LC-TOOL-2014-003
http://www-flc.desy.de/lcnotes

PATHFINDER

A track finding package based on Hough transformation

Isa Heinze
DESY, Hamburg

February 24, 2014

Abstract

PATHFINDER is a package which provides a global track finding algorithm using a Hough transformation. This document provides basic information on the software package which is needed to run it successfully.

1 Introduction

PATHFINDER (PAckage for Tracking with a Hough trafo FINDER) is a software package which provides a global track finding algorithm based on Hough transformation. It was mainly written to do the track finding for data taken with the Large Prototype TPC [1]. The algorithm can either be used standalone or in a software framework such as MarlinTPC [2].
In Section 2 the track parameters used in PATHFINDER will be presented. Section 3 contains an explanation of the Hough transformation and Section 4 gives information on the algorithms used in PATHFINDER. Finally, in Section 5, some notes on the usage of PATHFINDER will be made.
For more details or example code see [3, 9] or the examples coming with PATHFINDER.

2 Track Parameters

If a charged particle travels through a homogeneous magnetic field its trajectory will have a helicoidal shape. However, for particles with very high momentum the curvature is expected to be very small. Thus the track will almost be a straight line. Particles with very low momentum on the other hand will curl (more than one turn of a helix). All these types of tracks need to be described by the same set of parameters. The parameters chosen are perigee parameters as described in detail in [4] and [5] so this topic will be dealt with only briefly here.

The coordinate system is chosen such, that the z-axis points along the direction of the magnetic field and the xy-plane is the plane perpendicular to the z-axis. A track projected into the xy-plane is then, neglecting energy losses, a circle. In this projection the track parameters are chosen to be the distance of closest approach d_0, the angle ϕ_0 between the transverse momentum at the point of closest approach and a line parallel to the x-axis crossing the point of closest approach. The third parameter describing a circle in the xy-plane is the curvature $|\Omega| = 1/R$ where R is the radius of the circle.
The other two parameters needed to describe a helix are defined in the sz-plane. s is the arc length of the track in the xy-plane. It is zero at the point of closest approach and counted along the direction of the momentum of the particle. In this projection a helix is a straight

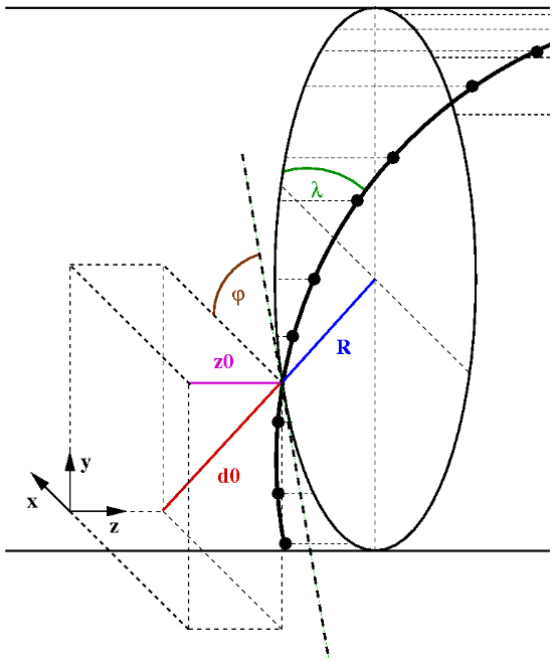

Figure 1: LCIO Track Parameters [4].

line described by the slope $\tan\lambda$ and the offset z_0. The parameters are shown in the sketch in Figure 1.
The parameters described above also work for straight lines since they are helices with $\Omega = 0$.

3 Hough Transformation

The Hough transformation [6] is a global pattern recognition method. This means that all hits (points in $3D$-space) enter into the algorithm at the same time and in the same way[1]. In this method for each hit in the event all possible tracks are calculated on which the hit can be. If several hits are on the same track, for each of those hits the same track can be found. An example for a straight line in a $2D$-plane

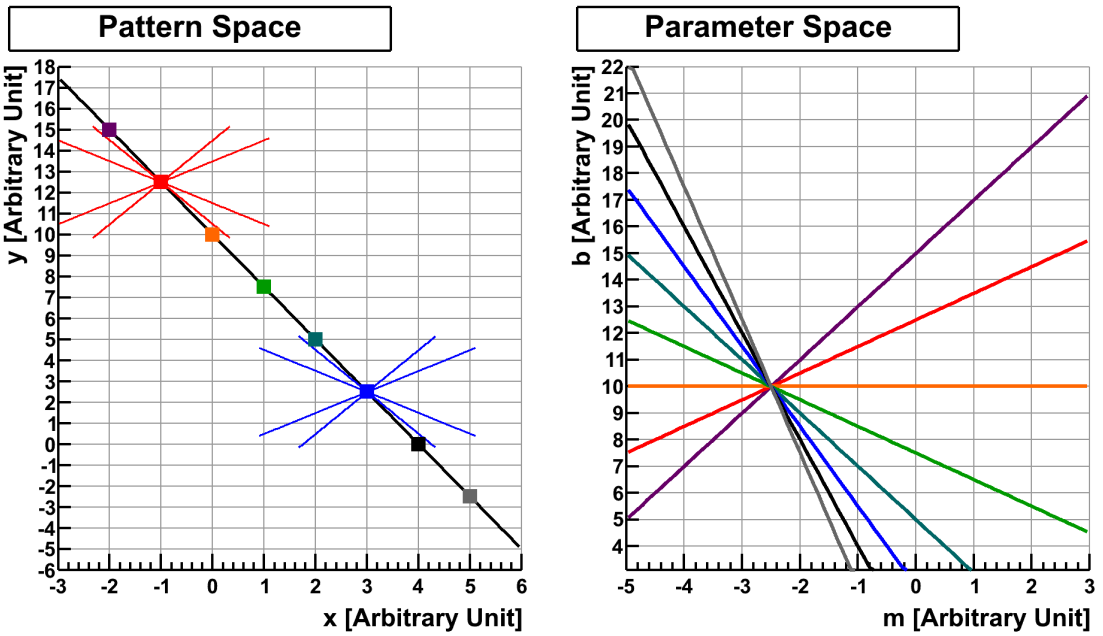

Figure 2: The idea behind the Hough transformation (here for straight lines in $2D$ space). Through each hit an infinit number of straight lines can be drawn (only a few are shown here and only for two hits). One straight line can be found which crosses all hits (left hand side). The functions calculated by the Hough transformation intersect in one point in parameter space. This point gives the parameter of the straight line the hits are on (right hand side).

is shown in Figure 2. The hits on a straight line in the pattern space (shown on the left hand side in Figure 2) are shown as dots with different colors. Through each of the hits an infinite number of straight lines can be constructed. A few of such straight lines are shown for two of the hits. If the straight lines the hits are on is described by

$$y(x) = mx + b \tag{1}$$

[1] In local methods the pattern recognition is started with a seed track consisting of a few hits, then more hits are added step by step.

with slope m and offset b, all straight lines a hit can be on is given by the function

$$b(m) = y - mx. \tag{2}$$

These functions are straight lines in the parameter space shown on the right hand side in Figure 2. If the hits are on a straight line, the functions in parameter space intersect in one point and the intersection gives the parameters of the straight line the hits are on.
As was explained in the previous section, the track parameters used here are defined in the xy-plane and the sz-plane. The parameters in the xy-plane are needed to calculate the arc length s. To do the search in sz the track in xy must be known. That is why the search for tracks is split into two parts: the xy-search and the sz-search. Two different types of patterns need to be found: straight lines (high p_T tracks in the xy-plane and tracks in the sz-plane) and circles (tracks with a p_T low enough to give them a significant curvature in the xy-plane).

Straight Lines The Hough transformation for straight lines in the xy-plane is done by calculating the parameter d_0 for all possible ϕ_0. This information can be used to define the Hough transformation for the center of the circle:

$$d_0(\theta) \quad = \quad \cos\theta \cdot x + \sin\theta \cdot y \tag{3}$$

which has the hit positions in x and y as parameters and where $\theta = \phi_0 - \frac{\pi}{2}$. Such a function exists for each hit. In case the hits are on a straight line the functions intersect in the Hough space (parameter space) and the point of intersection gives the parameters of the straight line the hits are on.
The search for straight lines in the sz-plane works in a similar way, but instead of x and y, s and z are used.

Circles The Hough transformation for circles is done in a different way. Here three parameters need to be taken into account. In order to do this in an efficient way the search is split into two steps. In the first step a search for the center of the circle is carried out as was proposed in [7]. For this task the information of one hit is not sufficient. More information can be added by not using single hits but pairs of hits. Two straight lines are constructed. The first one connects the two hits. The second one is perpendicular to the first straight line and crosses it half way between the two hits. If the two hits are both on the same circle the second straight line crosses the center of the circle. This can be described by a function

$$\frac{1}{D(\theta)} \quad = \quad 2 \cdot \frac{(y_1 - y_2)\sin\theta + (x_1 - x_2)\cos\theta}{(y_1^2 - y_2^2) + (x_1^2 - x_2^2)}, \tag{4}$$

where D is the distance of the center of the circle to the origin and $\theta = \phi_0 - \frac{\pi}{2}$. The inverse of D was chosen to avoid discontinuities[2]. The function has four parameters: the hit positions in x and y for two hits. Such functions are built for

[2]From version v00-03 on, before that $D(\theta) = \frac{1}{2}\frac{(y_1^2 - y_2^2) + (x_1^2 - x_2^2)}{(y_1 - y_2)\sin\theta + (x_1 - x_2)\cos\theta}$ was used.

all hit combinations and they intersect in one point if all hits are on the same circle. A vertex constraint can be introduced easily at this point by using a pair of vertex and hit instead of using a pair of hits.
Once the center is found the radius is determined by calculating the distances of the hits to the center of the circle.

4 Implementation of the Algorithm PATHFINDER

In real data the hits will not be exactly on a straight line or helix but shifted off slightly. So the functions presented in the previous section will not intersect in exactly one point. However there will be a region where they approach each other. How one can find tracks in this situation will be explained in this section.
The algorithm is shown graphically in Figure 3. As already indicated the search is done in several steps. As a first step the search is done in the xy-plane. The Hough space must be calculated first (for more details see Section 4.1). Once this is done the point of intersection needs to be found. This gives a rough estimate of the track parameters. Using these parameters one can determine which hits were on the track. To improve the result a very simple fit is done with the hits found on the track. Then it is checked if additional hits can be added to the track. At this point the search in xy is over. To do the search in the sz-plane s needs to be calculated for each hit (see Section 4.4). In this step the parameters in the xy-plane are needed. After having done this the search in sz is performed in an analog way as in the xy-projection, but when determining which hits were on the track only those hits are used which were already assigned to the track in the xy-search. Finally tracks are built, which consist of the hits on the track and the rough estimation of the track parameters coming from the Hough transformation. To match the parameters described in Section 2 the track parameters must be converted first.
This procedure is repeated on the remaining hits which were not assigned to any track (no matter at which step they were rejected) until no more tracks can be found.
In the following each of these steps will be described in more detail.

4.1 Calculating the Hough Space and Finding the Point of Intersection

The Hough space is calculated in such a way that in a certain range values for θ are chosen. How many such values are chosen depends on the number of bins given as steering parameter. Then for each of those values for θ and for each hit or pair of hits the value of the function (3) or (4) (depending on what shape needs to be found) are calculated. The result of this is a continuous spectrum of values. Those values are then binned according to the steering parameters set (number of bins and range in that direction). It is counted how often each bin was crossed by one of the functions. [3]

[3] All hits are weighted with 1, but in principle it is also possible to weight them with the error of the hit position (not implemented yet).

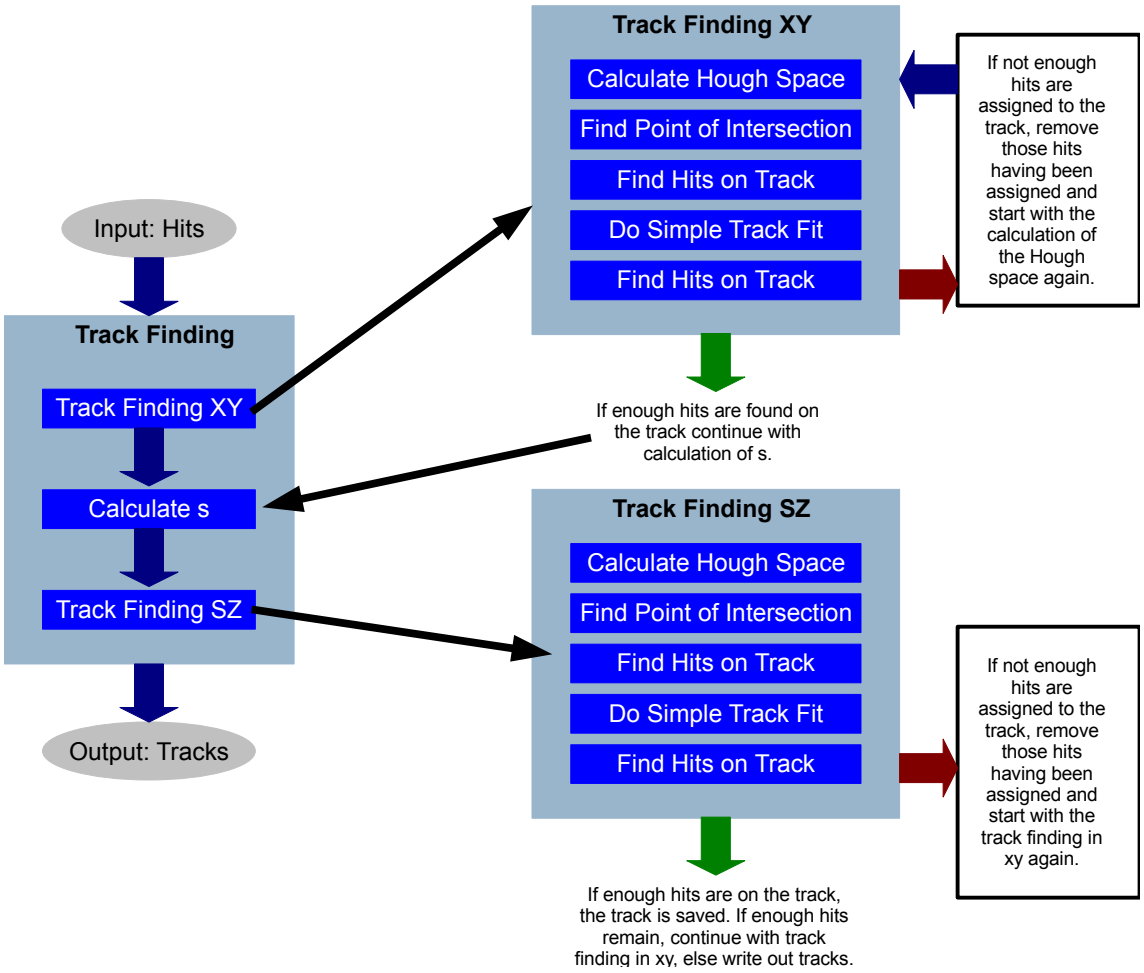

Figure 3: Track Finding Algorithm as implemented in `PATHFINDER`.

The point of intersection corresponds to the bin which was crossed most often. In other words one has to find the bin with the maximum number of entries. Since there might be ambiguities (bins with the same number of entries) for events with more than one track, an option was implemented to not only look for the bin with the maximum number of bins but also at the adjacent bins. This improved the the search in data taken with small prototypes with in the order of 10 hits per track. If still more than one intersection candidate remains the search algorithm works as follows: If there is more than one intersection candidate the first candidate found is used for the further search. After the track was found the Hough space is recalculated with the remaining hits which were not assigned to a track.

4.2 Finding Hits on the Track

To determine which hits were on the track the shortest distance between hit and track (in the xy-plane or the sz-plane) is calculated. If the distance is below a certain value (which can be set as a steering parameter, different values can be set for finding hits on the track before the fit and after the fit) the hit is assigned to the track, else it is rejected.

4.3 Track Fitting

In order to improve the track parameters obtained from the Hough transformation the found tracks are fitted with simple fitting algorithms in the xy-plane and in the sz-plane. For straight lines a linear regression is used, for circles the fit is done with the method described in [8]. The result is an improved estimate of the track parameters. The errors on the track parameters are calculated as well.

4.4 Calculate s

To calculate s the track parameters in the xy-plane are needed. It can thus only be done when the search in the xy-plane is complete. s is defined to be zero at the point of closest approach. This point can easily be calculated from the track parameters d_0 and ϕ_0:

$$\begin{aligned} x_{\text{pca}} &= -d_0 \cdot \sin(\phi_0) \\ y_{\text{pca}} &= d_0 \cdot \cos(\phi_0). \end{aligned}$$

Then for each hit the distance between the point of closest approach and the hit position in xy along the track can be calculated. For straight lines this equation reads

$$s = \pm\sqrt{(x_{\text{hit}} - x_{\text{pca}})^2 + (y_{\text{hit}} - y_{\text{pca}})^2}. \tag{5}$$

For circles the equation is

$$s = \frac{1}{|\Omega|} \cdot \left(\arctan\left(\frac{y_{\text{pca}} - y_{\text{center}}}{x_{\text{pca}} - x_{\text{center}}} \right) - \arctan\left(\frac{y_{\text{hit}} - y_{\text{center}}}{x_{\text{hit}} - x_{\text{center}}} \right) \right) \tag{6}$$

with the center of the circle

$$\begin{aligned} x_{\text{center}} &= \left(\frac{1}{\Omega} - d_0 \right) \cdot \sin(\phi_0) \\ y_{\text{center}} &= -\left(\frac{1}{\Omega} - d_0 \right) \cdot \cos(\phi_0). \end{aligned}$$

5 Usage of PATHFINDER

5.1 Input and Output

PATHFINDER needs input from the user to find tracks successfully. First of all it needs hits (points in $3D$ space) on which the pattern recognition should be done. Additionally steering parameters need to be provided. They are listed and explained in section 5.2. More details can be found in [3, 9] and in the examples coming with PATHFINDER [10].
PATHFINDER gives back a set of found tracks consisting of the track parameters and the hits on the track.

5.2 Steering Parameters

_isStraightLine, _isHelix: These steering parameters define what kind of track shape should be found. For straight lines _isStraightLine needs to be true, for helices _isHelix has to be true.

_findCurler: _findCurler has to be true if curlers (helices with more than one turn) should be found. It only has an effect if _isHelix = true. If it is set to false a curler will be found as several tracks, one track for each turn. For tracks with high energy losses the curler will not be found in one piece even if this parameter is set to true.

_minimumHitNumber: Via this parameter the minimum number of hits on the track can be specified. If there are less hits on the track, the track is rejected and the hits on that track are counted as noise hits.

_maxXYDistance, _maxSZDistance: The maximum allowed distance of the hits to the track in the xy-plane and the sz-plane before the fit can be set via these two parameters. They are needed to determine which hits are close enough to the track to be counted to be on the track.

_maxXYDistanceFit, _maxSZDistanceFit: Basically these two parameters are the same as the previous ones but are used after the fit. Since the track parameters after the fit are a better estimate than the ones before the fit the values for the maximum allowed distance after the fit can usually be set a bit lower than those used before the fit.

_numberXYDzeroBins, _numberXYThetaBins, _numberXYOmegaBins, _numberSZDzeroBins, _numberSZThetaBins: These steering parameters set the binning of the Hough spaces. θ here means the angle between the direction to the point of closest approach and the x-axis. In the xy-plane this means $\phi = \theta - \pi/4$. The parameters θ and d_0 are also used for the straight lines in the sz-plane instead of $\tan\lambda$ and z_0. The slope and the offset can in principle have infinitely large values which would lead to problems.
_numberXYOmegaBins only has an effect if _isHelix = true. The number of bins cannot be larger than 1000. How the values are chosen best depends on the data. If the bins are to wide, the track parameters do not come out correctly. In this case no hits can be assigned to the track. If the binning is too small the point of intersection might not be defined well enough to be found correctly. Also, the more bins are chosen, the longer the computation will take.

_maxDxy, _maxDsz: These are the maximum ranges of the Hough spaces and depend on the layout of the readout plane and the setup. The range is chosen symmetrically around the zero. At this point the user needs to think about how the data and the setup the data were taken with look like. For straight lines _maxDxy is the maximum distance of closest approach possible so that the track was still visible

on the readout plane. For circles it is the maximum possible distance of the center of the circle in the xy-plane so that the track could still be seen on the pad plane with a significant curvature [4] or the inverse of this [5].
_maxDsz is the maximum possible distance of closest approach in the sz-plane.

_useVertex, _VertexPosition: The default of _useVertex is false. _VertexPosition is a pair giving the vertex position in the xy plane. If it is set, _useVertex is set to true and the vertex constraint is used. If _VertexPosition is not set, the vertex constraint is not used. The vertex information is added by not using all combinations of hits (compare section 3) but combining every hit with the vertex. This speeds up the computation but only tracks coming from the origin can be found. It only has an effect if _isHelix = true.

_searchNeighborhood: An option to improve the search for the point of intersection of the functions in the Hough space. This might be helpful if there are very few hits in the event (10 or less). In case there was more than one point of intersection found [6], not only the point of intersection is used to determine if there was a track found, but also the surrounding area around the point of intersection is taken into account.

_saveRootFile: This is an option to save the Hough Spaces in a ROOT Tree [11] (for debugging only).

5.3 Further Notes on the Usage

It does not matter in which units the hit positions, the maximum allowed distances and the Hough space ranges are given, but they all need to have the same unit. Furthermore the hit positions have to be given in Cartesian coordinates.
If all hits are located rather far away from the origin the track finding might not work very well. In this case a shift of the hits towards the origin can be helpful. The track parameters found for the shifted hits do have a different point of reference than if the search would have been done without the shift. Therefore, after the track finding the point of reference of the track parameters has to be shifted. See e.g. [5].
`PATHFINDER` cannot find tracks parallel to the z-axis because such tracks look like points in the xy-plane. Also, `PATHFINDER` cannot find tracks parallel to pad rows since in this case only one single hit would have been reconstructed in that row.
No hit cleaning is done in `PATHFINDER`. This means, in case for a pad readout, it is in principle possible that two hits in the same row are assigned to the same track.
The track parameters are, due to the binning of the Hough space, only a rough estimate. A proper fit should be done after the track finding.

[4] `PATHFINDER` versions v00-01-00, v00-01-01 and v00-02
[5] From `PATHFINDER` version v00-03 on
[6] Due to binning effects this can happen even if there is only one track in the event.

5.4 PathfinderInterfaceProcessor

A processor using PATHFINDER is available in MarlinTPC [2]. It contains an option to shift the hits towards the origin before doing the track finding (see section 5.3). After the track finding the track parameters are shifted back so that they match the original hit positions. It is also possible to chose a point of reference different from the default one (which is 0,0,0).

6 Performance of PATHFINDER

In this section a short summary of the performance of PATHFINDER is given. For more details see [3].

6.1 Track Finding Efficiency

PATHFINDER has a good track finding efficiency for events with low track multiplicities and reaches the same efficiency as a local track finding algorithm [12]. In Figure 4 the track finding efficiencies for the two algorithms are shown. A simulation of single muons in the ILD [13] was used. A track was defined to be found correctly if at least 75 % of all hits are assigned to the reconstructed track correctly. The efficiency is the number of correctly found tracks over the number of simulated tracks.

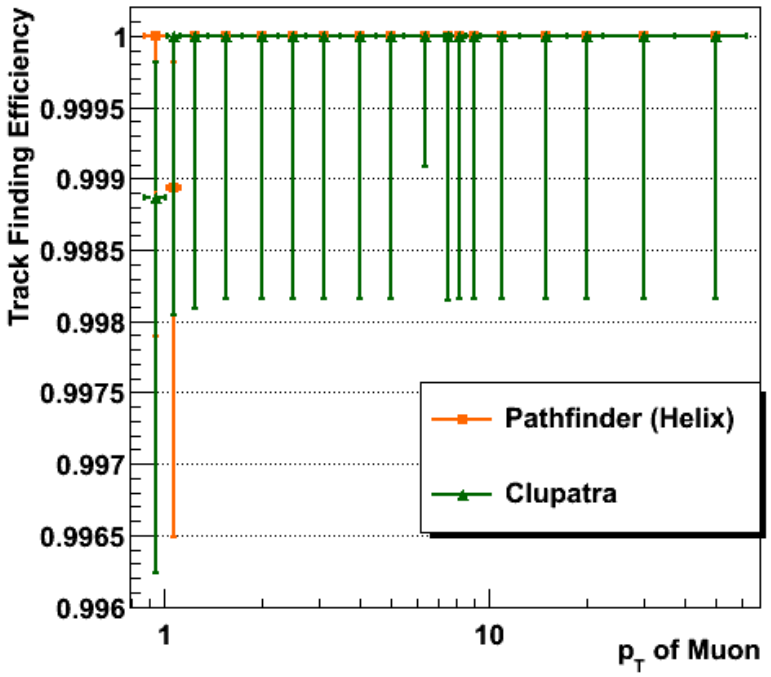

Figure 4: Track finding efficiency for PATHFINDER and a local track finding algorithm (Clupatra [12]). A simulation of single muons with different transverse momenta in the ILD [13] was used.

6.2 Computing Time

The Hough transformation is a global method and thus slower than algorithms following local track finding approaches. The part taking longest is the calculation of the Hough space. A good estimate for the time needed is how often an entry is filled into the Hough space, in other words how often the access operator is called. In case for straight lines and helices with vertex constraint the number of calls of the access operator can be described by a polynomial of order $\mathcal{O}(2)$, in case for a helix without using a vertex constraint it is a polynomial of order $\mathcal{O}(3)$. For example (on an average computer): Finding one track with 222 hits in one event takes about 2.5 seconds, 50 tracks in one event would take about 18 minutes.
To conclude, single track events (as expected in testbeam data) can be reconstructed using `PATHFINDER`. Reconstructing full physics events is not possible in a reasonable amount of time with the current version of `PATHFINDER`.

References

[1] P. Schade, J.Kaminski on behalf of the LCTPC collaboration, A large TPC prototype for a linear collider detector, Proceedings of the 12th International Vienna Conference on Instrumentation, Nucl. Inst. & Meth. in Phys. Res. A 628, Issue 1, 1 February 2011, 128-132

[2] Christoph Rosemann, Executive Summary: Recent Software Developments for Time Projection Chambers, LC-DET-2012-071, 2012

[3] Isa Heinze, Development of a Hough Transformation Track Finder for Time Projection Chambers, DESY-THESIS-2013-055, December 2013

[4] Thomas Krämer, Track Parameters in `LCIO`, LC-DET-2006-004, 2006

[5] J. Alcaraz, Helicoidal Tracks, L3-Note 1666, February 18, 1995

[6] Paul V. C. Hough et al., Method and Means for Recognizing Complex Patterns, US Patent 3,069,654, 1962

[7] J. Illingworth and J. Kittler, The Adaptive Hough Transformation, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9, 690-698, September 1987

[8] V. Karimäki, Effective circle fitting for particle trajectories, Nuclear Instruments and Methods in Physics Research A305 (1991) 187-191

[9] `http://www-flc.desy.de/flc/flcwiki/PATHFINDER` (last accessed on January 24, 2014)

[10] `PATHFINDER` Repository: `https://svnsrv.desy.de/websvn/wsvn/General.pathfinder?` (last accessed on January 28, 2014)

[11] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also `http://root.cern.ch` (last accessed on November 16, 2012)

[12] F. Gaede, Clupatra - Topological TPC pattern recognition, AIDA WP2 Meeting 2011, CERN, October 2011. See also `http://indico.cern.ch/getFile.py/access?contribId=9&resId=0&materialId=slides&confId=159340` (last accessed on February 28, 2013)

[13] The International Linear Collider Technical Design Report - Volume 4: Detectors, 2013, CERN-ATS-2013-037, DESY 13-062, ILC-REPORT-2013-040