

Some Thoughts about Hits, Geometry etc

Rob Kutschke, Caroline Milstene,
Hans Wenzel
Fermilab
March 27, 2007

Overview

- The existing org.lcsim classes that represent hit info:
 - do not have the info needed for track finding and fitting.
 - or do not have it in a convenient form.
 - For example, See talks by:
 - [Nick and Dima at the Feb 13 meeting](#)
 - [Hans at Feb 27 meeting](#)
 - [Rich at March 6 meeting](#)
- Private, interim solutions have been implemented.
 - Difficult for others to use.
 - Unlikely to play nicely with each other.
- To remedy this we need to:
 - Modify some of the existing classes
 - Create new classes to represent new ideas.
 - Include all interested parties so we develop solutions that will be widely adopted.
- **This talk outlines a plan we can use for the next steps in this discussion.**

Guiding Principle

- We need a clear separation between geometry information and hit information.
 - A hit should not contain geometry info.
 - A geometry object should not contain hits.
- But hits are connected to geometry
 - We need a way to represent that connection.
 - The 64 bit channel ID is not a useful solution.
 - What's the channel ID of a cluster centroid?
 - The connection needs to be fast.

Reminders

- The native language of planar sensors is the (ruvw) local coordinates introduced by [Rich at the March 6 meeting](#).
 - r – Displacement of local origin.
 - u – unit vector along measurement direction
 - v – unit vector along the strip
 - w – unit normal to the plane.
- This information may not be present in the native geometry representation.
 - But can easily be derived from it.
- How many sensors do we have, pixels + strips?
 - I think its $O(20,000)$.
 - Small enough to compute derived information once and then cache it.

Where are We Now?

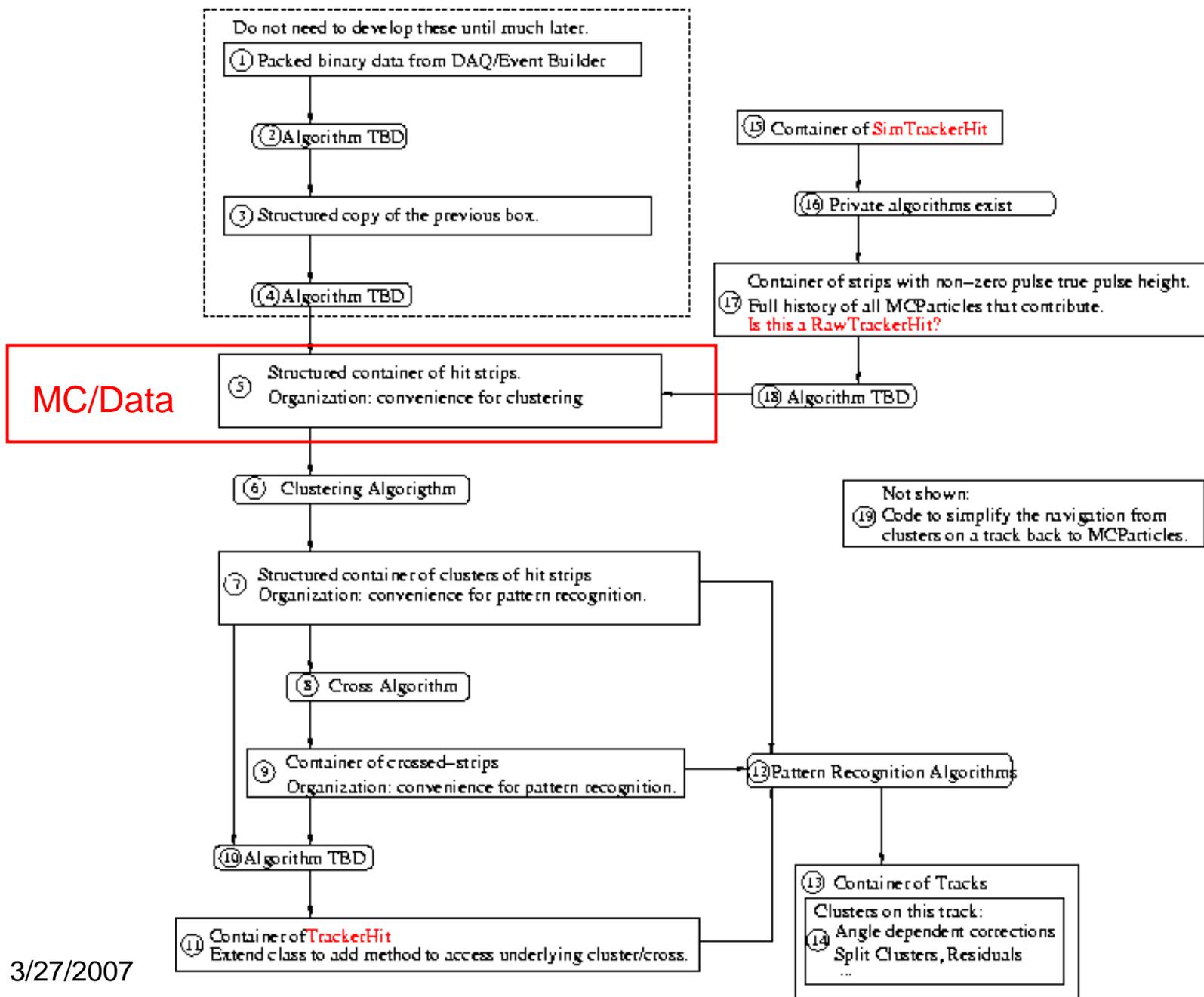
- Existing classes:
 - SimTrackerHit, RawTrackerHit, TrackerHit.
- Missing ideas:
 - Clusters of hit strips/pixels.
 - Maybe crosses of crossed strip clusters.
 - The native information of strips/pixels is (ruvw) not (xyz) but it is difficult to extract (ruvw) from existing classes.
 - Clear definition of the common point for data flow in real data and MC events?
 - Classes that organize clusters for fast access by pattern recognition and fitting codes.
 - I understand that RawTrackerHit is a flat container.
 - Chain of MC truth trace back is incomplete.

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.
- Understand where the data and MC processing streams meet.

Sensors

- Assume planar sensors.
 - Is this safe long term? Safe for now?
- One bookkeeping unit of tracking software is the sensor.
 - Need to organize hit strips by sensor in order to do clustering.
 - Sensors have position and orientation.
 - Same for all strips on wafer.
 - Alignment will be done on a per sensor basis.
 - Hot/dead channels may be maintained on per sensor or per readout chip basis.
- Various pattern recognition codes will need different organizations of the same hit information.
 - For example:
 - All clusters on a given sensor.
 - All clusters on a specified (layer, z segment, phi segment).
 - How do we support multiple organizations?



General Comments

- Next pages discuss the figure on the previous page.
- Discussion and figure are shown for strips.
 - Can be generalized to include pixels.
- Remember that this outline 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 but do know about the calibration database.
 - 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.**
 - Objects in box 5 do not contain geometry info, but one can find that info using the **SensorId.**
 - Strips measure the u-coordinate.
 - Strip object has no information global coordinates.
 - If MC, must link back to box 17.
 - If data, must link back to data in box 3.
 - Link may be implicit if box 4 does not throw out information?

What is SensorID?

- Something that exists for the purposes of lookup and cross-reference 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: Multi-index object:
 - Vertexer/Tracker; Forward/Barrel; Layer; phi-segment; z-segment?
 - Option 2: A dense integer that indexes into an indirection array?
 - Option 3: ???
- We only want a single copy of this derived sensor geometry in memory.
- A SensorId should be a small object that can be added as member to data to many objects.
- The operation of getting geometry information given a SensorID must be fast because it will be done often.
 - Any complexity needed for fast access should be implemented within the geometry system not within the hit system.
 - To a hit, a sensor id is just a magic key to the geometry.

- 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 me a container of all clusters on layer n of the tracker, within the some specified bounds on z and phi.
 - Maybe this is a method in some other class, not a method of box 7
???? If so, then box 7 only needs to know about the sensorid, not other properties of the geometry.
 - 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 object has no information in global coordinates.
 - 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.
 - Probably specified in global coordinates.
 - Conceptually similar to traditional 2-sided Si. But:
 - Sensors are offset longitudinally (r or z).
 - Sensors might be staggered transversely.
 - Strips might be at arbitrary angle, not just 90°.
 - For now don't bother to define this further.
 - Need to be able to get back to the underlying clusters.
 - 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.
 - Pulse height correlation less powerful since two sensors.
- I think that crosses will be useful for pattern recognition but that fitting will be done using two individual clusters.
 - Because the two sensors are at different locations, unlike traditional double-sided Si.

- **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.
 - Legacy code still needs to run.
 - 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 it 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 info that can be found by following precursors to their 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.
 - Track should also know about its residuals.
 - The existing reconstructed track objects will need to be extended to get all this in.
- **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?

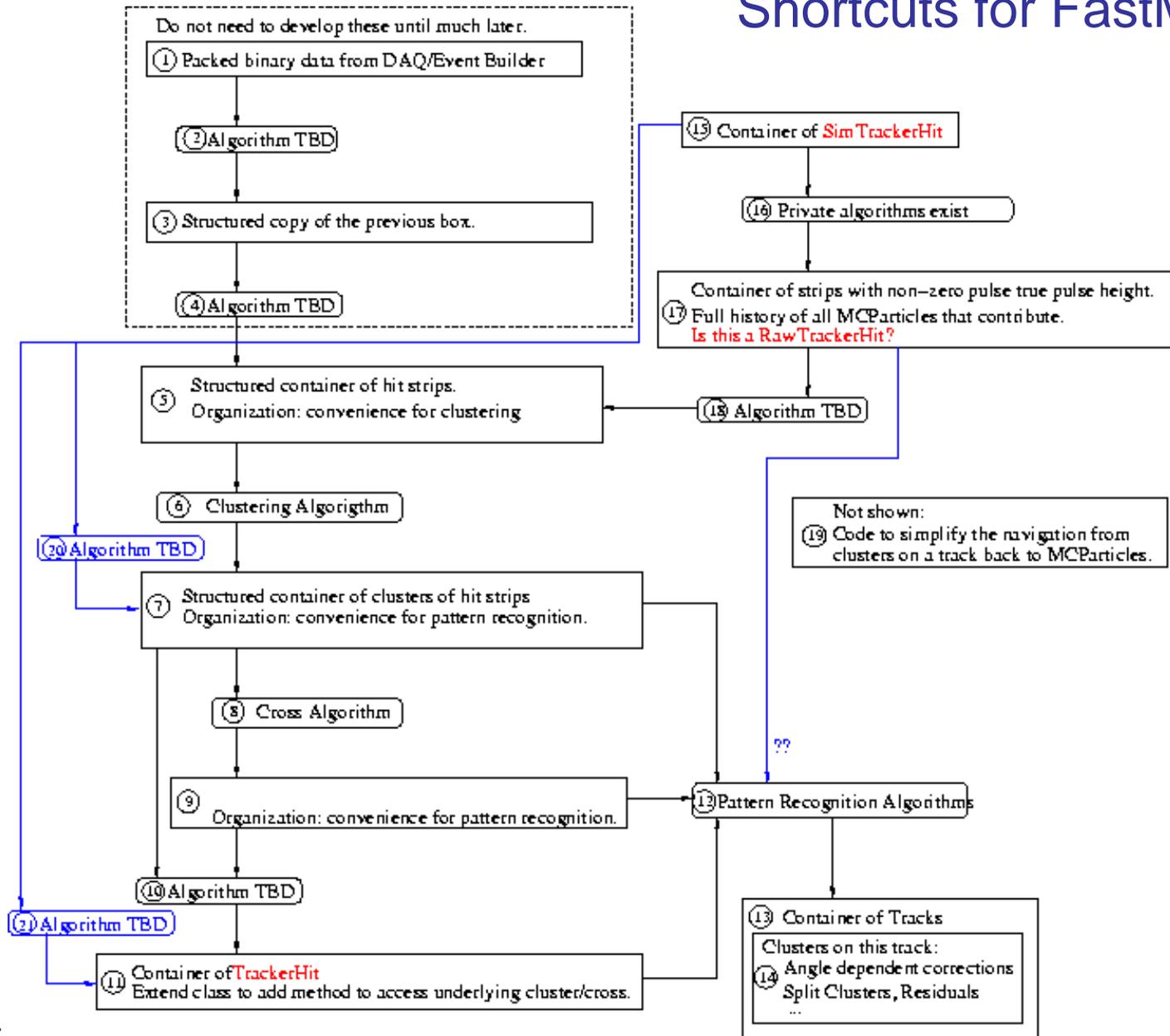
- Different internal organizations:
 - Box 17 is organized for the convenience of MC hit creation.
 - Box 5 is organized for the convenience of clustering algorithms.
- They understand different geometry abstractions:
 - Box 17 knows cellids.
 - Box 5 knows sensor ids.
- Much of the information present in box 17 is not present in data. It is MC specific.
 - I don't like the idea that objects representing real data need to know about MC classes in order to be instantiated.
 - I need to know more about Java to before discussing options here.

- Box 18:
 - First version of this can be very simple.
- 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.

Short Cuts for FastMC

- We still need to support various short cuts for fast MC.
 - These are shown on the next page.
- In particular we do need to make perfect, gaussian clusters, straight from the SimTrackerHits, in order to certify that pattern recognition and vertexing code are correct.

Shortcuts for FastMC



Polymorphism and Inheritance

- How are hit pixel and hit strip classes related?
- In Boxes 5 and 7:
 - What is the internal organization of the containers
 - 4 containers: strips/pix, forward/barrel?
 - Is there one container with everything?
 - 2 containers?
 - My guess is that internally we should maintain 4 separate containers but provide interfaces that allow access in all ways.
 - We can change the internal organization as we learn about real access patterns.
- My experience: it produced more problems than it solved to treat strips as pixels that just happen to one very long dimension: adding a v measurement at the mid point of the strip can, in some circumstances, poison the fitted track parameters.
 - There is still lots of room for polymorphism and inheritance but we need to be smart about it.
 - I do think that the geometry information needed for a pixel sensor and a strip sensor are the same: the quantities in Rich's talk last week.

Persistency

- Modified existing classes need to be persistable.
- For development purposes it would be great if boxes 5,7,9 were persistable.
- For production there are many things we would choose not to write out.
- In the long run we will need to understand what can be recomputed as needed so it does not need to be written out.

Visualization

- I have not thought this through yet. Just a few thoughts for now.
- It can be useful to visualize a hit strip or a cluster as a line segment.
 - Will need some control to say which strips you want to visualize and which to suppress or else the picture can get too busy to be useful.
- Clusters of pixels and crosses should be straightforward for visualization.

Summary

- **Need to introduce the concept of SensorId**
 - Used to connect hits to geometry.
- Geometry system needs to provide the (ruvw) information Rich mentioned March 6, accessible by SensorId.
- Need new classes to represent:
 - Structured containers of hit strips or pixels.
 - Structured containers of clusters of strips or pixels.
 - Structured container of crosses (strips only).
 - These classes talk (r,u,v,w); no info in global coordinates.
- Some existing classes will need to be extended:
 - RawTrackerHit: return a list of contributing SimTrackerHits
 - TrackerHit: link back to its precursors.
 - Reconstructed track: need to add a container of HitOnTrack information.
- Need new tools to ease the navigation from reconstructed back to generated objects, and vice versa.
- New stuff needs to be persistable.
- Need to think of visualization.
- Need to understand where polymorphism and inheritance in the hit classes and hit container classes are useful.

What Next?

- Do people agree on the general outline?
 - If not, let's evolve it to get agreement.
- If yes then:
 - Define the per sensor geometry information.
 - Rich's suggestion is complete or close to it.
 - Who needs/wants to be in on the design of:
 - SensorId?
 - Modified classes?
 - New classes
 - What do we need to do to get persistency?
 - Who will do which work?
 - What is an aggressive but believable time scale for getting this done? A few weeks? Longer?
- Who, outside of this group, needs to be in on this?
 - Calorimeter and muon people? Non-SID users of LCIO?