Integrated Tracking-Clustering

Dmitry Onoprienko
Fermilab

2009 Linear Collider Workshop of the Americas.
Albuquerque, September-October 2009
What is ITC

Evolved from Calorimeter-Assisted track finder for SiD

Immediate task: non-prompt track reconstruction

Can do more...

Two-way information exchange between tracker and calorimeter

Requirements:
- extremely flexible
- decoupled from any particular algorithm on either side
- extendable
Basic Algorithm

For each Rosary:
- Propagate trajectory to the next layer
- Try to attach Dot (may branch)
- If unsuccessful, try to attach Bead
- If unsuccessful,
  - If this is a single Dot Rosary with no attached track, try to attach Dot based on proximity
  - Otherwise, attach Crack
- Check for Rosary termination condition
  - If first termination, reverse direction
- If not terminated, update trajectory

Rosary – chain of clusters, possibly with a track attached on one end.

Cluster hits in each calorimeter layer into Dots (MIP trace) and Beads
Seed first Rosaries from tracks
Thread created Rosaries
- For each Calorimeter Layer:
  - Create new Rosaries from unattached Dots
  - Thread created Rosaries
- Assign quality ratings to Rosaries, cut
- Remove duplicate parts

See if there is orphan track nearby – if yes, try to stitch.
Main Driver + Libraries

Pre-configured drivers:
- drivers
- BasicITC
- MipStubFinder
- SeedTrackerConfirm

Skeleton Driver:
- RosaryClusterer
  - FindDotsAndBeads
  - ChooseSeedLayers
  - SeedFromTrack
  - SeedFromDot
  - Step
  - ComputeTrajectory
  - SearchForDots
  - SearchForBead
  - SearchForProximityDots
  - ContinueStepping
  - ReverseThreadingDirection
  - AcceptThreadedRosary
  - SelectRosaries

Plug-Ins (more available, new can be added):
- FindDotsAndBeads_Basic
- ChooseSeedLayers_FirstN
- SeedFromTrack_Basic
- SeedFromDot_Basic
- Step_Basic
- ComputeTrajectory_Basic
- SearchForDots_NN
- SearchForBead_Basic
- SearchForProximityDots_NN
- ContinueStepping_Basic
- ReverseThreadingDirection_Basic
- AcceptThreadedRosary_Basic
- SelectRosaries_Basic
Example of Performance

SiD02 detector, ttbar @ 500GeV

Out-of-the-box MipStubFinder

Findable $K_S^0$

Non-prompt tracks
Examples of Use Scenarios
( focusing on tracking )

Follow-up on standalone track finder:
  • Run SeedTracker with default settings
  • Run ITC with built-in track finder and fitter

Fake rate reduction:
  • Run SeedTracker with default settings
  • Run ITC with no track finder
  • Run SeedTracker with default set of strategies but relaxed cuts on found tracks
  • Remove tracks not matched to ITC-produced seeds

Seed standalone track finder:
  • Run SeedTracker with default settings
  • Run ITC with SeedTracker as a track finder

Integrate with standalone track finder:
  • Run SeedTracker with default settings
  • Run ITC with no track finder
  • Run SeedTracker with outside-in set of strategies and relaxed cuts, use ITC-produced seeds instead of confirmation layer

All of the above in a single reconstruction job!
Interoperability with Other Packages

Implemented as a package in org.lcsim

Uses its own object model:

org.lcsim.contrib.onoprien.data
LCIO-compatible, WIRED-compatible
Converters provided to/from other hit/track classes used by several SiD algorithms

Geometry independent:

- Adapter available, can be customized for efficiency
- Knows how to find next sensor given the trajectory, neighbors of a sensor, etc.
- ITC code
- Tracker
- Calorimeter
- Virtual Segmentation
- Sensor
- CalGeometry
- Surface, Trajectory, Intersection

Small Print:
ITC expects to have standard org.lcsim geometry services available
Some classes in the algorithms library are wrappers for other people code
Status & Plans

Core package – functional and documented
  • need to provide example drivers
  • waiting for better standalone fitter to become available in org.lcsim

Integration with SeedTracker – prototype functional
  • releasable version is in the works

Interoperability – requires use of converters
  • need to standardize object model and infrastructure!
  • waiting for LCIO 2.0?

Miscellaneous supporting packages – functional and documented
  job services and management, MC truth access, performance testing and configurable cheaters, geometry services, auxiliary drivers, etc.
Backup slides
**Object model**

Simulation output
- SimTrackerHit
  - IRawTrackerData
    - ITrackerHit
      - ITrackerPulse
      - ITrackerHit
      - ITrackerHit
      - ITrackerHit
      - ITrackerHit
  - ITrackerHit extends ITrackerObject:
    - List<ITrackerHit> getParentHits()
    - List<ITrackerHit> getClusters()
    - List<ITrackerPulse> getPulses()  
    - IRefFrame getRefFrame()
    - Type getType()
    - Hep3Vector getDimensions();
    - double getdEdx()
    - double getTime()
    - Hep3Vector getLocalPosition()
    - Hep3Vector getLocalErrors()
    - Hep3Vector getPosition(IRefFrame referenceFrame)
    - SymmetricMatrix getCovMatrix(IRefFrame referenceFrame)
    - Sensor getSensor()

Truth pulse height in each channel
- IRawTrackerData
- ITrackerHit
- ITrackerPulse
- ITrackerHit
- ITrackerHit
- ITrackerHit
- ITrackerHit

DAQ output
- IRawTrackerData
- ITrackerHit
- ITrackerPulse
- ITrackerHit
- ITrackerHit
- ITrackerHit

Calibrated signal in each channel
- IRawTrackerData
- ITrackerHit
- ITrackerPulse
- ITrackerHit
- ITrackerHit
- ITrackerHit

Cluster: group of pulses that cannot be unambiguously separated + measurement
- IRawTrackerData
- ITrackerHit
- ITrackerPulse
- ITrackerHit
- ITrackerHit
- ITrackerHit

Derived measurement
- IRawTrackerData
- ITrackerHit
- ITrackerPulse
- ITrackerHit
- ITrackerHit
- ITrackerHit

LCIO based persistency
- MCTruthDriver
- MCTruth
Extendable library of algorithms. Includes both fast simplified algorithms (smearing, etc.) and full digitization (Tim Nelson, Nick Sinev)

Shortcut drivers available

Data exchange format: TrackerHitMap<K extends Identifiable, V>
Object model interfaces do not have any Monte Carlo specific methods.

MC information bookkeeping is done automatically by hit processing framework, and accessible through MCTruth object.

RecType:
- TRACKER_CLUSTER
- TRACKER_HIT
- TRACK_SEED
- TRACK
- CAL_HIT
- CLUSTER
- PARTICLE

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td><code>findMCParticles(RecType type, Object reconstructedObject)</code></td>
</tr>
<tr>
<td><code>&lt;T&gt; WeightedList&lt;T&gt; get(RecType type, MCParticle mcParticle)</code></td>
</tr>
<tr>
<td><code>WeightedList&lt;MCParticle&gt; getMCParticles(RecType type, Object reconstructedObject)</code></td>
</tr>
<tr>
<td><code>&lt;T&gt; WeightedTable&lt;MCParticle, T&gt; getMCParticleTable(RecType type, Collection&lt;? extends T&gt; collection)</code></td>
</tr>
<tr>
<td><code>&lt;T&gt; WeightedTable&lt;MCParticle, T&gt; getMCParticleTable(RecType type, String... collectionNames)</code></td>
</tr>
<tr>
<td><code>List&lt;SimTrackerHit&gt; getSimTrackerHits(IRawTrackerHit hit)</code></td>
</tr>
<tr>
<td><code>List&lt;SimTrackerHit&gt; getSimTrackerHits(ITrackHit hit)</code></td>
</tr>
<tr>
<td><code>List&lt;SimTrackerHit&gt; getSimTrackerHits(ITrackPulse pulse)</code></td>
</tr>
<tr>
<td><code>List&lt;SimTrackerHit&gt; getSimTrackerHits(MCParticle mcParticle)</code></td>
</tr>
<tr>
<td><code>&lt;T&gt; void setMCParticleTable(RecType type, WeightedTable&lt;MCParticle, T&gt; map)</code></td>
</tr>
</tbody>
</table>
Virtual segmentation

Extendable Library of SensorTypes

Define how sensitive volumes are tiled with Sensors.

Extendable Library of Segmenters

+ machinery for chaining

Applies on top of arbitrary simulation geometry

Hit processing framework

VSRawTrackerData
VSRawTrackerHit
VSTrackerPulse
VSTrackerHit

Sensor

SensorType

Defines geometry and other parameters of a Sensor, how it is tiled with channels, what Type of hits it produces.
Driver :

- Extended version of Driver

JobManager :

- Accepts listener registration and dispatch events that trigger geometry dependent initialization in client classes. Listeners can be registered along with dependencies among them - JobManager guaranties that listeners specified as "prerequisites" for some other listener will receive the event first.
- Allows registration and retrieval of singleton objects of any type.
- Provides access to the default AIDA object that can be used for histogramming, plotting, etc.
- Accepts `HepRepCollectionConverter` registration for visualizing Lists of objects in the event record using Wired event display.
- Can print message and/or save AIDA tree every specified number of events.
Cheaters and performance analysis tools.

See:

- Description

## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDefinition</td>
<td>Interface to be implemented by classes that define &quot;findable&quot;, &quot;reconstructed&quot; MCParticle, and &quot;fake&quot; reconstructed object - used by cheaters and performance analysis classes.</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisDriver</td>
<td>Base driver for testing event reconstruction algorithms performance.</td>
</tr>
<tr>
<td>CheatClusteringDriver</td>
<td>Cheater that attaches hits and tracks to reconstructed particles.</td>
</tr>
<tr>
<td>CheatRecoParticleCreator</td>
<td>Driver that creates a tree of reconstructed particles and vertices based on Monte Carlo truth information, packs it into RecParticleList, and puts it into the event record.</td>
</tr>
<tr>
<td>CheatRecoParticleDriver</td>
<td>Driver that uses MC truth info to produce a list of reconstructed particles.</td>
</tr>
<tr>
<td>CheatTrackFinderDriver</td>
<td>Driver that uses MC truth info to produce a list of tracks.</td>
</tr>
<tr>
<td>CheatVertexFinderDriver</td>
<td>Driver that uses MC truth info to produce a list of reconstructable vertices in the event, and to associate them with previously reconstructed tracks.</td>
</tr>
<tr>
<td>Definition</td>
<td>Configurable implementation of IDefinition.</td>
</tr>
</tbody>
</table>