

Some Thoughts about Hits, Geometry etc

Rob Kutschke, Hans Wenzel

Fermilab

March 13, 2007

Overview

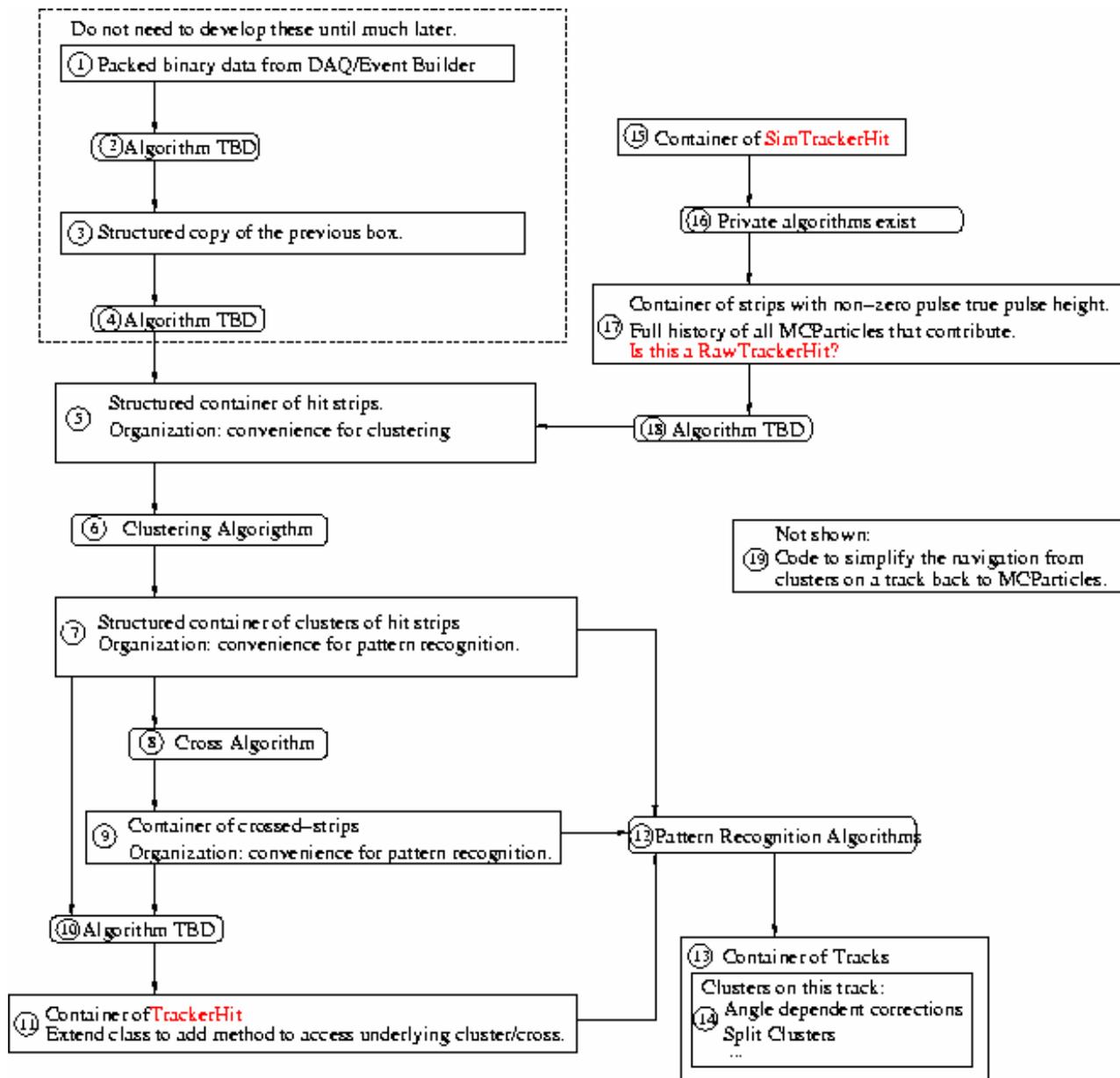
- Extend our ideas about “hits” so that we can write the next round of pattern recognition and fitters.
 - Includes moving between hits and geometry.
- Building from:
 - Existing code.
 - Dima’s observations from a few weeks ago.
 - Rich’s suggestions from last week.
 - Our own ideas of what we want for development of forward tracking code.
- So far in software we have: SimTrackerHit, RawTrackerHit, TrackerHit.
- These do not have the richness for all that we need to do.

Goals for This Talk

- List all of the ideas that are important for hits and sketch how they are interrelated.
 - Do not need all details today.

Sensors

- Do we want to consider non-planar sensors?
- If not, then fundamental unit of tracking software is the sensor.
 - Sensors have position and orientation.
 - Alignment will be done on a per sensor basis.
 - Hot/dead channels may be maintained on per sensor or per readout chip basis.
- How many sensors in SiD?
 - $O(20,000)$, including pixels + strips?
- Last week Rich presented a list of frequently used information that might not be present in the “native” geometry representation and needs to be derived from it.
- I agree that this is the information that we need.



General Comments

- Discussion is shown for strips.
 - Can be generalized to include pixels.
- Remember that this framework should work not just for MC but also for actual ILC data and for testbeam data.

- Boxes 1...4
 - Design driven by electronics and DAQ.
 - No need to implement these now.
 - These objects know nothing of geometry.
 - Algorithm in box 4 might throw out strips with a pulse height below threshold?
- Box 5
 - The meeting point between data and MC.
 - Internal organization is driven by the needs of the clustering algorithms.
 - Need to be able to ask this object:
 - Give me a container of all hit strips/pixels on a single sensor.
 - How do identify which sensor: SensorId ...next page.
 - Strips measure the u-coordinate.
 - If MC, must link back to box 17.
 - If data, should probably link back to data in box 3?

What is SensorID?

- Something that exists for the purposes of **fast** lookup and x-ref of “hits” and of geometry info.
- Probably a derived product of the geometry system that can be created only after the geometry is instantiated.
- Not a 64 bit cell Id.
 - I really mean that it should be implemented at the sensor level.
 - So we have about 20,000 of them.
- Possible implementations:
 - Option 1: Multiple indexed container:
 - Vertexer/Tracker; Forward/Barrel; Layer; phi-segment; z-segment?
 - Option 2: A dense integer that indexes into an indirection array?
 - Option 3: ???
- In any case we only want a single copy of the sensor geometry in memory.

- Box 6:
 - May need separate methods for strips and pixels?
- Box 7: Clusters
 - Organization of this container needs to be chosen to optimize pattern recognition algorithms.
 - Need to be able to ask things like:
 - Give me a container of all clusters on this sensor.
 - Give a container of all clusters on layer n of the tracker, within the the specified bounds on z and phi.
 - Maybe this is a private method of a particular pattern recognition method, not of box 7 ????
 - Cluster must be able to say which strips it includes.
 - Need to be able to navigate from cluster to sensor geometry via sensorid.
 - Clusters measure the u-coordinate.
 - Cluster may contain a single strip.
 - At this level, clusters know nothing about tracks.
 - Do we require each strip/pix is in exactly one cluster? Probably at this level?

Aside: Crosses

- In forward tracker we are considering two layers of crossed strip sensors.
 - Cross: intersection point of two clusters, one from each sensor.
 - Conceptually similar to traditional 2-sided Si.
 - But:
 - Sensors might be staggered.
 - Sensors are not at same z.
 - Strips might be at arbitrary angle, not just 90° .
 - For now don't bother to define this further.
 - Need to be able to navigate quickly to geometry of both sensors.
- A given cluster can belong to many crosses.
 - There will be ghost crosses.

- Box 8:
 - Only needed for strips, not pixels.
- Box 9: Crosses
 - Same comments as for box 7 (clusters).
- Box 10:
- Box 11:
 - Can fill TrackerHits from clusters or crosses.
 - TrackerHit can link back to its precursor cluster or cross.
 - Dima agrees that this would solve his geometry access problems.
 - This will provide the link to the geometry that Rich was talking about last week.
 - Rich: does this solve your problem?

Aside: Forwarding Functions?

- I said that TrackerHit links back to its precursor.
 - I think it should have a method to return its precursor (by reference, not value).
 - I don't think that TrackerHit should inherit from cross or cluster.
 - I don't think that it should forward methods from its precursor. If you forward when do you stop? Does it forward all of the geometry?

- Box 12:
 - Box should have been called pattern recognition and track fitting.
 - There are already many pattern recognition and fitting codes. I presume they will coexist for a while and then one or two will win out or will incorporate the others.
 - At FNAL we plan that our forward tracking code will be driven by clusters and, maybe, crosses.
 - Existing TrackerHit based code can still run and can be extended for a proper treatment of strips.

- **Box 13:**
 - All tracking codes should produce the same track objects.
 - The track objects should contain a list of which clusters, crosses, or TrackerHits that they contain.
- **Box 14:**
 - Once a cluster is assigned to a track we can compute the fully corrected centroid.
 - Corrected info is maintained here, not in box 7.
 - If a cluster wants to be on two tracks, we might want to split the cluster and this is the place to maintain that bookkeeping.
 - When a cluster is split, we will not modify box 7.

The MC Side

- Box 15:
 - These are made by SLIC.
- Box 16:
 - I understand that prototype algorithms exist but that they are only in private code?
- Box 17:
 - I know what I want this to be but I am not sure that it is really a RawTrackerHit.
 - If it is RawTrackerHit, then that class must be modified to return a container of contributing SimTrackerHits, not just a single SimTrackerHit.]
 - We must also provide a method to return how much electronic noise was added to the true pulse height.
 - See next page.

Why both Boxes 5 and 17?

- Different internal organizations:
 - Box 17 is organized for the convenience of MC hit creation.
 - Box 5 is organized for the convenience of clustering algorithms.
- Lots of the information present in MC in box 17 is not present in data.
 - I don't like the idea that real data objects carry around a lot of MC related interface.
 - I am willing to be convinced otherwise if I have missed some advantages.

- Box 18:
- Box 19:
 - A reminder that we need to provide convenience functions for navigating from the reconstructed tracks back through the full reco history and the full MC history.
 - We can talk later about where these methods live. Are they part of the existing classes or are they their own classes or even free functions.
 - Keeping with my comment on forwarding functions, I think that I will vote for one of the last two options.

Shortcuts for FastMC

