Status on Modularized seeding algorithm development

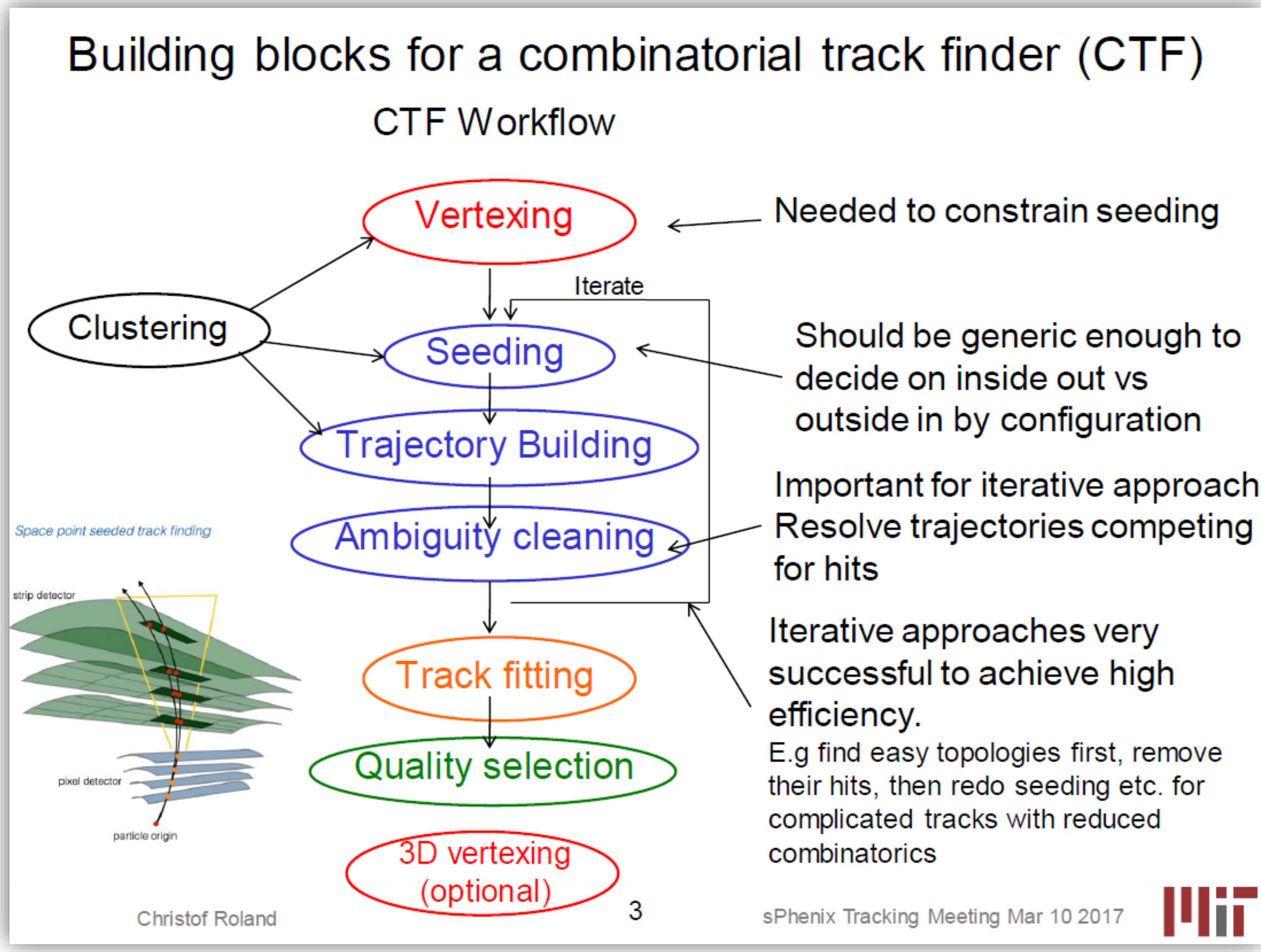
sPHENIX simulation meeting (July 11, 2017)

Anthony Frawley (FSU)

Christof Roland (MIT)

Sookhyun Lee (ISU)

Current status on tracking software development



Building blocks for a combinatorial track finder (CTF)

CTF Workflow

Needed to constrain seeding

Should be generic enough to decide on inside out vs outside in by configuration

Important for iterative approach Resolve trajectories competing for hits

Iterative approaches very successful to achieve high efficiency.

E.g find easy topologies first, remove their hits, then redo seeding etc. for complicated tracks with reduced combinatorics

Christof Roland 3 sPhenix Tracking Meeting Mar 10 2017

- (On Left) Overview of tracking software development plans proposed by Prof. Christof Roland
- Not much work has been done on developing algorithms for initial vertexing & modularized track seeding.

Plans for modularized track seeding
: discussed during tracking workfest (June 10~11)

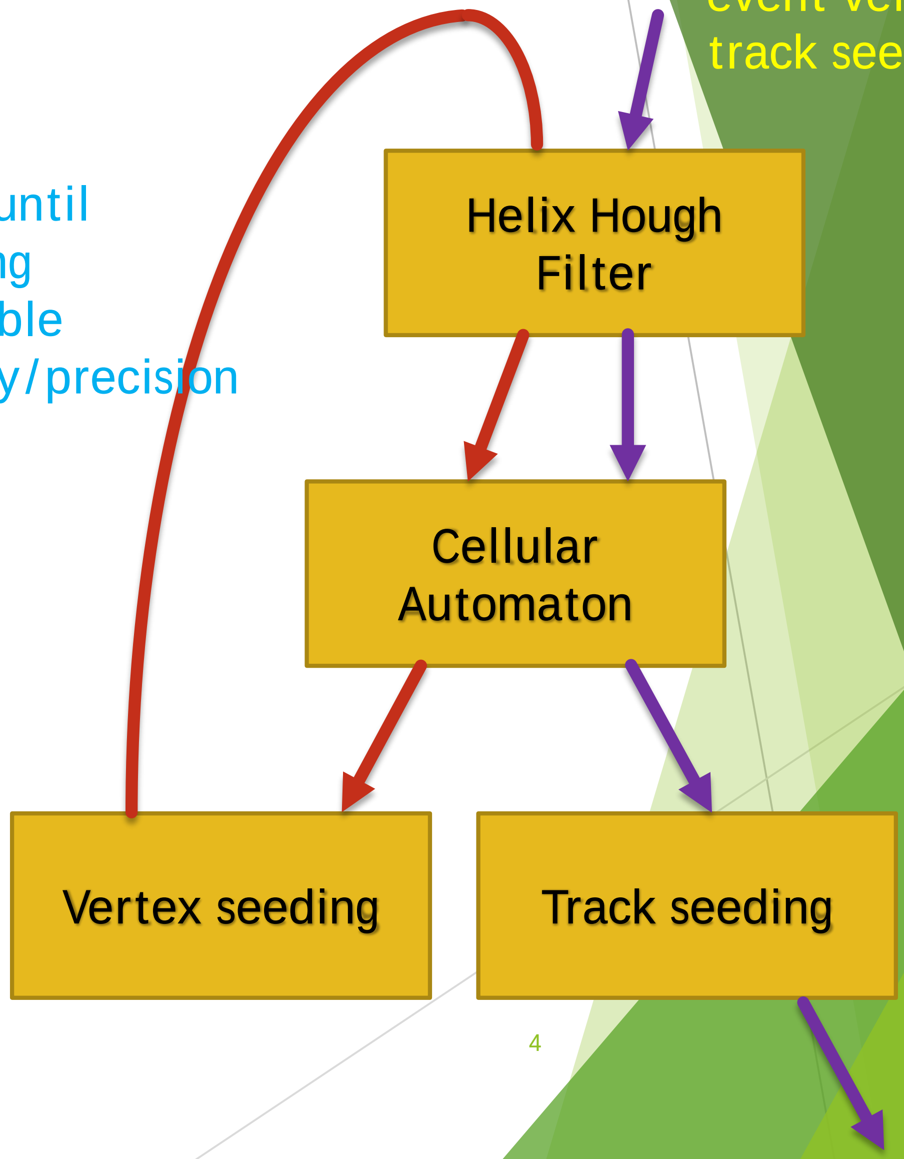
- Seeding modules will have to be modularized enough to be able to handle vertex seeding as well as track seeding.
- Less dependence on Hough seeding due to concerns of potentially introducing bias in tracking performance.
- Progressive track finding strategy: easier ones first, near perfect tracking efficiency for tracks originating from primary vertex.

Procedures for vertex & track seeding

- Use Hough transformation to select clusters whose helix parameters fall within constrained kinematic range, i.e. high pT, positive/negative eta ranges, primary vertex ranges and etc.

- Run sampled clusters through Cellular Automaton for fast vertex finding. (Previously, Cellular Automaton will be performed indiscriminately on random triplets, therefore took quite a long time to achieve a reasonable precision.)

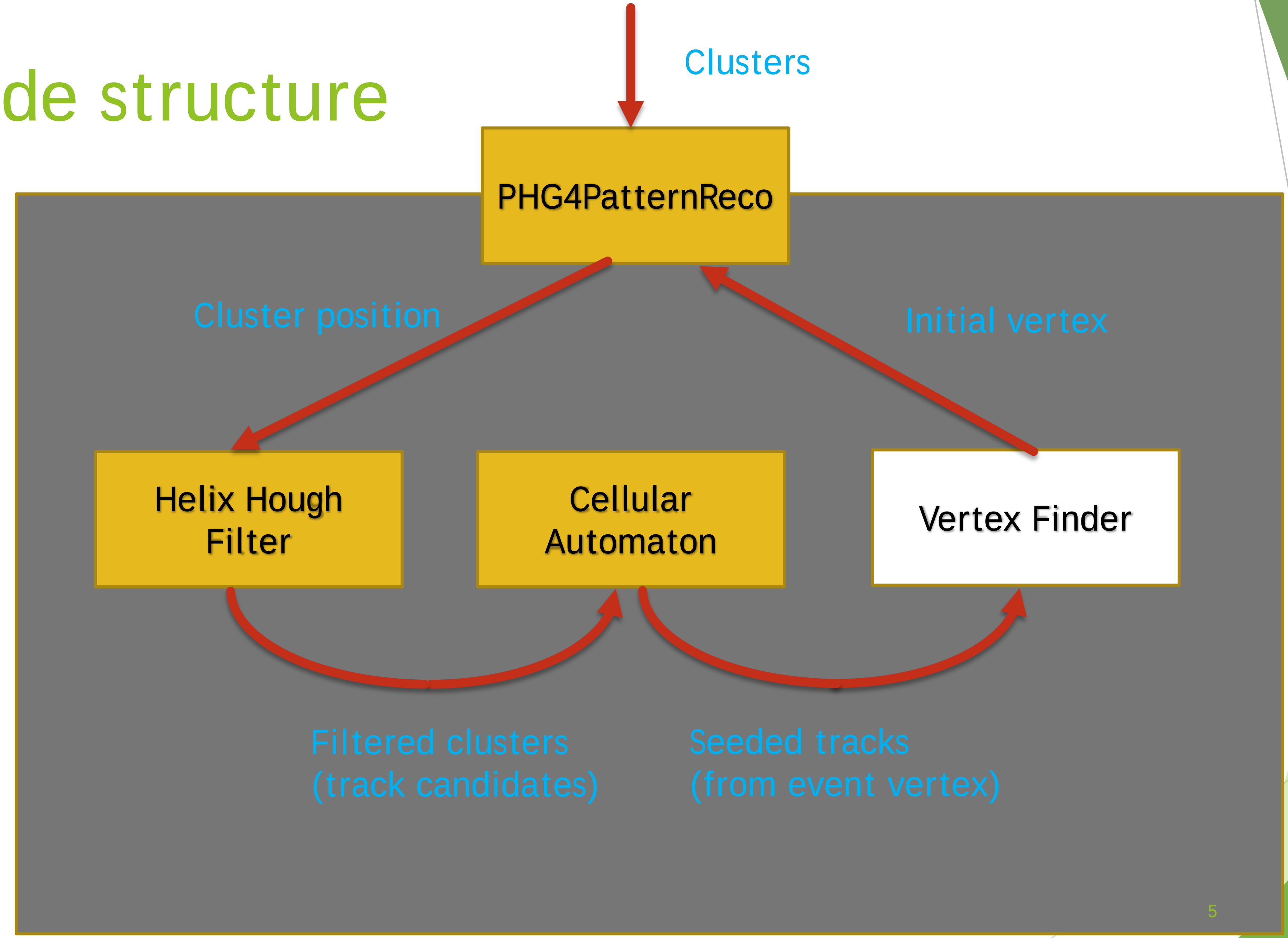
- Hough filter for selective tracks suggested by Tony & Christof



Iterate until achieving reasonable accuracy/precision

Once vertex is found, place clusters on event vertex & start track seeding

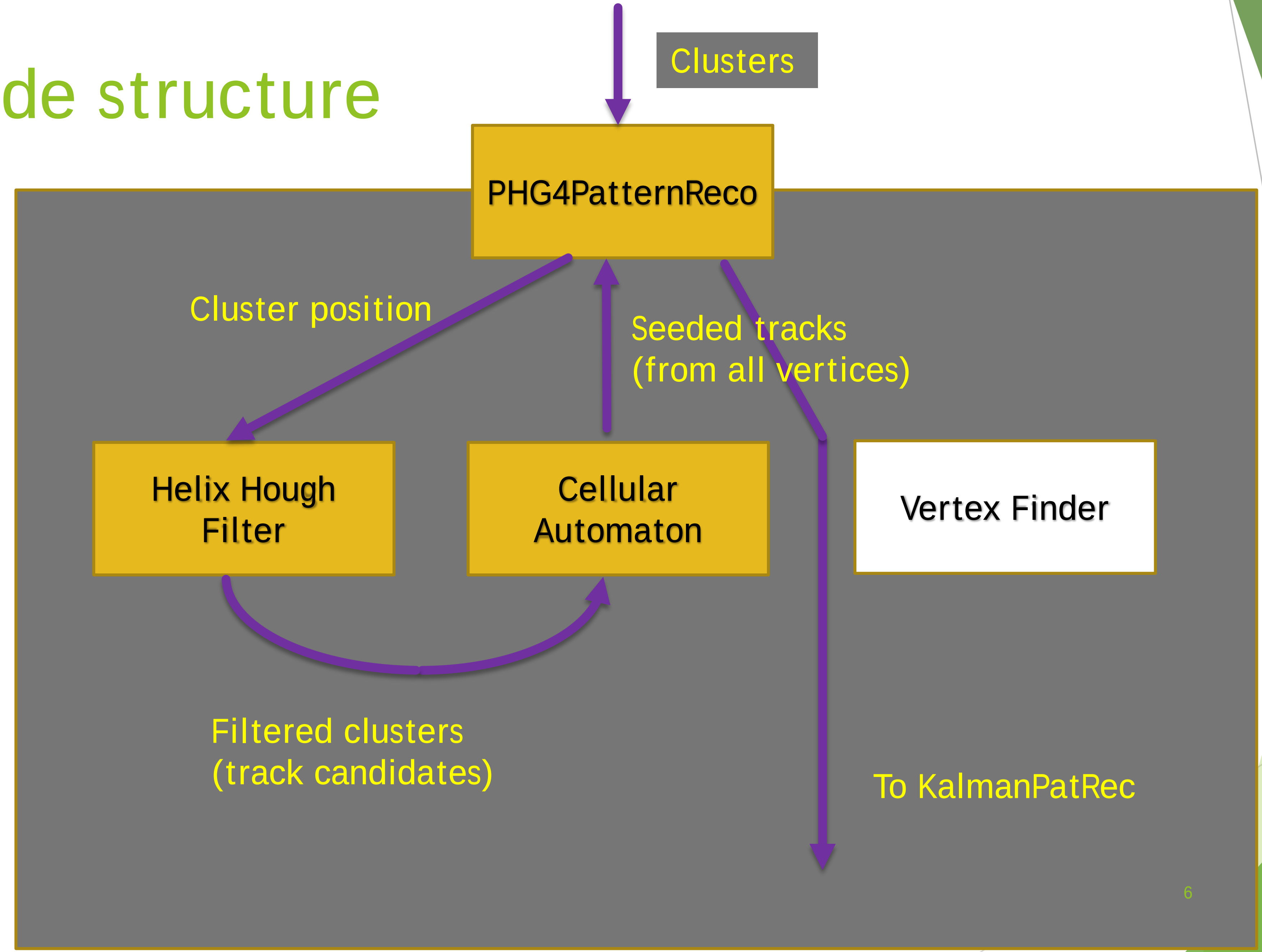
Code structure



🟨 : Module written

⬜ : Module not written

Code structure



Codes (currently all under g4hough)

- HelixHoughBin
- HelixHoughSpace
- HelixHoughFuncs

- Data encapsulation.
- Do not carry around vectors of clusters for counting.
- Voting and pruning steps can be added as we wish (output tracks can be checked at each step)

- PHG4PatternReco

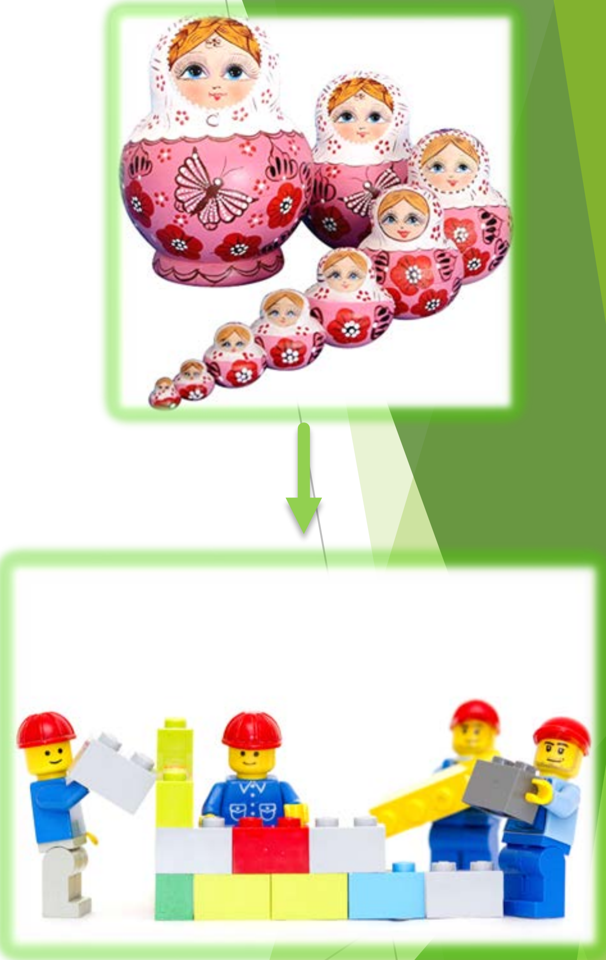
- CellularAutomaton

- Fast & simple seeding
- Remove bad clusters

- VertexFinder

- Cluster3D
- Track3D

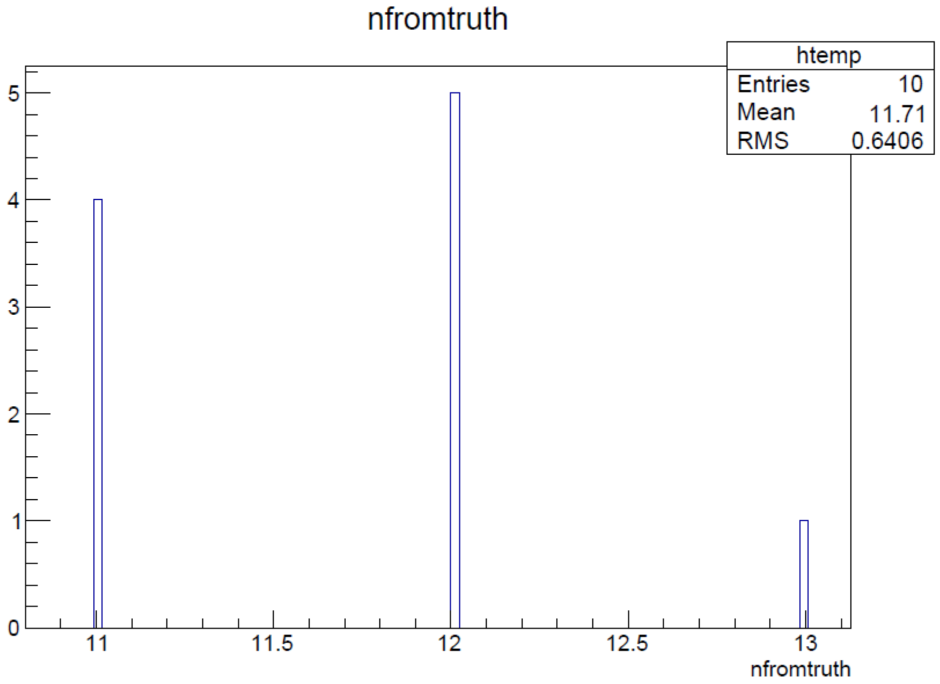
- Names changed from SimpleHit3D & SimpleTrack3D to avoid conflicts in future.

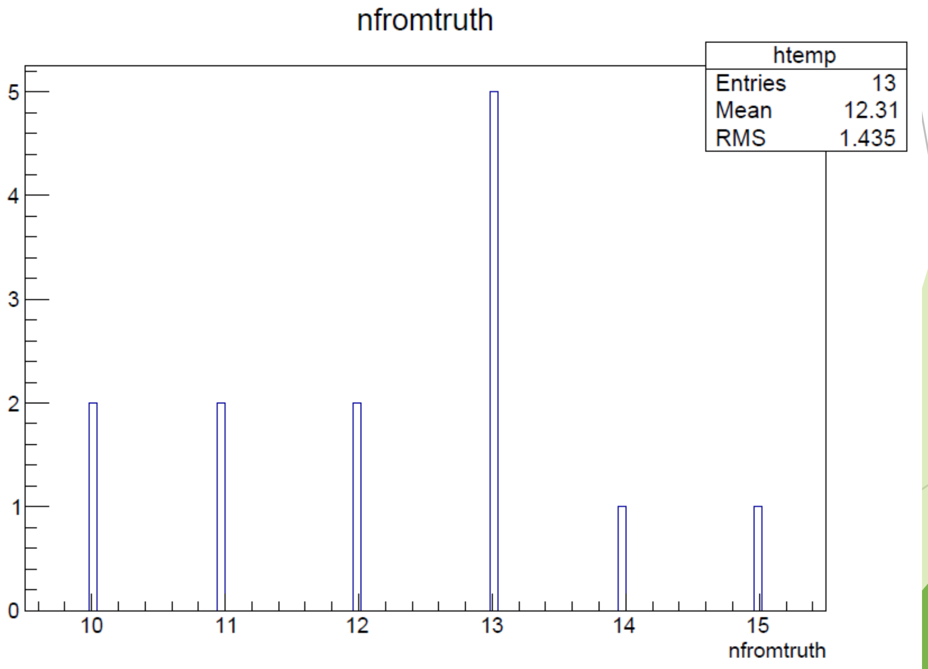


From Russian doll style
to Lego block style.

Hough filter seems to easily find ~10 seeded tracks at 1 iteration

(and of course we need a lot more iterations for hundreds of tracks)





12 seeding layers/
11 hits required

10 positrons between
5~10 GeV/c

15 positrons between
5~10 GeV/c

Summary

- Codes still under development.
- Trying to make certain each module works correctly before moving to the next module.
- Need input and feedback!