Proposed Pattern Finding Framework

Jack Carlton
University of Kentucky

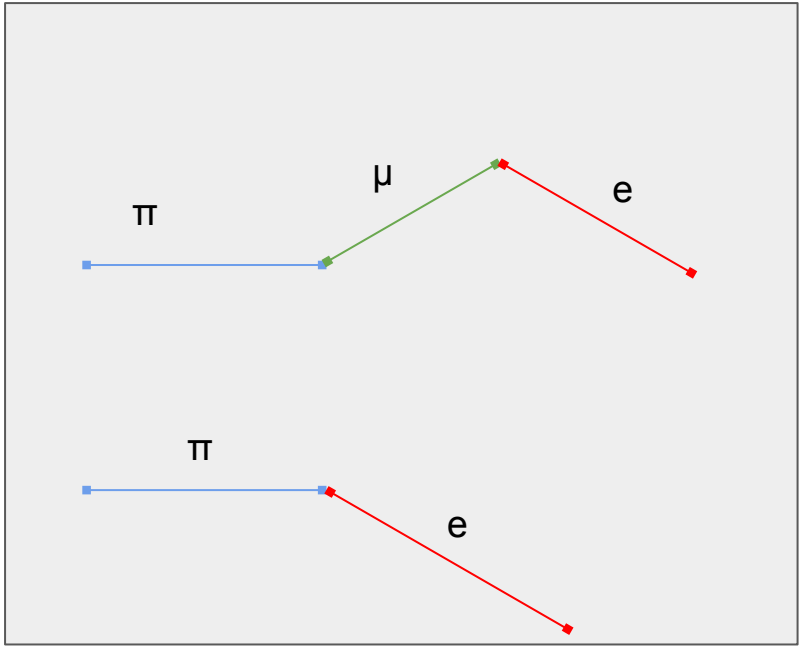
Information Framework

EventPatterns

Member Variables:
- Set of Pattern classes

Methods:
- Anything for event level pattern info, examples:
    - getPatterns()

Notes:
This is effectively just a wrapper around a set of patterns but provides structure to add more
(ex. Write to fRecEvent method)

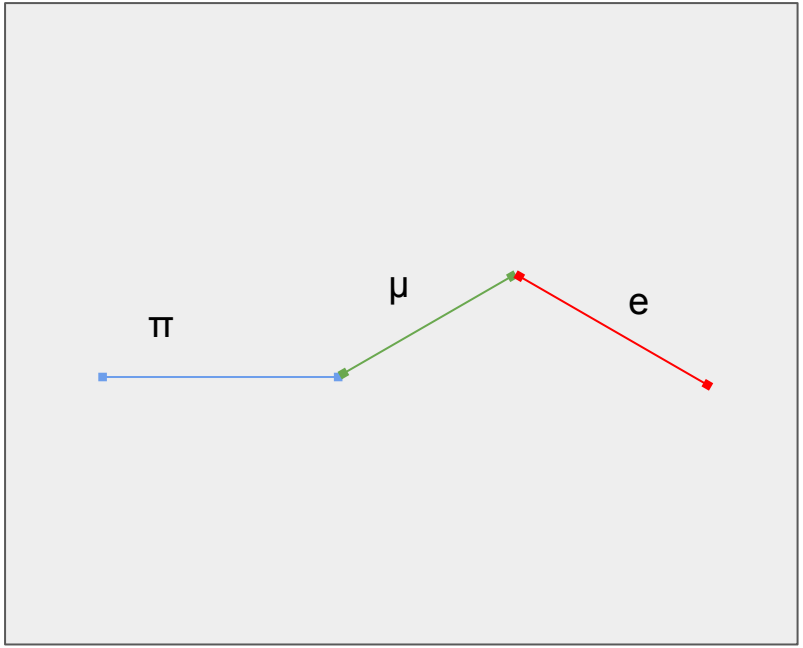


Pattern

Member Variables:
- Set of Vertex classes

Methods:
- Anything for pattern level info, example:
    - getVertices()

Notes:
This is effectively just a wrapper around a set of vertices but provides structure to add more



Vertex

Member Variables:

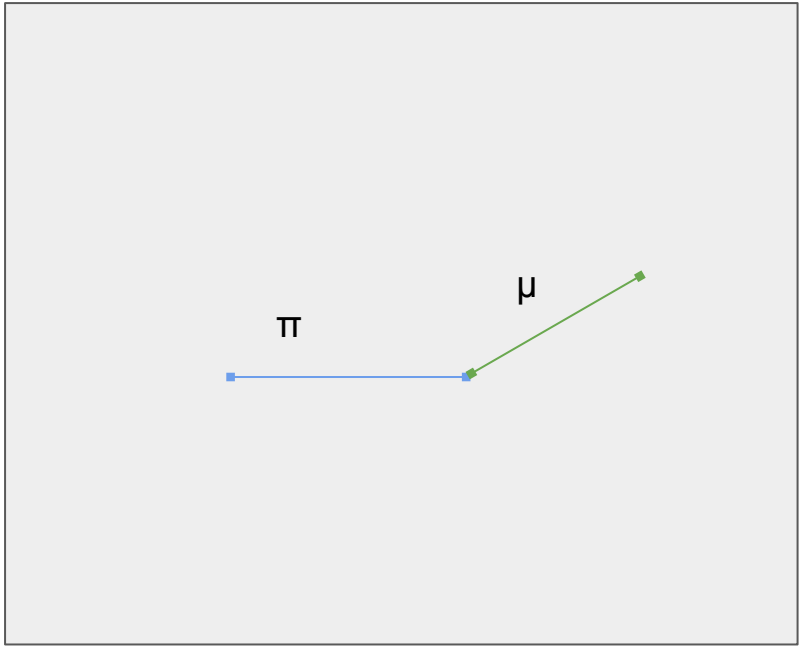
- Set of Tracklet† classes
- PatternCreator tags (json map or similar)

Methods:

- Anything for vertex level info, examples:
    - getTracklets()

Notes:

This is effectively just a wrapper around a set of tracklets but provides structure to add more



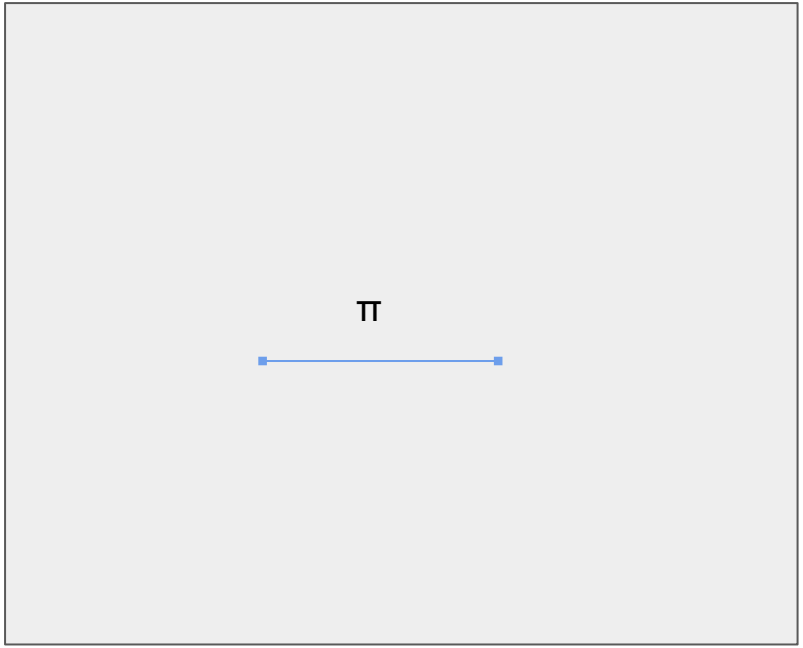
† This will likely be a wrapper around what comes out of the tracklet finding algorithm, not the actual tracklet from tracklet finding

Tracklet†

Member Variables:
- Hits (or endpoints only?)
- Fitter function (or abstract class?)
- Fit results (json map or similar)
- In/out tracklet boolean?
- VertexCreator tags (json map or similar)

Methods:
- Anything for tracklet level pattern info, examples:
    - getEndpoints()
    - fit()

Notes:
We may need a “PatternFinderTracklet” object or similar to wrap around what Jessie gives us.



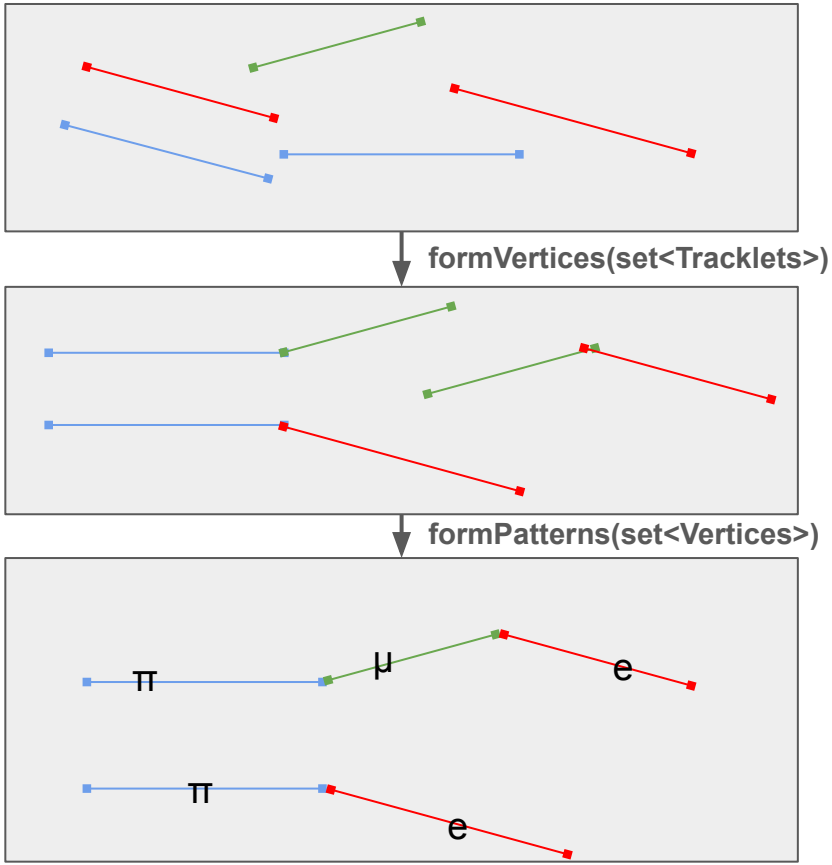
Algorithm Framework

PatternFindingHelpers (Helper class)

Member Variables:
- VertexCreator abstract class
- PatternCreator abstract class

Methods:
- formVertices(set<Tracklets>)
- formPatterns(set<Vertices>)

Notes:
More or less just a container for logic to create patterns. Has member variables to “hot swap” algorithms.



VertexCreator

Member Variables:

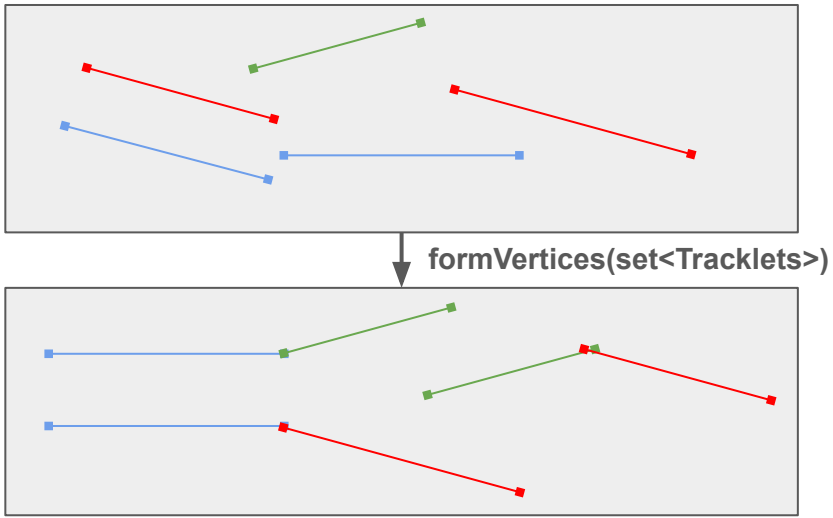
- …

Methods:

- Virtual formVertices(set<Tracklets>)

Notes:

Can implement any algorithm with any parameters needed by creating a class derived from VertexCreator.



PatternCreator

Member Variables:

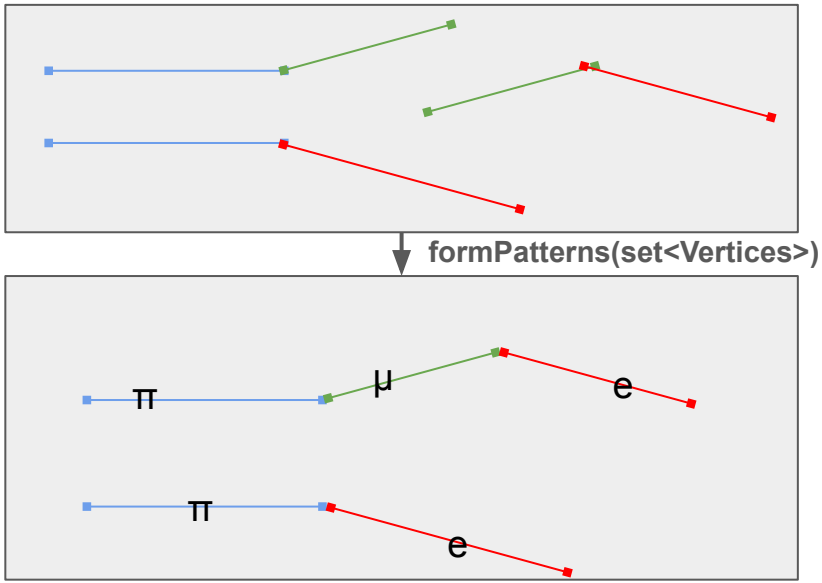
- …

Methods:

- Virtual formPatterns(set<Vertex>)

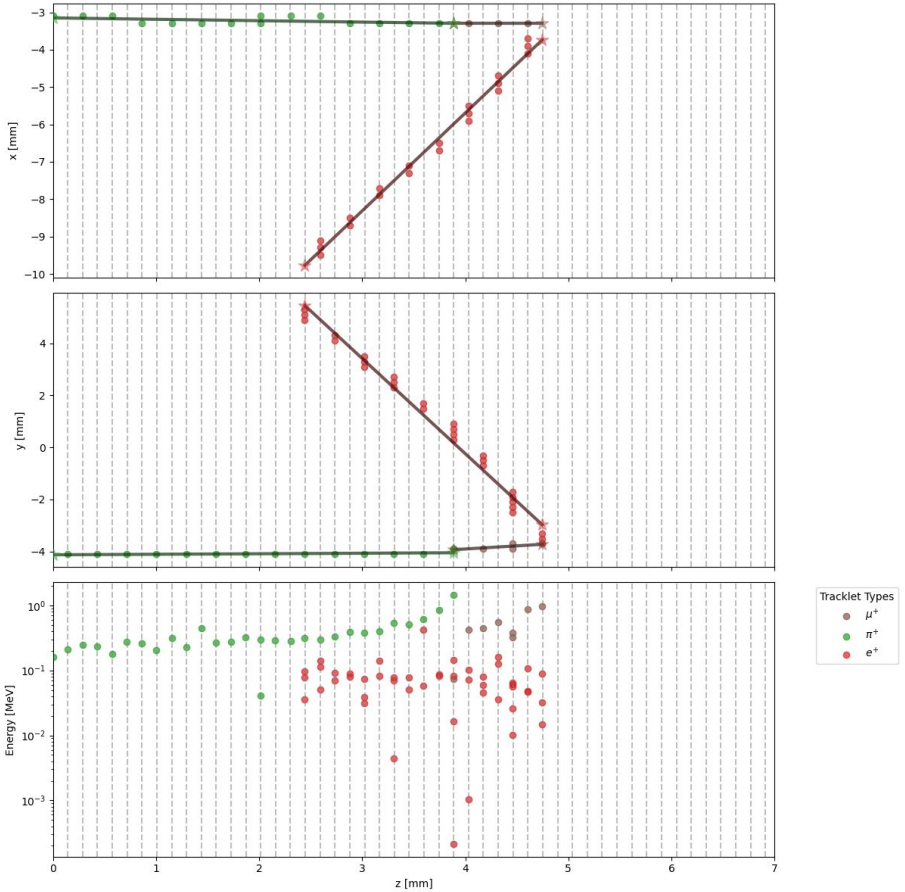
Notes:

Can implement any algorithm with any parameters needed by creating a class derived from PatternCreator.



Pattern Finding Playground

- Started creating python framework for testing algorithms
  - Github repo
  - Using python allows for quicker testing, switching, and validation of algorithms we design (from a development perspective)
- Next steps:
  - Test and develop algorithm in python
  - Port into C++ simulation framework



Auxiliary Slides

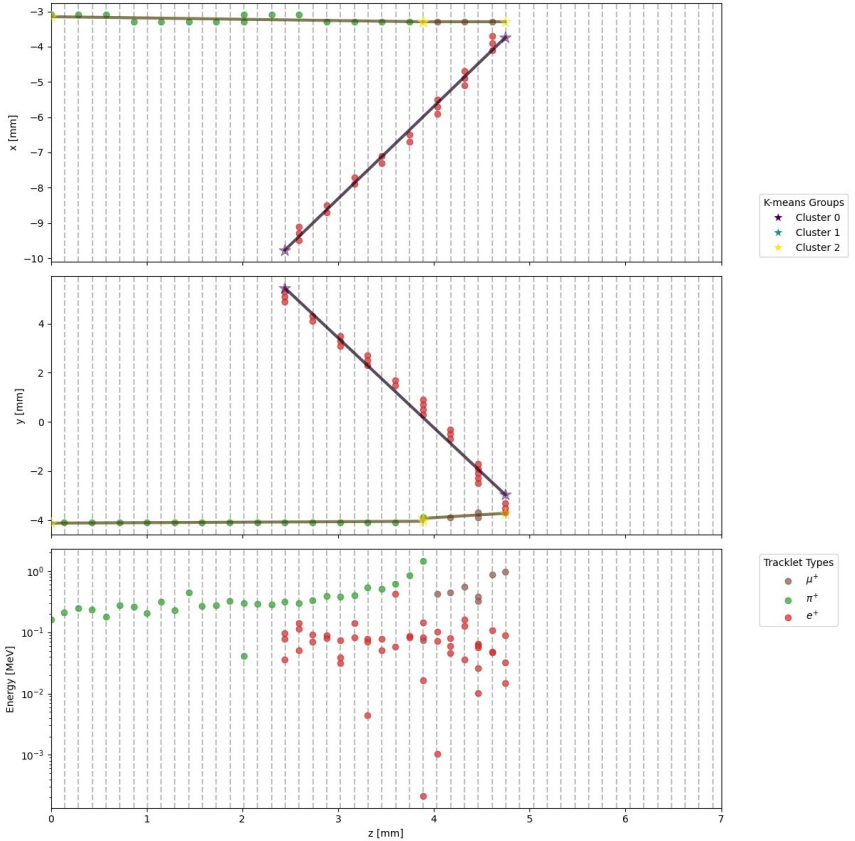
formVerticies Method (vague) Ideas

- Could treat it as clustering of endpoints. Let endpoints have form:
  (x,y,z,E,t, … whatever else …)
  - Kmeans? See how endpoints get grouped?
- Similarly, could define some “measure”

$$D(\text{Tracklet}_1, \text{Tracklet}_2) = \text{”score of closeness”}$$

Simple Example (spatial distance):

$$D(\text{Tracklet}_1, \text{Tracklet}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 - (z_1 - z_2)^2}$$



Kmeans groupings k=3 for endpoints (x,y,z)