PathFinder - A Package for Global Track Finding using Hough Transformation

Isa Heinze

DESY

LCTPC Collaboration Meeting

March 26, 2012

Content

Track Finding with Hough Transformation

Finding Tracks:

- One needs: Points in three dimensional space.
- There are different possibilities to do track finding.
- Local methods: Start with a track candidate with few hits. Step by step more hits are added.
- Global Methods: All hits enter the search algorithm at the same time and in the same way.
- In praxis: Combination of local and global methods.
  - First local method: fast method for tracks that are easy to find.
  - Then global method: slow method for tracks that are harder to find.

Hough Transformation:

- Global method.
- Works for any track model that can be described with one set of parameters.
- Here: either straight lines (no magnetic field or high p_T) or helix.
- Search is done in two projections.
  - xy-projection (readout plane): straight line or circle.
  - sz-projection (s is the arc length in the xy-projection): straight line.

Hough Transformation for Straight Lines

Parametrization of straight lines:

- r and θ

Search for straight lines:

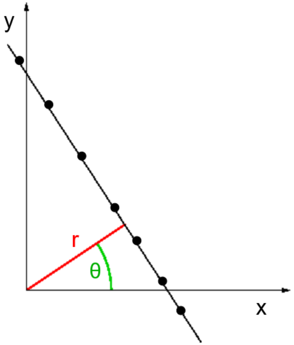
- Equation for Straight Lines:

$$y(x) \quad = \quad -\frac{\cos\theta}{\sin\theta}\cdot x + \frac{r}{\sin\theta}$$

- For every hit straight lines with different parameters are constructed on which the hit can lie. This can be expressed by the following function:

$$\Rightarrow r(\theta) \quad = \quad \cos\theta\cdot x + \sin\theta\cdot y$$

- The hit positions are inserted in the function $r(\theta)$.
- If the hits are on a straight line, the functions intersect in one point in Hough space (parameter space).
- The point of intersection delivers the parameters of the straight line.



Hough Transformation for Straight Lines

Parametrization of straight lines:

- r and θ

Search for straight lines:

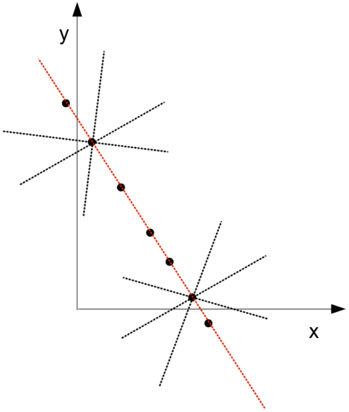
- Equation for Straight Lines:

$$y(x) = -\frac{\cos\theta}{\sin\theta}\cdot x + \frac{r}{\sin\theta}$$

- For every hit straight lines with different parameters are constructed on which the hit can lie. This can be expressed by the following function:

$$\Rightarrow r(\theta) = \cos\theta\cdot x + \sin\theta\cdot y$$

- The hit positions are inserted in the function $r(\theta)$.
- If the hits are on a straight line, the functions intersect in one point in Hough space (parameter space).
- The point of intersection delivers the parameters of the straight line.



Hough Transformation for Straight Lines

Parametrization of straight lines:

- r and θ

Search for straight lines:

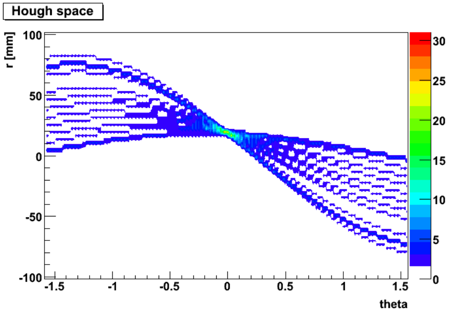
- Equation for Straight Lines:

$$y(x) = -\frac{\cos\theta}{\sin\theta} \cdot x + \frac{r}{\sin\theta}$$

- For every hit straight lines with different parameters are constructed on which the hit can lie. This can be expressed by the following function:

$$\Rightarrow r(\theta) = \cos\theta \cdot x + \sin\theta \cdot y$$

- The hit positions are inserted in the function $r(\theta)$.
- If the hits are on a straight line, the functions intersect in one point in Hough space (parameter space).
- The point of intersection delivers the parameters of the straight line.



Hough Transformation for Circles

First idea:

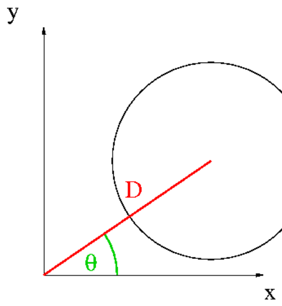
- Use Equations for Circles: $R^2 = (x - x_C)^2 + (y - y_C)^2$
- Following steps as in Hough Transformation for straight lines: Functions $R(x_C, y_C) \to$ 3D Hough Space.
- The Functions $R(x_C, y_C)$ look very similar in large regions. It is difficult to find the point of intersection.

New idea: Search for circles in two steps.

- Search for the center of the circle: 2D Hough transformation.
- Search for radius of the circle: 1D Hough transformation.
- Faster than the 3D Hough transformation.

Parametrization of center of the circle:

- D and θ

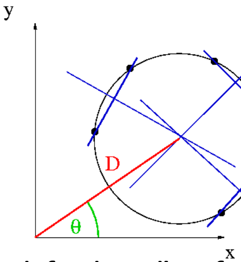


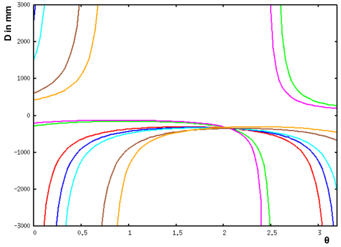
Hough Transformation for Circles

Search for center of the circle:

- Two hits in the *xy*-Plain: (x_1, y_1), (x_2, y_2).
- Construct a straight line with both hits on it. Calculate the point (x_h, y_h) on the straight line, half way between the two hits.
- Construct a second straight line with the center of the circle and (x_h, y_h) on it. This second straight line is perpendicular to the first straight line.

$$D(\theta) = \frac{1}{2} \cdot \frac{(y_1^2 - y_2^2) + (x_1^2 - x_2^2)}{(y_1 - y_2) \cdot \sin\theta + (x_1 - x_2) \cdot \cos\theta}$$





Search for the radius of the circle:

- Determining the radius by calculating the distances of the hits to the center of the circle.

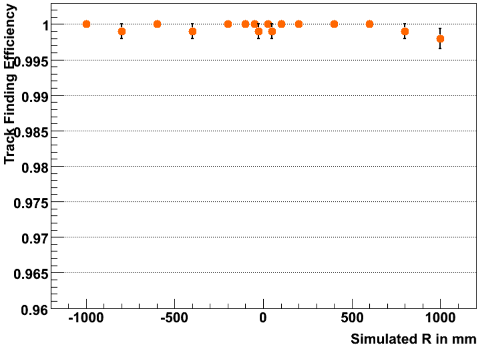
Hough Transformation for Helices: Algorithm used in PathFinder

The algorithm is split into two steps.

- **Search in xy projection:**
  - Tracks are either straight lines or circles in this projection.
    - Straight lines: Do a Hough transformation for straight lines
    - Circles: Find the center of the circle, after that find the radius of the circle.
- Determine which hits are on the found track in the xy projection.
- Do a simple fit to improve the result.
- Only hits being on the track in the xy projection enter the search in the sz projection.
- **Search in sz projection:**
  - s is the arc length in the xy projection.
  - Tracks are always straight lines in this projection.
- Determine which hits are on the found track in the sz projection.
- Do a simple fit to improve the result.

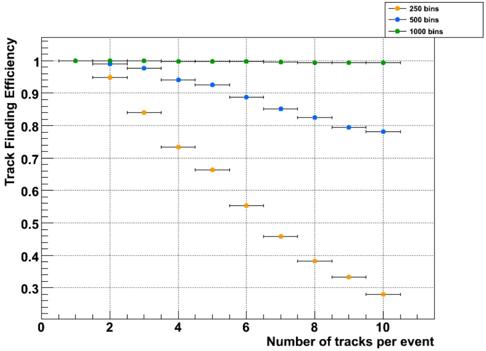
Efficiency Study: Radius of a Helix

- Simulated tracks; radius R fixed, other parameters randomly chosen; 1000 events per simulated R, one track each, up to 200 hits over track, hits exactly on track, no noise hits.
- Efficiency: all hits of the simulated track have to be on the reconstructed track.
- Tracks that are not found were not entirely on the pad plane but cut off (rather few hits on track).



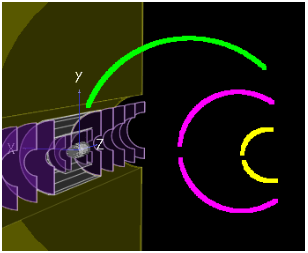
Efficiency Study: Number of Tracks per Event

- Simulated straight tracks; track parameters randomly chosen; 1000 event er number of simulated tracks, 200 hits on track, hits exactly on track, no noise hits.
- Efficiency: all hits of the simulated track have to be on the reconstructed track.
- Number of bins of Hough space have a strong influence on how well tracks can be found.
  large number of bins = high efficiency = long computing time.

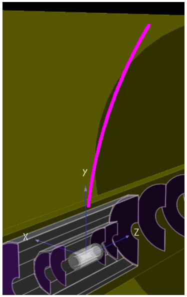


Efficiency Study: Single Muons in ILD TPC

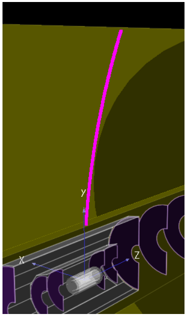
- Simulated single muons in the ILD with different p_T using Mokka.
- Cut out all events with more than one MC particle.
- Cut out all events with more than 1000 hits in the TPC (curler), else track finding takes too long.
- Efficiency: all hits of the simulated track have to be on the reconstructed track.



1 GeV Muon.

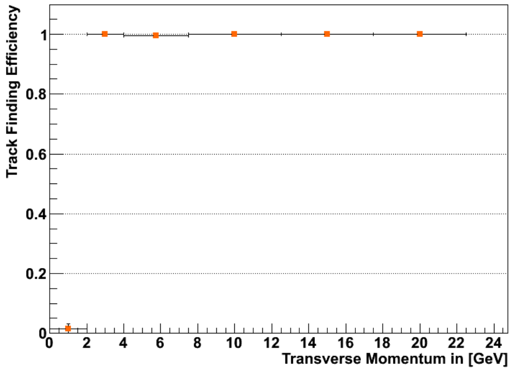


3 GeV Muon.



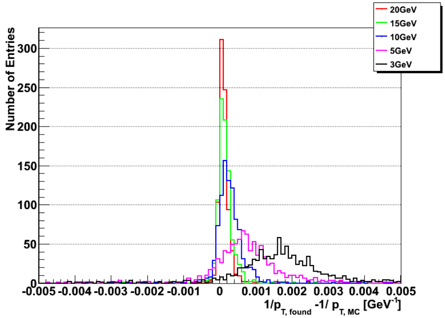
5 GeV Muon.

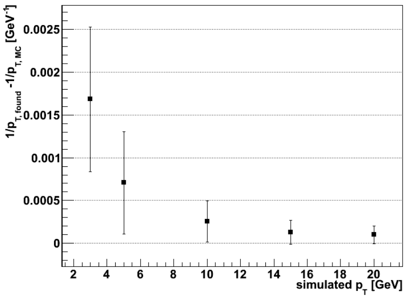
Efficiency Study: Single Muons in ILD TPC



1 GeV Muons are curling back into the TPC. Tracks can only be found in segments.

Efficiency Study: Single Muons in ILD TPC

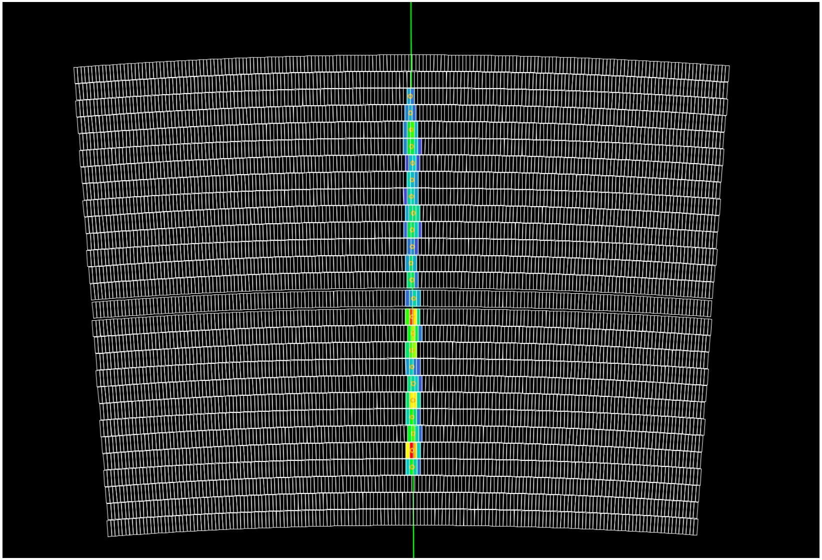




- Overall the transverse momentum is found correctly.
- Differences of transverse momenta probably due to energy loss in the vertex detector and in the SIT.

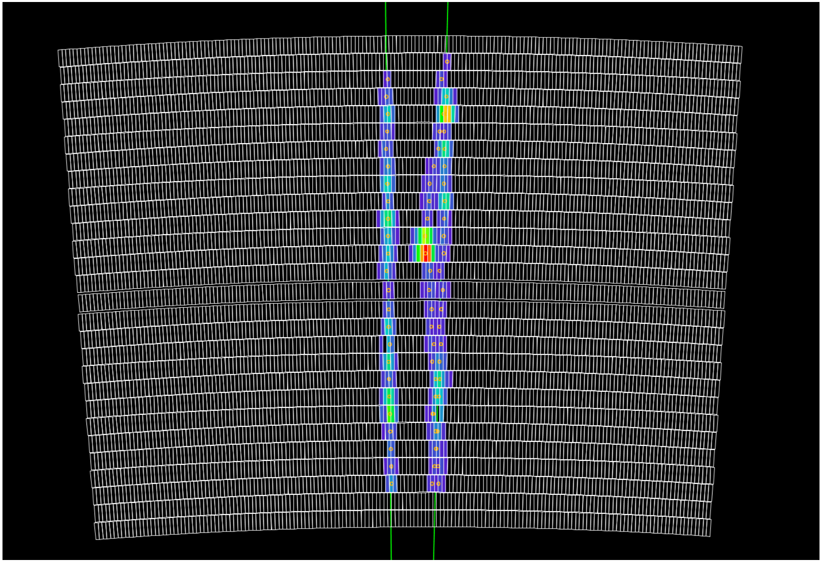
Hough Transformation used on Test Beam Data

Data were taken with the Large Prototype, one readout module (DESY module) was installed.



Hough Transformation used on Test Beam Data

Data were taken with the Large Prototype, one readout module (DESY module) was installed.



PathFinder Package

- PathFinder is a package which provides an algorithm for track finding using Hough transformation.
- It can find
  - Straight lines
  - Helices
- A simple track generator is included to test the algorithm.
- Limitations:
  - Tracks parallel to the z-axis cannot be found (track parameter $\tan\lambda \rightarrow \infty$).
- It can be used stand alone, examples are included in PathFinder.
- It can be used in software frameworks such as MarlinTPC.
- There is a MarlinTPC processor using PathFinder (PathFinderInterfaceProcessor).
- More information (especially on usage) can be found here: http://www-flc.desy.de/flc/flcwiki/PATHFINDER

Summary and Outlook

Summary:

- A Hough transformation for track finding was implemented.
- The package is called PathFinder.
- Two track models:
  - Straight Line
  - Helix
- It can find:
  - More than one track per event.
  - Tracks in events where noise hits are present.
  - Tracks were hits are not exactly on the track.
- The algorithm was successfully used on Large Prototype test beam data.

Outlook:

- It is not possible to run on events with many hits in a reasonable amount of time. Calculating the Hough space is what takes longest.
- Currently an adaptive Hough transformation is being implemented (Initial binning rather coarse, finder binning in interesting regions). This is expected to be faster.