Geometric 3D Vertex Fitter

Giuseppe Cerati
LArSoft Coordination Meeting
Sep. 26, 2017

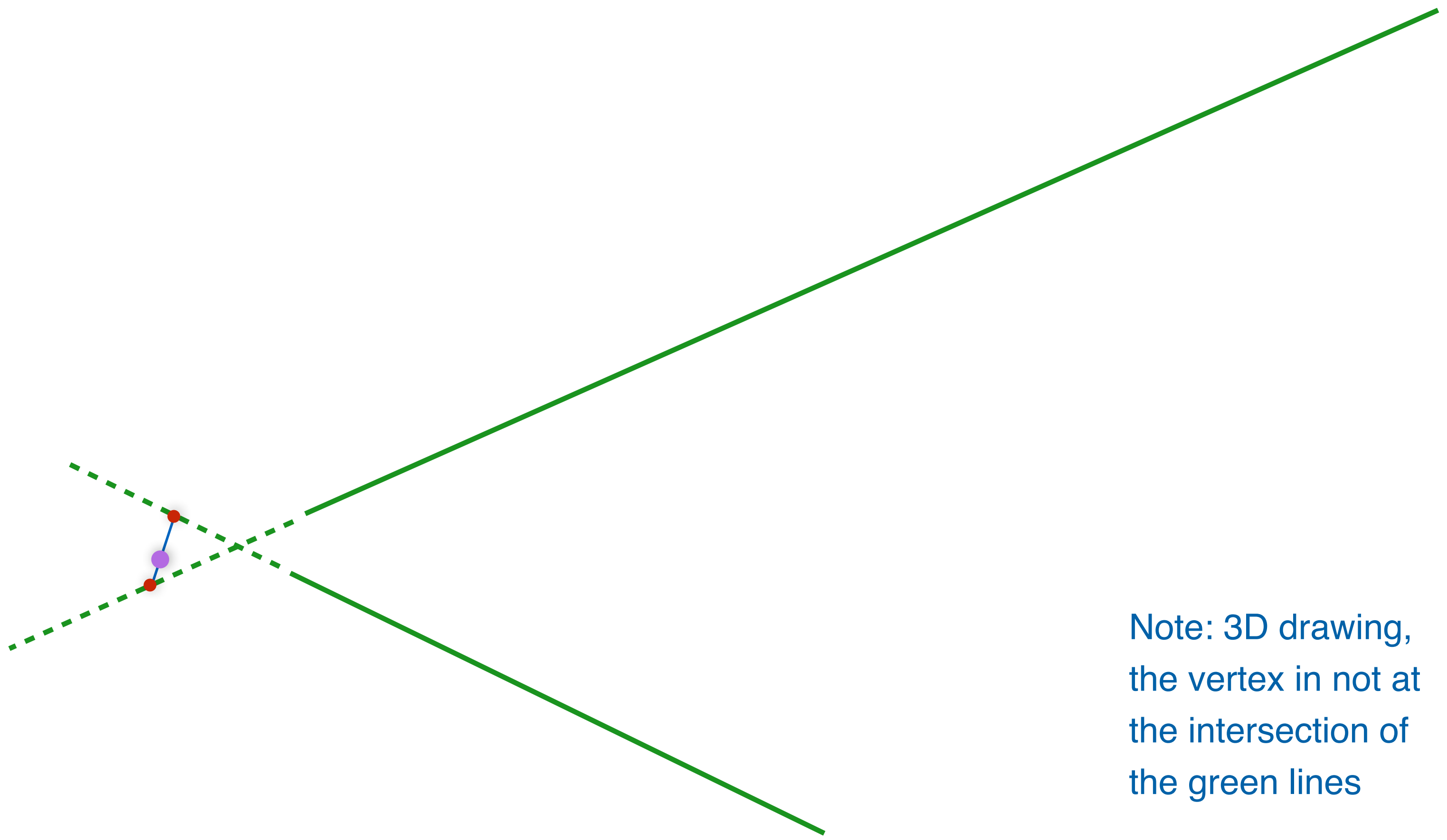
Introduction

- Kalman Filter track fits available in LArSoft since about half a year
- They provide updated track parameters and full covariance matrix
  - as well as fit ndof and chi2
- The most natural use for fitted tracks is to use them as input to fit vertices

- This talk:
  - short introduction about a simple vertex fit algorithm
  - discussion about changes to recob::Vertex needed to store the fit results

Two tracks vertex fit

- Consider the lines defined by the track start position and direction
- Find the two points along the lines with minimum distance
- Propagate the track uncertainties to the two points
- The vertex (position and uncertainty) is computed from the weighted average of the two points

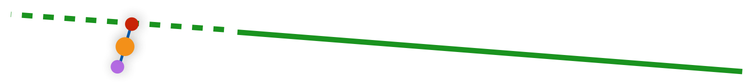


Note: 3D drawing, the vertex in not at the intersection of the green lines

Note: the algorithm neglects that the point with minimum distance on one track depends on the parameters of the other track

Vertices with >2 tracks

- In case a vertex has more than two tracks, tracks are sorted by number of hits
- The first two are fitted as before to get the 2-track vertex, the others are added as follows
- Consider the line defined by the 3rd track start position and direction
- Find the point along the line with minimum distance to the 2-track vertex
- Propagate the track uncertainties to the point
- The updated vertex (position and uncertainty) is computed from the weighted average of the 2-track vertex and point
- Repeat for 4th track (with 3-track vertex), etc.



Note: the algorithm neglects that the point with minimum distance on the track depends on the position of the 2-track vertex

Features of the fitted vertex

- Fitted vertices contain the following information:
  - Position in 3D
  - Covariance matrix
  - Chi2
  - Propagation distance from track start
    - expected to be backward or ~0
  - Impact parameter (distance from vertex to closest point along track direction)
    - expected to be ~0

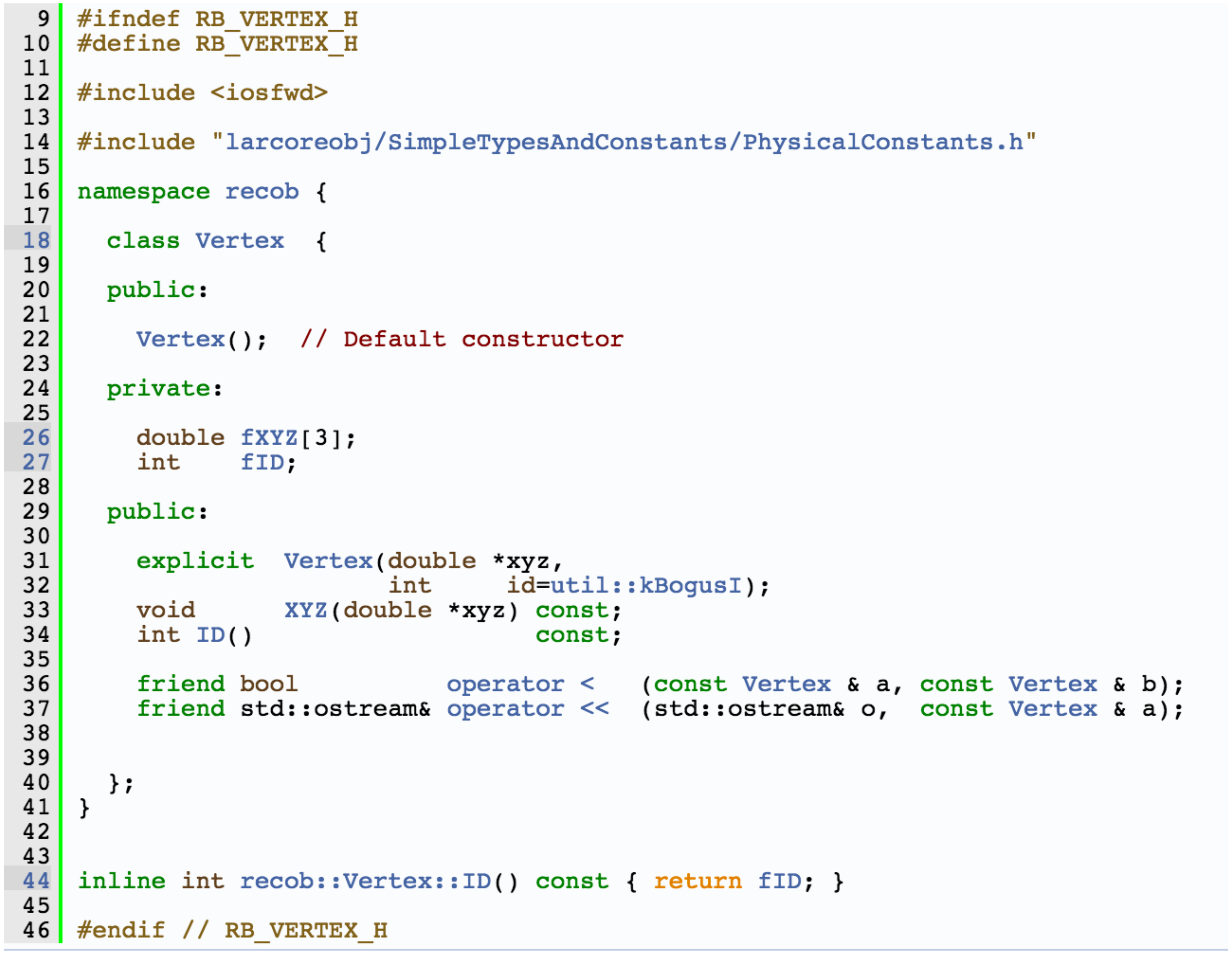
- They can be used to study:
  - goodness of vertex fit
  - compatibility of a track or a shower with the vertex

Technical details

- Code living in larreco feature/cerati_vertex-fit-devel branch
- Producer: larreco/VertexFinder/VertexFitter_module.cc
- Algorithm: larreco/RecoAlg/Geometric3DVertexFitter.h/cxx
  - fit function takes PFP and associated tracks as input
  - could become a tool so that the producer can easily change algorithm
- Current recob::Vertex object not adequate for storing fitted vertex results

recob::Vertex

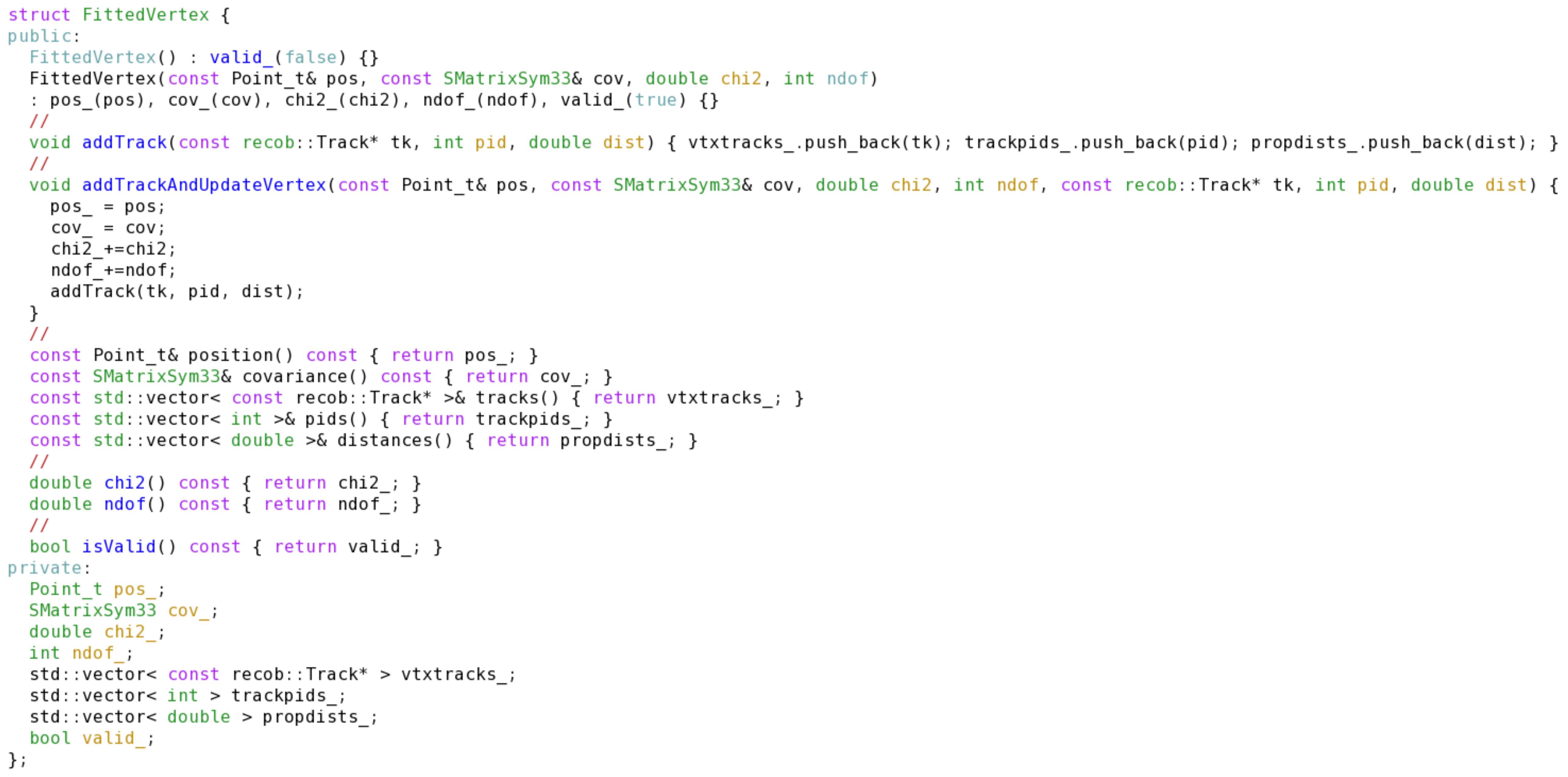
- Current recob::Vertex is just 3D position (array of double) and an ID (int)



```
#ifndef RB_VERTEX_H
#define RB_VERTEX_H

#include <iosfwd>

#include "larcoreobj/SimpleTypesAndConstants/PhysicalConstants.h"

namespace recob {

  class Vertex  {

  public:

    Vertex();  // Default constructor

  private:

    double fXYZ[3];
    int    fID;

  public:

    explicit  Vertex(double *xyz,
                     int     id=util::kBogusI);
    void      XYZ(double *xyz) const;
    int ID()                   const;

    friend bool           operator <   (const Vertex & a, const Vertex & b);
    friend std::ostream& operator <<  (std::ostream& o,   const Vertex & a);


  };
}


inline int recob::Vertex::ID() const { return fID; }

#endif // RB_VERTEX_H
```

Local version: FittedVertex

- Temporarily storing fit results in a dedicated struct



```
struct FittedVertex {
public:
  FittedVertex() : valid_(false) {}
  FittedVertex(const Point_t& pos, const SMatrixSym33& cov, double chi2, int ndof)
  : pos_(pos), cov_(cov), chi2_(chi2), ndof_(ndof), valid_(true) {}
  //
  void addTrack(const recob::Track* tk, int pid, double dist) { vtxtracks_.push_back(tk); trackpids_.push_back(pid); propdists_.push_back(dist); }
  //
  void addTrackAndUpdateVertex(const Point_t& pos, const SMatrixSym33& cov, double chi2, int ndof, const recob::Track* tk, int pid, double dist) {
    pos_ = pos;
    cov_ = cov;
    chi2_+=chi2;
    ndof_+=ndof;
    addTrack(tk, pid, dist);
  }
  //
  const Point_t& position() const { return pos_; }
  const SMatrixSym33& covariance() const { return cov_; }
  const std::vector< const recob::Track* >& tracks() { return vtxtracks_; }
  const std::vector< int >& pids() { return trackpids_; }
  const std::vector< double >& distances() { return propdists_; }
  //
  double chi2() const { return chi2_; }
  double ndof() const { return ndof_; }
  //
  bool isValid() const { return valid_; }
private:
  Point_t pos_;
  SMatrixSym33 cov_;
  double chi2_;
  int ndof_;
  std::vector< const recob::Track* > vtxtracks_;
  std::vector< int > trackpids_;
  std::vector< double > propdists_;
  bool valid_;
};
```

Fermilab

A proposal for a new recob::Vertex

- Up for discussion!

- 3D position (ROOT GenVector, aka recob::tracking::Point_t)
- 3D covariance matrix (ROOT SMatrix, aka recob::tracking::SMatrixSym33)
- chi2 (double), ndof (int)
- Assn to tracks used in vertex fit
- Additional info for tracks used in fit:
  - 1) distance from track start to closest point to vertex along track direction
  - 2) impact parameter of track to vertex
  - 3) significance of impact parameter of track to vertex
  - info should be unbiased: vertex computed without the track in question
  - can be stored as meta data in Assn
- Provide functionality for computing those quantities for any track
  - ideally also for showers

- Need to create I/O rules for backward compatibility

- A façade or wrapper may be provided for more user friendly access

Conclusions

- Presented a first version of Geometric3DVertexFitter algorithm
  - simple approach based only on 3D position and direction of tracks (with uncertainties)
- Provides additional handles for assessing vertex quality and compatibility with tracks or showers
- Its result do not fit into current recob::Vertex data produce
- A new version of recob::Vertex is being proposed
  - collecting inputs before proceeding with final implementation