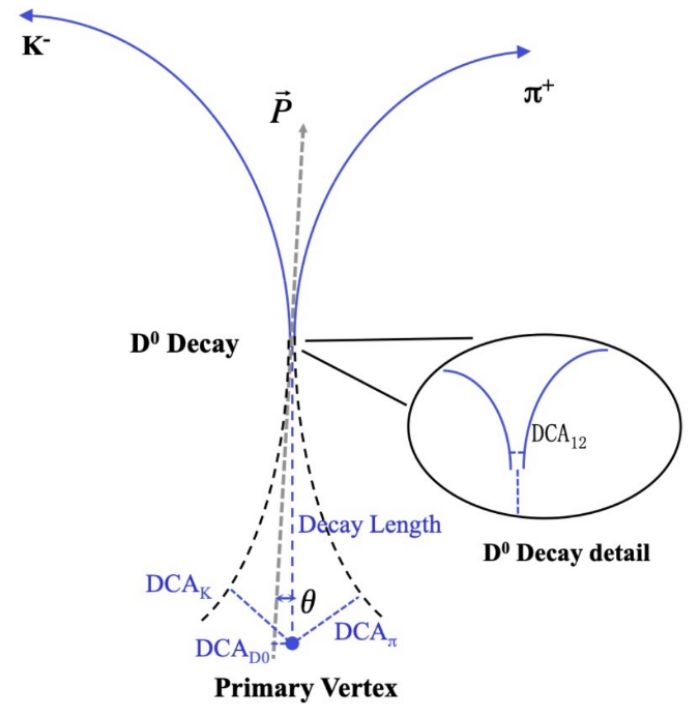
Nature of the Study

- The `ACTS` framework is broadly used in `EICrecon` for tracking and PV
- Implemented the `ACTS::AdaptiveMultiVertexFinder`
- Calculate both the primary and secondary vertices using AMVF
- Feed in `ReconstuctedParticleCollection, Trajectories`
- Preliminary results form `Helix swimming`
- Standalone dev repo of AMVF exists on the main repo
- `PR` has been submitted and currently work-in-progress
- Input file: `pythia8NCDIS_18x275_minQ2=10_beamEffects_xAngle=-0.025_hiDiv_vtxfix_1.hepmc3.MC.root`

Leading Order Process



D^0 Toplogical Reconstruction

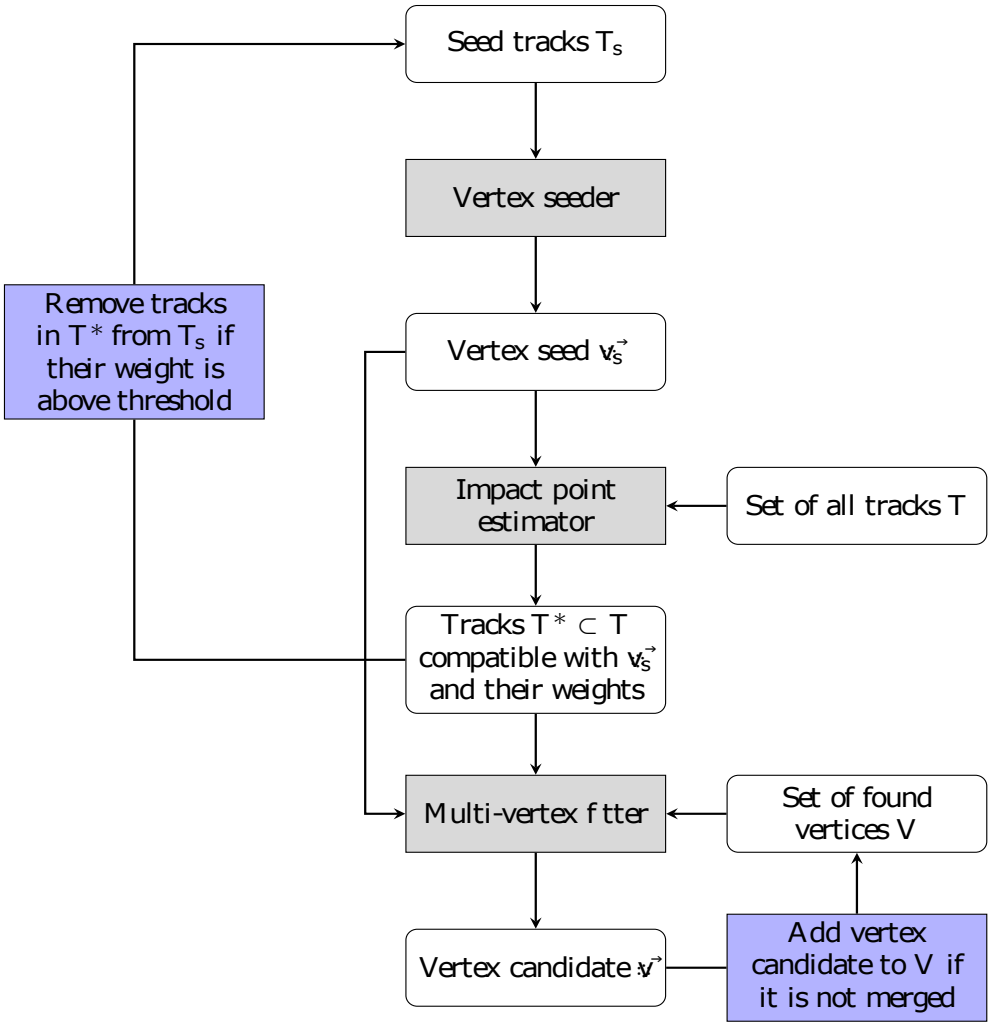


STAR, Phys. Rev. C 99, 034908 (2019)

- Topological variables
  - DCA_π, DCA_K, DCA_{12}
  - DCA_{D^0}, decay length, $cos(\theta)$
- Calculated based on helix swimming in a constant magnetic field
  - Adapted from `STAR software`
  - B = -1.7 T

Rongrong

Adaptive Multi-VertexFinder (AMVF)



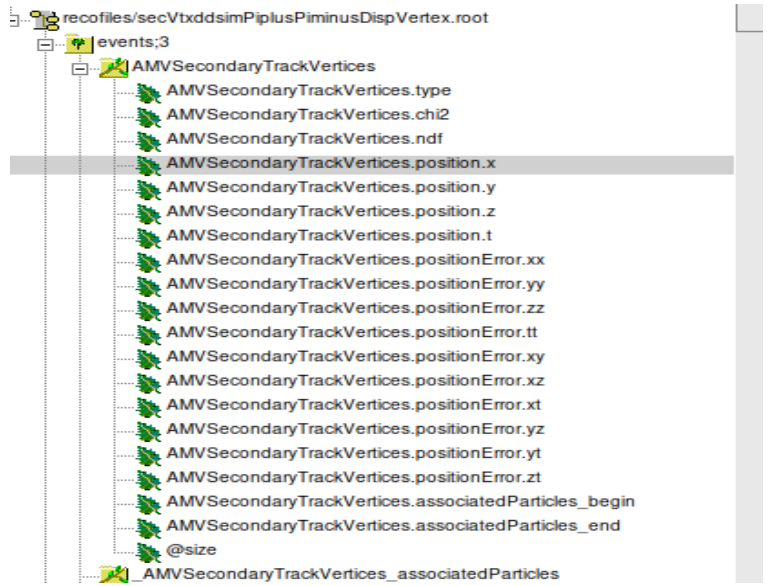
- Based on weighted adaptive Kalman filter with deterministic annealing
- Removeable beamspot constraint (useful for secondary vertices)
- Gaussian track density seed finder to estimate vertex position
- Check compatibility of tracks with seeder (assign weight if compatible)
- Simultaneous refit of all previously found vertices and vertex seed
- After convergence of fit, check whether the vertex candidate is merged with other vertices, discard if merged
- Remove seed tracks if their weights are above threshold

Identification of b-jets and investigation of the discovery potential of a Higgs boson in the $WH \rightarrow \ell\nu b\bar{b}$ channel with the ATLAS experiment
N. G. Piacquadio (2010)

Secondary Track Vertex Factory

```cpp
#include <vector>

#include "algorithms/tracking/SecondaryVertexFinderConfig.h"
#include "algorithms/tracking/SecondaryVertexFinder.h"
#include "extensions/jana/JOmniFactory.h"
#include <spdlog/logger.h>

namespace eicrecon {

class SecondaryVertexFinder_factory
    : public JOmniFactory<SecondaryVertexFinder_factory, SecondaryVertexFinderConfig> {

private:
  using AlgoT = eicrecon::SecondaryVertexFinder;
  std::unique_ptr<AlgoT> m_algo;

  PodioInput<edm4eic::ReconstructedParticle> m_reco_input{this};
  Input<ActsExamples::Trajectories> m_acts_trajectories_input{this};
  PodioOutput<edm4eic::Vertex> prm_vertices_output{this};
  PodioOutput<edm4eic::Vertex> sec_vertices_output{this};

  ParameterRef<int> m_maxVertices{this, "maxVertices", config().maxVertices,
                                  "Maximum num vertices that can be found"};
  ParameterRef<bool> m_reassignTracksAfterFirstFit{
      this, "reassignTracksAfterFirstFit", config().reassignTracksAfterFirstFit,
      "Whether or not to reassign tracks after first fit"};
  ParameterRef<float> m_tracksMaxZinterval{this, "tracksMaxZinterval", config().tracksMaxZint
erval,
                                           "Max z interval for Acts::AdaptiveMultiVertexFinde
r."};
  ParameterRef<float> m_maxIterations{this, "maxIterations", config().maxIterations,
                                      "Max iterations for Acts::AdaptiveMultivertexFinder"};
  ParameterRef<float> m_maxDistToLinPoint{
      this, "maxDistToLinPoint", config().maxDistToLinPoint,
      "Max disttance to line point (pca) for Acts::AdaptiveMultivertexFinder"};

  Service<ACTSGeo_service> m_ACTSGeoSvc{this};

public:
  void Configure() {
    m_algo = std::make_unique<AlgoT>();
    m_algo->applyConfig(config());
    m_algo->init(m_ACTSGeoSvc().actsGeoProvider(), logger());
  }

  void ChangeRun(int32_t /*run_number*/) {}

  void Process(int32_t /*run_number*/, uint64_t /*event_number*/) {
    std::tie(prm_vertices_output(), sec_vertices_output()) =
        m_algo->produce(m_reco_input(), m_acts_trajectories_input());
  }
};

} // namespace eicrecon
```

```cpp
app->Add(new JOmniFactoryGeneratorT<SecondaryVertexFinder_factory>(
    "SecondaryTrackVerticesAMVF", {"ReconstructedParticles", "CentralCKFActsTrajectories"},
    {
        "PrimaryTrackVerticesAMVF",
        "SecondaryTrackVerticesAMVF",
    },
    {}, app));
```



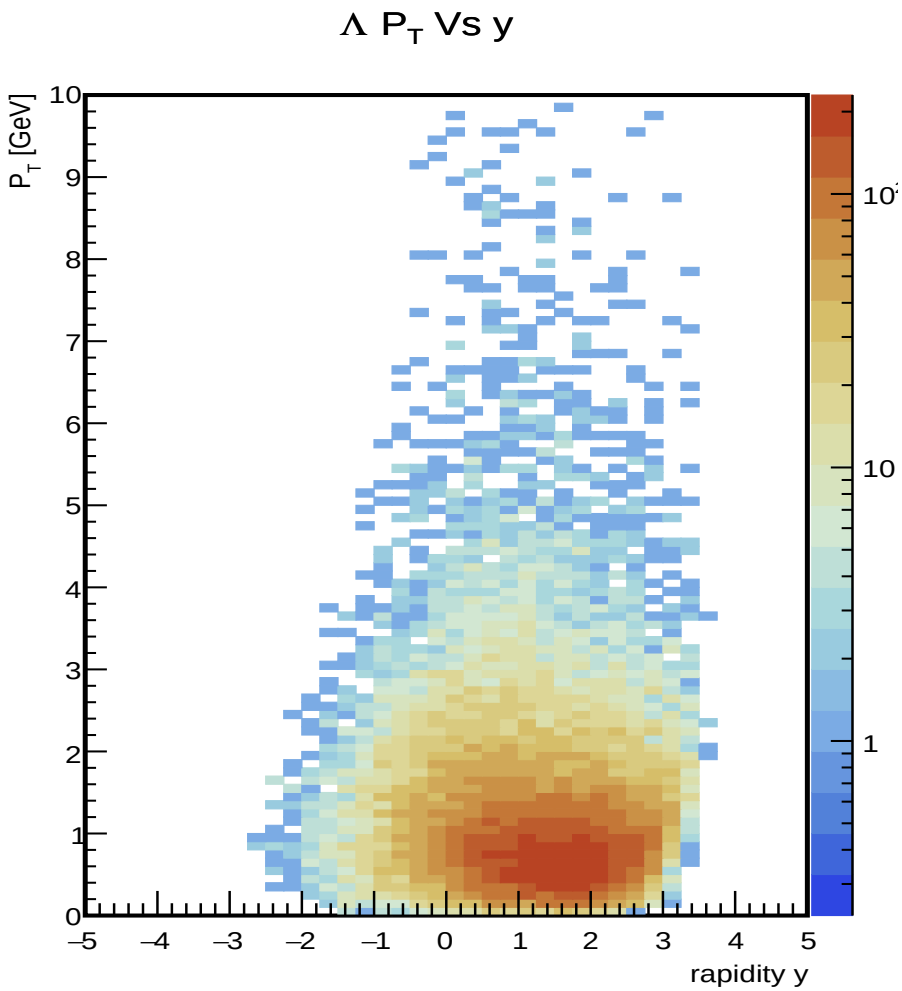
- Secondary vertex Factory
- Might need to create a custom Collection for our Secondary Vertices to play with removal of overlapping vertices

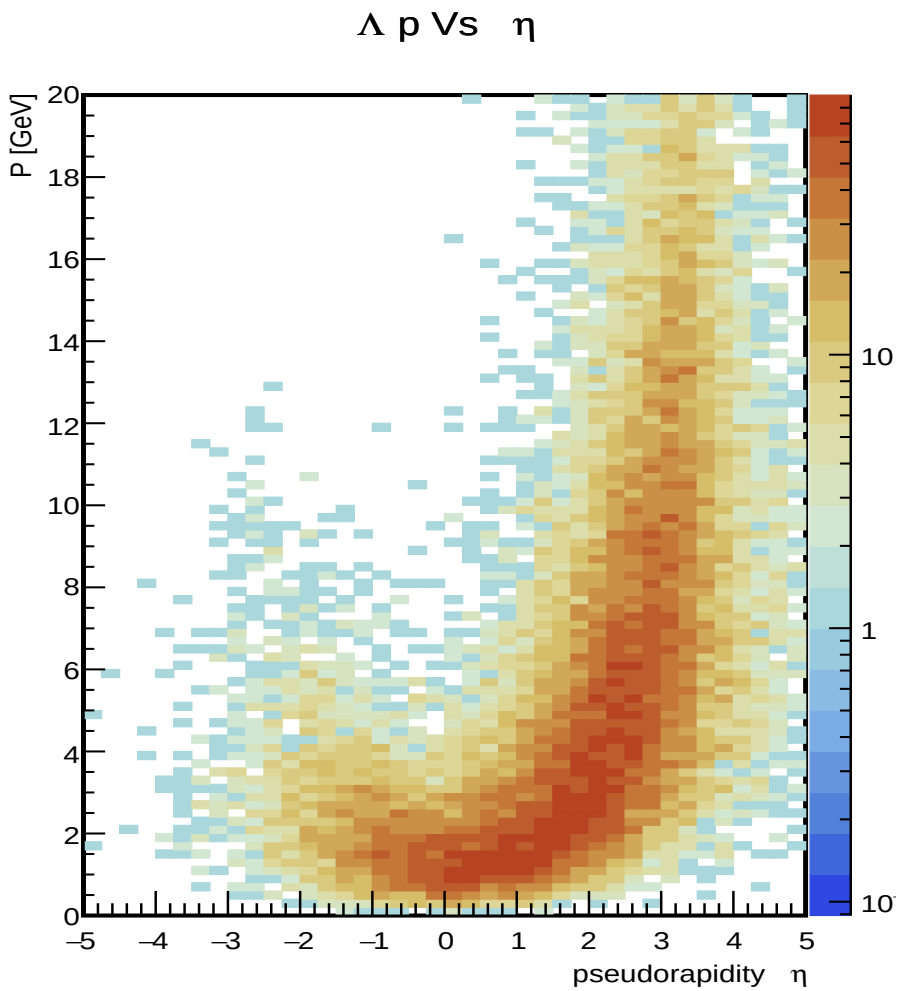
Secondary Track Vertex Factory

```cpp
#include <vector>

#include "algorithms/tracking/SecondaryVertexFinderConfig.h"
#include "algorithms/tracking/SecondaryVertexFinder.h"
#include "extensions/jana/JOmniFactory.h"
#include <spdlog/logger.h>

namespace eicrecon {

class SecondaryVertexFinder_factory
    : public JOmniFactory<SecondaryVertexFinder_factory, SecondaryVertexFinderConfig> {

private:
  using AlgoT = eicrecon::SecondaryVertexFinder;
  std::unique_ptr<AlgoT> m_algo;

  PodioInput<edm4eic::ReconstructedParticle> m_reco_input{this};
  Input<ActsExamples::Trajectories> m_acts_trajectories_input{this};
  PodioOutput<edm4eic::Vertex> prm_vertices_output{this};
  PodioOutput<edm4eic::Vertex> sec_vertices_output{this};

  ParameterRef<int> m_maxVertices{this, "maxVertices", config().maxVertices,
                                  "Maximum num vertices that can be found"};
  ParameterRef<bool> m_reassignTracksAfterFirstFit{
      this, "reassignTracksAfterFirstFit", config().reassignTracksAfterFirstFit,
      "Whether or not to reassign tracks after first fit"};
  ParameterRef<float> m_tracksMaxZinterval{this, "tracksMaxZinterval", config().tracksMaxZint
erval,
                                           "Max z interval for Acts::AdaptiveMultiVertexFinde
r."};
  ParameterRef<float> m_maxIterations{this, "maxIterations", config().maxIterations,
                                      "Max iterations for Acts::AdaptiveMultivertexFinder"};
  ParameterRef<float> m_maxDistToLinPoint{
      this, "maxDistToLinPoint", config().maxDistToLinPoint,
      "Max disttance to line point (pca) for Acts::AdaptiveMultivertexFinder"};

  Service<ACTSGeo_service> m_ACTSGeoSvc{this};

public:
  void Configure() {
    m_algo = std::make_unique<AlgoT>();
    m_algo->applyConfig(config());
    m_algo->init(m_ACTSGeoSvc().actsGeoProvider(), logger());
  }

  void ChangeRun(int32_t /*run_number*/) {}

  void Process(int32_t /*run_number*/, uint64_t /*event_number*/) {
    std::tie(prm_vertices_output(), sec_vertices_output()) =
        m_algo->produce(m_reco_input(), m_acts_trajectories_input());
  }
};

} // namespace eicrecon
```

```cpp
std::vector<Acts::InputTrack> inputTracks;
for (unsigned int i = 0; i < trajectories.size() - 1; i++) {
  auto tips = trajectories[i]->tips();
  if (tips.empty()) {
    continue;
  }
  for (unsigned int j = i + 1; j < trajectories.size(); j++) {
    auto tips2 = trajectories[j]->tips();
    if (tips2.empty()) {
      continue;
    }
    // Checking for default DCA cut-condition
    for (auto& tip : tips) {
      /// CKF can provide multiple track trajectories for a single input seed
      inputTracks.emplace_back(&(trajectories[i]->trackParameters(tip)));
    }
    for (auto& tip : tips2) {
      inputTracks.emplace_back(&(trajectories[j]->trackParameters(tip)));
    }
    // run the vertex finder for both tracks
    std::vector<Acts::Vertex> verticesSec;
    auto resultSecondary = vertexfinderSec.find(inputTracks, vfOptions, stateSec);
    if (resultSecondary.ok()) {
      verticesSec = std::move(resultSecondary.value());
    }

    for (const auto& secvertex : verticesSec) {
      edm4eic::Cov4f cov(secvertex.fullCovariance()(0, 0), secvertex.fullCovariance
()(1, 1),
                         secvertex.fullCovariance()(2, 2), secvertex.fullCovariance
()(3, 3),
                         secvertex.fullCovariance()(0, 1), secvertex.fullCovariance
()(0, 2),
                         secvertex.fullCovariance()(0, 3), secvertex.fullCovariance
()(1, 2),
                         secvertex.fullCovariance()(1, 3), secvertex.fullCovariance
()(2, 3));
      auto eicvertex = secVertices->create();
      eicvertex.setType(0); // boolean flag if vertex is primary vertex of event
      eicvertex.setChi2(static_cast<float>(secvertex.fitQuality().first)); // chi2
      eicvertex.setNdf(static_cast<float>(secvertex.fitQuality().second)); // ndf
      eicvertex.setPosition({
          static_cast<float>(secvertex.position().x()),
          static_cast<float>(secvertex.position().y()),
          static_cast<float>(secvertex.position().z()),
          static_cast<float>(secvertex.time()),
      });                                 // vtxposition
      eicvertex.setPositionError(cov); // covariance
```

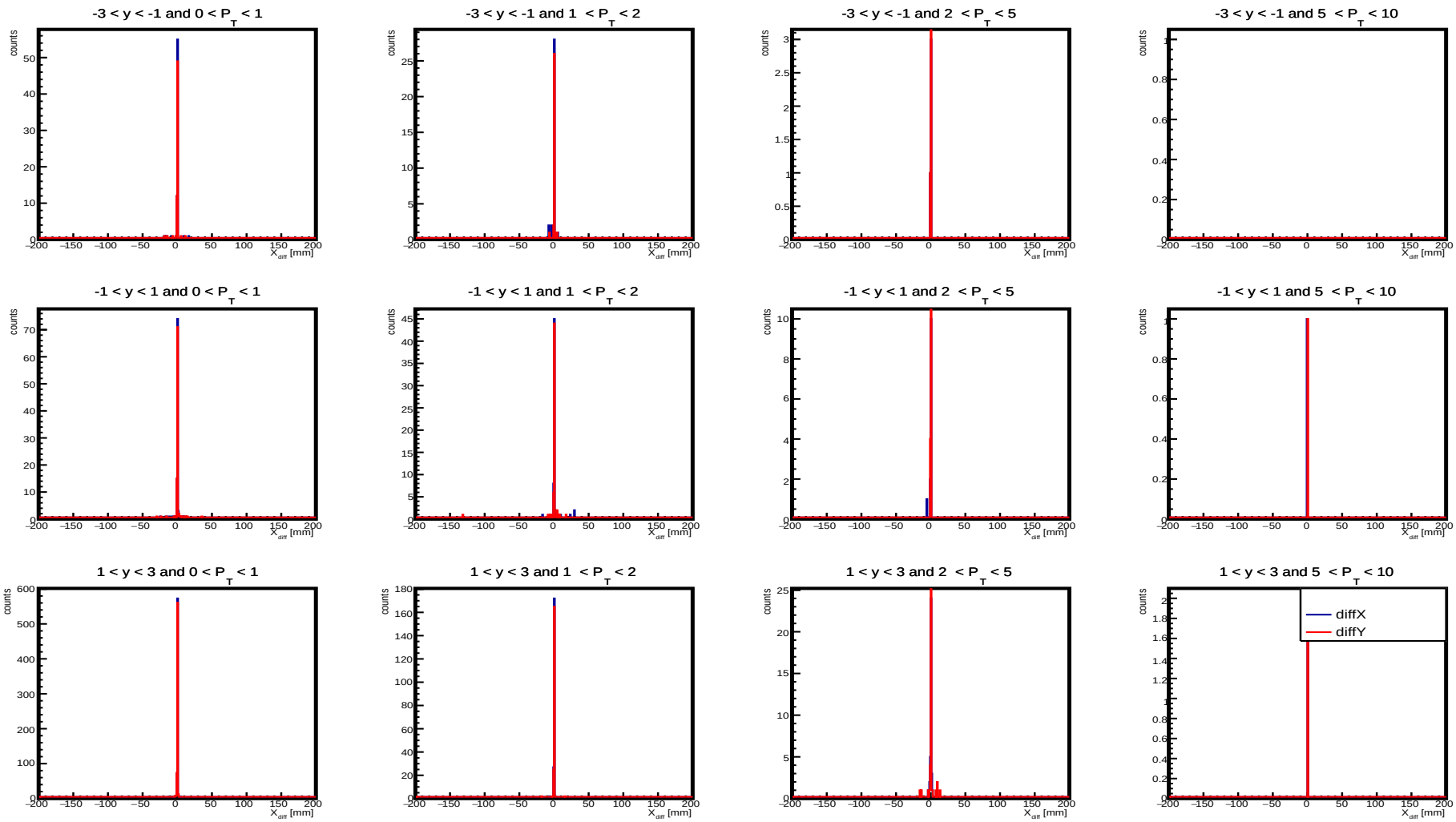
Kinematic Plots





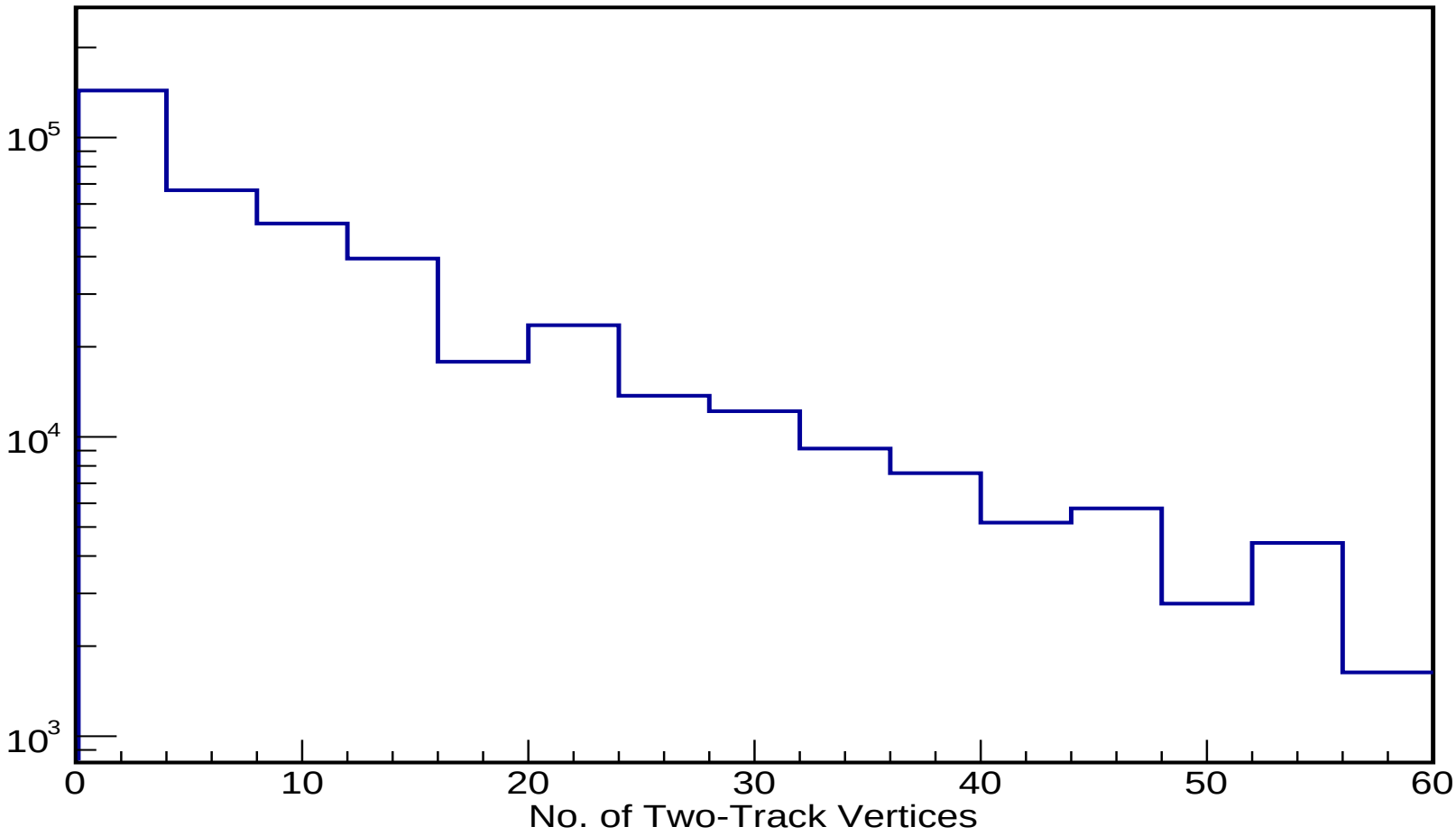
- Λ kinematic plots

Difference between Primary Vertices

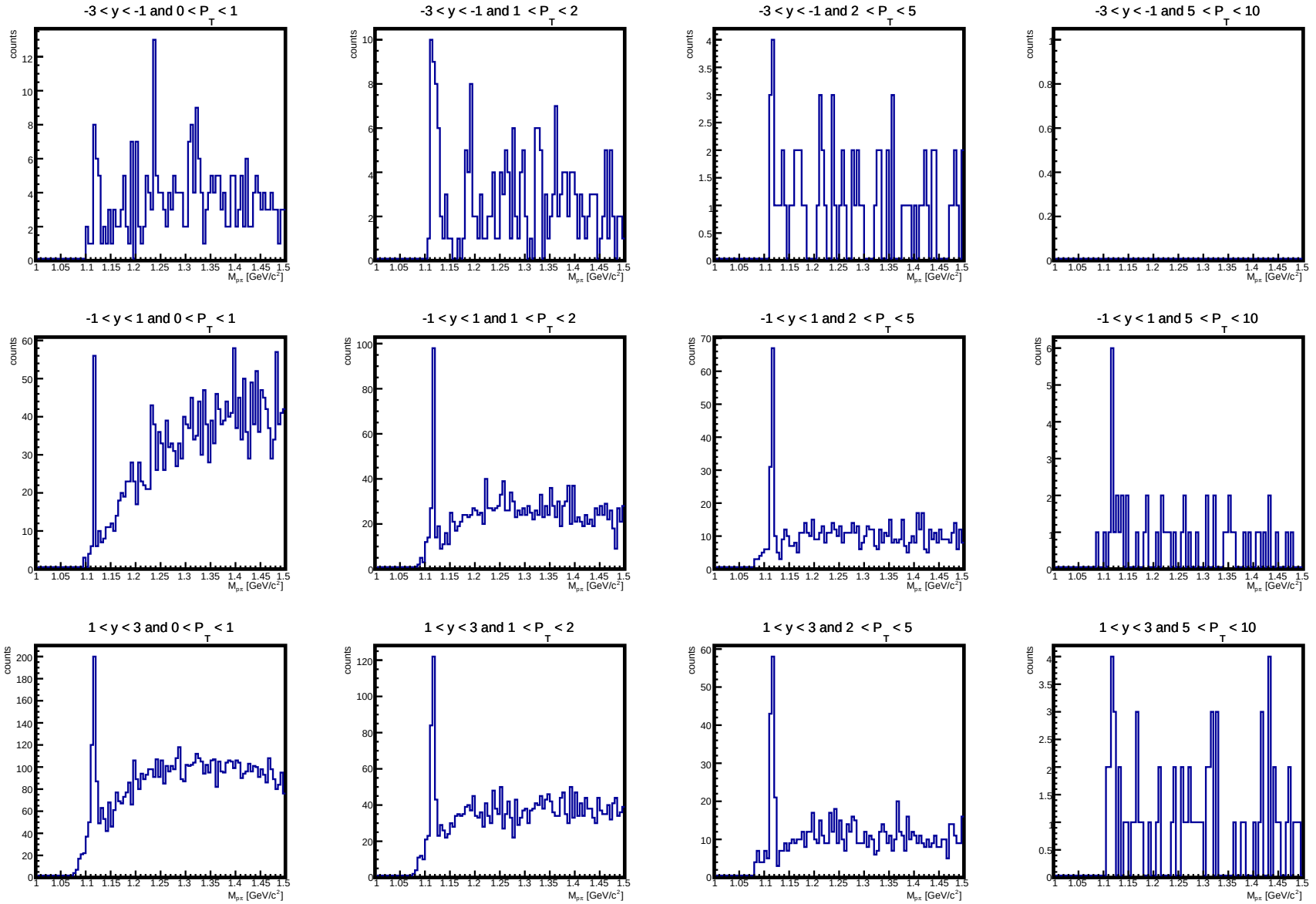


- Distributions of $PV_{AMVF} - PV_{CTV}$ in XY
- Distributions at p_T and y cuts

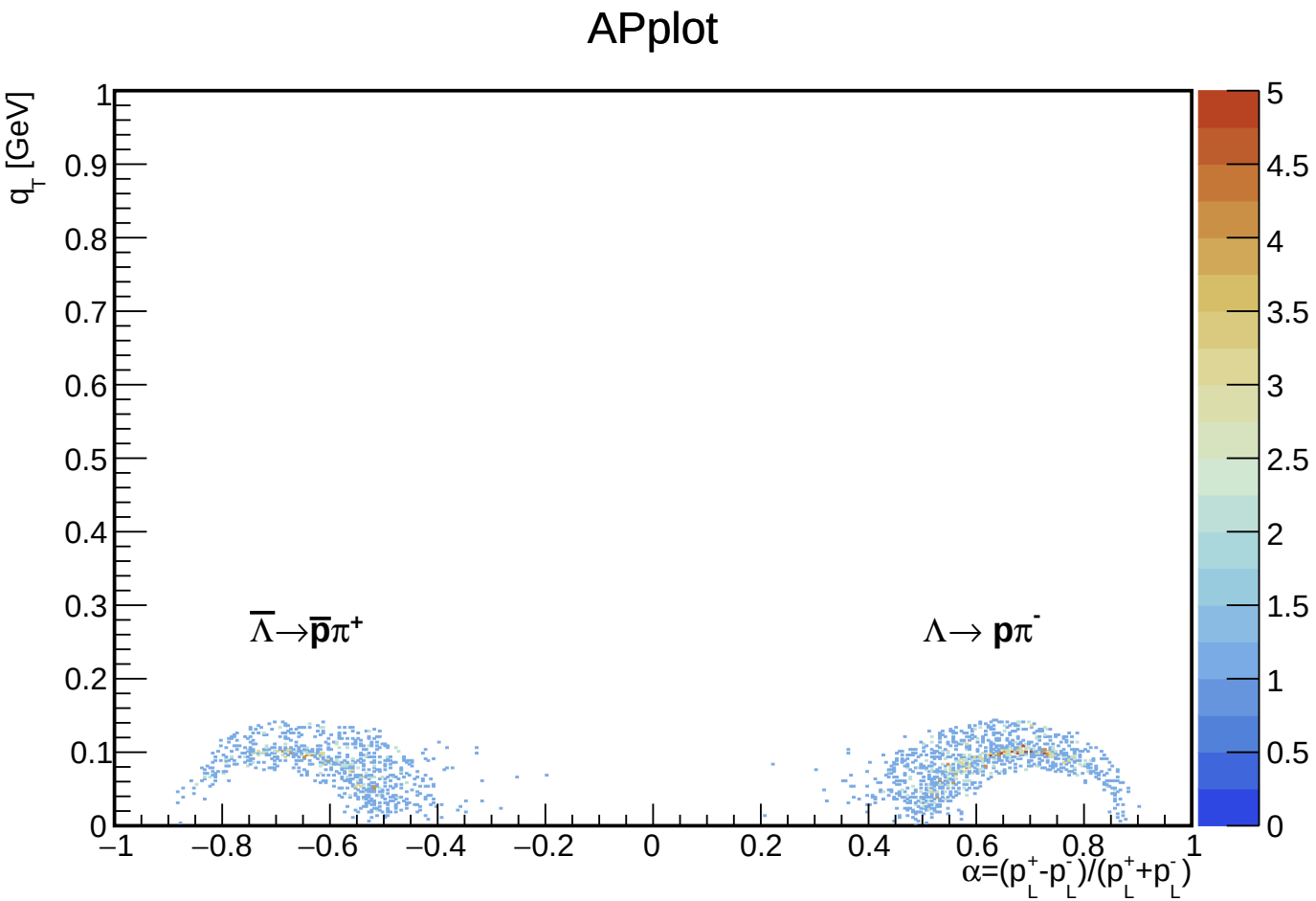
SV Multiplicity



M_Λ Distribution in p_T and y bins

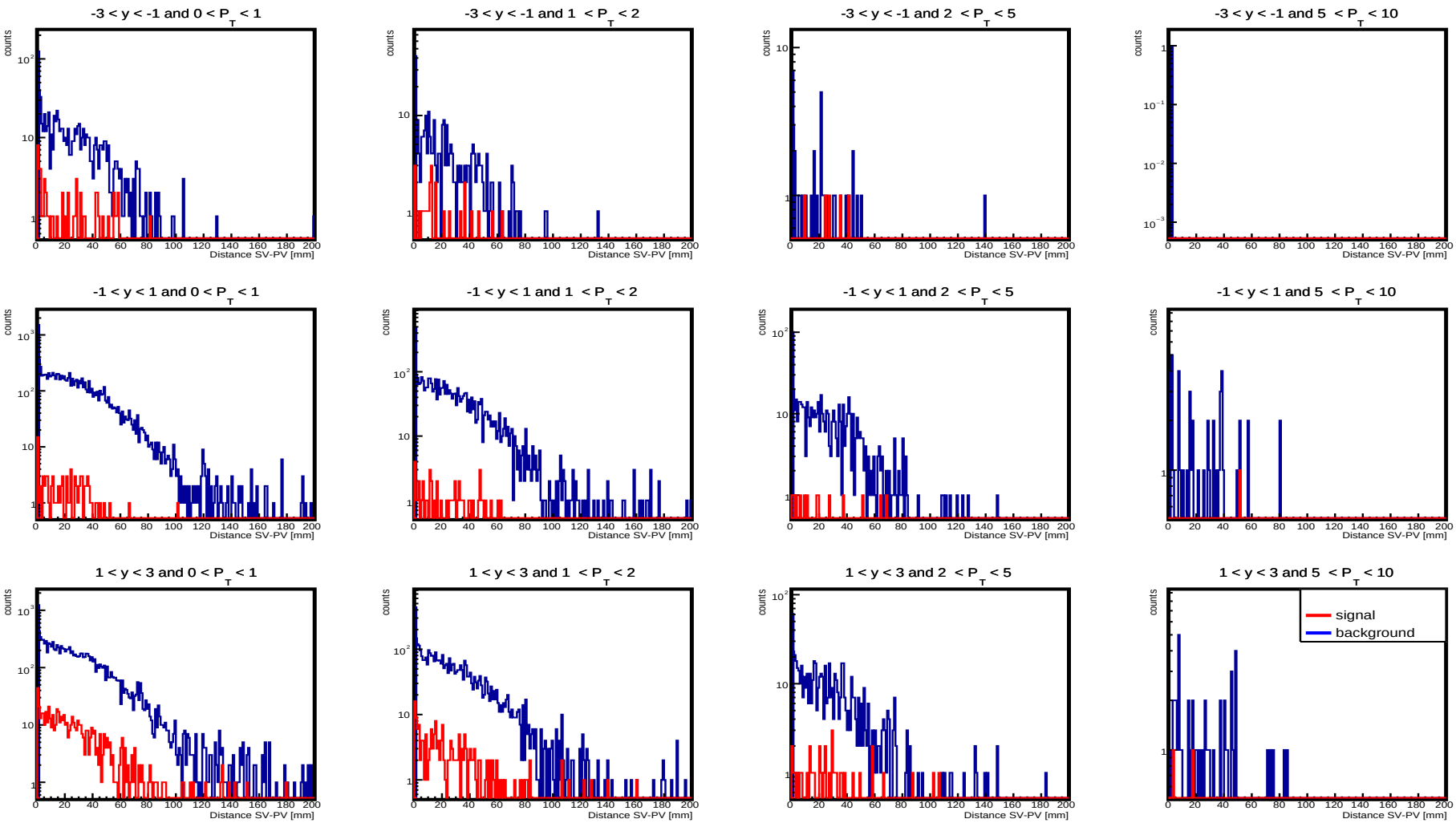


Armenteros-Podolansky Plot



- PID plot of daughters p_L asymmetry relative to the parent particle

Decay Length: SV-PV



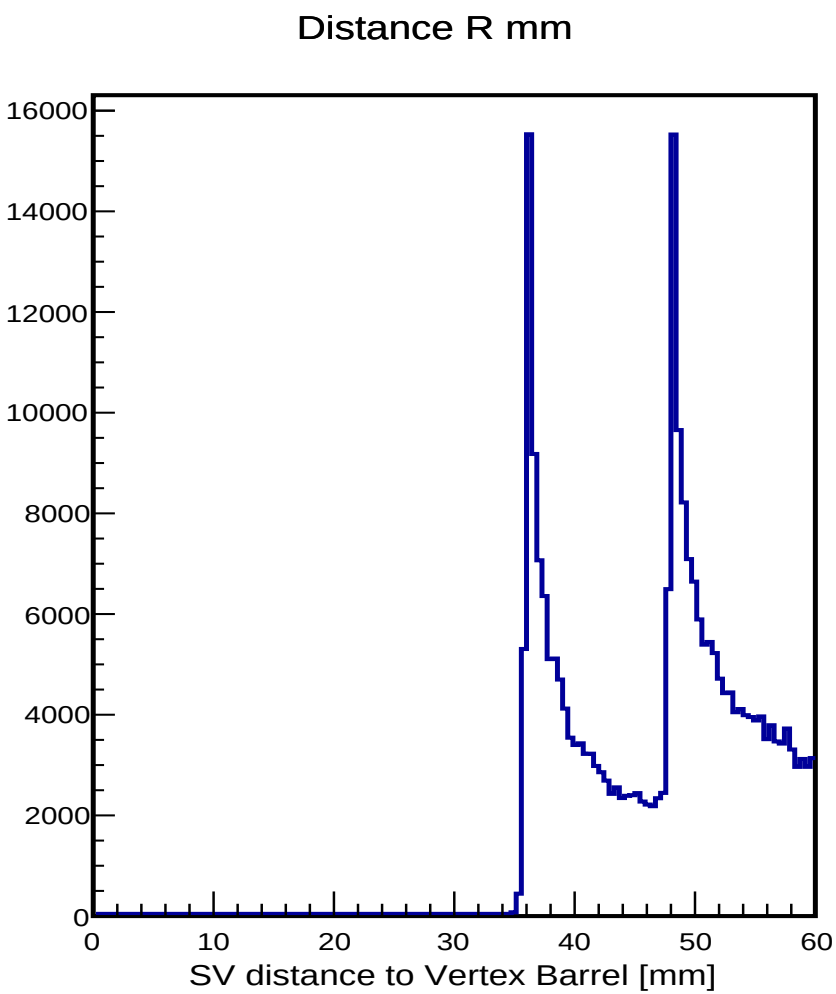
- Signal here refers to narrow cuts around M_Λ
- Need more stats

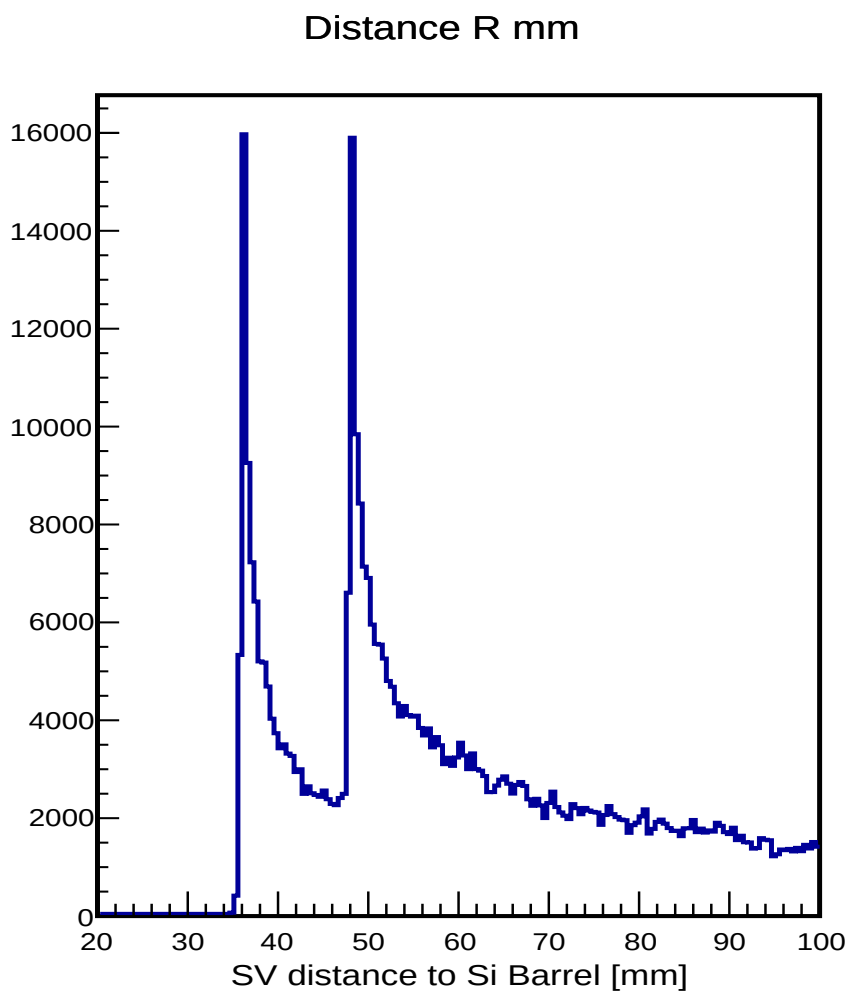
Summary and Beyond

- Heavy flavor production at EIC provides unique access to gluon distributions, especially at large x
- $\Lambda \longrightarrow p\pi$ two-track vertex
- Consistency check: PV evaluation with AMVF and compare to existing results
- D^0 studies with SV now underway and compare with Helix fitter results $\longrightarrow$ Use enriched D^0 samples
- Directly use **Secondary vertex** to find $\Lambda \longrightarrow p\pi^-$

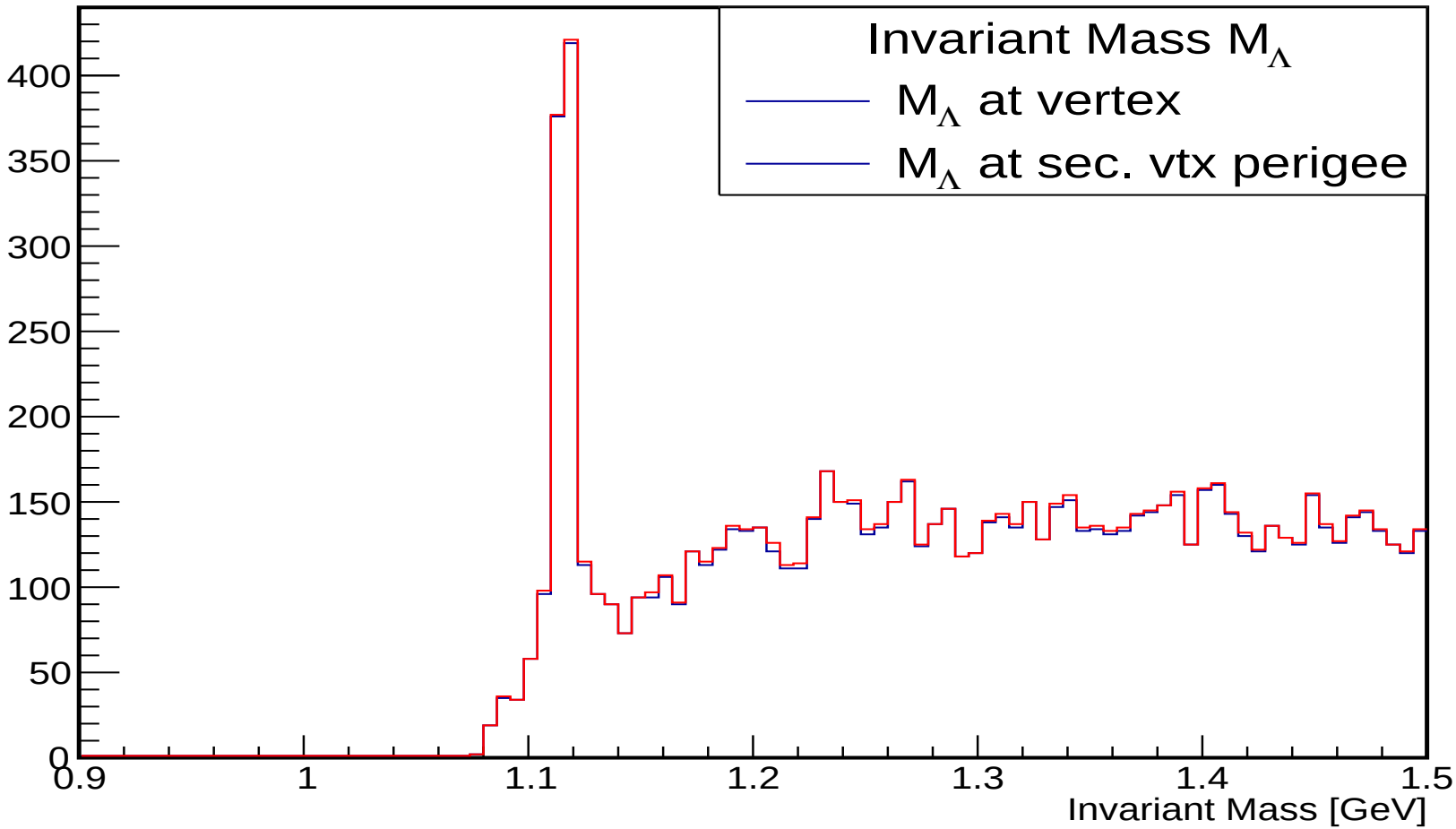
Back Up

SV Distance to Material





M_Λ


Invariant Mass M_Λ
M_Λ at vertex
M_Λ at sec. vtx perigee
400
350
300
250
200
150
100
50
0
0.9
1
1.1
1.2
1.3
1.4
1.5
Invariant Mass [GeV]

Using Secondary