Particle Track Estimation

Andreas Spanopoulos (andrewspanopoulos@gmail.com)
Supervisors: Andreas Salzburger, Noemi Calace
CERN, Geneva, Switzerland

Abstract
In this report, we formulate the problem of reconstructing charged particles using hits from a detector, and we assess the performance of the Hough Transform algorithm, which was implemented from scratch. A comparison is made with the results from both the ideal and realistic datasets and Machine Learning based methods are discussed in order to enhance our predictions, thus setting a strong baseline for the solution of this problem.

Keywords
particle track, hough transform, machine learning

1 Introduction

With the creation of enormous particle colliders such as the **Large Hadron Collider (LHC)** and Detectors such as **Atlas (A Toroidal LHC Apparatus)**, the data produced by CERN annually is in the scale of tens of petabytes [1]. Analyzing this data is of outermost importance in order to provide insights in the field of particle physics, and unravel the mysteries of our universe.

One specific task that gives useful information to physicists is the identification of particles from raw hit spatial-data, generated in tracking detectors when high-energetic particles collide. The goal is the *reconstruction of charged particles*. Since we don't know a priori the number of particles that are generated in a physics event, the problem reduces to assigning each hit to a cluster of hits that were likely produced by the same particle.

Since the data is effectively in the form of a three dimensional point cloud, a way to create groups of hits that likely belong to the same particle, is to try to reconstruct the trajectories of those particles. The key to this idea is that the trajectories obey to physics laws. We can model the spatial position of a particle using information such as its transverse momentum p_T, charge q and emittance angle ϕ. Therefore the problem of computing the trajectory of a particle boils down to finding its origin, also known as the vertex (x_0, y_0, z_0) and the values for the p_T, q, ϕ parameters. From now on we will refer to the trajectories as *tracks*.

2 Dataset

In a recorded physics event, there is no ground truth information that can help us assess the performance of any of our algorithms. That's why physicists and developers have turned to simulation software in order to produce their datasets. The **ACTS toolkit** [2] is used to simulate the samples used to assess the performance of the algorithms presented in this documents. Events are generated using **Pythia8** [3]. These samples contain the ground truth regarding how many particles are present in an event, along with the particle that every hit belongs to and other useful information such as the initial momenta and charge.

2.1 Visualizing the datasets

The most common ways to visualize the data of a given dataset is to plot the '**x-y**' and '**r-z**' **views**. The former refers to plotting the x and y coordinates of each hit, and the latter to the r and z coordinates

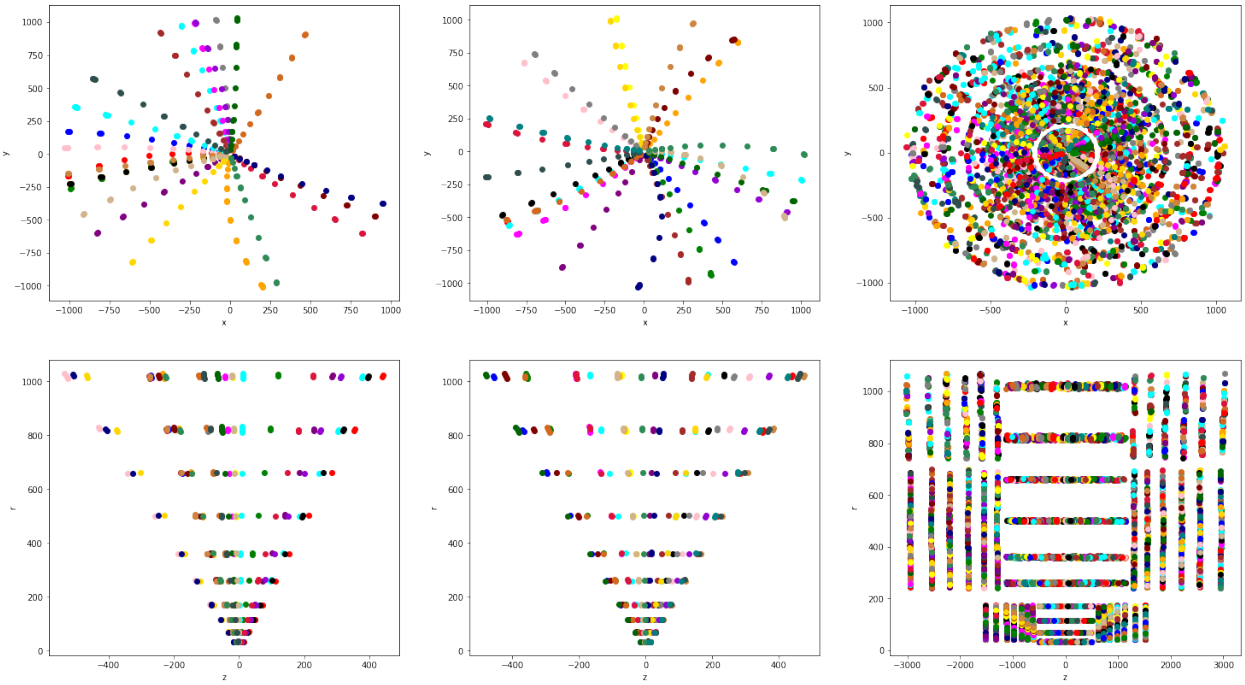

Fig. 1: x-y and r-z views for the ideal dataset with muons (left images), for the dataset with pions (middle images) and for a realistic physics event (right images).

respectively. Note that r is computed as $r = \sqrt{x^2 + y^2}$. Both can give views us different insights on what might have happened during an event.

In Fig. 1 we show examples of those views for 3 different datasets: One for 25 muons (pdg = 13) with $0.5 \leq p_T \leq 10$ GeV and $-0.5 \leq \eta \leq 0.5$, where there is no material interaction and the magnetic field is constant to 2T, one for 25 pions (pdg = 211) with $0.5 \leq p_T \leq 50$ GeV and $-2.5 \leq \eta \leq 2.5$ where there are material effects (i.e. multiple scattering) and the magnetic field is non homogenous, and one where a real collision is simulated, with pileup[1] 50 and soft QCD.

For a more detailed introduction to event display and visualization, as well for more advanced methods such as V-plots, the reader can check the work of Drevermann et al. [4].

We can also take a look at the 3d projections of those hits in Fig. 2, though the plots are very dense and not easy to comprehend.

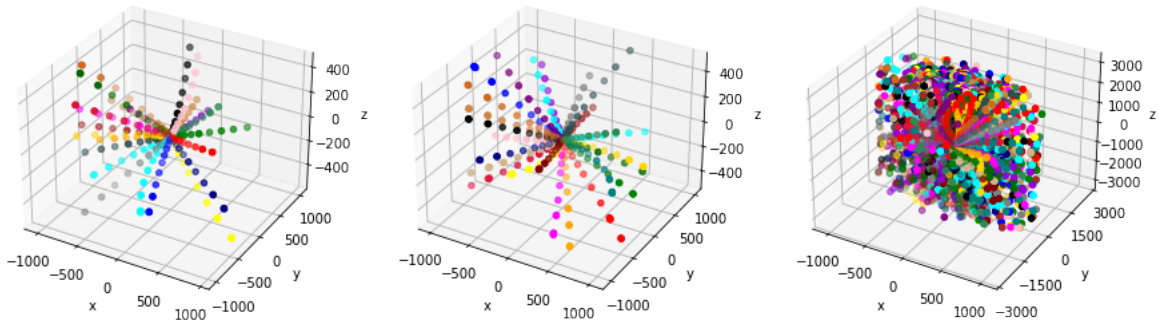

Fig. 2: 3D views for the muon, pion and realistic datasets respectively.

[1] *Pileup* refers to the number of additional proton-proton interactions that take place during an event.

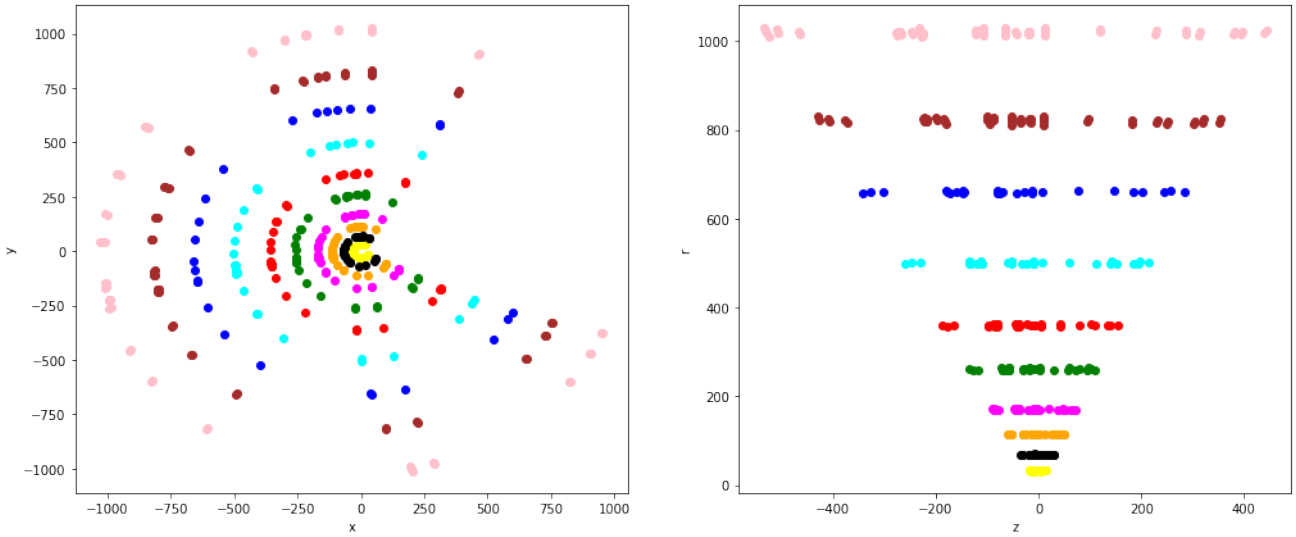

Fig. 3: Distribution of detector regions. Each color represents a layer-volume combination.

2.2 Additional Data

But how do detectors record this spatial data? In a very simplified version, detectors are a set of high resolution cameras where each one of them points to **completely** dark spot. Once a high energetic particle goes through a spot, the corresponding camera will notice a change in its frame and will record the hit. Thus, the detector is divided in regions called layers, where each layer covers a specific area. Layers are also divided into small substructures, but there is need to analyze this further in the context of what discussed in this document. The reader can check Ref. [5] for more information.

One additional piece of information that is also available in real world physics event, is which sensor of the detector has been hit. This can be quite useful as we will see in the next sections, because the track of a charged particle will go through each layer at most once[2]. Therefore, if we have 2 hits that were identified by the same layer, then we are sure that they do not belong to the same particle. In Fig. 3 we show how the regions of the detector are distributed.

2.3 Towards realistic datasets

There were different datasets to study and analyze throughout the project. First, an ideal dataset where only one particle type (e.g. muon) was available, with 25 particles per event. This dataset was simulated without material interactions and had constant axial magnetic field of 2T. Later, we explored datasets that took into account material effects and didn't have a homogenous magnetic field. Other particles were tried as well, like pions and electrons. The final datasets (that were simulated using Pythia8) with high pileup values and soft QCD, closely resembled a recorded event, by having particles of all types and many more tracks. Hadronic interactions with detector material were not taken into account.

3 Hough Transform

The **Hough Transform (HT)** is a feature extraction algorithm that is used in the fields of Image Analysis and Computer Vision. It was originally introduced by Paul Hough at 1962, and was later improved by Duda et al. [6] in 1972. The idea behind the algorithm is to detect specific shapes in images. The first version was able to detect straight lines. Later, more extensions of the algorithm were published, such as the **Circular Hough Transform (CHT)** and the **Generalized Hough Transform (GHT)** by Ballard [7].

[2]This not always true in real life scenarios, as in some layers, some sensors might overlap.

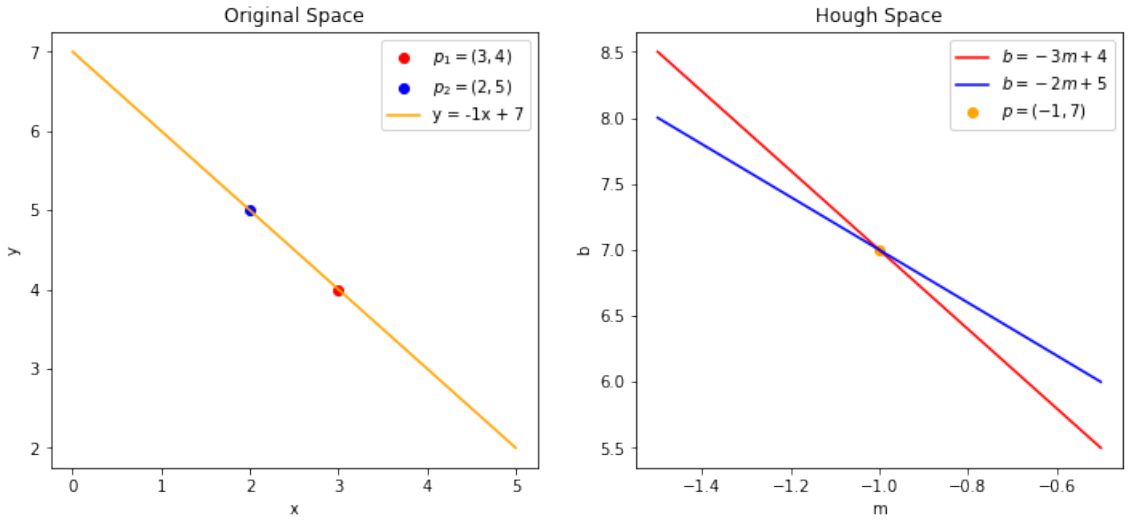

Fig. 4: Original Space (left) and Hough Space (right) Visualization. The line that connects points p_1, p_2 in the original space, is the intersection point of the lines defined by p_1, p_2 in the Hough Space.

3.1 A brief introduction to the algorithm

Given a set of points (or pixels for images) $A = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, the goal is detect which points might lie on the same straight line. Since any two points are connected by a line, we are looking for lines that contain at least a specific amount of points (e.g. 100).

One idea that comes to mind is to enumerate all $\binom{n}{2}$ lines and see how many points each one contains. This wouldn't be very practical if n were too large, and also it wouldn't work if the data had noise. To overcome this issue, we can use the Hough Transform algorithm.

It is known that the equation of a line is represented by $y = mx + b$, where m is the slope of the line and b is the intercept. Suppose we have a point $p = (x_0, y_0)$. If we plug it in the equation of the line, we get:

$$y_0 = mx_0 + b \;\Leftrightarrow\; b = -x_0 m + y_0 \tag{1}$$

which basically is itself a line equation with abscissa m, ordinate b, slope $-x_0$ and intercept y_0. That is, m, b become the variables and $(-x_0, y_0)$ become the coefficients. The solutions to equation 1 effectively give us the coefficients of all the lines that go through point p. Any pair (m, b) that satisfies 1, is a straight line that goes through p. Out of all the straight lines, we would like to find the one that will go through other points as well. The range of values for the (m, b) parameters is referred to as the Hough Space (or Parameter Space).

This can be thought of in the following way: every point in the original x-y plane (Original Space), corresponds to a line in the Hough Space. Every point in the Hough Space, corresponds to a line in the Original Space. Further visualizations can be found in Fig. 4.

Therefore, the goal is to find the most-common intersection points. Since the data might be a bit noisy, thus not allowing for an *exact* intersection point, we can discretize the Hough Space in small bins (i.e. use **binning**) to identify hot spots. We have to keep track as to which points fall into which bins, in order to later see how many hits fall inside the bin in total. For this we use an **accumulator array**. We can then pick the bins with the most hits inside them and consider them as our *estimated* tracks.

A Hough Transform is used here in the longitudinal (r-z) plane to detect the tracks of the particles. The results we got (for 1 event) can be seen on Fig. 5.

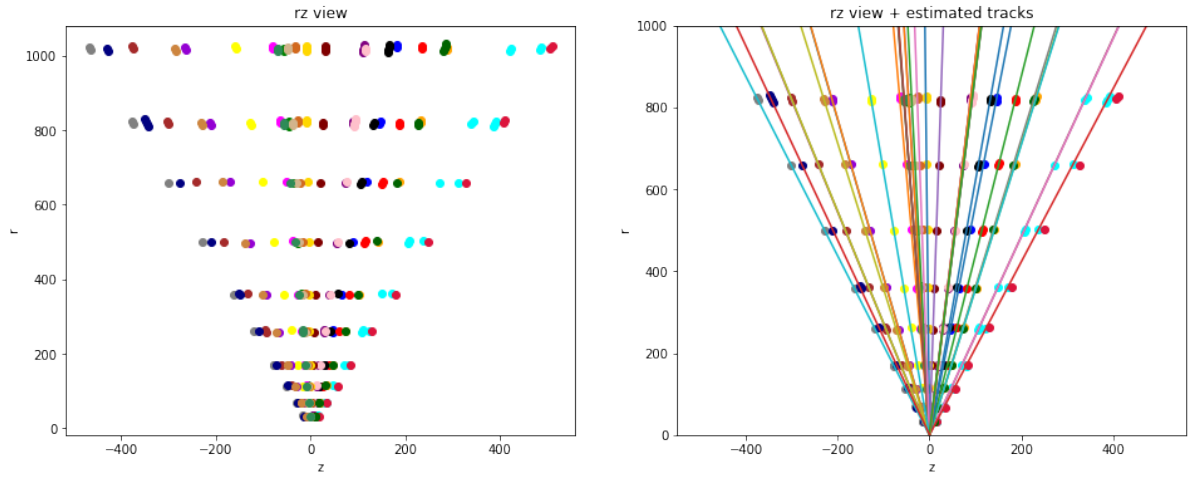

Fig. 5: r-z view of the ideal dataset (left) and the estimated tracks for it (right).

3.2 Extending the Hough Transform to the Helical Conformal Mapping Transform

With the hypothesis that the magnetic field is constant in the (0, 0, 1) direction, the particle will perform a helical trajectory. Thus, in the x-y plane it will leave a circular track, obeying the following formula [8]:

$$\frac{qA}{p_T} = \frac{\sin(\phi_0 - \phi_1)}{r_1} \tag{2}$$

where ϕ_1 is the angle θ between a hit, the vertex and the x-axis, r_1 is the distance from the vertex and $A = 3 \times 10^{-4}$ GeVc^{-1}mm^{-1}e^{-1}. If we are interested in particles where the difference $\phi_0 - \phi_1$ is small, then we can approximate $\sin(\phi_0 - \phi_1) \approx \phi_0 - \phi_1$, thus giving the final equation:

$$\frac{qA}{p_T} = \frac{\phi_0 - \phi_1}{r_1} \tag{3}$$

For particles with high p_T, we can assume $\phi_0 - \phi_1$ to be small, thus the approximation holds. The results (for the ideal dataset) can be found in Fig. 6.

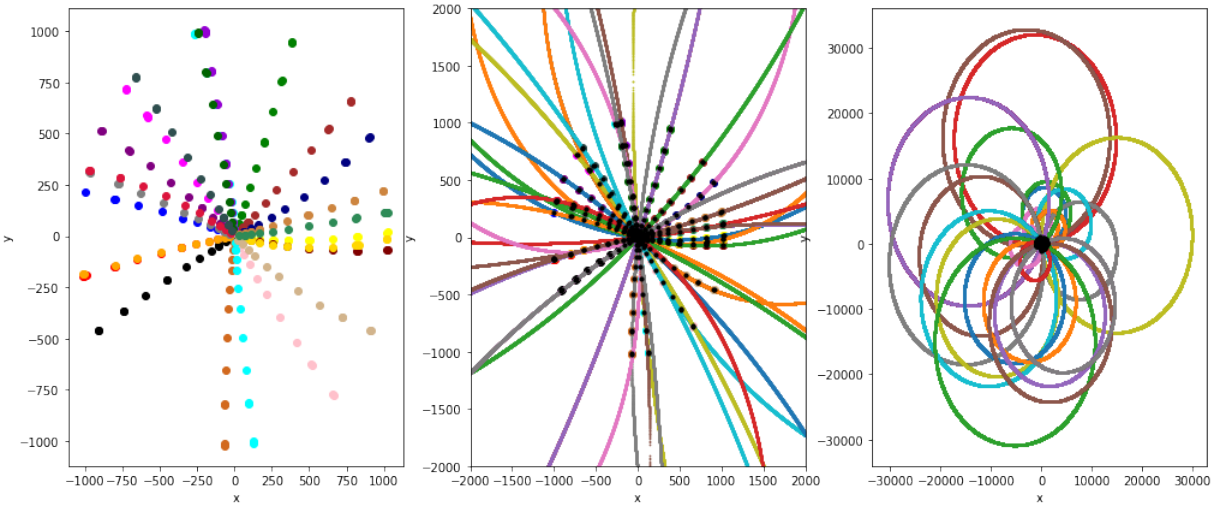

Fig. 6: x-y view of the ideal dataset (left), zoomed-in est. tracks for it (middle) and complete est. tracks (right). Note that the detectors are small, which means that the right setup occurs only for low p_T tracks (with small radii).

HT Type	bin-width	bin-height	nhits
Helical	0.001	0.2	10
Longitudinal	0.04	20	6

Table 1: Optimal values found for the hyperparameters of the HT.

3.3 Hough Transform Hyperparameters

During the development of the HT, some hyperparameters had to be tuned in order to achieve better results. Those hyperparameters are the **bin size (width x height)** and **minimum number of hits inside a bin (nhits)**, in order for that bin to be considered a candidate estimation.

Trial and error showed that as the bin size *increases*, the ER, FR and DR (defined in the next section) drop. The ER might oscillate a bit, but in general the bigger bin sizes lack in precision, hence the loss. If the *nhits* parameter decreases, then the ER, FR, DR increase. Bigger nhit values will help in decreasing duplicates, but might lose some efficiency. The optimal values found can be seen in Table 1.

4 Results

Even though the results might look promising at first glance, we need a way to automatically measure the performance of our reconstruction algorithm, using the ground truth labels from the simulation software. For this, we use the metrics reported in the next section.

4.1 Metrics: Efficiency, Fake and Duplicate Rates

First we have to define the notions of *weight* and *Matching Probability*.

Each hit in a dataset, is assigned a **weight**. This happens because some hits are *more important* than others (i.e. it is preferred to identify the first and the last hits belonging to the same particle in order to better estimate the transverse momentum). The higher its weight, the more crucial a hit is considered.

Given a set of hits R that are supposed to reconstruct a track, the **Matching Probability** of a particle is defined as the ratio of the sum of the weights of the hits found over the total sum of the weights of all the hits for that particle. Namely:

$$\text{MP}(R, \text{particle}) \;=\; \frac{\sum\limits_{h_i \in \text{particle}} w_{h_i} \times \text{found}(h_i, R)}{\sum\limits_{h_i \in \text{particle}} w_{h_i}} \tag{4}$$

where w_{h_i} is the weight of hit h_i and $\text{found}(h_i, R)$ is equal to 1 if h_i is in R, else 0. Now we are ready to define the metrics we used. Note that with E we denote set of all the reconstructed tracks and with P is the set of all particles.

4.1.1 Efficiency Rate

The efficiency rate is defined as the ratio of the number of particles reconstructed over the total number of particles. An estimated track R is considered to reconstruct a particle p, if the **leading particle** of R (i.e. the particle that has the highest sum of weights of hits present in R) is p and $\text{MP}(R, p) \geq$ threshold. Usually we set threshold = 0.5. Formally:

$$\text{ER}(E, P) = \frac{|\{p \in P \mid \exists R \in E : \text{MP}(R, p) \geq 0.5\}|}{|P|} \tag{5}$$

4.1.2 Fake Rate

The fake rate is defined as the ratio of number of fake reconstructed tracks over the total number of reconstructed tracks. A reconstructed track is considered to be fake if the Matching Probability of the leading particle is smaller than a specified threshold, i.e. 0.25. That is, the track is considered fake because effectively it doesn't reconstruct anything. Formally:

$$\text{FR}(E, P) = \frac{|\{R \in E \mid \text{MP}(R, \text{leading\_particle}(R, P)) < 0.25\}|}{|E|} \tag{6}$$

Single muon ideal dataset				Single pions, material and non uniform B				Realistic event, pileup 50, soft QCD			
HT Type	**ER**	**FR**	**DR**	**HT Type**	**ER**	**FR**	**DR**	**HT Type**	**ER**	**FR**	**DR**
Helical	1	0.12	0.98	Helical	0.99	0.15	0.97	Helical	0.92	0.2	0.99
App. Hel.	1	0.12	0.95	App. Hel.	0.98	0.15	0.95	App. Hel.	0.91	0.2	0.99
Longitudinal	0.85	0.3	0.54	Longitudinal	0.8	0.4	0.5	Longitudinal	0.72	0.4	0.99
Comb. 1	1	0.21	0.9	Comb. 1	1	0.3	0.9	Comb. 1	-	-	-
Comb. 2	0.875	0.09	0.51	Comb. 2	0.834	0.09	0.51	Comb. 2	-	-	-
Comb. 3	0.834	0.09	0.53	Comb. 3	0.834	0.09	0.53	Comb. 3	-	-	-

Table 2: Hough Transform algorithm average (over 100 events) results for the ideal dataset (left), for the pions with material effect and non-homogenous magnetic field dataset (mid) and for the realistic dataset (right).

4.1.3 Duplicate Rate

The duplicate rate is defined as the ratio of the number of duplicate tracks over the total number of estimated tracks. Two estimated tracks are considered to be duplicates if they have the same leading particle. With some algebraic manipulations, the duplicate rate can be equivalently defined as:

$$\mathrm{DR}(E,P) = 1 - \frac{|\{p \in P \mid \forall R \in E : \text{leading\_particle}(R,P) = p\}|}{|E|} \tag{7}$$

4.2 Comparison

Different combinations of the Helical and straight line Hough Transform were tried in order to best fit the given data.

The first combination is running the Helical HT first, removing the found hits, running the longitudinal with the remaining hits and then aggregating the results. For the second combination, both the Helical and Longitudinal HTs are run, and then an "intersection" is taken, that is, only the estimated tracks that have a common track in the other transform, are kept. The third combination is similar to the first, except that the hits aren't removed (i.e. the predictions of the Helical HT are enhanced by the longitudinal HT). The results of these algorithms in each one of the datasets described before, are located in Table 2. As the events get more difficult, we can notice the decrease in performance of the HT (i.e. efficiency drop, fake/duplicate rates increase).

One thing that is of particular interest is to see how the Approx. version of the Helical HT works with low p_T tracks. It would make sense to see the efficiency drop as p_T drops, because the difference $\phi_0 - \phi_1$ wouldn't be near 0, and thus the sinus approximation wouldn't work. In Fig 7, the ER for particles with different p_T (in the pions dataset) can be seen. As expected, the efficiency drops for lower p_Ts. We also show the ER for the different values of pseudorapidity η, which seem constant.

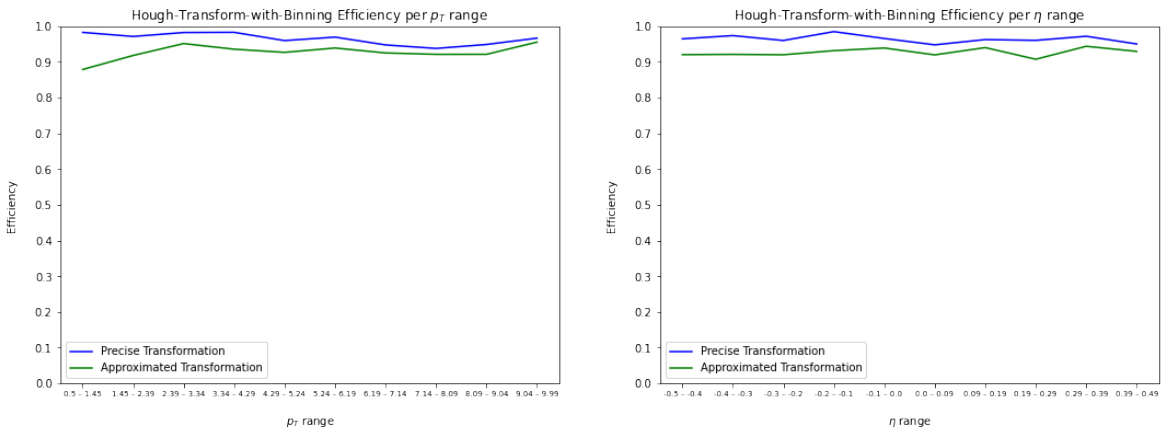

Fig. 7: Efficiencies per p_T and η ranges in the pions dataset.

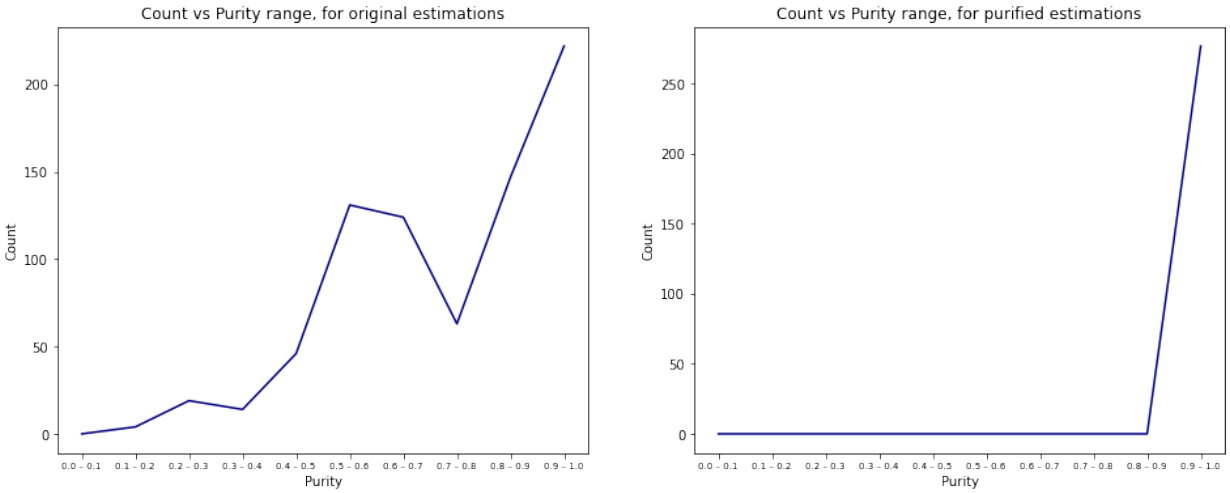

Fig. 8: Count vs Purity plot for the ideal dataset before applying purification (left) and after (right). Similar results hold for the other datasets as well.

5 Improvements

During the development of the HT algorithm, there were 2 main issues that had to be addressed. The first one was the high duplicate rate in the estimations, and the second one was the slow runtime of some specific functions, such as the metrics.

5.1 Purification - Duplicate Removal

In the beginning, different duplicate removal algorithms were tried, in order to decrease the DR. The goal was to not lose efficiency as well. Even though the implementation of the duplicate-removal algorithms seemed correct, the results were not great. After some analysis, the issue came down to be in the *purity* of estimations.

Given a set oh hits R that are supposed to reconstruct a track, the **Purity** of that set is defined as the ratio of the sum of the weights of the leading particle over the total sum of the weights of all the hits inside the set. Namely:

$$\text{Pur}(R, P) = \frac{\sum\limits_{h_i \in \text{leading\_particle}(R,P)} w_{h_i} \times \text{found}(h_i, R)}{\sum\limits_{h_i \in R} w_{h_i}} \tag{8}$$

On Fig 8, the purity values for the estimations can be seen. They aren't very good. Ideally, they should be around 1.0, which does happen after the purification process.

In order to **purify** the estimations, the longitudinal HT was used. Specifically, for each bin chosen in the Helical HT, the longitudinal HT was used for the hits on those bins. This turned out to work perfectly, as in the datasets that were used, usually there are no tracks that are very close in all three axes. A visualization can be seen in Fig. 9.

Now that the estimated tracks have been purified, the duplicate removal algorithm that was used is very straightforward: if two estimated tracks are **close within a threshold distance** (e.g. 100 bins) and they **contain a specific number of common hits** (e.g. at least 15%), then they are marked as duplicates, and the one with less hits is discarded from the estimations. Table 3 can be checked for further information.

Dataset	ER	FR	DR
Ideal	1	0	0.01
Pions	0.98	0	0.01
Realistic	DNF	DNF	DNF

Table 3: Purified + Duplicate-Removal Hough Transform Results. Note that for the realistic dataset, the purification process would take to long to compute, as it would need to run the longitudinal HT once for every bin, which is quite expensive.

5.2 Metric computation speedup

The early implementation of the metric methods made use of Hash Tables (python dictionaries), which turned out to be a very bad choice. The time it took to compute the metrics was very slow, and this was due to the constant hashing of the same objects being done over and over again. The solutions were 2: vectorization and arrays, and they could be combined.

The vectorization term refers to giving the ability to apply to matrices a function that gets applied in scalars. This is done using parallel programming, and quite often it provides very good speed ups [9].

The arrays term refers to switching from hash tables to an array data structure for the handling of the data. This usually improves the performance of an algorithm, as arrays are always stored in the RAM, so no cache misses will occur (like in other data structures), and accessing/editing can be done in $\mathcal{O}(1)$. Therefore the whole philosophy of the book-keeping mechanism for the results of the algorithm had to be changed from lookup-based to index-based (basically the indices in the arrays). This was easy and quick to do. In general, it is a good practice in Data Science to use arrays when dealing with Big Data and when possible.

With these changes, the compute time of the efficiency for pileup 200 dropped from 1-2 years to 1 minute.

6 Future Work

One task that could definitely be done is improving the runtime of the purification mechanism. For this, we could carefully craft the longitudinal HT (i.e. running it on a specific range for each bin, depending on the hits inside) and also use multithreading programming. This would allow for much better estimations in larger events.

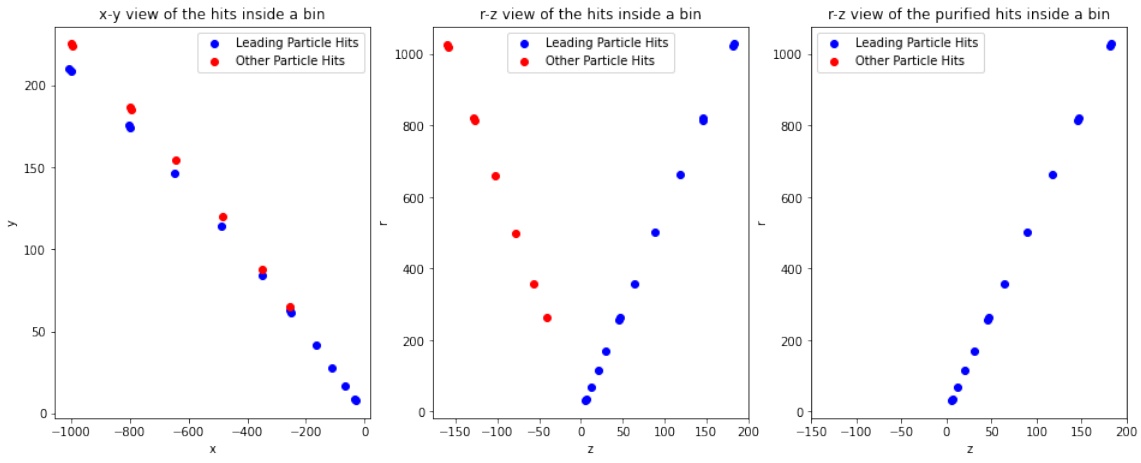

Fig. 9: A short visualization of the purification process for one estimated track in the ideal dataset.

One aspect that is also very interesting and worth exploring is the usage of Deep Learning models for classification and inference purposes. The first idea that comes to mind is creating a model for classifying whether a bin correctly reconstructs a particle (fake classifier). One other could be for purification: to classify each hit of a bin as correctly placed or not (actually belonging to the leading particle). Another idea is to create a model that classifies whether 2 bins are duplicates or not. As a last idea, we could implement a model that given the hits of a particle, learns to infer its properties, like the vertex and the momentum.

7 Conclusion

After many tests and evaluations of the HT and its variations that were implemented, it is safe to conclude that even though it's not a state-of-the-art algorithm, it achieves remarkable results. The purification process is quite slow though. There are many candidates ways to improve this, i.e. careful crafting of the longitudinal HT and parallelization.

The metric scores that were achieved set a strong baseline for future solutions. Supposing that compute is not a limiting factor, the performance of the HT could be further enhanced, thus coming closer to fully solving the particle tracking problem. There are many ideas for this project, though currently Machine Learning and Computer Vision techniques seem the right direction to head to for improvement.

8 Acknowledgements

Firstly, I would like to thank CERN for selecting me for this programme. I am really grateful for having been given the opportunity to work alongside such brilliant minds, like my mentors Andreas Salzburger and Noemi Calace. They taught me the basics of particle physics that were needed to carry on with the project, and then assisted me throughout the whole summer internship by providing me with great insights on the results I was getting. I also thank them for buying me lunch at Luigia, waiting for you guys in Greece to take you to the best restaurants.

Bibliography

[1] CERN Storage information: `https://home.cern/science/computing/storage`

[2] ACTS Common Tracking Software: `https://github.com/acts-project/acts`

[3] Pythia8 Simulation Software: `https://pythia.org/`

[4] H.Drevermann, D.Kuhn, B.S.Nilsson, Is there a Future for Event Display? CERN, Geneva (1993)

[5] Atlas Detector: `https://atlas.cern/discover/detector`

[6] Richard O. Duda and Peter E. Hart, Use of the Hough Transformation To Detect Lines and Curves in Pictures Stanford Research Institute, Menlo Park, California (1972)

[7] D. H. Ballard, Generalizing the Hough Transform to detect abritrary shapes, Computer Science Department, University of Rochester, Rochester, NY 14627, U.S.A.

[8] Mikael Mårtensson, A search for leptoquarks with the ATLAS detector and hardware tracking at the High-Luminosity LHC, Acta Universitatis Upsaliensis Uppsala (2019)

[9] An interesting blogpost on vectorization: `https://medium.com/@rajathbharadwaj/what-is-vectorization-19bf4f531ef8`