# PERFORMING TRACK RECONSTRUCTION AT THE ALICE TPC USING A FAST HOUGH TRANSFORM METHOD

# Outline

- The ALICE experiment and AliceO$^2$
- The O$^2$ Computing System
- Track reconstruction using the Hough Transform algorithm
- Execution on the O$^2$ computing system

# The ALICE experiment

The ALICE experiment is planned to be upgraded for 2018 (LHC LS2) (*Technical Design Report for the Upgrade of the Online-Offline Computing System - Buncic, P et al – CERN-LHCC-2015-006*)

The **tracking precision** of the experiment will be improved at both central and forward rapidity with Pb-Pb collisions at rates of 50kHz, sampling the p-p and p-Pb at up to 200kHz

**Continuous read-out** will be implemented for some of the detectors to deal with event pile-up and avoid trigger-generated dead time - a substantial change from current practice

The data are not delimited by a physics trigger but are composed of several constant data streams to be transferred to the computing system

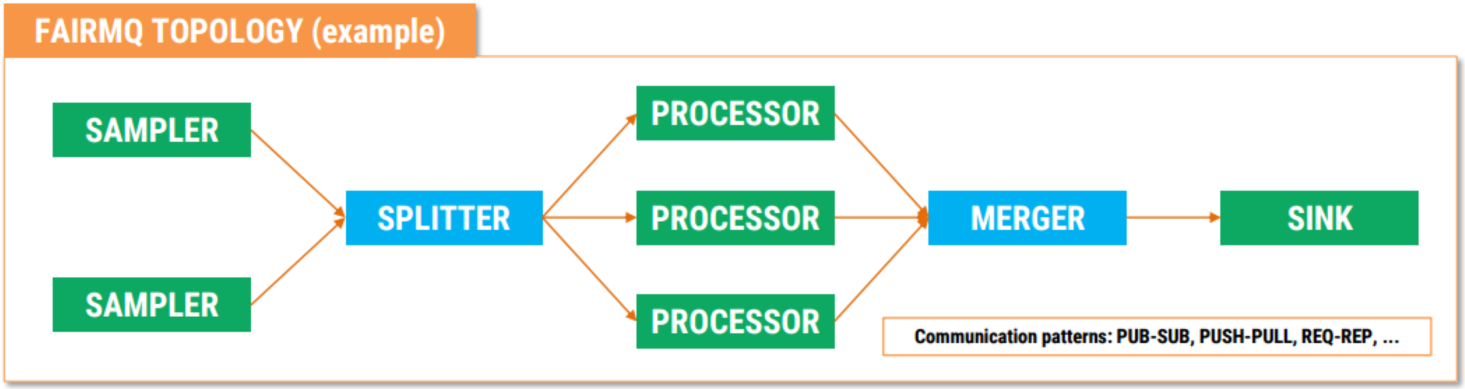
The **integration** of **online and offline** data processing requires a common software framework and a common computing facility dedicated to both data collection and processing

# AliceO$^2$

The AliceO$^2$ software framework is currently under development as a collaboration between CERN and FAIR

It is built on top of ALFA, a **modular** set of packages including FairMQ, DDS as well as configuration and management tools

**See M. Al-Turany's presentation tomorrow**
*(ALFA: Next generation concurrent framework for ALICE and FAIR experiments)*

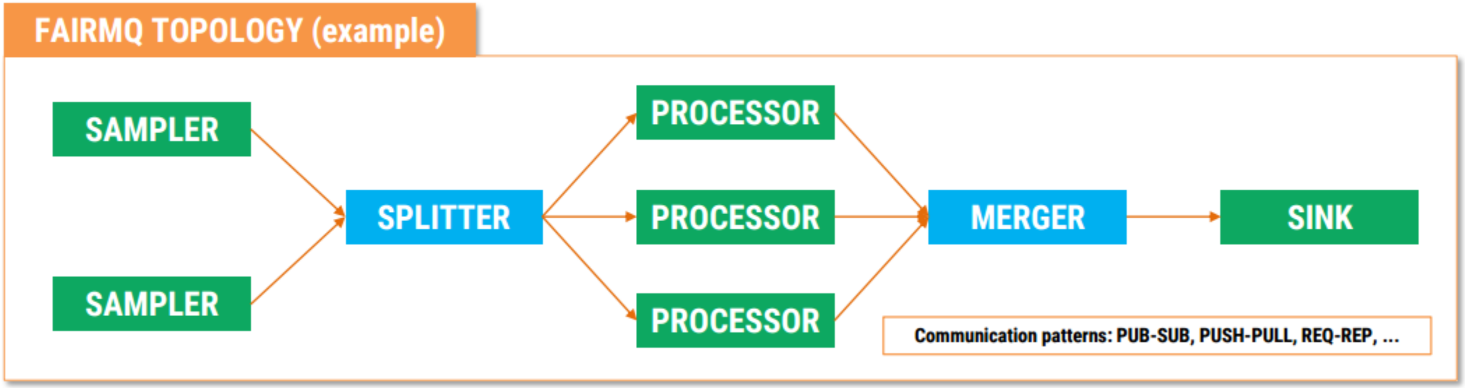


# AliceO$^2$

Processing tasks are organized into **topologies,** consisting of independent processes called **devices**

Devices communicate on either inter-process or distributed fashion using **asynchronous message queues**

It uses a data-flow based model. Multi-process parallelization is achieved with the help of the message queues

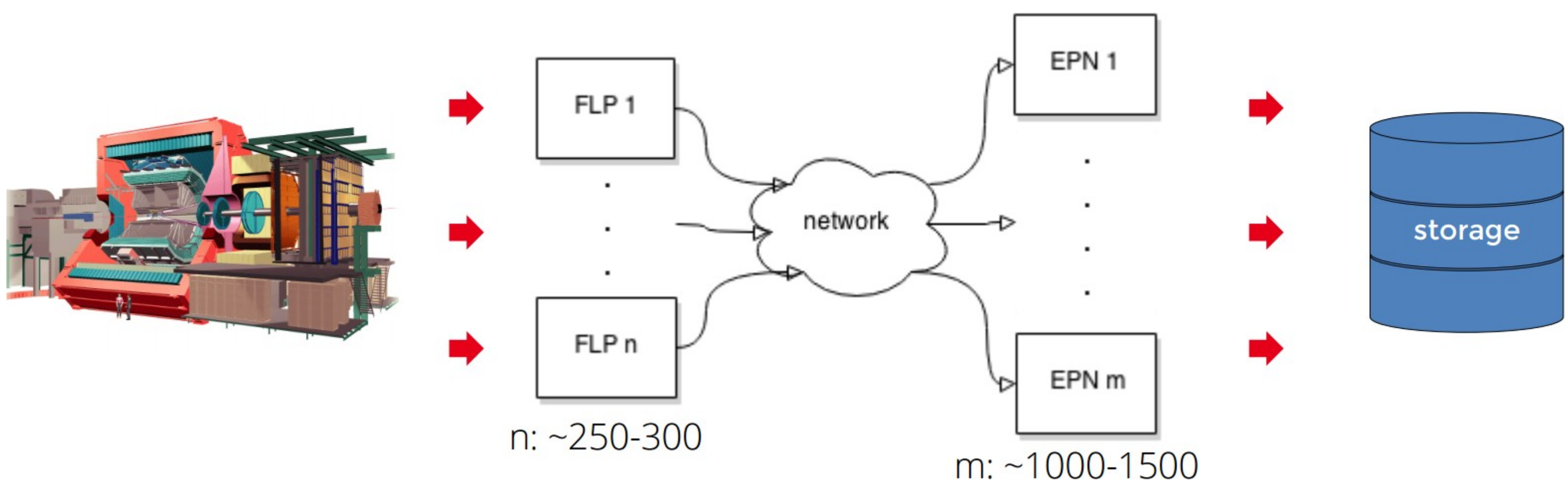


# The $O^2$ Computing System

Data and processing flow:

Data are produced by the detectors in **continuous** or **triggered** read-out mode, synchronized by the trigger system

The data from the detector front-end electronics will arrive via multiple links to the FLP (**First Level Processor**) nodes

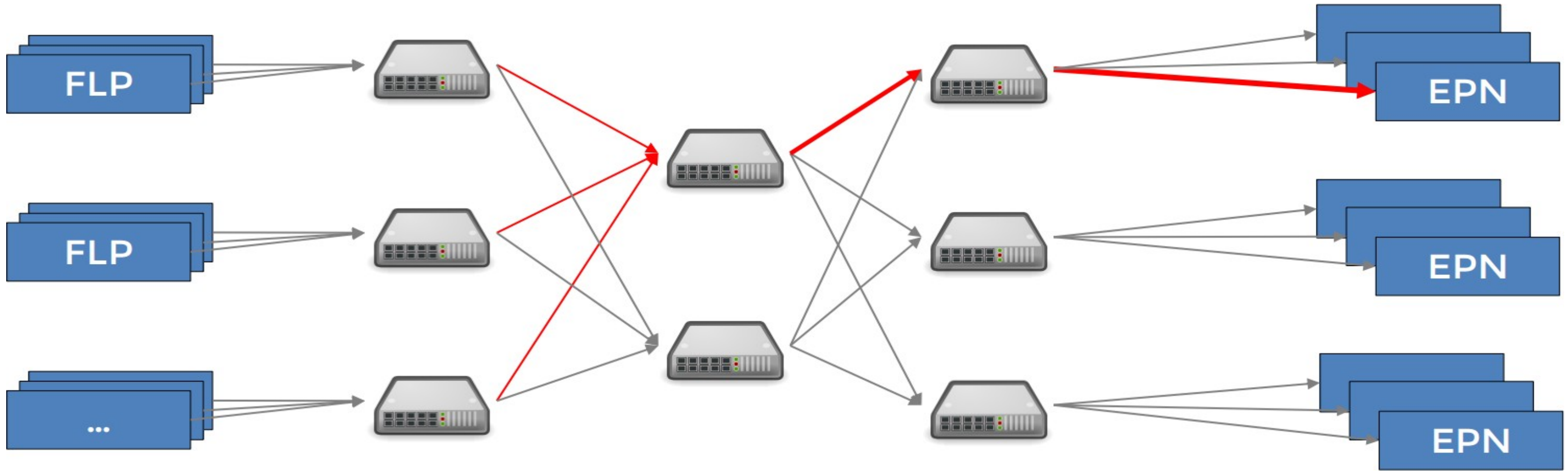


# The $O^2$ Computing System

The FLPs read out the raw data samples and achieve a first data **reduction** and **compression** by performing the local data processing tasks on data fragments

Example of data processing: local cluster reconstruction on hardware accelerator cards in real-time on the input streams

The continuous streams of data samples are split into data frames, using as a reference clock arbitrary **heartbeat** triggers embedded in the raw data streams

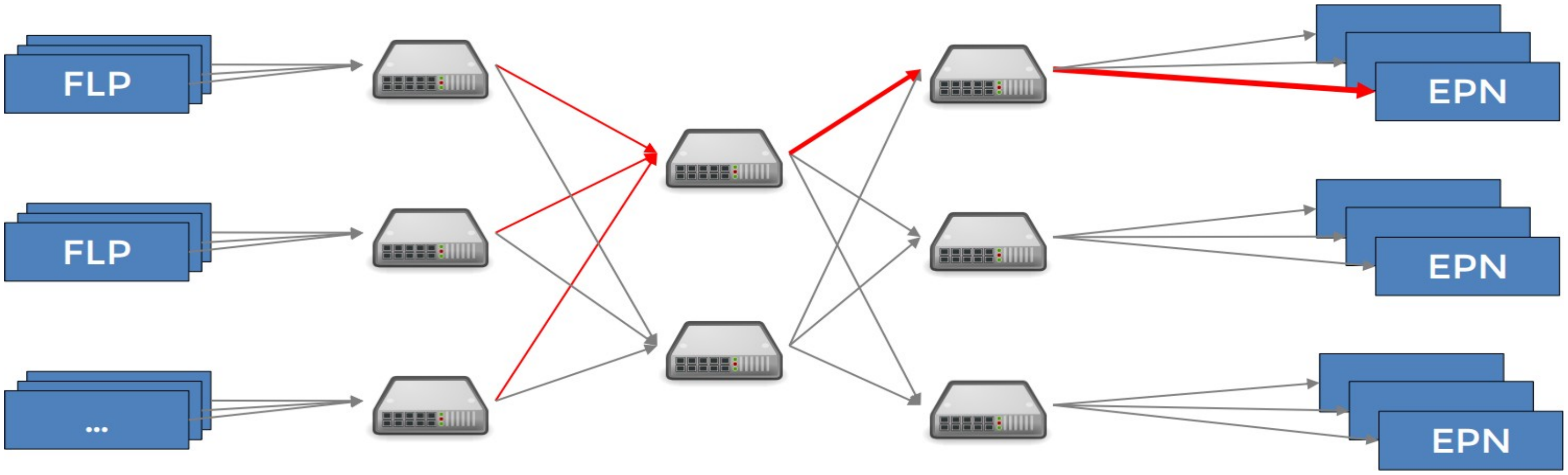


# The $O^2$ Computing System

The data frames are then dispatched to the EPNs (**Event Processing Nodes**) for aggregation via the network

The data frames related to the same time period and from all FLPs are received by the same EPN and aggregated into **time frames**

EPNs perform the reconstruction for each detector. The fully compressed time frames are then stored on disks in the $O^2$ facility or Tier 0 / Tier 1 data centers



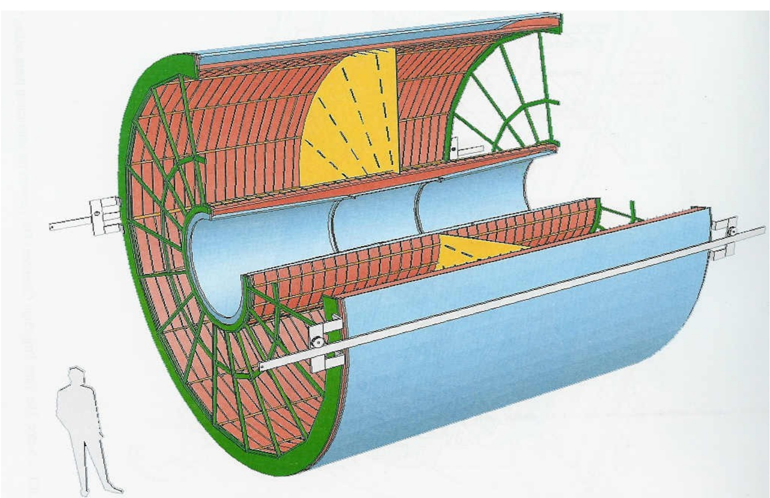
# The Hough Transform algorithm

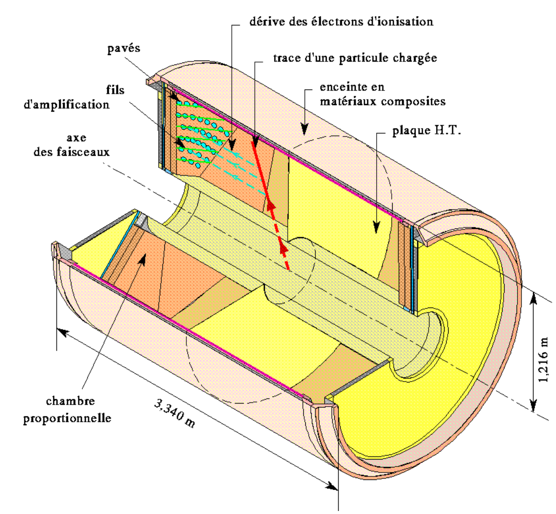
TPC is the main tracking detector of the central barrel of ALICE. 92.5% of the data is generated by the TPC

The overall acceptance covers the pseudorapidity range of $|\eta| < 0.9$

The positive ions created in the ionized gas move towards the surrounding electrodes, inducing a positive current signal

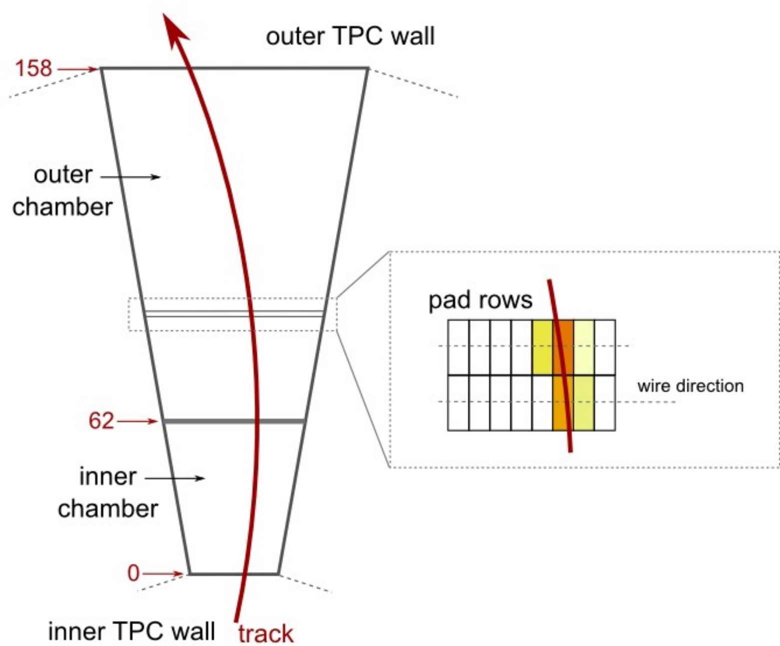
The readout of the signal is performed by the inner and outer pads (570132) located at the TPC end plates

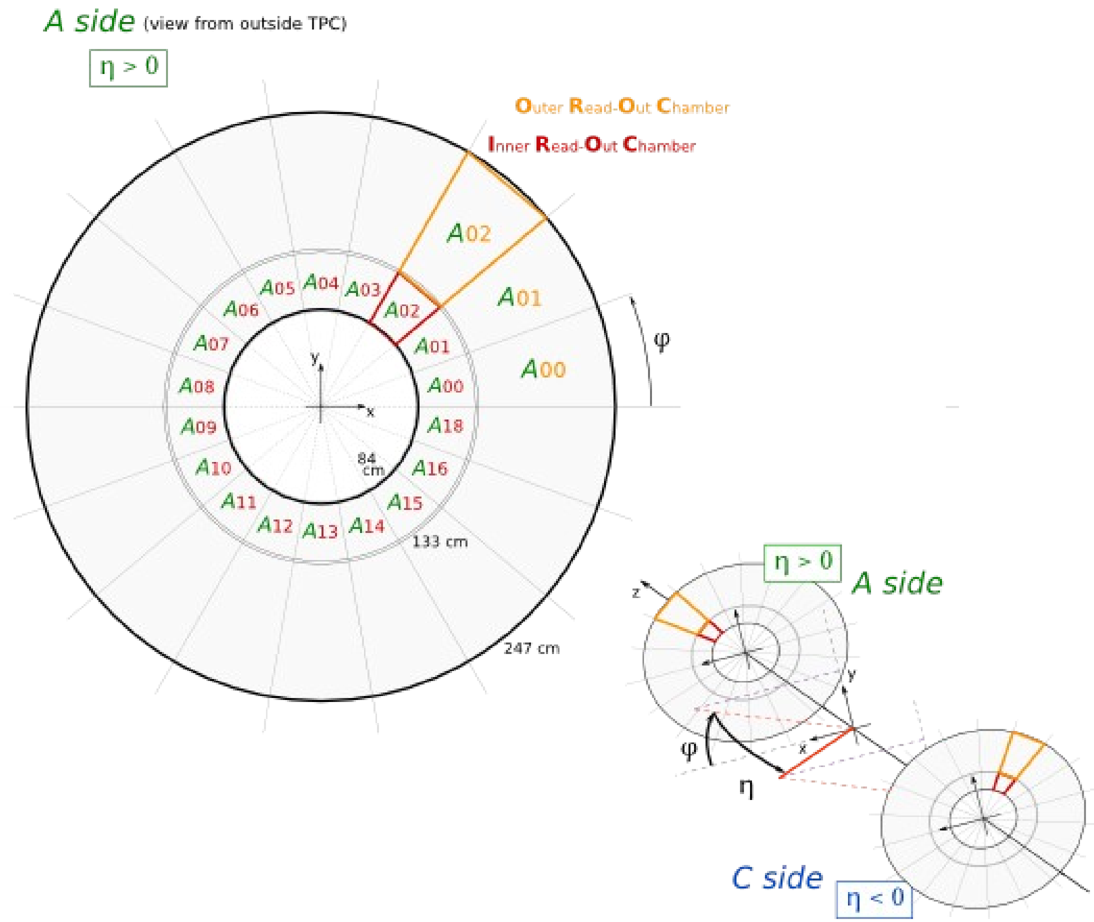




# Algorithm implementation

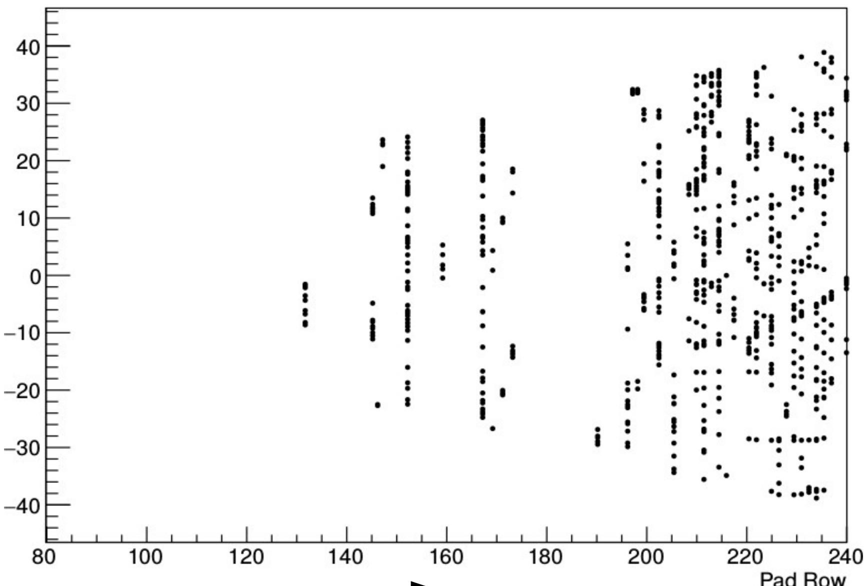
TPC data is divided into sectors, partitions and η slices

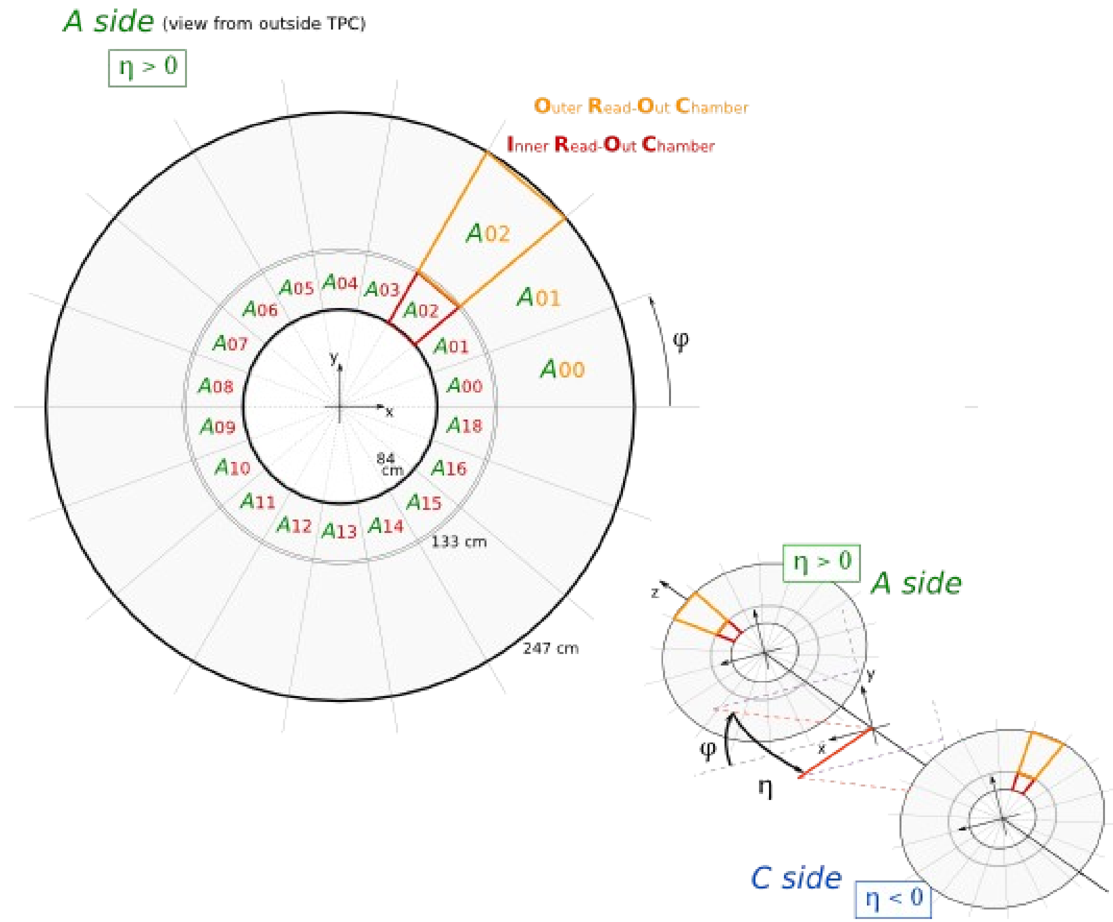


# Algorithm implementation

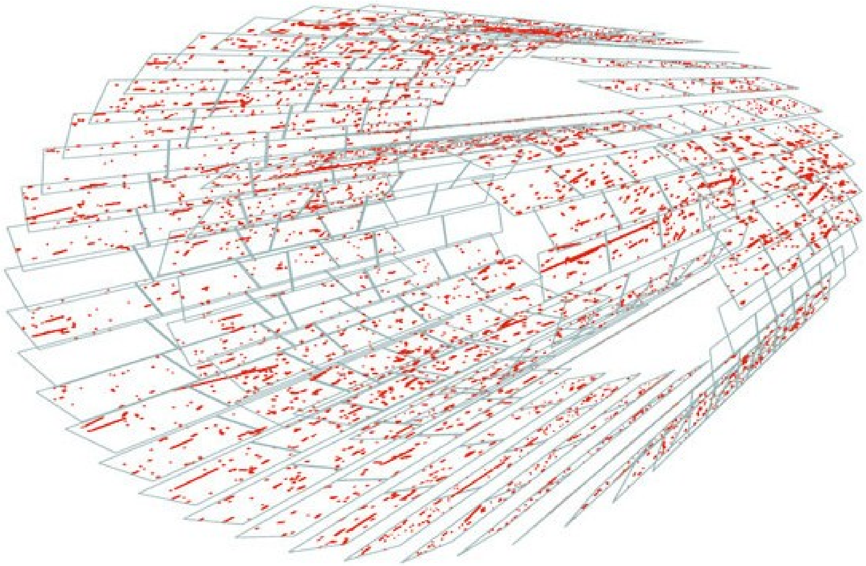
TPC data is divided into sectors, partitions and η slices



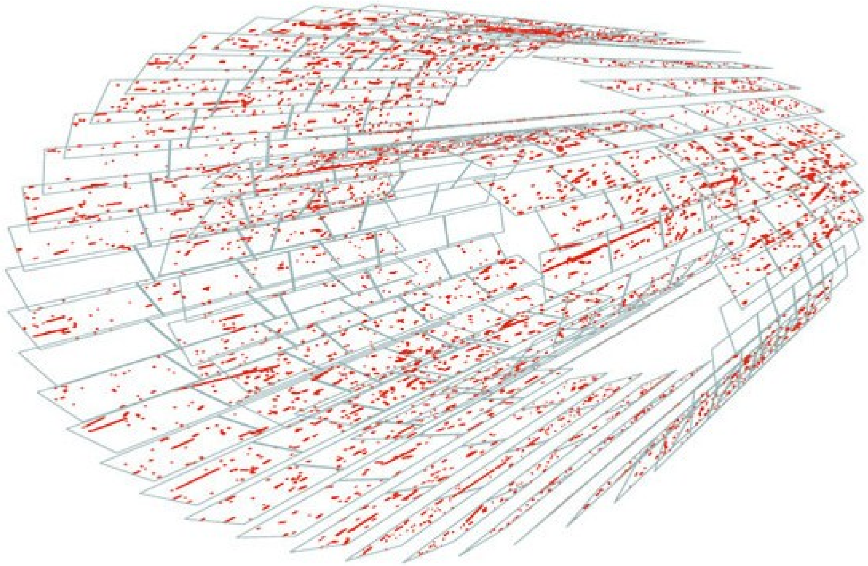

Clusters for a given TPC sector

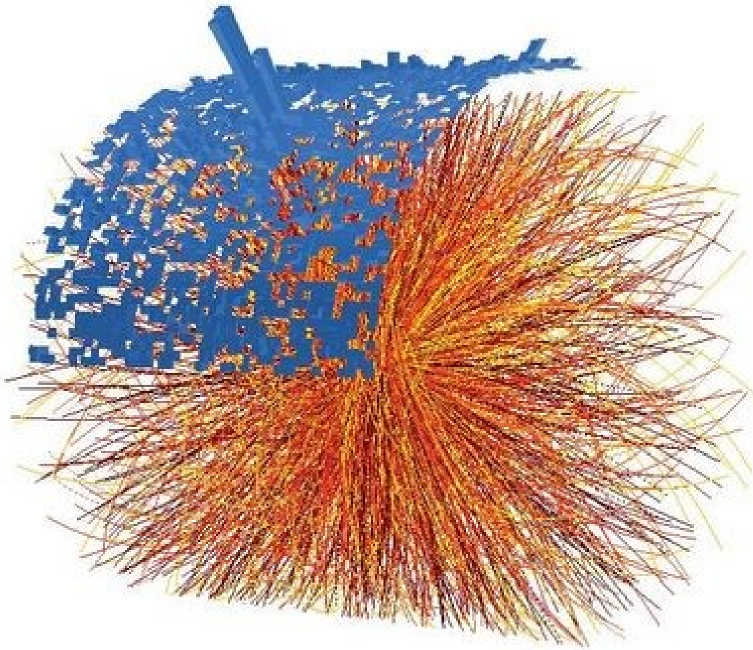
# The Hough Transform algorithm



How to get the original particle tracks reconstructed?

# The Hough Transform algorithm





How to get the original particle tracks reconstructed?

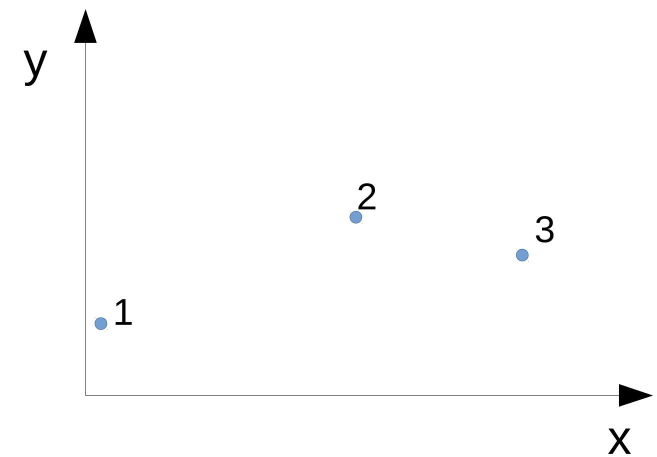
# The Hough Transform algorithm

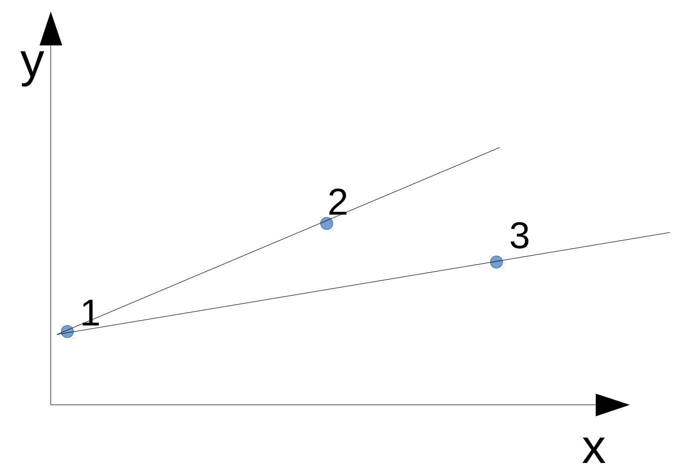
The Hough Transform algorithm can be used for **track reconstruction** on TPC

The main ideas are taken from Cvetan Cheshkov's prior work
(*Fast Hough Transform Algorithm - Cvetan Cheshkov - Presentation at HLT Workshop 6-9 June 2004*)

A computer vision algorithm. The aim is to find all the possible lines in an **image** based on a given set of **pixels**

Can be used for Track Reconstruction if image equals **readout** and pixels equal **cluster coordinates**



# The Hough Transform algorithm

The method is based on a global pattern recognition by identification of local patterns in a **parameter space**. The main idea is to consider the line - not as discrete image points (x1, y1), (x2, y2), etc., but instead in terms of its parameters

*Example line parameterization:* $\rho = x\cos\theta + y\sin\theta$

The parameter space must be carefully selected as it impacts both the track reconstruction efficiency and the unbiased extraction of the track parameters

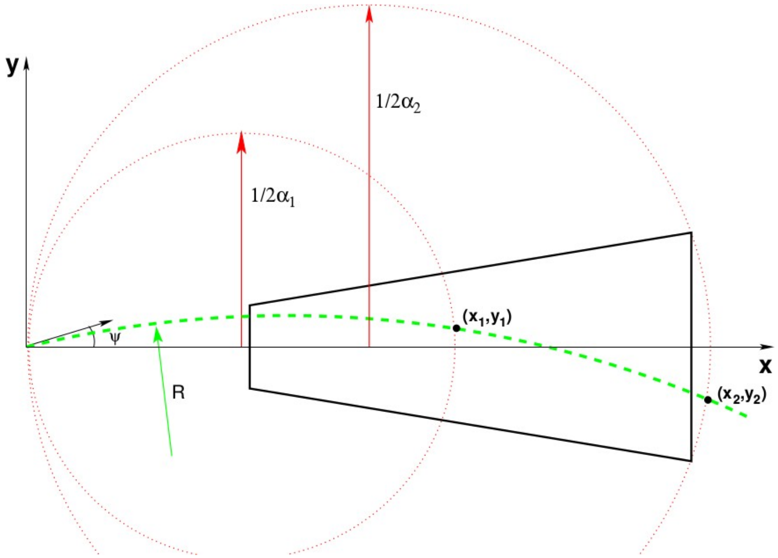
On the other hand, the performance of the algorithm is determined by the complexity of the arithmetic operations

With parametrization, a given cluster coordinate is transformed into a parameter space **curve** which corresponds to all possible track candidates this signal can belong

# The Hough Transform algorithm

One way to define the parameter space is by using an angle-curvature definition

That way, the track trajectory in the transverse plane is represented by the **track emission angle** Ψ and the **track curvature** k = 1 / R



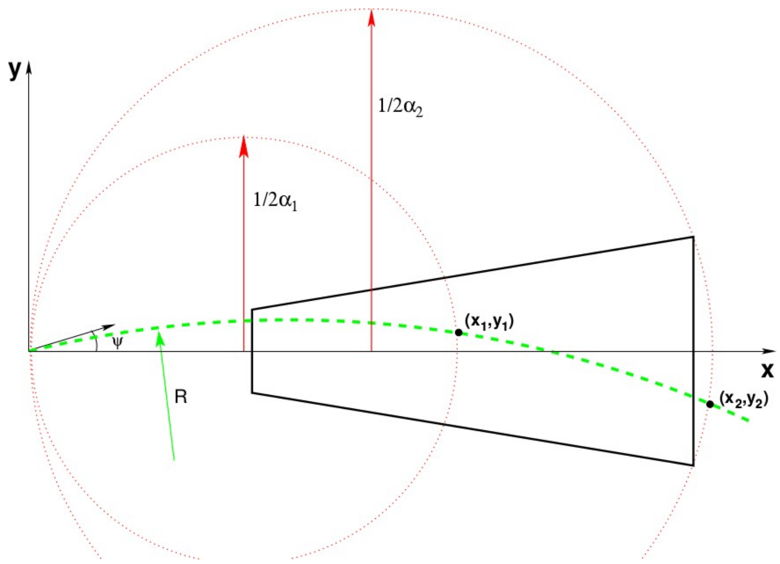

Obvious choice but with serious disadvantages:

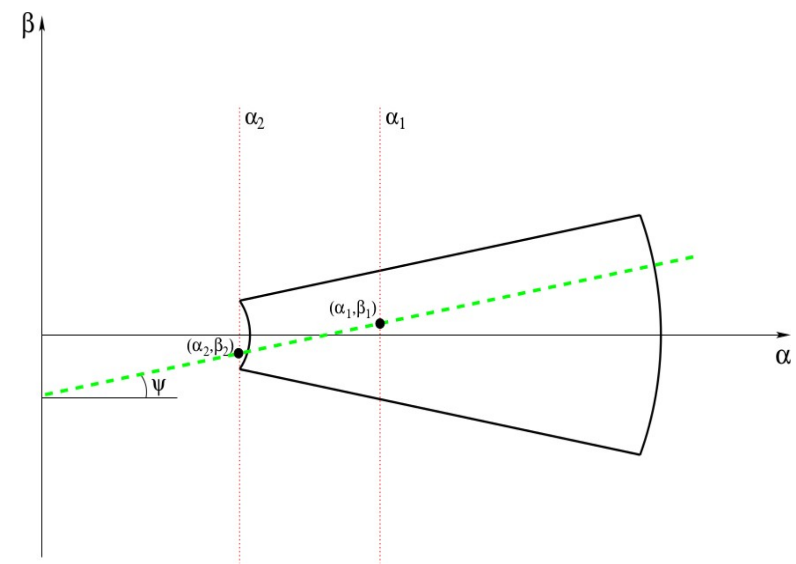
- The sinusoidal form of the track trajectory equation results in expensive floating point operations
- Ψ and k are intrinsically correlated resulting in a relatively high amount of fake tracks
- It would force the use of a slow and complex peak finding algorithm

# The Hough Transform algorithm

The curved track trajectories can be converted to **lines** by transforming the coordinates from the Cartesian system to a Conformal Mapping system

$$a=\frac{x}{x^2+y^2}\quad b=\frac{y}{x^2+y^2}$$



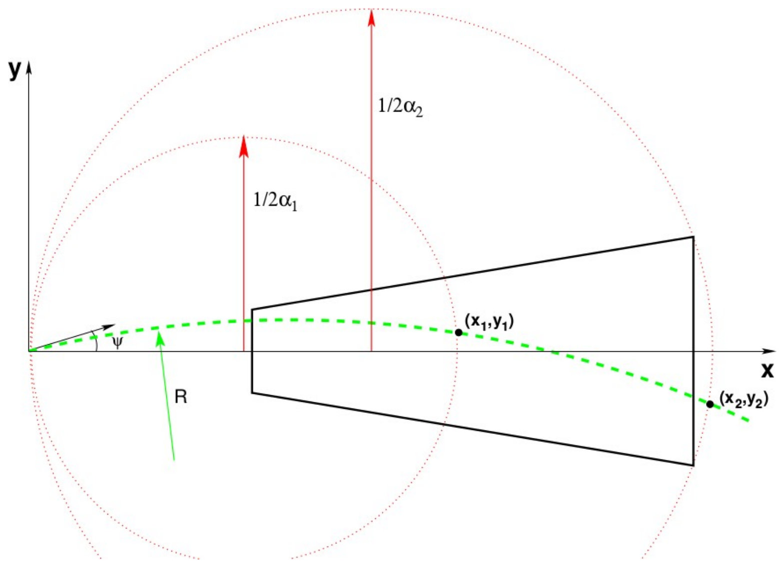



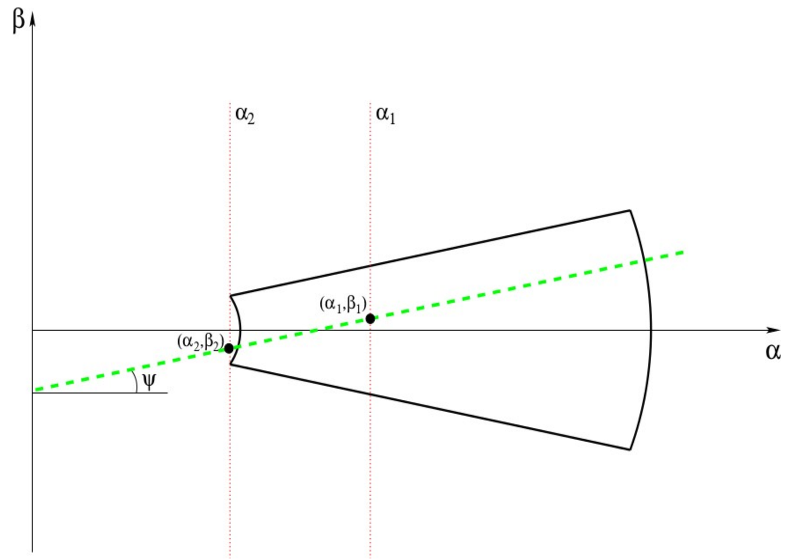
# The Hough Transform algorithm

Two circles -**straight lines** in CM space- can be defined given the constants $\alpha_1$ and $\alpha_2$

$$\alpha 1=\frac{x}{x^2+y^2}\quad \alpha 2=\frac{x}{x^2+y^2}$$

A track crossing the TPC sector can then be represented by two points ($\alpha_1\beta_1$ and $\alpha_2\beta_2$) on these circles



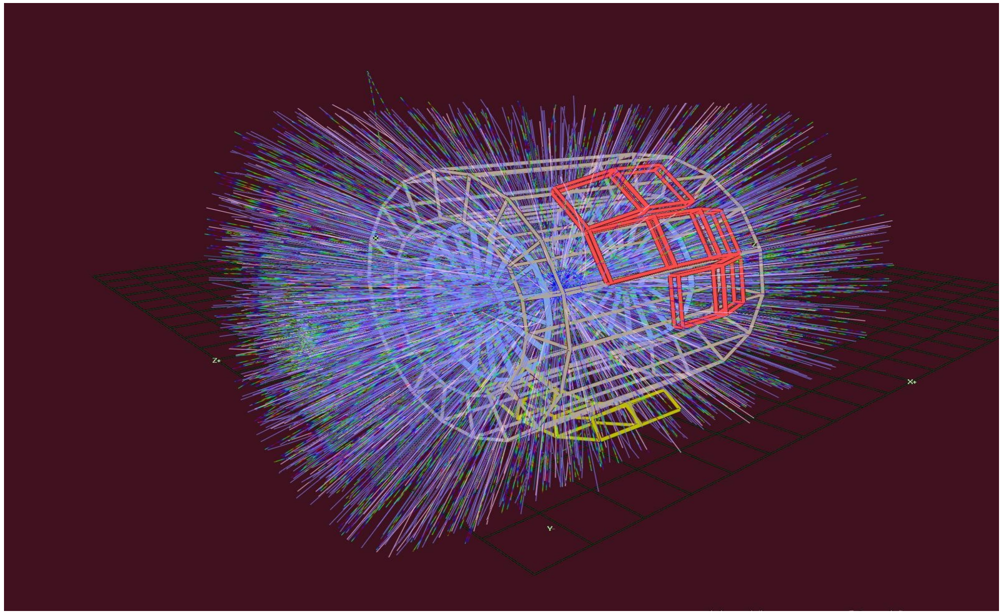



# The Hough Transform algorithm

In practice, the parameter space is discretized and managed as a **histogram**

The value for each histogram bin is increased based on the signal curves crossing it

A **peak finder** is used to locate the peaks in the parameter space and reconstruct the original tracks



# The Hough Transform algorithm

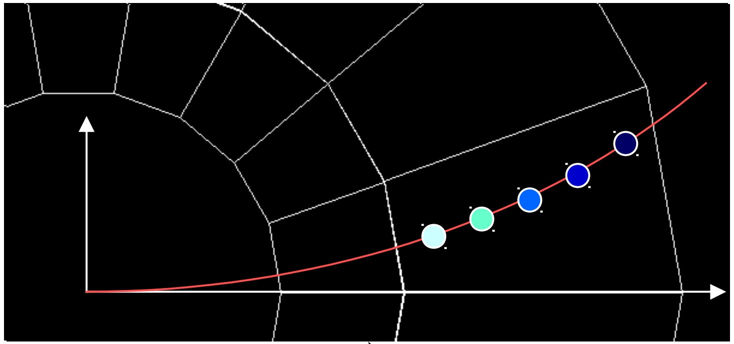


*Image space*
*TPC sector*

# The Hough Transform algorithm

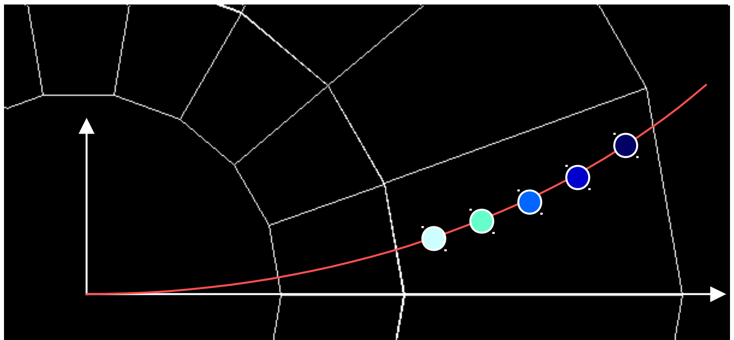
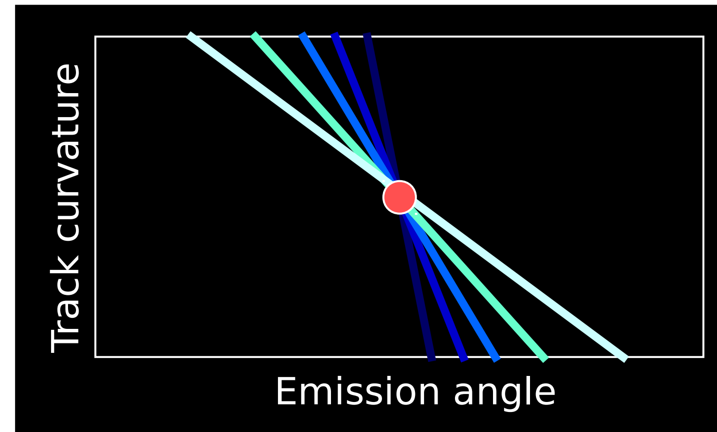
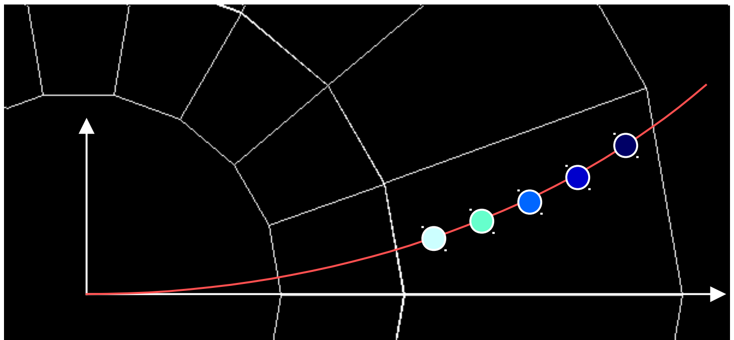


*Image space*
*TPC sector*




*Parameter space*

# The Hough Transform algorithm



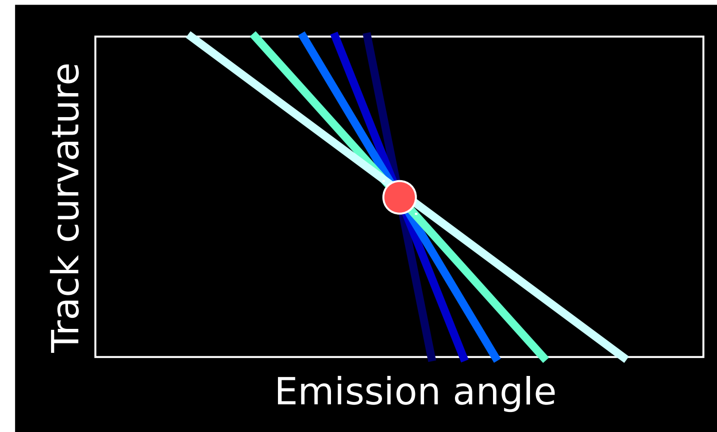
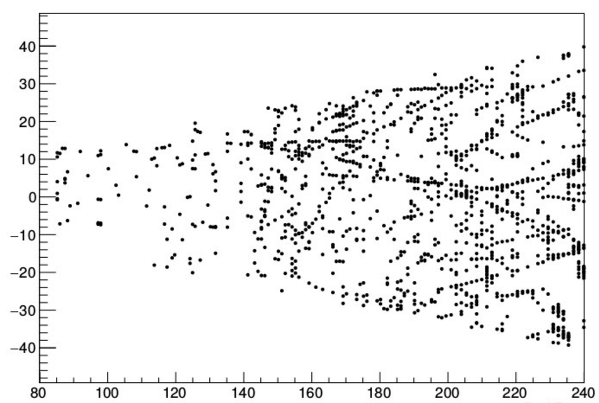
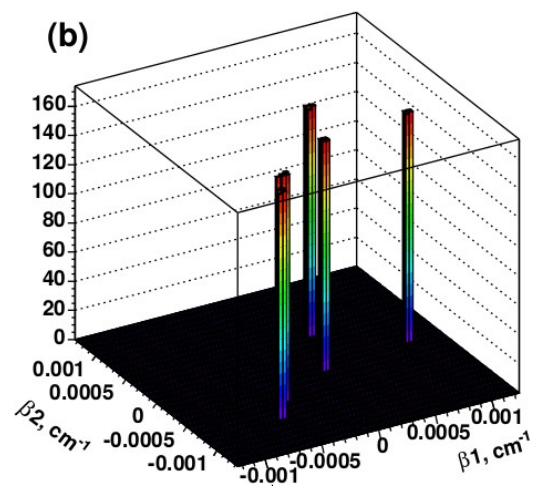



*Image space*
*TPC sector*




*Parameter space*

*A resulting histogram*

# The Hough Transform algorithm

To improve the efficiency of the algorithm, the idea of **gaps** is used*:*

A good track candidate must have not only some amount of cluster charge along its trajectory, but must also pass through consecutive TPC rows

Then the histogram is filled by the following *weight*:

$$\frac{number\ of\ rows}{number\ of\ gaps}$$

The number of gaps for a given track must be below a given threshold

# Algorithm execution

**TPC clusters**

On the FLP:

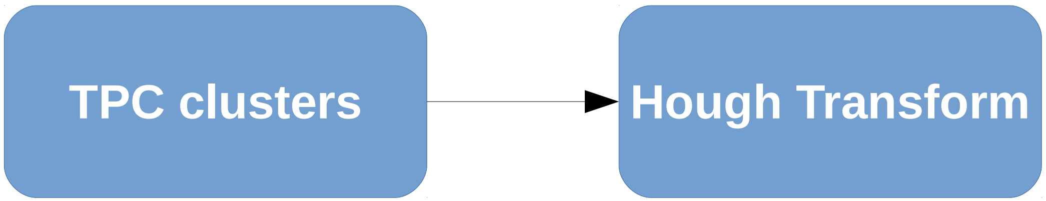
Retrieve cluster information from the raw data of the TPC

Clusters will originate from within the same TPC sector

Due to the nature of the **continuous readout, a time window** will be used to separate different events by splitting cluster data on the **time axis**

There is no guarantee at this stage that the selected clusters will be part of the same event

# Algorithm execution




On the FLP:

The Hough Transform is performed on the clusters

The parameter space is binned and stored to histograms

# Algorithm execution


On the FLP:

Local peaks are identified by the peak finder

The output of the peak finder is a number of **partial** and **candidate** tracks

The tracks can be **partial** if they cross more than one TPC sectors

The tracks can be **rejected** if the clusters exist across multiple events

# Algorithm execution

**TPC clusters** → **Hough Transform** → **Local Peak Finder** → **Track fit**

On the EPN:

Merge the output of the individual Peak Finder executions for a full event

Receive triggering information from ITS and use it to verify or reject **candidate** tracks

Associate the tracks to a collision

# Conclusions

We **develop** a Hough Transform algorithm implementation to perform Track Reconstruction on the TPC for RUN3 as part of $AliceO^2$

We will **evaluate** its performance and efficiency comparing to other alternatives (i.e. Cellular Automata)

The algorithm will be used as a **device** on the FLPs and EPNs

Parallelization opportunities will be used where possible

# Additional Information

**See A. Rybalchenko's presentation on FairMQ**
(Data Transport for Online & Offline Processing, ALICE Offline week, July $1^{st}$, 2015)

**See M. Richter's presentation on the AliceO2 computing facility**
(A design study for the upgraded ALICE $O^2$ computing facility, CHEP 2015)

**The Alice$O^2$ software repository**
**https://github.com/AliceO2Group/AliceO2**

**See M. Al-Turany's presentation on ALFA**
*(ALFA: Next generation concurrent framework for ALICE and FAIR experiments)*

**See C. Cheshkov's presentation on the Hough Transform algorithm**
(*Fast Hough Transform Algorithm - Cvetan Cheshkov - Presentation at HLT Workshop 6-9 June 2004*)