The BinGo Pattern Recognition Package

Michael Marsh
University of Illinois at Urbana-Champaign
February 8, 2000

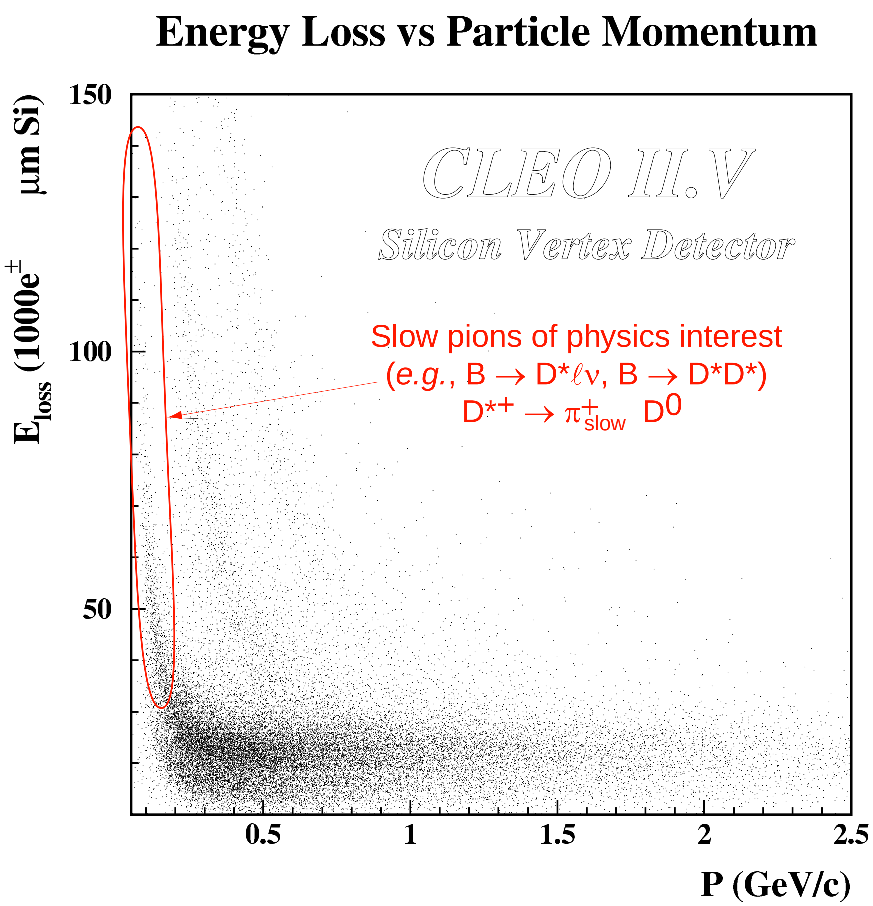
Outline

- Introduction
- The `Scungili` Pattern Recognition and Fitting Manager
- `LayerSets` for Pattern Recognition (PR)
- `TrackFilters` for Track Candidate Manipulation
- The `BinGo` Pattern Recognition Algorithm
- Some Basic `TrackFilter` Implementations
- Initial Performance Checks
- Ongoing Efforts and Summary

Introduction

Goals

- Flexible Use
  - Isolate detector information in *interface* layer
  - Standard interface for multiple algorithms
    - Fast implementation of new algorithms
    - Allow for specialty algorithms, *eg.*:
      - Custom curler pattern recognition
      - Use 'non-geometrical' information (*eg.*, dE/dx)
      - Pattern recognition after partial event reconstruction (*eg.*, use vertex information, search cones)
- Allow for sequential pattern recognition — output of one PR pass can be input to another
- Uniform output from all algorithms
- Ability to merge output with the output from other trackers



Code Overview

- Object-oriented C++ infrastructure
- Singleton (*Design Patterns*, p.127) interface for use with procedural languages
- Bookkeeping handled automatically for the *user*
- Interface code minimized for the *user*

- `Scungili`
  - Master class
  - Takes hit information from the *user* and stores in `TrackHit` objects
  - Manages `LayerSet` PR objects
  - Manages `TrackFilter` fitting/filtering objects
  - Returns `TrackCand` objects to *user*
- `TrackHit`
  - Provides a standard interface to data
- `TrackCand`
  - Holds `LayerSet` and `TrackFilter` output
  - Provides a standard internal representation of hits and information describing a track
- LayerSet
  - Class to carry out PR operations
- TrackFilter
  - Class to manipulate `TrackCand` objects
  - Provides fitting and filtering operations

Scungili

- Coordinates simultaneous operation of multiple track finders and track filters.
- *All* book-keeping for found and fitted candidates handled automatically
- Standard set of methods $\leftrightarrow$ standard HEP analysis operations:
  - beginJob()
  - beginRun()
  - event()
  - endRun()
  - endJob()
  - status()
  - reset()

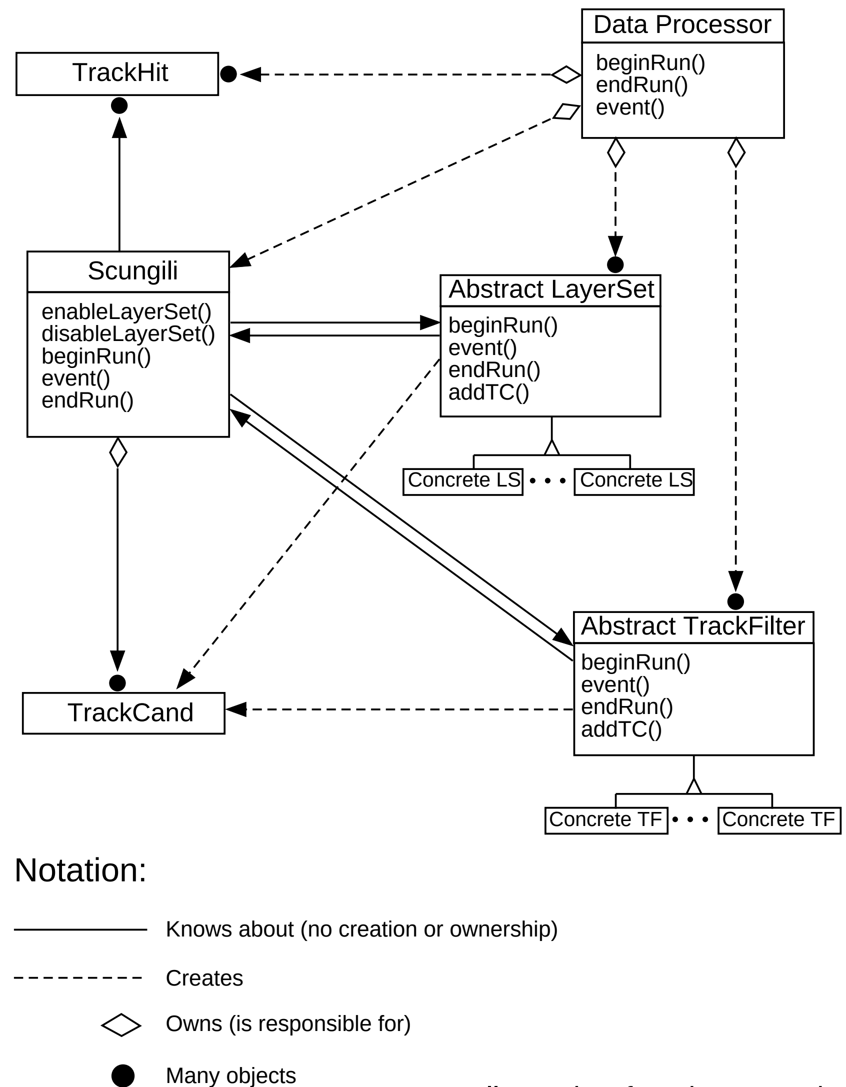
- *USER* tasks:
  - Implement specific track finders and track filters
  - Activate track finders and track filters of interest
  - Provide raw data to `Scungili`
  - Ensure that `Scungili` methods are executed
  - Receive list of track candidates

Using Scungili

Scungili Event Flow:

1. *USER* prepares vector of `TrackHit` objects to pass to `Scungili`
2. *USER* calls `Scungili::event()`
3. `Scungili` calls each `LayerSet::event()` method
   - Provides each `LayerSet` with `TrackHit` objects
   - Stores any found `TrackCand` objects
4. `Scungili` calls each `TrackFilter::event()` method
   - Provides access to found `TrackCand` objects
   - Stores `TrackCand` objects returned by `TrackFilters`
5. Fitted track candidates available to *USER*

Scungili Class Relationships



Not all member functions are shown

LayerSet AND TrackFilter CLASSES

LAYERSET

- Pattern Recognition Class
- Primary Components:
  - List of Tracking Layers
  - Algorithm
- Inputs `TrackHit` objects
- Outputs`TrackCand` objects
- Implemented `LayerSet`:
  - MARK III Dictionary-based Algorithm
- `LayerSets` in Progress
  - Curler Identification
  - Silicon-only stubs with event vertex information

TRACKFILTER

- Track Candidate Processing Class
- Primary Component:
  - Algorithm
- Inputs `TrackCand` objects
- Outputs updated `TrackCand` objects
- Implemented `TrackFilters`:
  - Preliminary Fitter (line, circle, *helix*)
  - Segment Matching (track parameter χ^2 comparisons)
- `TrackFilters` In Progress
  - Hit Addition Algorithm
  - Track List Merging Algorithm (with outside source)
  - Final Fitter (kalman)

BinGo

- Mark III Dictionary-based Algorithm
  - J. Becker, *etal.*, *Nucl. Instrum. Methods*, **A235**,502(1985)
- Integer pattern recognition

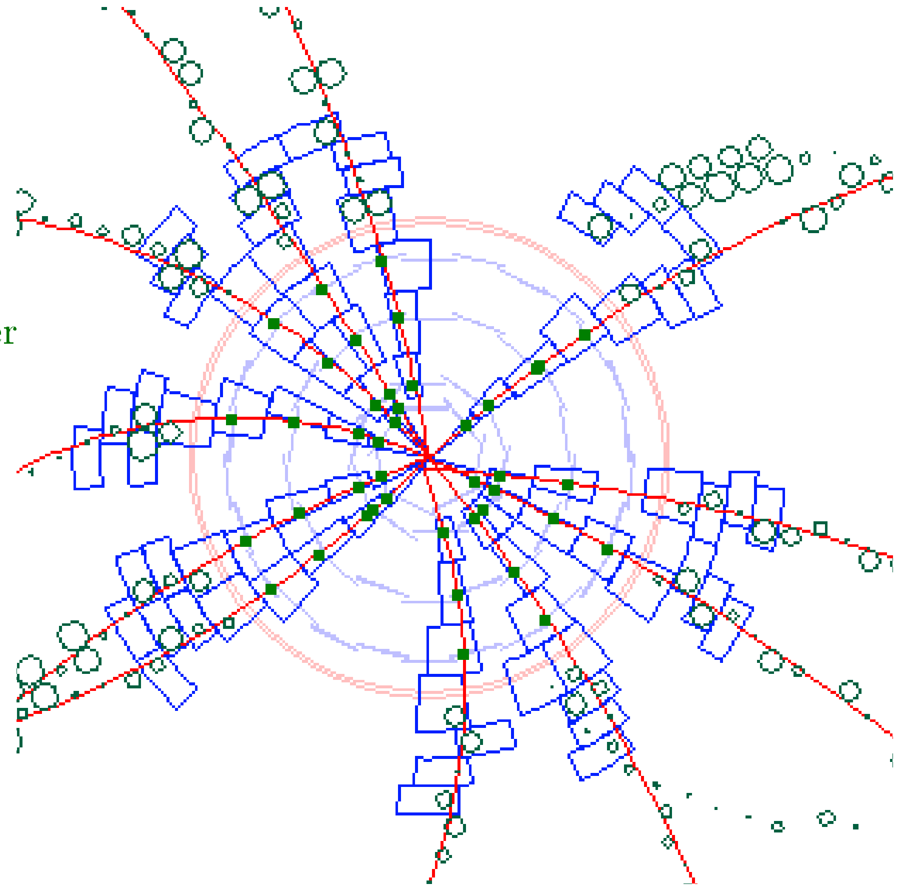
  ⇒ memory intensive but *fast*
- Operates off of bins in each tracking layer

  ⇒ Easy to apply to arbitrary detector geometries
- Up to 8 tracking layers per `LayerSet`
- *USER* must supply:
  - Binning function for each tracking layer
  - Dictionary of allowed tracks
- Dictionary Generation
  - Methods supplied to generate dictionary from single-track detector MC

DICT_ID	1								
NTRKLYR	8								
LYRLIST		1	2	3	4	9	10	11	12
LYRBINS		36	31	37	27	33	33	37	37
NLEGPAT	37								
NTRACKS	3751								
NSUBTRK	57760								
LEG_PAT		1	1	1	1	1	1	1	1
LEG_PAT		0	1	1	1	1	1	1	1
LEG_PAT		1	0	1	1	1	1	1	1
				...					
LEG_PAT		0	0	1	1	1	1	1	1
TRKBINS	0	1	1	2	2	2	1	2	2
TRKBINS	1	1	2	1	1	31	30	34	33
TRKBINS	2	1	2	1	26	31	30	34	33
TRKBINS	3	1	2	2	1	1	32	1	36
				...					

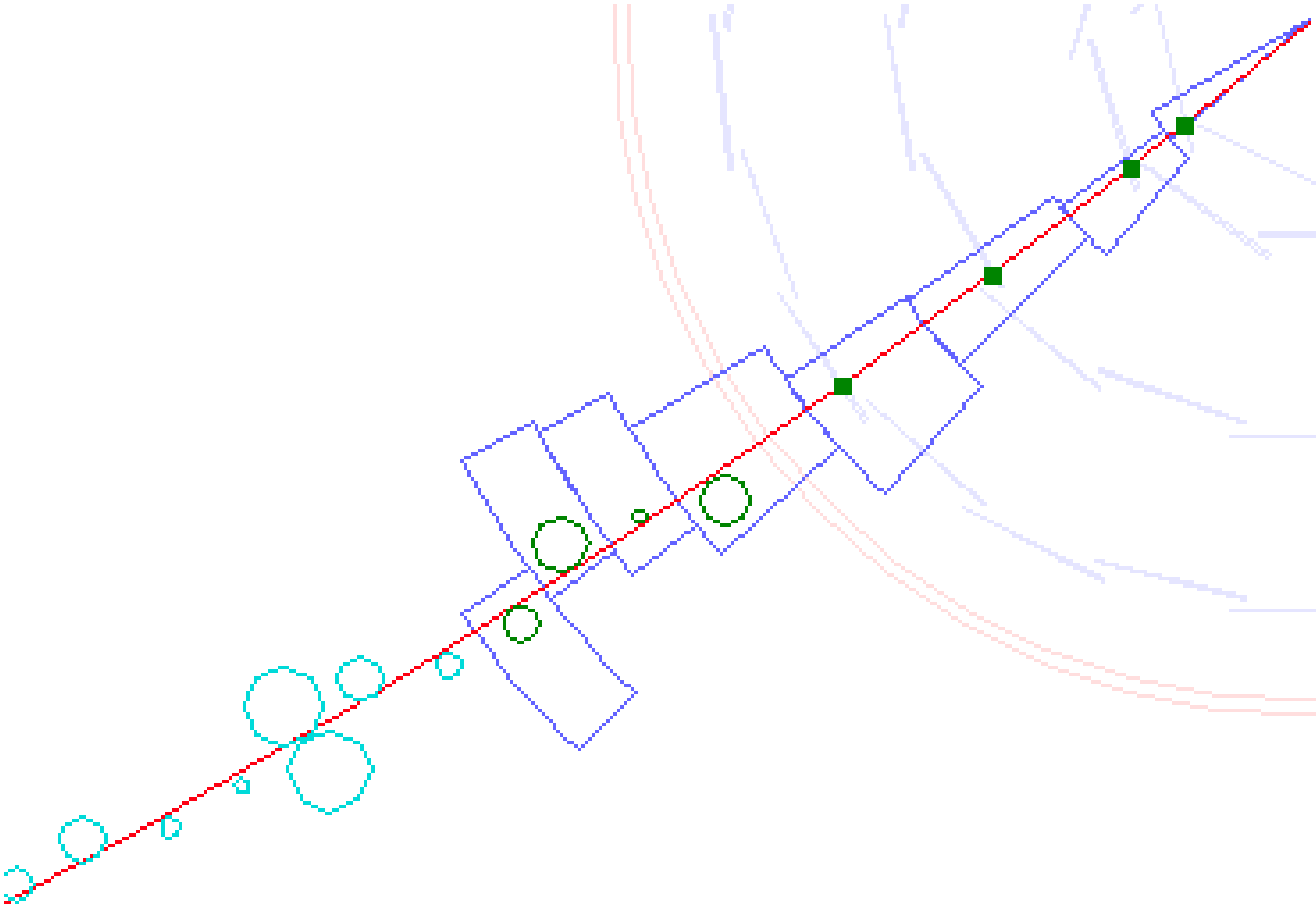
BinGo

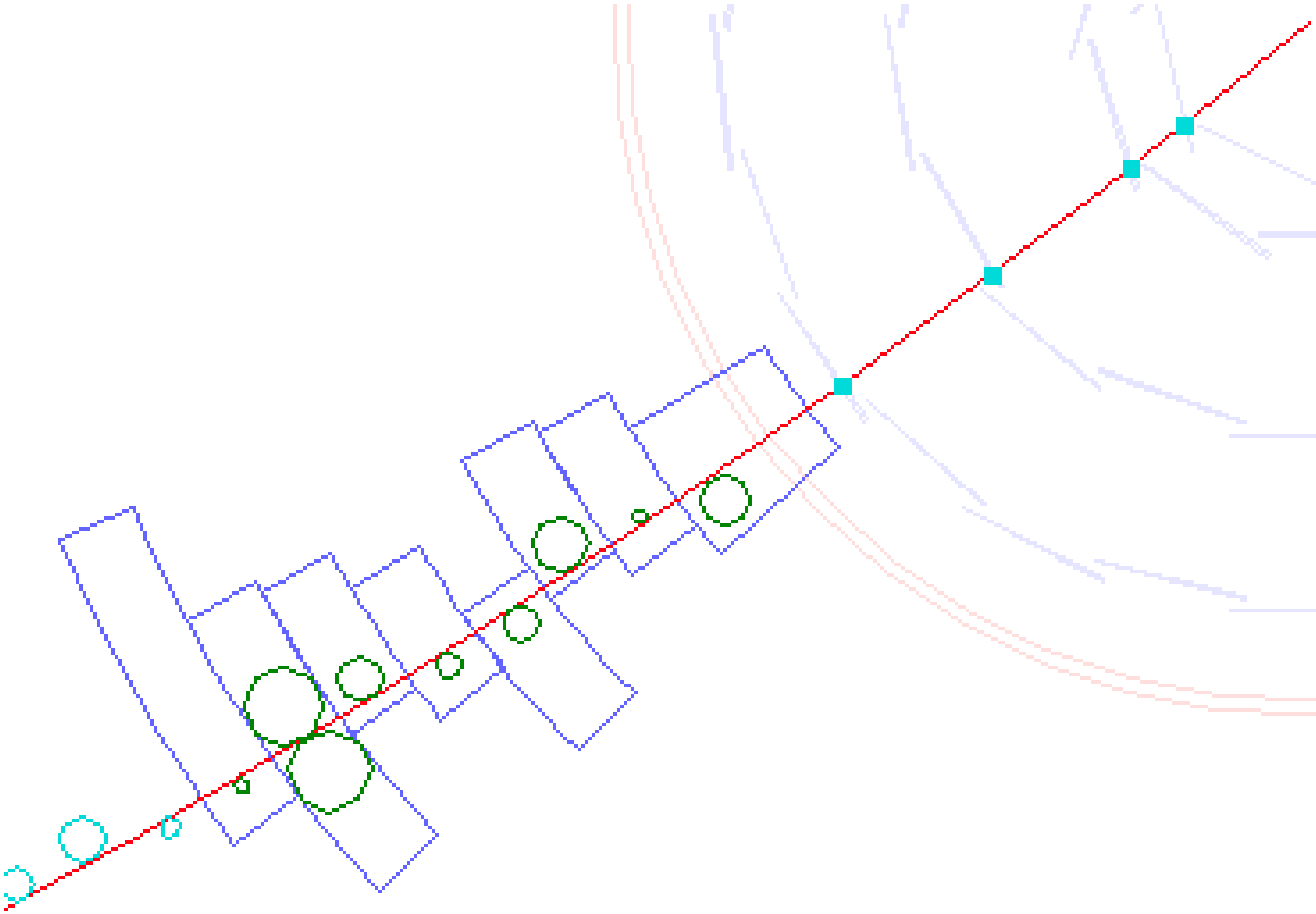
- CLEO III Implementation
  - $r\phi$ pattern recognition
    - 3 `LayerSets`
    - 4 silicon layers and 1st 16 drift chamber layers
  - rz pattern recognition
    - 1 `LayerSet`
    - 4 silicon layers and DR cathode layer



Preliminary Track Fitting

- Operates on Found Track Candidates from individual `LayerSets`
- Considers all hits specified in each tracking layer
- Fast circle fit
  - Applied to layers with $r\phi$ strips, axial wires, U or V stereo wires
  - Hit ambiguities resolved
- Line fit in rz
  - Applied to layers with rz strips and cathode pads
- All combinations with acceptable χ^2 added to output list of Fitted Track Candidates





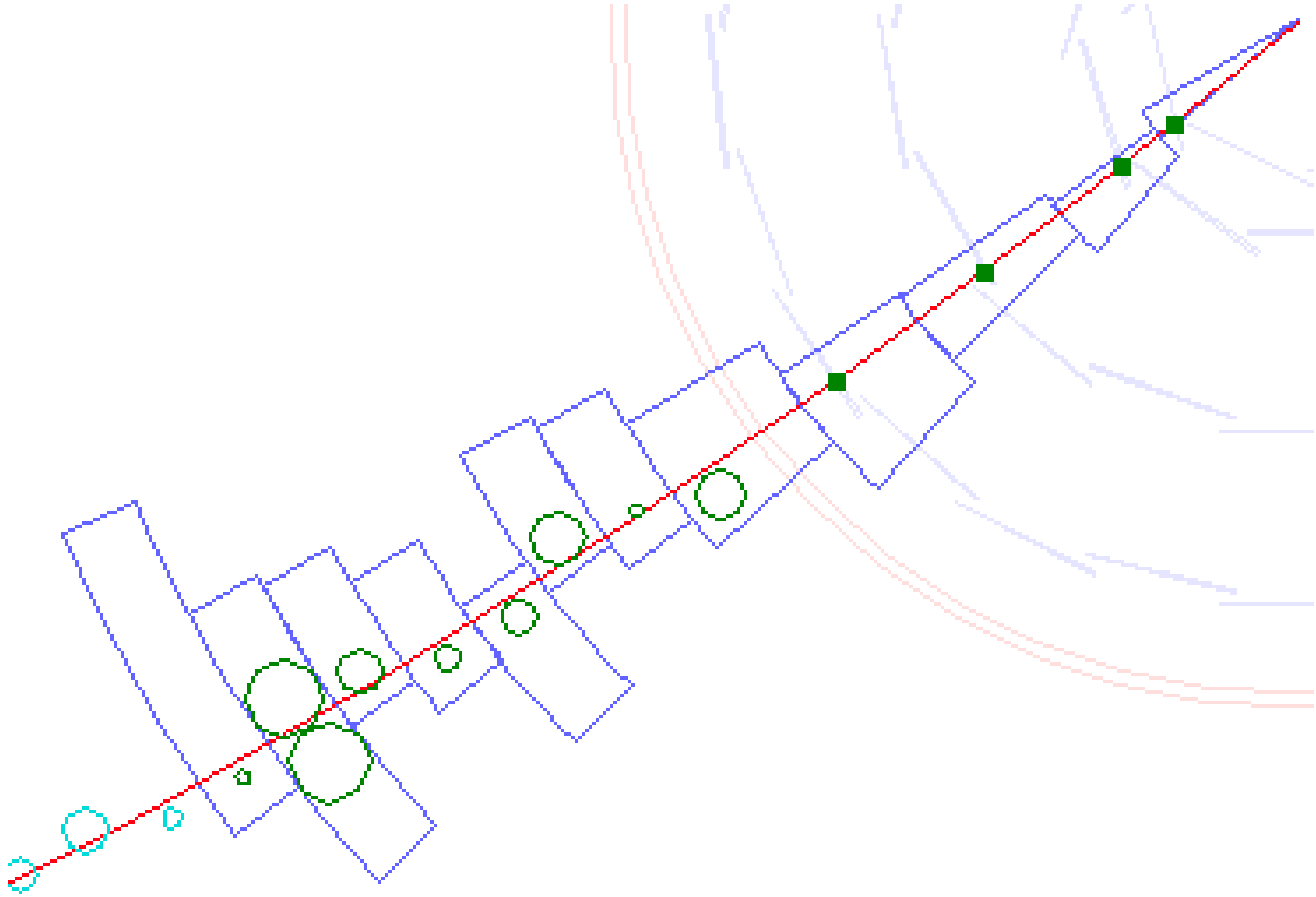
χ^2 Matching

- Match Fitted Track Candidates from distinct `LayerSets`
- Apply χ^2 matching criteria:

  $\chi^2 = \Delta\eta(\mathbf{V_i} + \mathbf{V_j})^{-1}\Delta\eta$

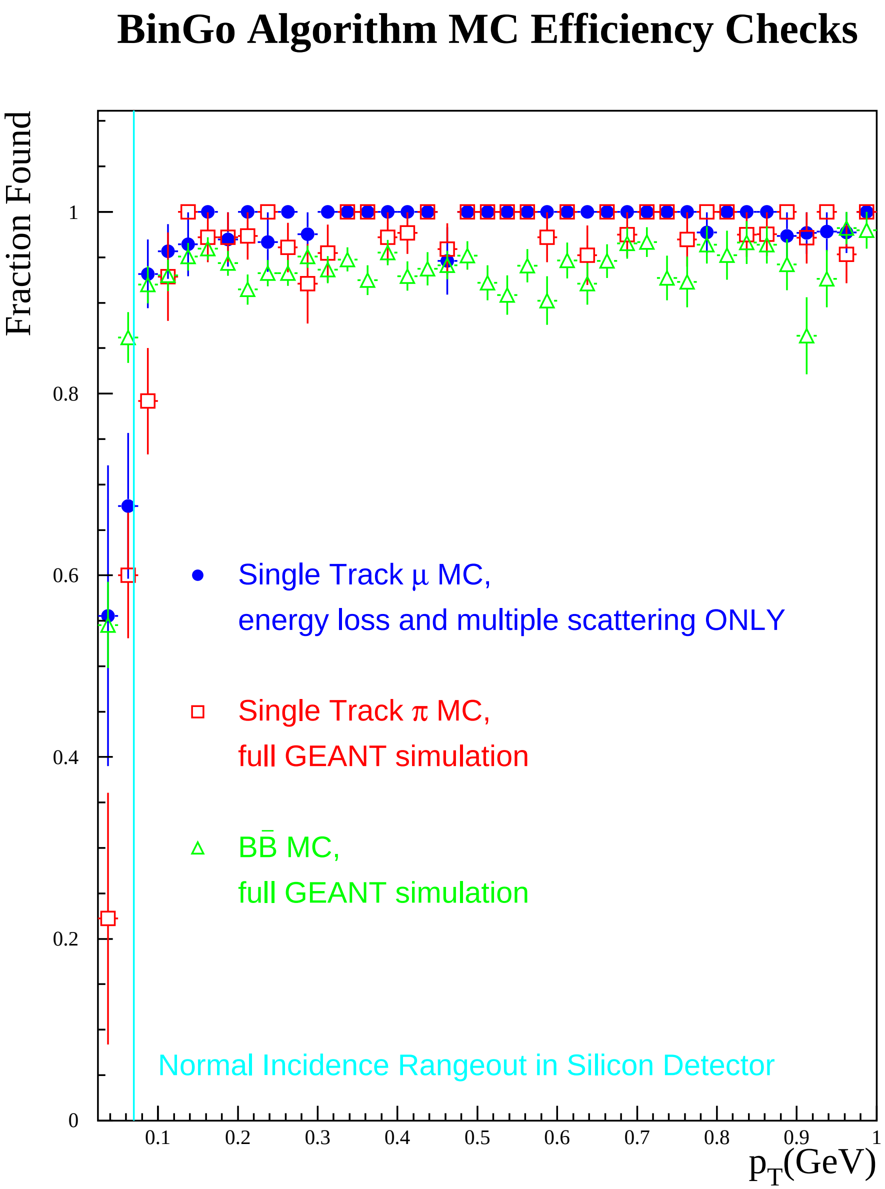
  where $\mathbf{V_i}$ is the track error matrix of candidate i

  $\Delta\eta = \eta_i - \eta_j$ is the difference of the track parameter vectors of candidates i and j
- Merge acceptable pairs into more complete track segment
- Refit new candidate and pass to `Scungili`



Preliminary Efficiency Studies

- Algorithm Efficiency Checks
  - Check completeness of generated dictionaries
  - Verify performance of algorithm
- Minimum PR requirement: hits found in 6 of 8 innermost layers (4 SV and 4 DR)
- Algorithm performance consistent with rangeout expectations
- Use of only innermost layers gives sensitivity to event environment
- Stable performance below 1 GeV
- Use of additional algorithms will help overall efficiency



Ongoing Efforts

- Complete verification of the infrastructure
- Optimization of algorithms (*e.g.*, insure completeness of `BinGo` dictionaries)
- Development of specialty pattern recognition passes
  - Curler Identification (including multiple passes)
  - Silicon-only stubs at very low momentum
  - Incorporation of silicon dE/dx and event vertex information
- Utility `TrackFilters`
  - Merging algorithm for internal and external sources of track candidates
  - Higher level fitting (*e.g.*, Kalman filter)
  - Allow for local hit addition as well as merging of complete track candidates
- Verify portability to non-CLEO software environment

SUMMARY

- We have implemented an object-oriented pattern recognition package offering:
  - Easy adaptation to new detectors and coding environments
  - Automatic bookkeeping for pattern recognition results
- Infrastructure allows for:
  - Simultaneous use of multiple pattern recognition algorithms and provides for generating a unified set of output
  - Easy implementation of new algorithms
    - Specialty algorithms — possibly physics mode specific
    - Algorithms suited for particular detectors
- Updates and further information can be found at:

  `http://www.lns.cornell.edu/~palmer/PatternRecognitionTools.html`