Displaced Vertex

Bishoy H. Dongwi
CFNS Edward Bouchet Fellow

Stony Brook University, Stony Brook NY 11794

July 10, 2025

Secondary Vertex Finder

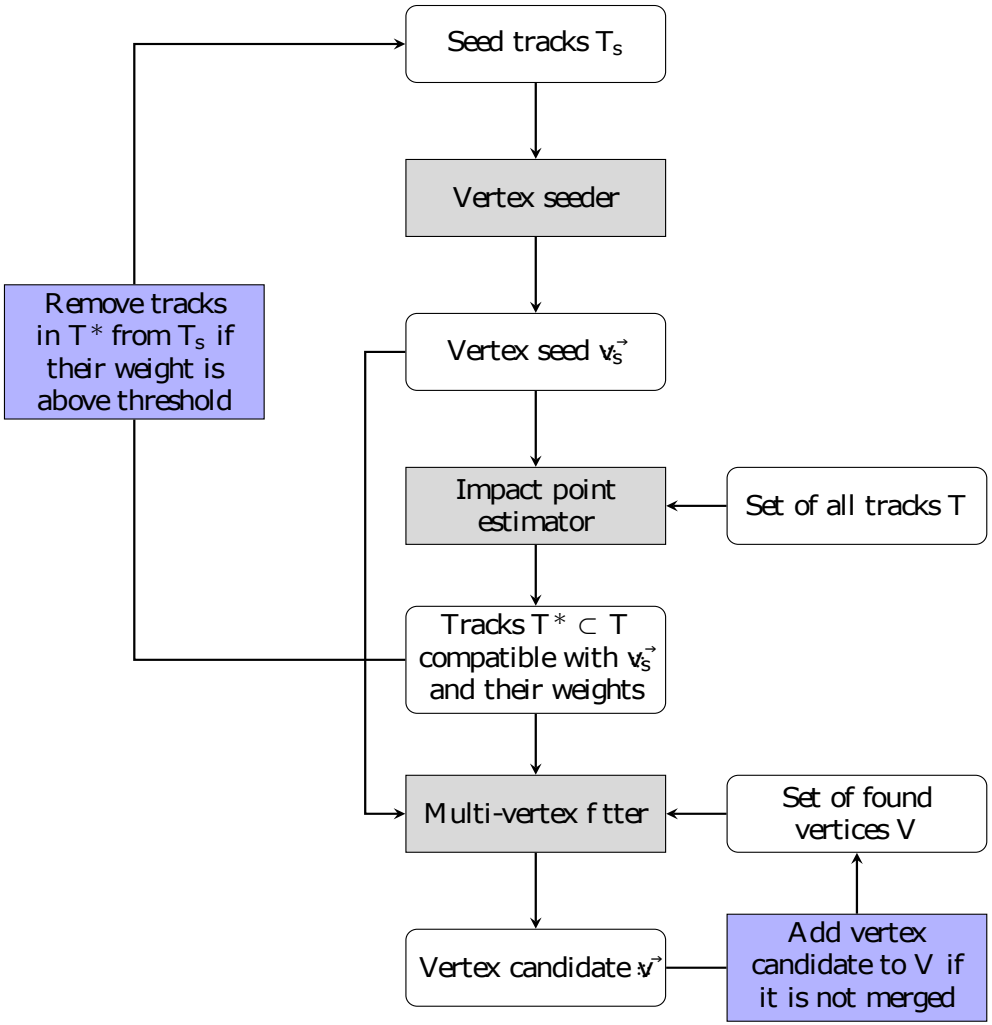
Nature of the Study

- Implemented the `ACTS::AdaptiveMultiVertexFinder`
- Calculate both the primary and secondary vertices using AMVF
- Write both vertices to output `ROOT` file (seems like there is overwritting going on)
- Input file: enriched D^0 sample

  `pythia8NCDIS_18x275_minQ2=10_beamEffects_xAngle=-0.025_hiDiv_1.hepmc3.MC.root`

  (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
- Cleaned up naming of factories

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().tracksMaxZinterval,
                                           "Max z interval for Acts::AdaptiveMultiVertexFinder."};
  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
```

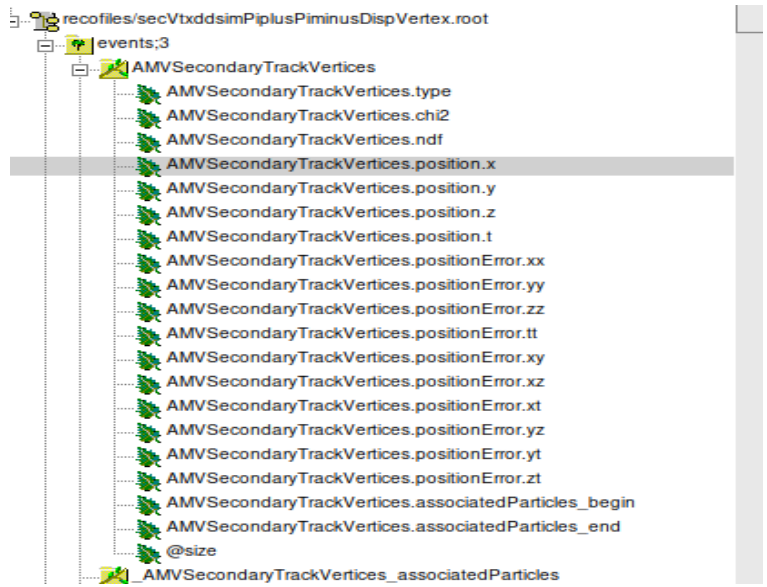
```cpp
#include "Acts/Vertexing/AdaptiveGridTrackDensity.hpp"
#include "Acts/Vertexing/AdaptiveMultiVertexFinder.hpp"
#include "extensions/spdlog/SpdlogToActs.h"

void eicrecon::SecondaryVertexFinder::init(std::shared_ptr<const ActsGeometryProvider> geo_svc,
                                           std::shared_ptr<spdlog::logger> log) {

  m_log    = log;
  m_geoSvc = geo_svc;
  m_BField =
      std::dynamic_pointer_cast<const eicrecon::BField::DD4hepBField>(m_geoSvc->getFieldProvider());
  m_fieldctx = eicrecon::BField::BFieldVariant(m_BField);
}

std::tuple<std::unique_ptr<edm4eic::VertexCollection>, std::unique_ptr<edm4eic::VertexCollection>>
eicrecon::SecondaryVertexFinder::produce(
    const edm4eic::ReconstructedParticleCollection* recotracks,
    std::vector<const ActsExamples::Trajectories*> trajectories) {
  auto primaryVertices = std::make_unique<edm4eic::VertexCollection>();
  auto outputVertices  = std::make_unique<edm4eic::VertexCollection>();

  Acts::EigenStepper<> stepperSec(m_BField);

  //Need to make sure that the track container is not actually empty
  if (!trajectories.empty()){
    // Calculate primary vertex using AMVF
    primaryVertices = calculatePrimaryVertex(recotracks, trajectories, stepperSec);
    // Primary vertex collection container to be used in Sec. Vertex fitting
    outputVertices = calculateSecondaryVertex(recotracks, trajectories, stepperSec);
  }

  return std::make_tuple(std::move(primaryVertices), std::move(outputVertices));
}
```

- Source code of sec. vertex finder
- Output is a `tuple` of the primary & secondary vertex

Secondary Track Vertex Factory

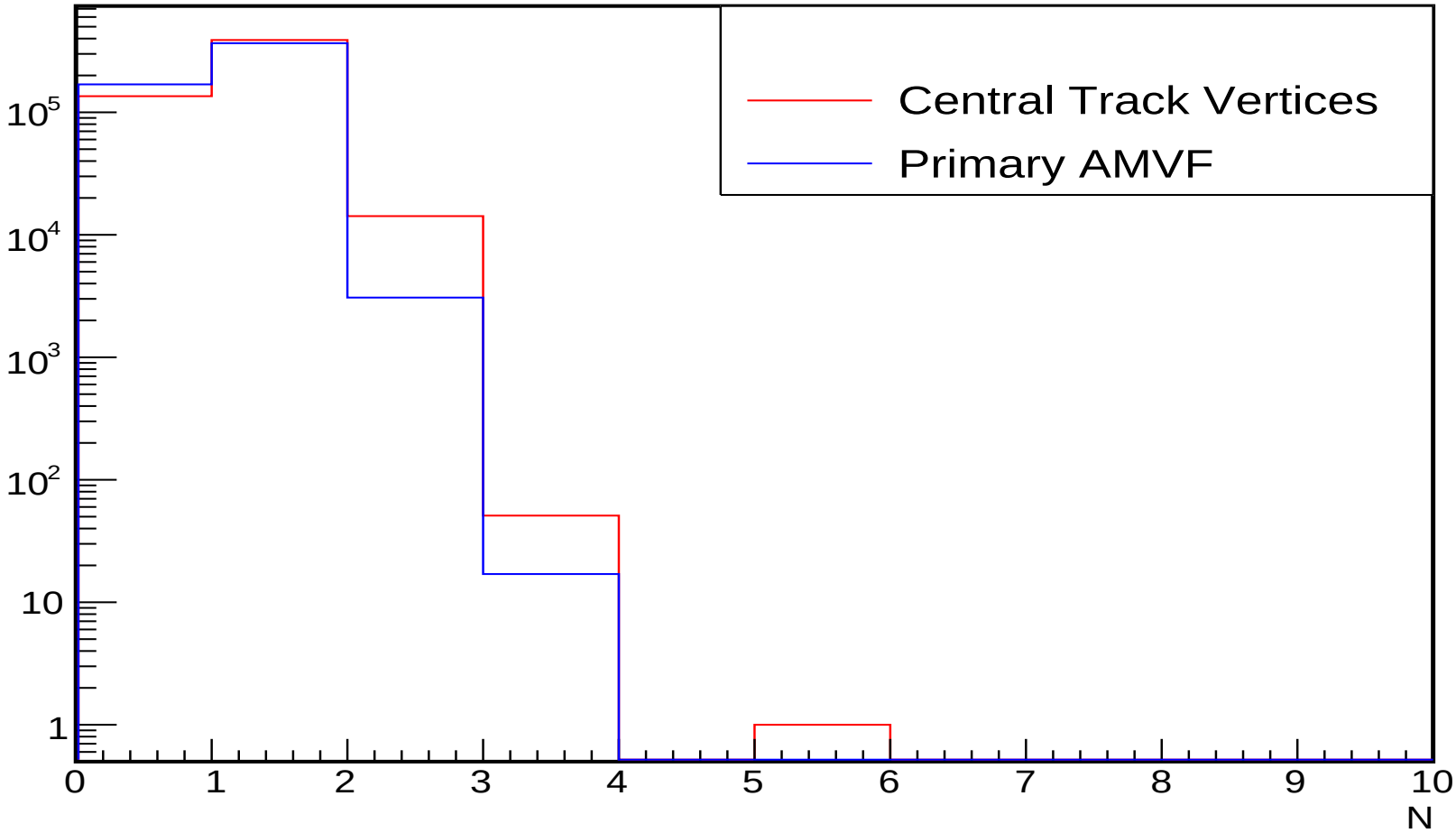
```cpp
#include <vector>

#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));
```



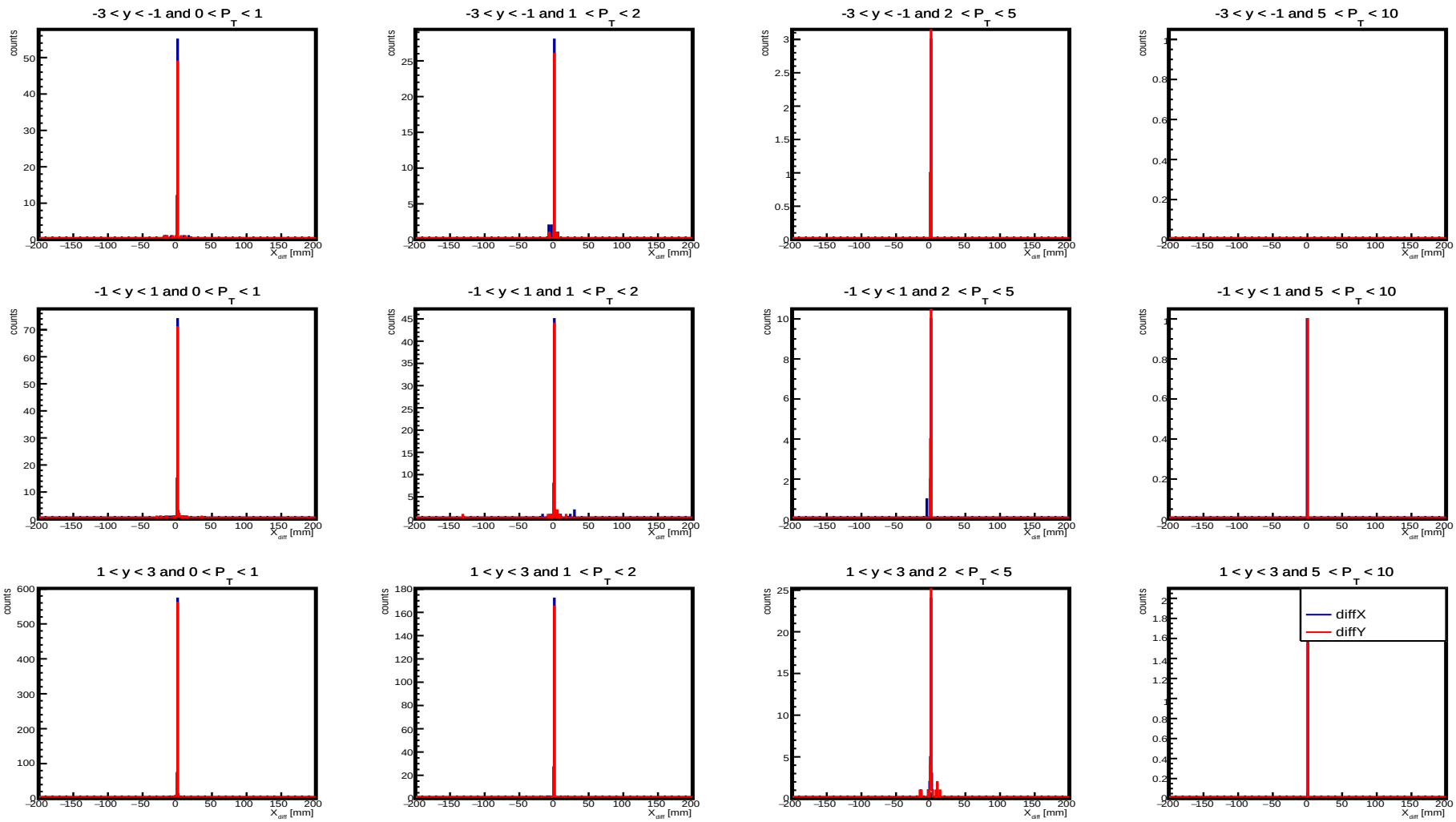
- Might need to create a custom Collection for our Secondary Vertices to play with removal of overlapping vertices

Primary Vertex Multiplicity



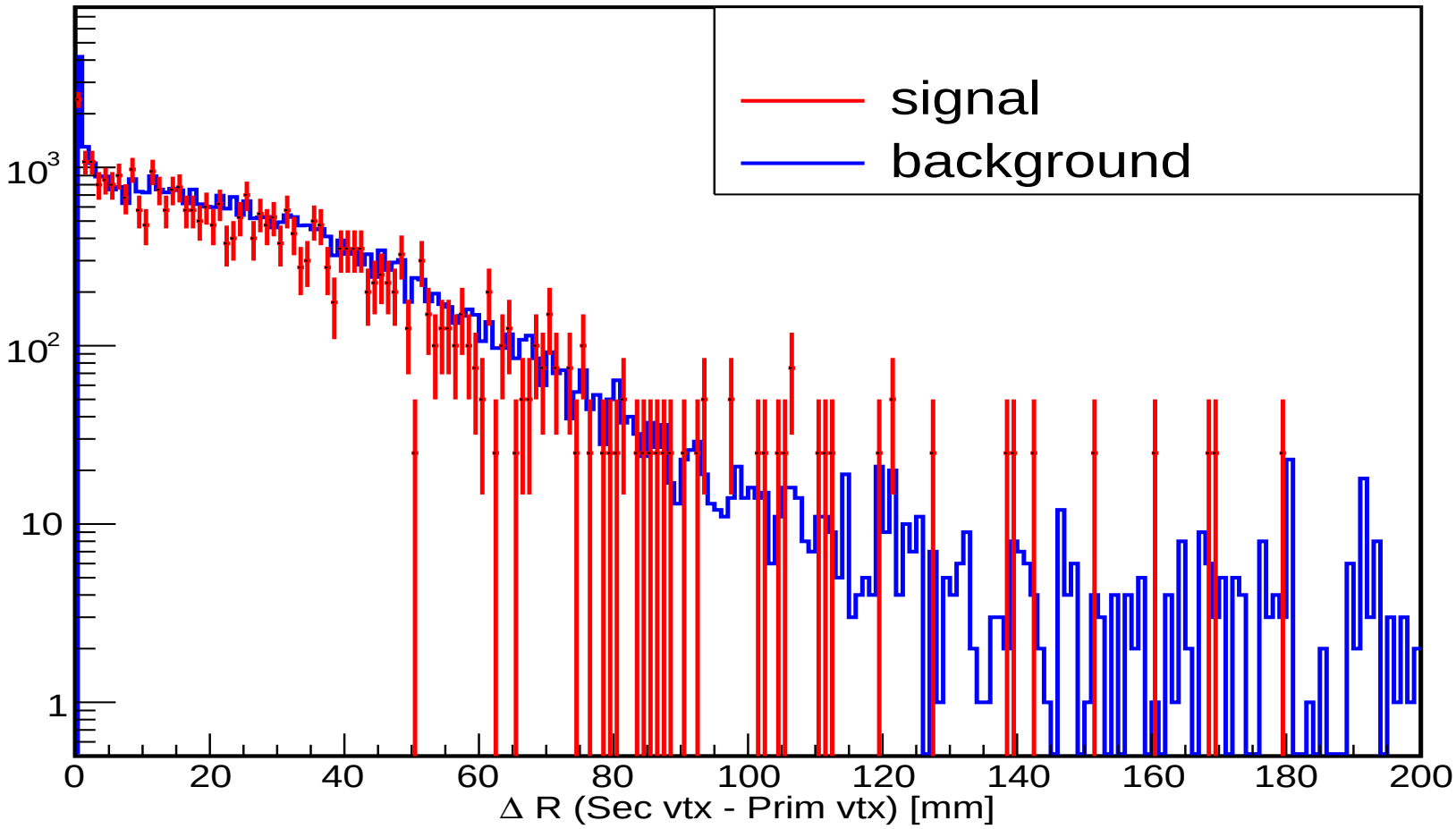
- Good agreement in Multiplicity between both primary vertex finder

Difference between Primary Vertices



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

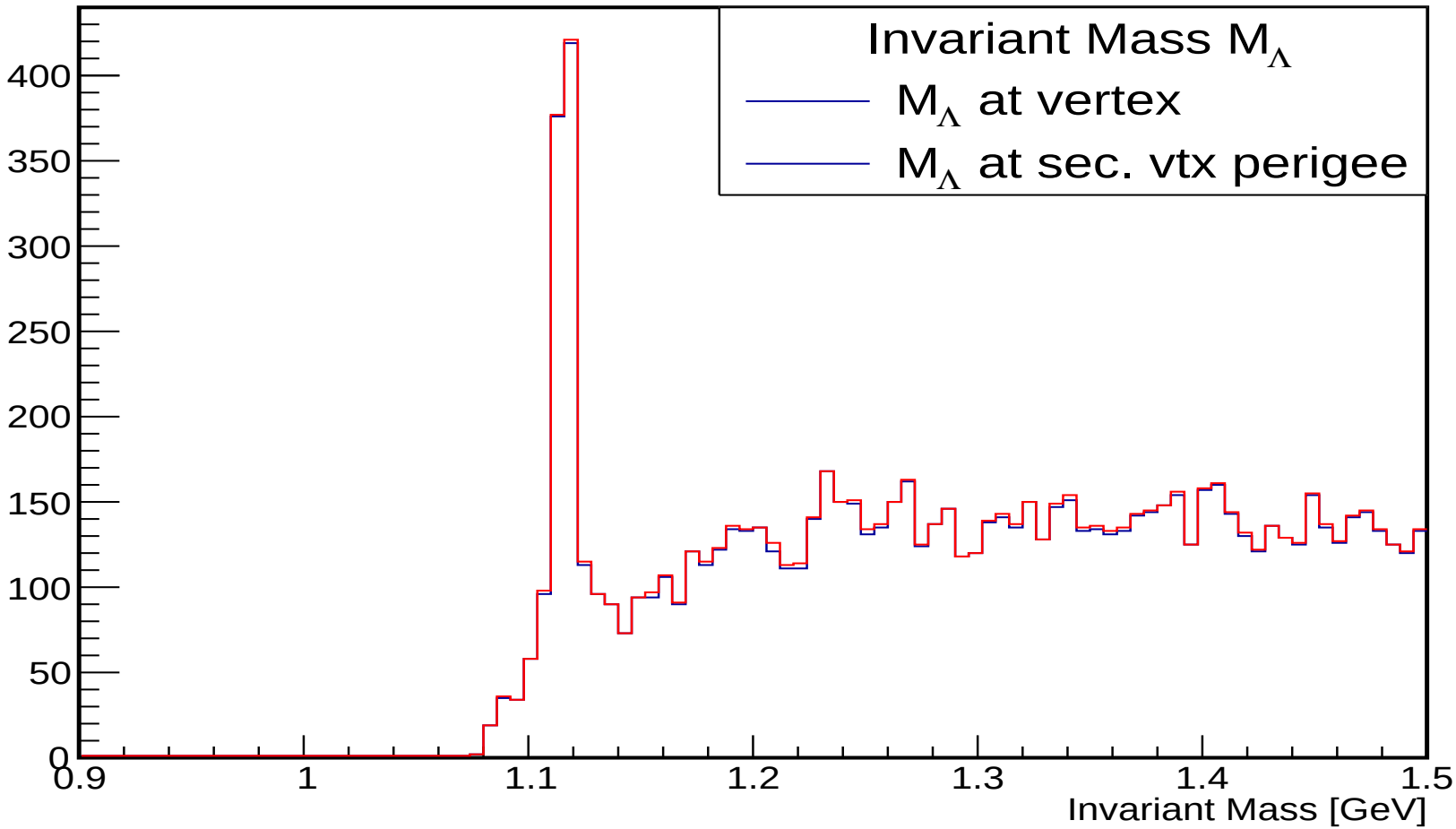
SV-PV



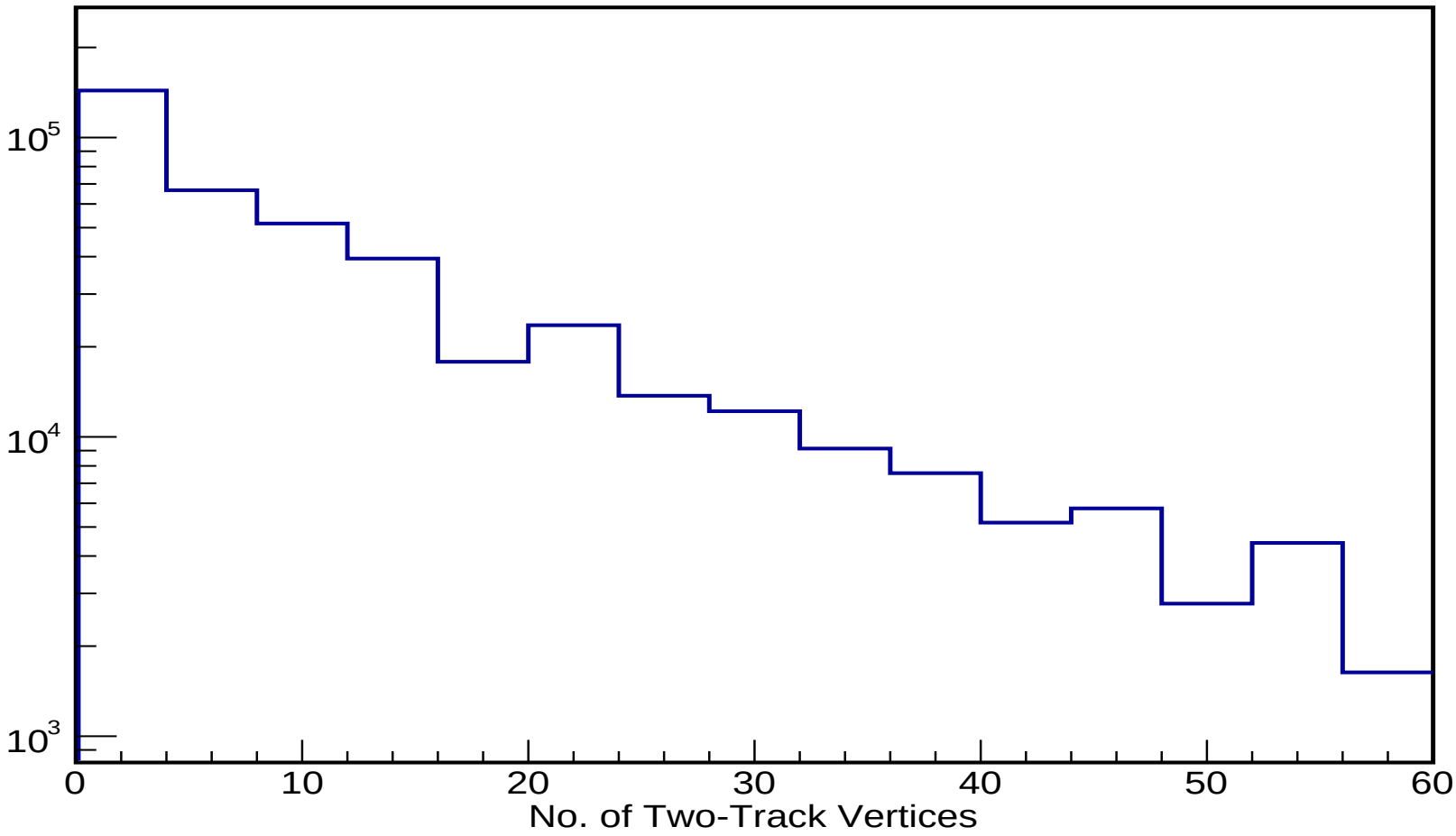
- Signal here refers to narrow cuts around M_Λ
- Need more stats

Back Up

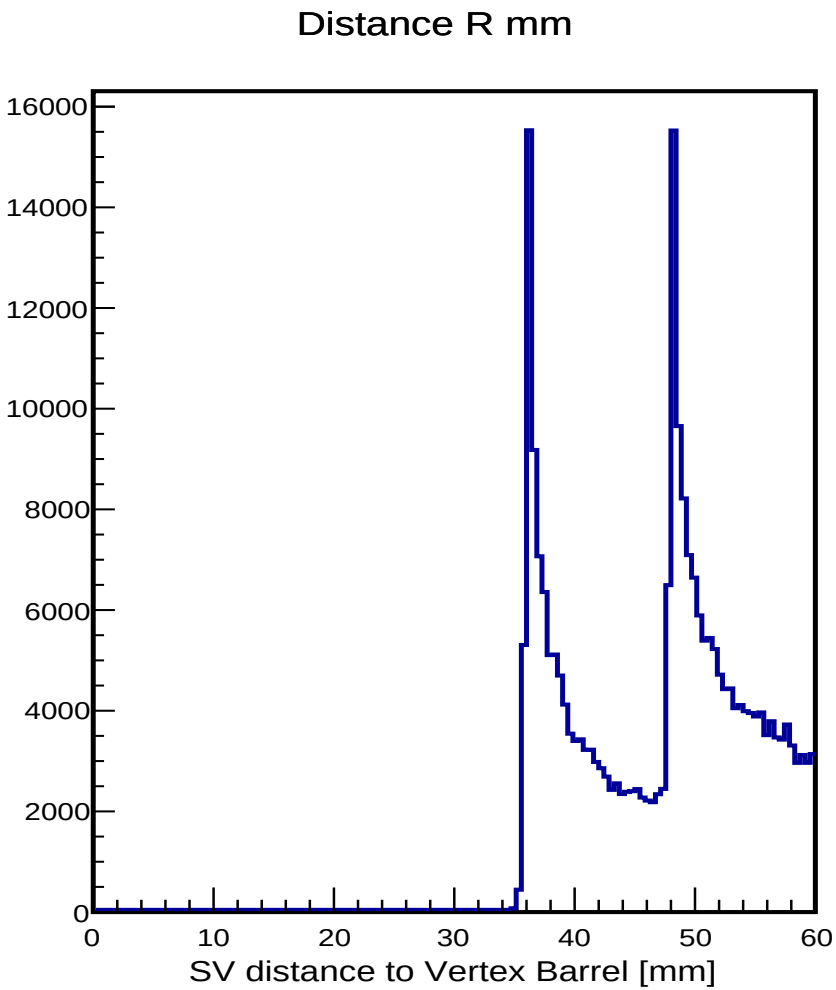
M_Λ

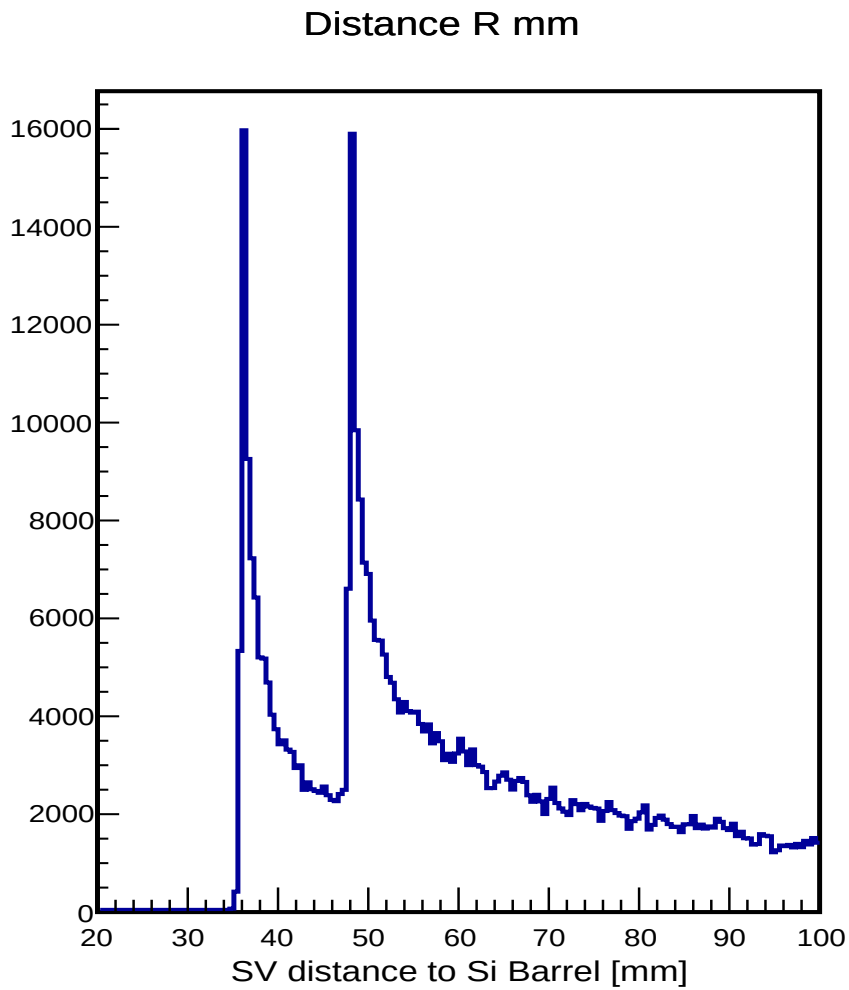

Invariant Mass M_Λ
M_Λ at vertex
M_Λ at sec. vtx perigee
400
350
300
250
200
150
100
50
0
0.9
1
1.1
1.2
1.3
1.4
1.5
Invariant Mass [GeV]

SV Multiplicity



SV Distance to Material





Using Secondary