Millones de círculos por segundo - RICH@CERN

Daniel Hugo Cámpora Pérez
dcampora@cern.ch

Seminario (I+A)A, 8 de junio de 2018

Universidad de Sevilla
CERN

Introducción

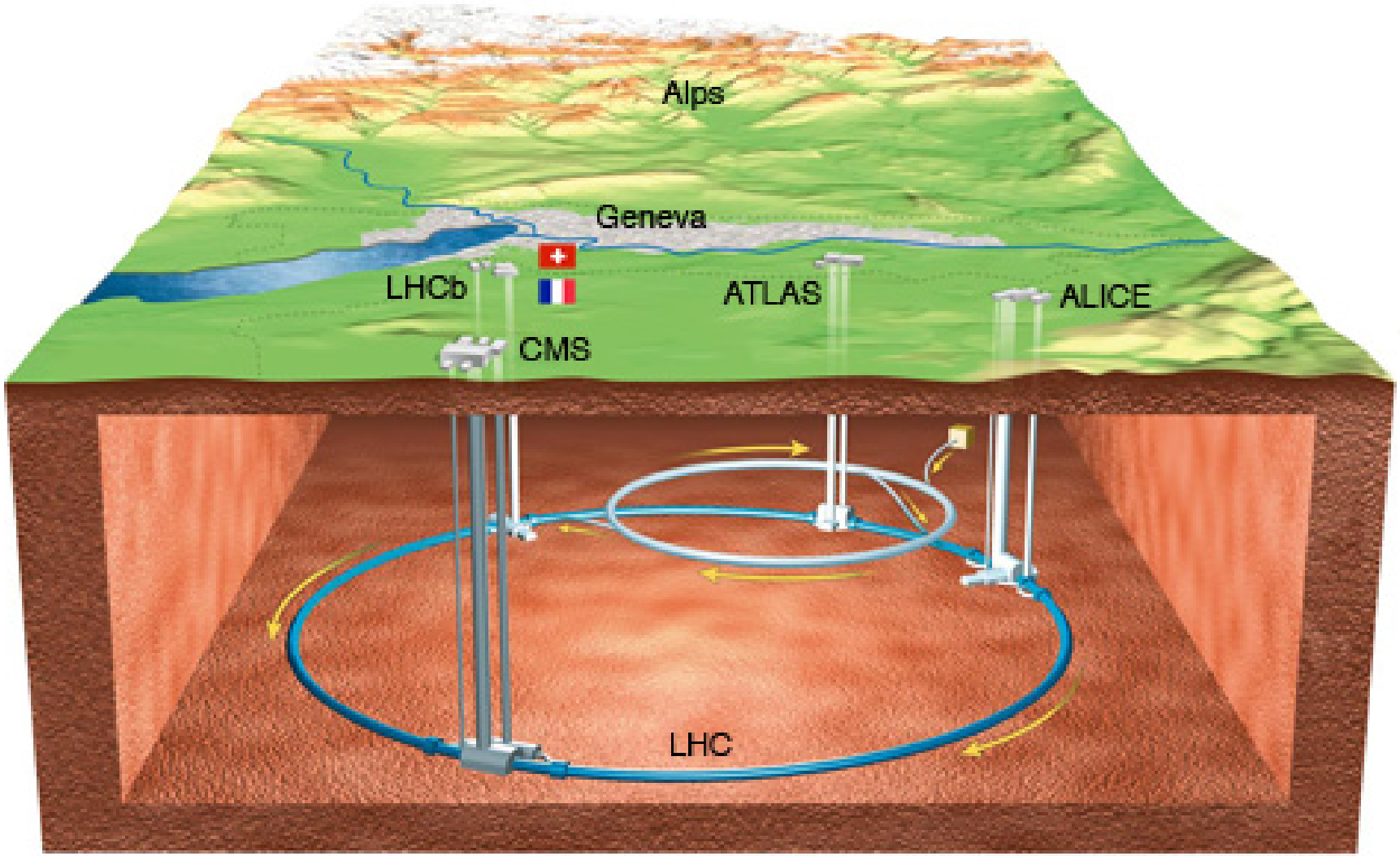
Un poco sobre mí

- Summer Student, Technical Student, Fellow
- Actualmente Doctoral Student @ CERN, con la US

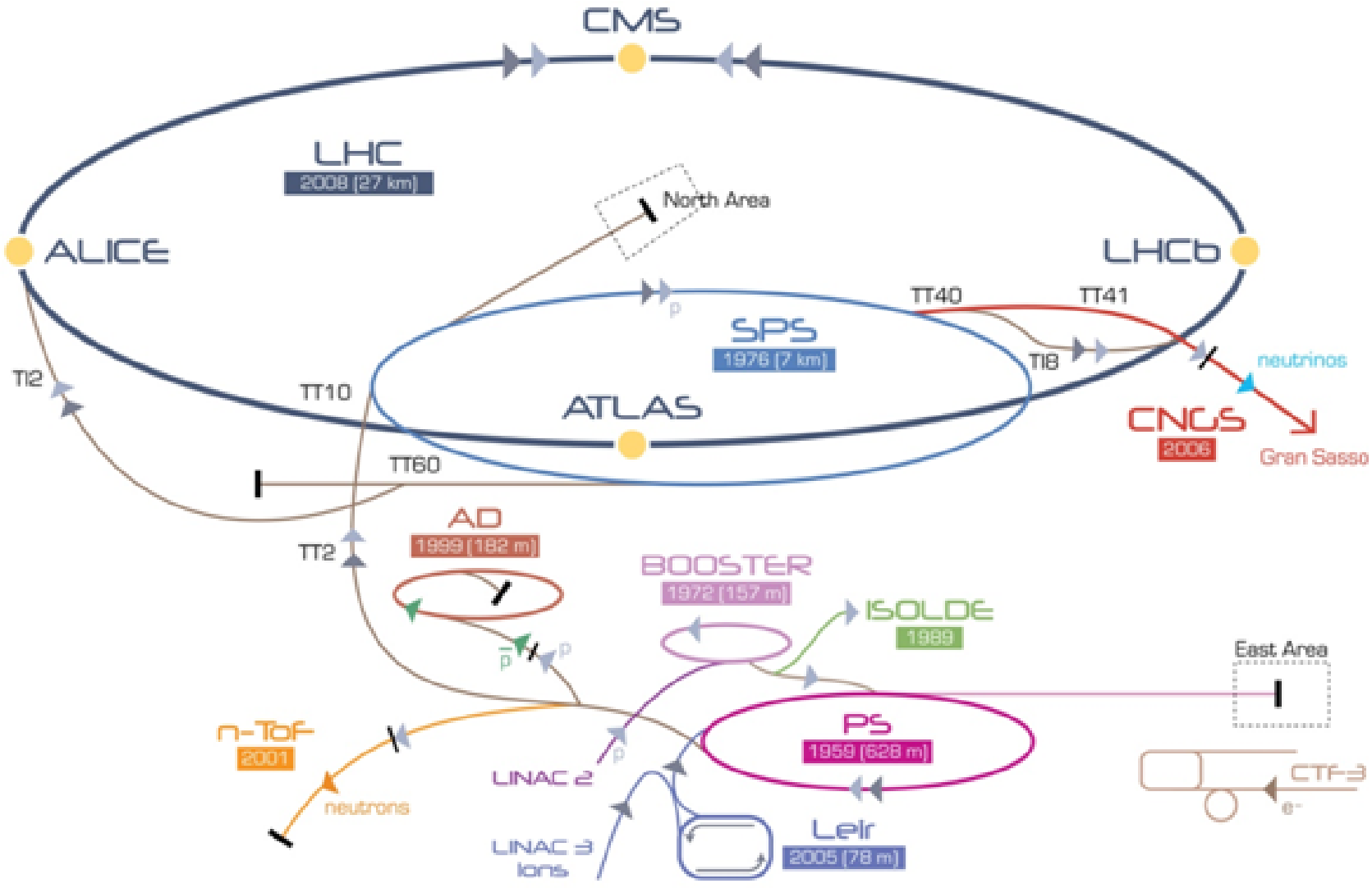
- Data Acquisition
- HPC
- AI

- Cross-architecture Kalman filter
- Full HLT1 reconstruction on GPUs
- RICH reconstruction using CNNs

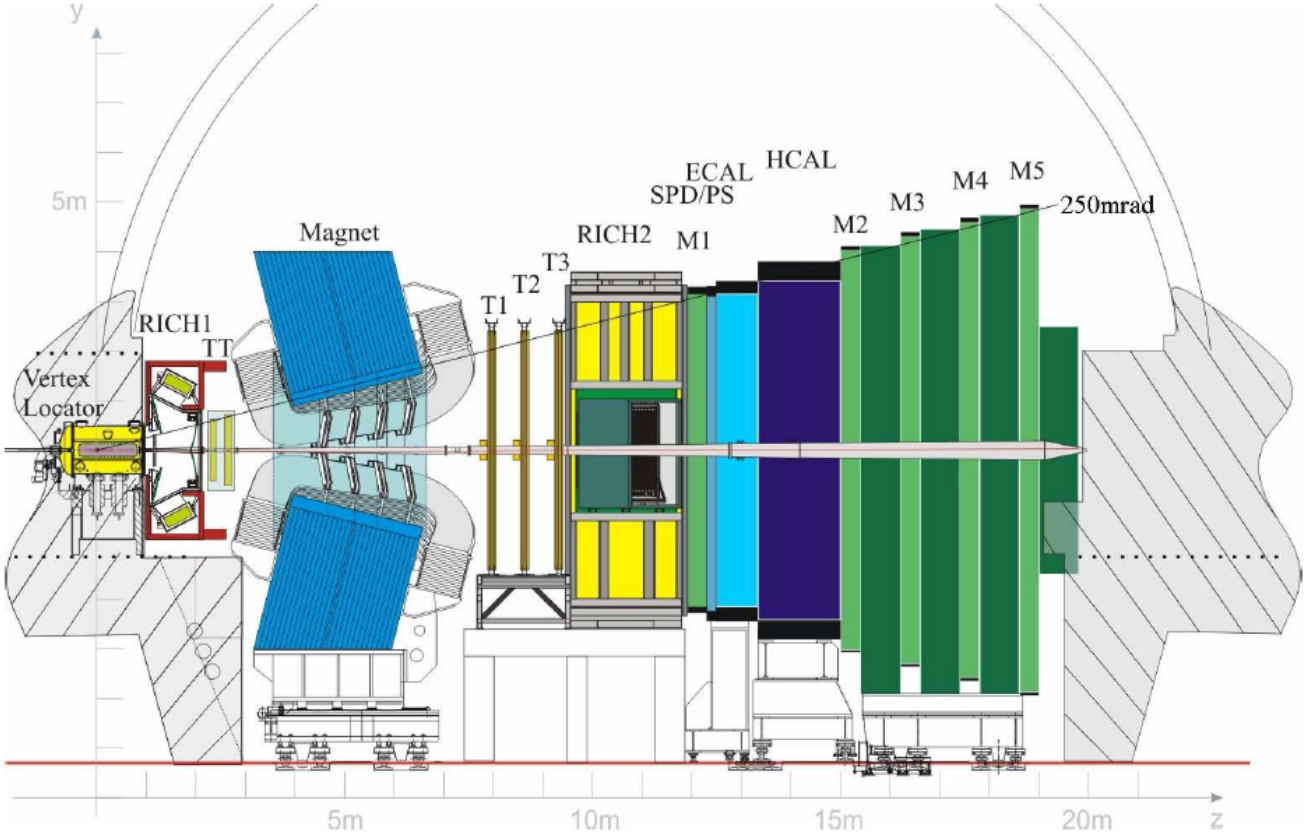
CERN



Un conjunto de aceleradores

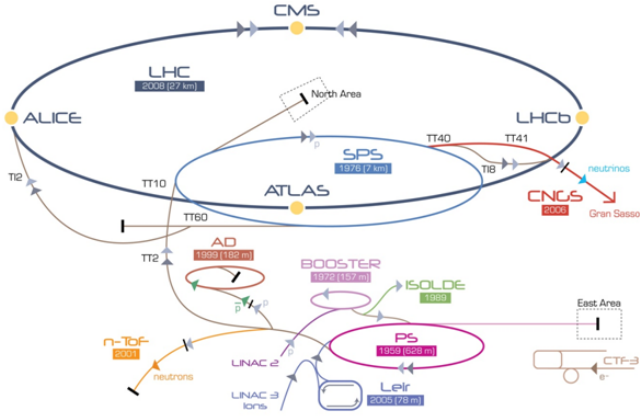


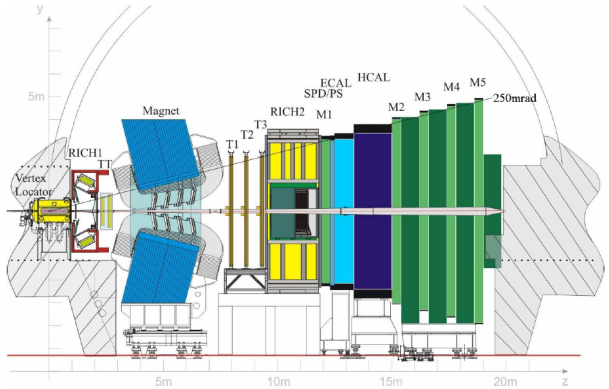
LHCb

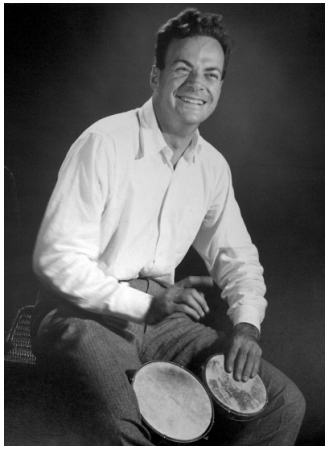


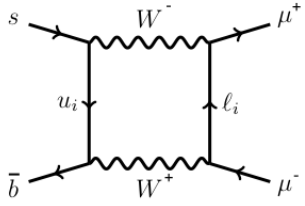
- Detectores de trazas
- Detectores RICH
- Calorímetros

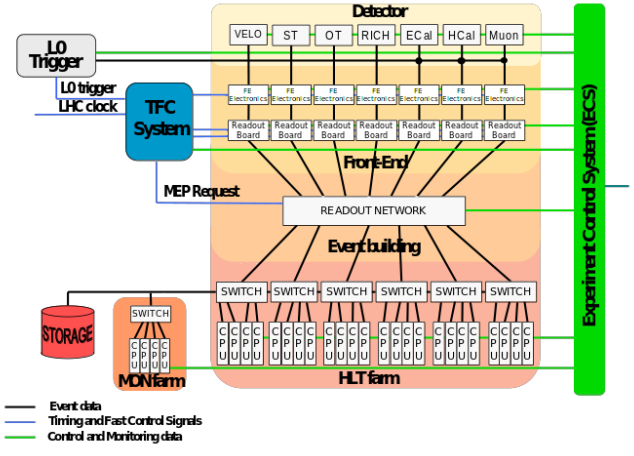
La cadena completa











RICH

Algunas preguntas

¿Cuál es la velocidad de la luz?

Algunas preguntas

¿Cuál es la velocidad de la luz?

$299\,792\ km/s$.

¿En qué medio?

Algunas preguntas

¿Cuál es la velocidad de la luz?

$299\,792\ km/s$.

¿En qué medio?

En el vacío.

¿Qué le pasa a la luz cuando entra en el agua?

Algunas preguntas

¿Cuál es la velocidad de la luz?

$299\,792\ km/s$.

¿En qué medio?

En el vacío.

¿Qué le pasa a la luz cuando entra en el agua?

Se refracta.

¿Por qué se refracta?

Algunas preguntas

¿Cuál es la velocidad de la luz?

299 792 km/s.

¿En qué medio?

En el vacío.

¿Qué le pasa a la luz cuando entra en el agua?

Se refracta.

¿Por qué se refracta?

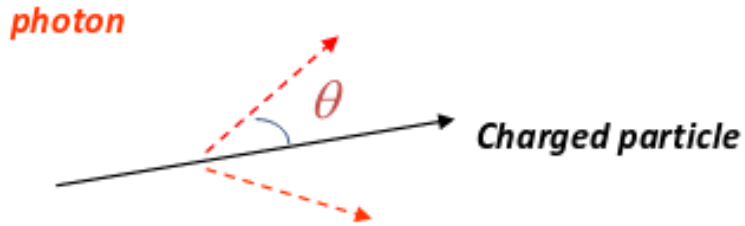
Porque cambia su velocidad.

Algunas preguntas (2)

¿Qué pasa si una partícula va más rápido que la velocidad de la luz?

La radiación de Cherenkov

La radiación de Cherenkov es un tipo de radiación electromagnética producida cuando una partícula va más rápido que la velocidad de la luz.



$\cos(\theta) = 1/(n\,\beta)$ where n = Refractive Index = $c/c_M = n(E_{ph})$

$$\beta = v/c = p/E = p/(p^2 + m^2)^{0.5} = 1/(1+(m/p)^2)^{0.5}$$

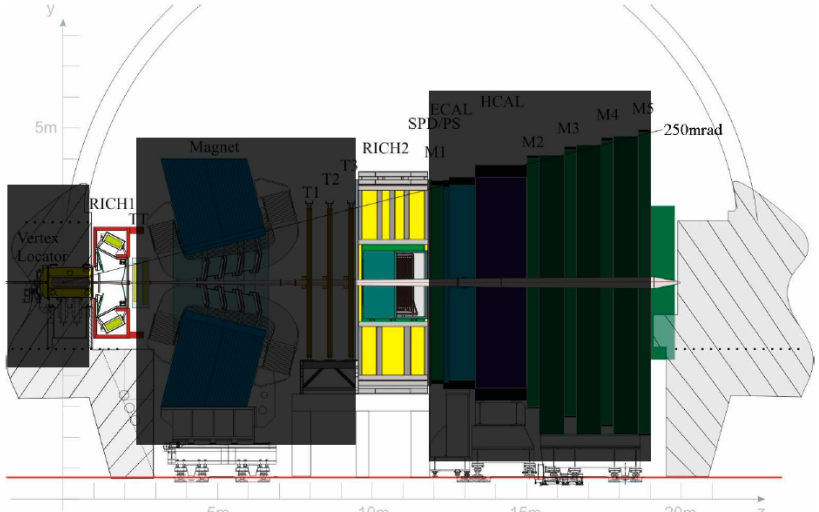
β= velocity of the charged particle in units of speed of light (c) vacuum
p, E ,m = momentum, Energy, mass of the charged particle.
C_M = Speed of light in the Medium (Phase velocity) ,
E_{ph} = Photon Energy, λ=Photon Wavelngth.

- Theory of Cherenkov Radiation: Classical Electrodynamics by J.D.Jackson (Section 13.5)

La radiación de Cherenkov (2)

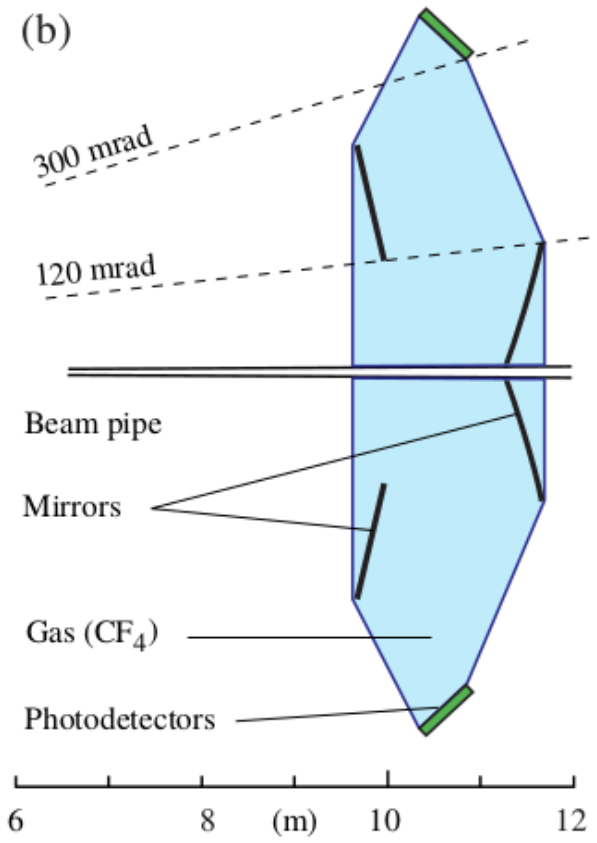
En concreto, el ángulo de los fotones producidos depende de la masa y la cantidad de movimiento de la partícula, con lo que al detectar el diámetro de los círculos, estaremos más cerca de detectar qué partícula lo originó (protón, kaón, pion, electrón, muon o deuterón).

El cono de luz generado por la radiación de Cherenkov se detecta mediante detectores RICH (Ring Imaging Cherenkov detectors). En LHCb:



Formulación del problema

Dadas unos tracks de partículas, unos fotones y una geometría del detector, determinar la hipótesis más probable para cada partícula de entre: Pion, electrón, muon, kaón, protón y deuterón.



Visualmente

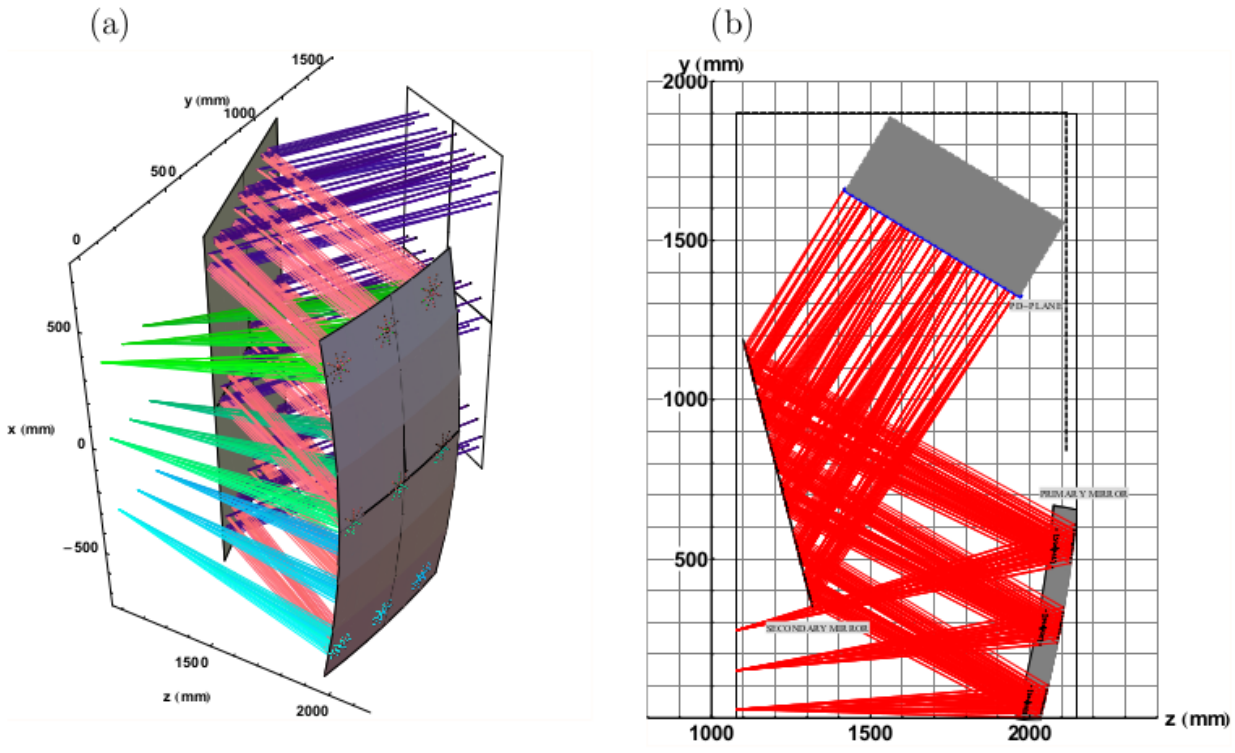
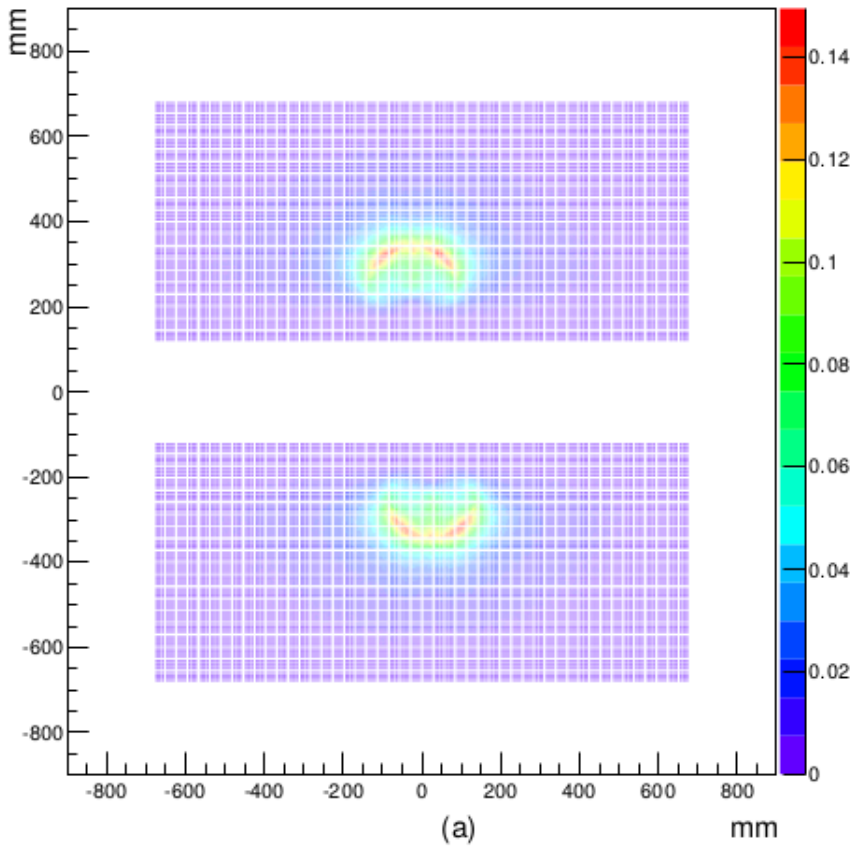
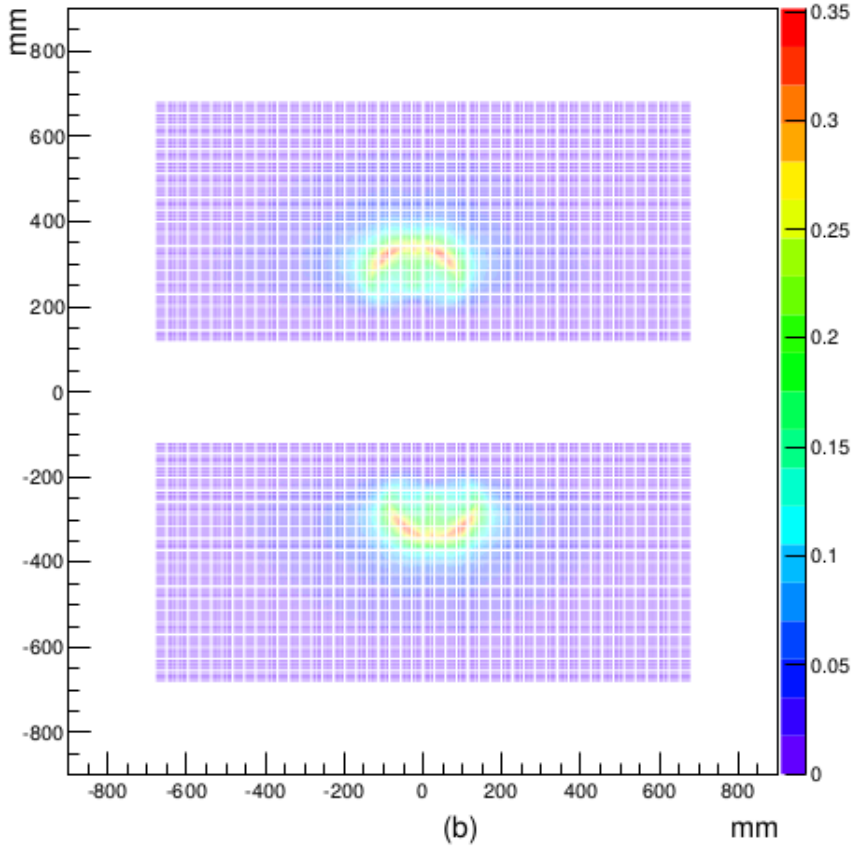


Figure 2.21: Simulated Cherenkov photons in the upper half of the upgraded RICH 1; (a) 3D view, (b) 2D view in the vertical plane.

Visualmente (2)





¿Cómo resolver el problema?

- Solución analítica
- Reconstrucción de círculos
- Elastic Neural Network

Solución analítica

La solución analítica consiste en reconstruir las posibles trayectorias de los fotones para cada partícula, una a una, y asignar una probabilidad (likelihood) a la asignación. Después, se cambia la asignación track-partícula, maximizando la probabilidad.

Es decir, se compone de dos partes:

- Reconstrucción de fotones
- Maximización de probabilidad

Reconstrucción de fotones

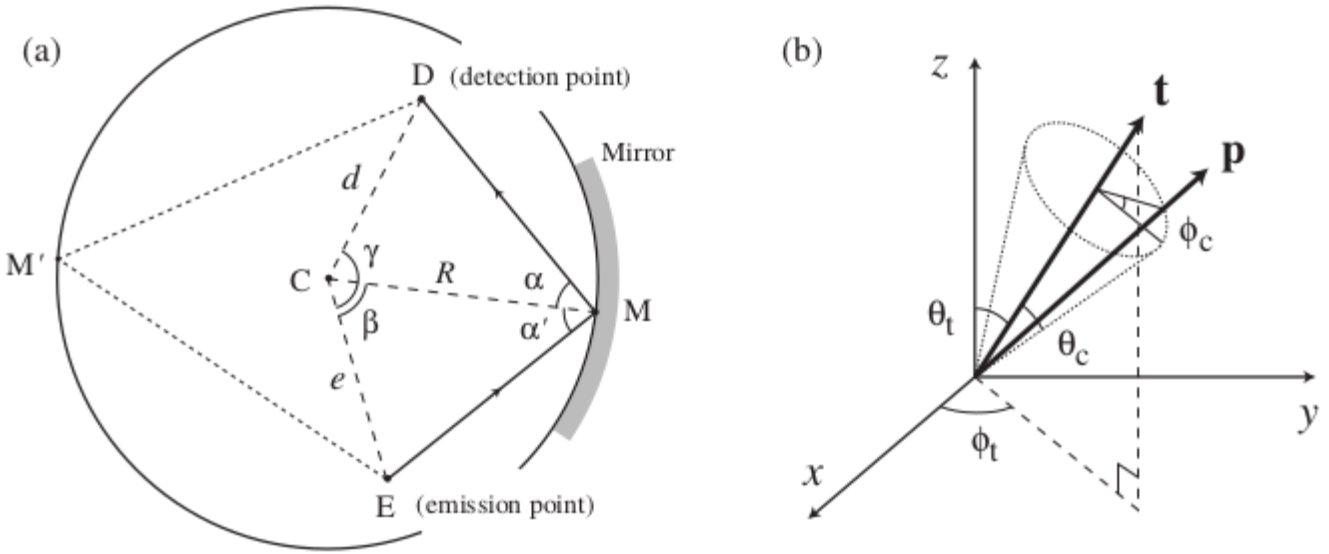


Figure 3: (a) Definition of parameters used in the reconstruction of the Cherenkov angles, for a photon emitted from point E, reflected in a spherical mirror at M and detected at D. (b) Definition of the angles describing the direction of a track **t** in the lab frame, and a photon **p** emitted by the track.

Maximización de probabilidad

Para cada track t_i, $i \in 0, ..., n$, se debe probar con las hipótesis $\{$pion, electrón, muon, kaón, protón, deuterón$\}$. Existen 6^n posibilidades.

No existe una heurística clara, así que se comienza con la hipótesis más probable para todas las partículas (pión), y se prueban partícula a partícula las otras cuatro posibilidades, asignando aquella que maximice la probabilidad.

Esto reduce la búsqueda a una búsqueda local, testeando únicamente $6n$ posibilidades.

Problemas de esta solución

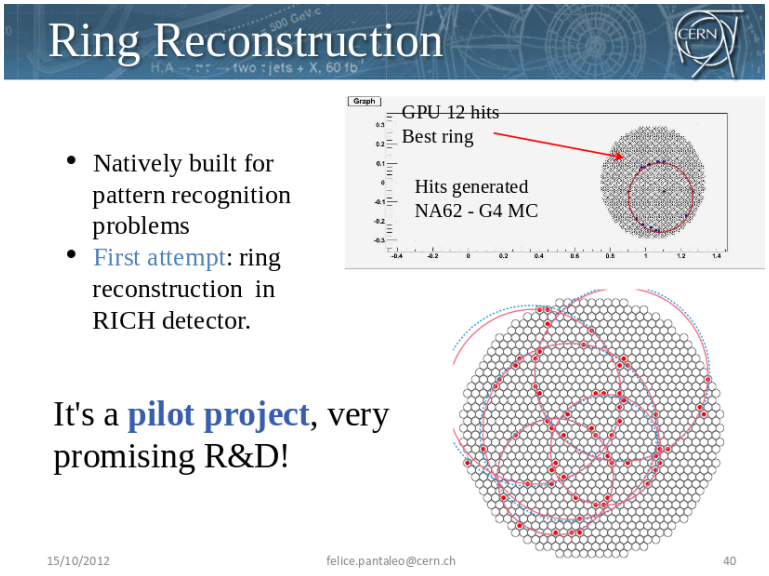
Tanto la reconstrucción de fotones como la maximización de probabilidad son procesos costosos. En particular, para cada fotón, hay que calcular el reflejo en un espejo esférico, lo que implica resolver la ecuación:

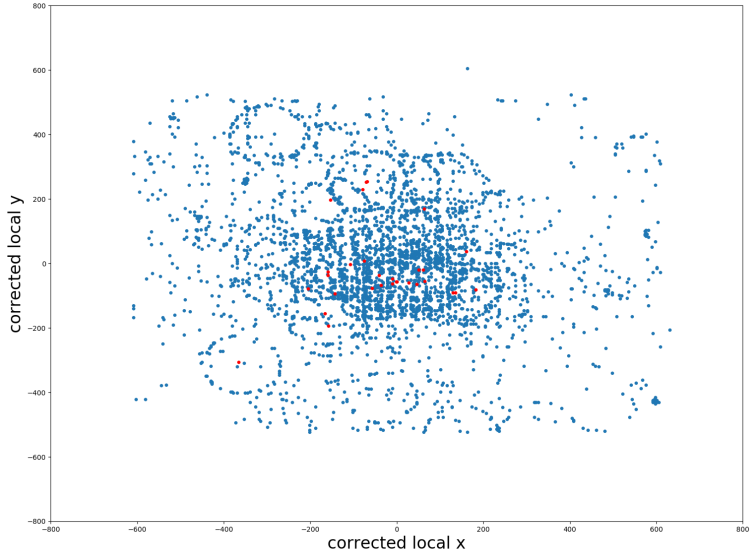
$$4e^2d^2\sin^4\beta \;-\; 4e^2d_yR\sin^3\beta \;+\; \left(d_y^2R^2+(e+d_x)^2R^2-4e^2d^2\right)\sin^2\beta$$
$$+\; 2ed_y(e-d_x)R\sin\beta \;+\; (e^2-R^2)d_y^2 \;=\; 0\,. \qquad (3)$$

This can be solved, using for example a routine in the CERN library [6], and gives four solutions for $\sin\beta$, two complex and two real. Of the real solutions one is the "backward" reflection (that would exist if the mirror were a complete sphere, shown as M′ in Fig. 3); the other is the desired solution, and can be selected from knowledge of the RICH detector geometry. The value of $\cos\beta$ can then be extracted using Eq. 2, and the coordinates of the reflection point M determined.

Otras técnicas: Reconstrucción de círculos

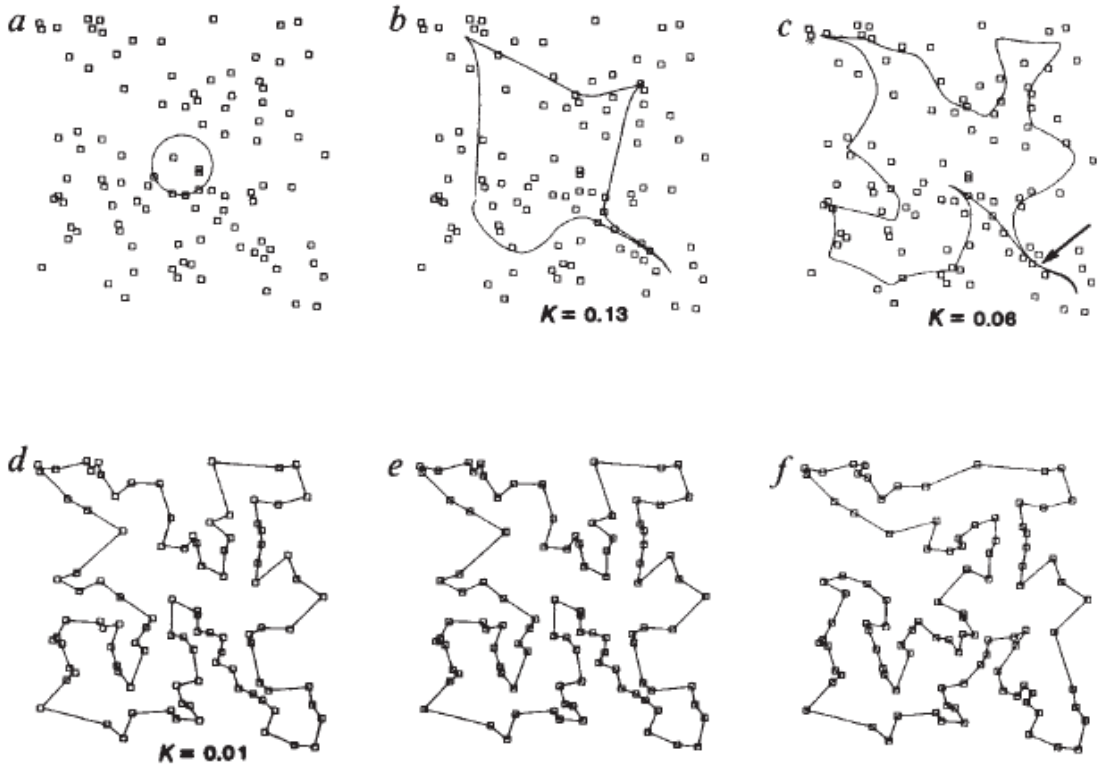
Una idea completamente diferente sería partir de la imagen con los fotones, y tratar de reconstruir círculos. Esto funciona en otros detectores, pero en LHCb *la cosa cambia*:





Otras técnicas: Elastic Neural Networks

En 1987, se publicó una heurística para el problema del viajante de comercio (TSP), consistente en empezar con una figura, y modelar cada ciudad como un punto que ejerce una fuerza atrayente sobre la figura, deformándola.



Elastic Neural Networks

Basándose en esta idea, el problema de encontrar círculos en el RICH puede también partir de una figura que se pueda deformar, permitiendo solo círculos y solo que crezca.

At the recognition step a special energy function is used to separate ring hits from others.

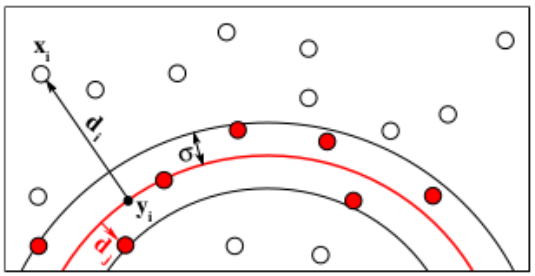


Figure 3: Selection of ring hits at the recognition step

Let us define the following variables (figure 3):
$\mathbf{x}_i$ — the i-th hit;
$\mathbf{y}_i$ — the elastic net point closest to the i-th hit;
$\mathbf{d}_i = (\mathbf{x}_i - \mathbf{y}_i)$ — distance between the hit and the net.

With these variables:

- the external forces are defined by minimizing the sum of (not squared) distances to hits

$$E = \sum |\mathbf{d}_i| , \tag{4}$$

¿Está toda esperanza perdida?

- Solución analítica
- Reconstrucción de círculos
- Elastic Neural Network
- ...


Episode IV
A NEW HOPE

Una nueva idea

Empecemos con los datos iniciales. Tenemos:

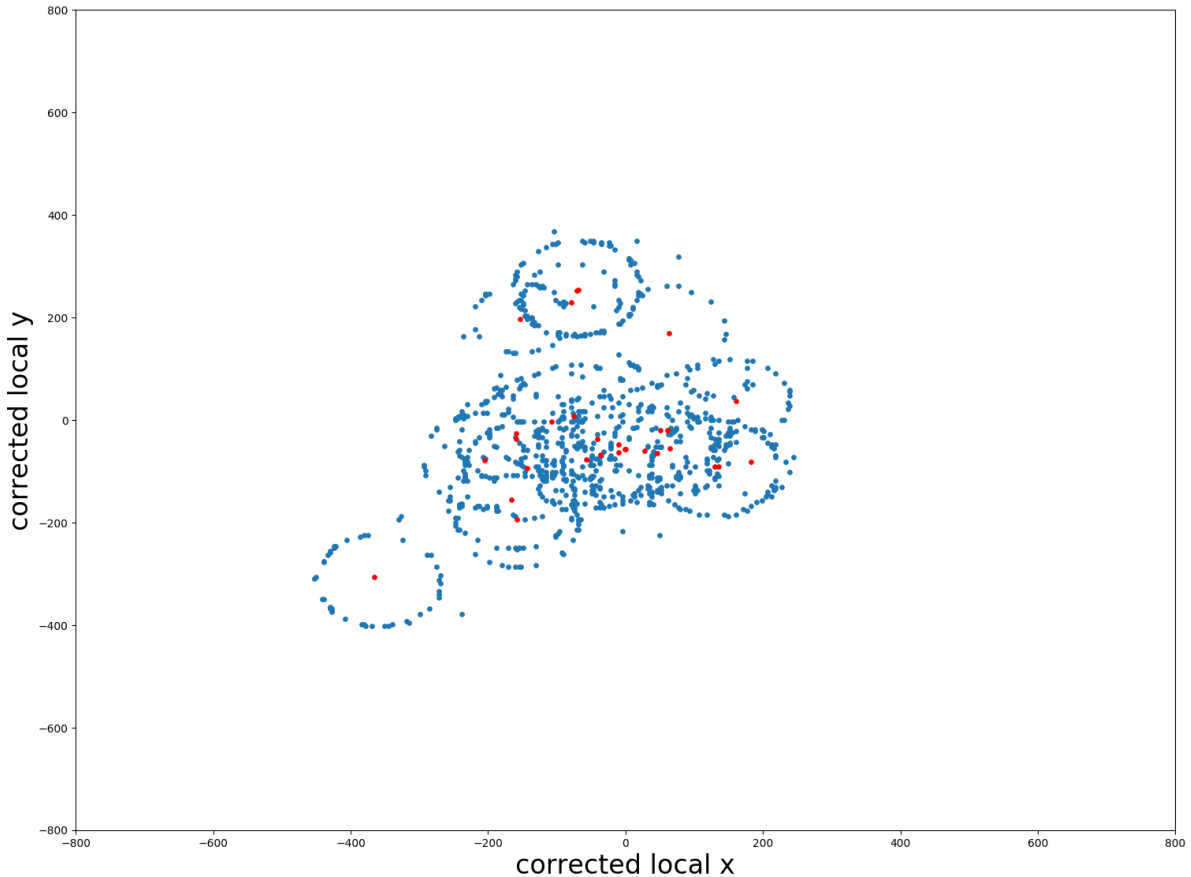
- Píxeles activos en una matriz bidimensional
- Trayectorias de las partículas cuya naturaleza queremos identificar
- Momento de las partículas

Use the ~~force~~ velocity and momentum

$$p = mv$$

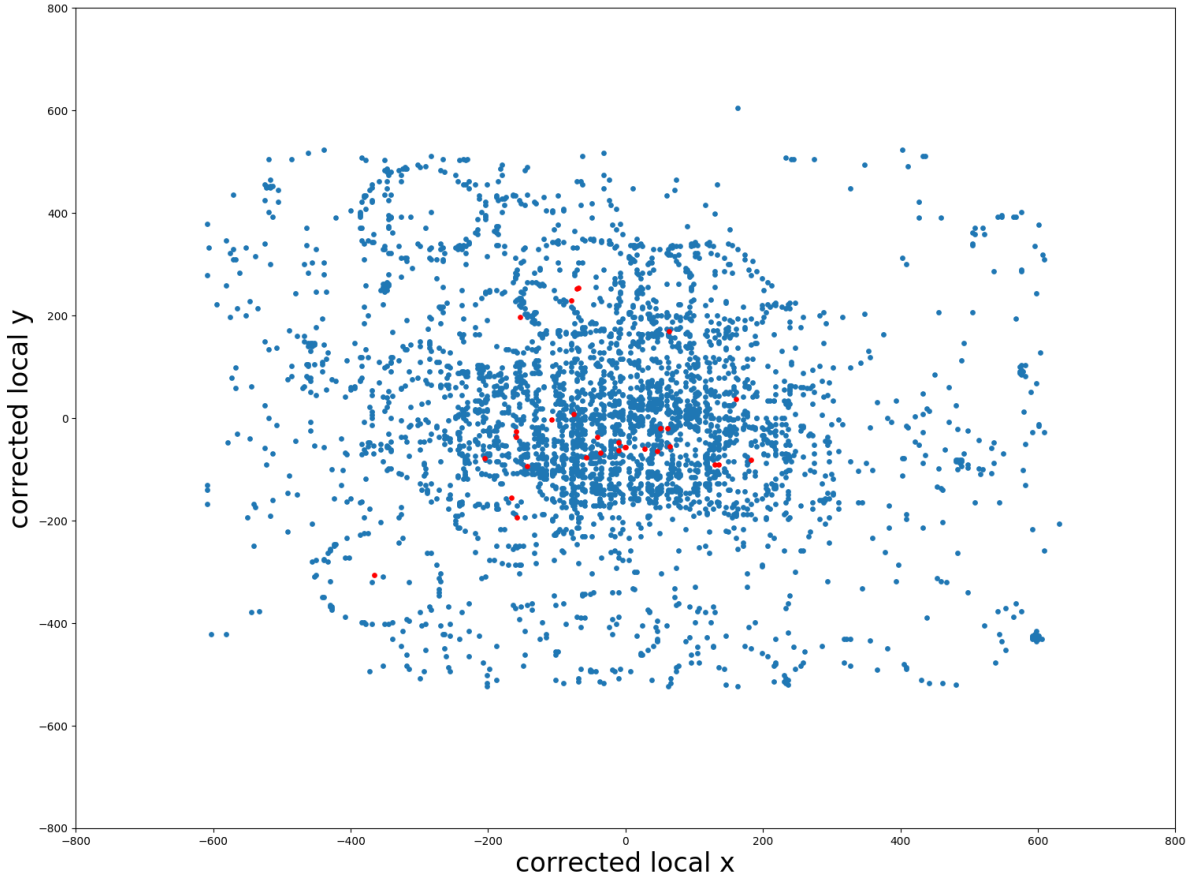
En detalle

Podemos extrapolar las trayectorias de las partículas en el plano de detección. En detalle, estamos buscando los círculos asociados a cada una de ellas:



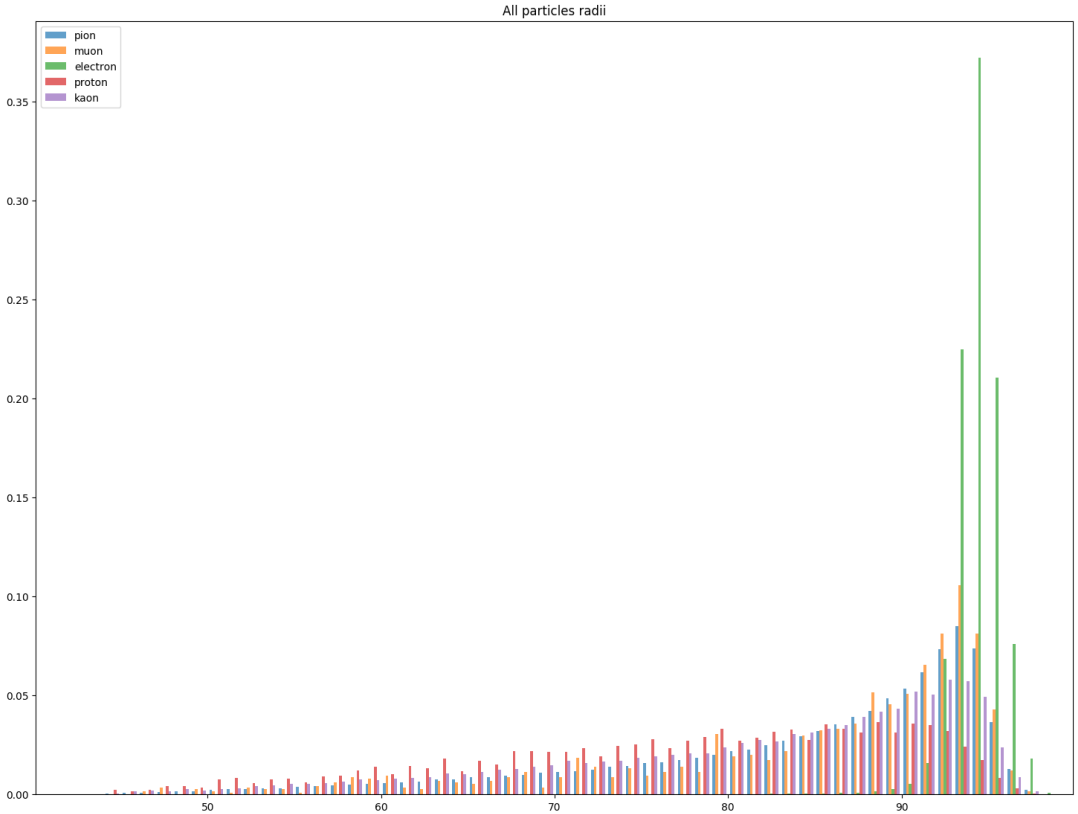
No es tan fácil

Aunque claro, tenemos ruido también. Así que no es tan fácil.



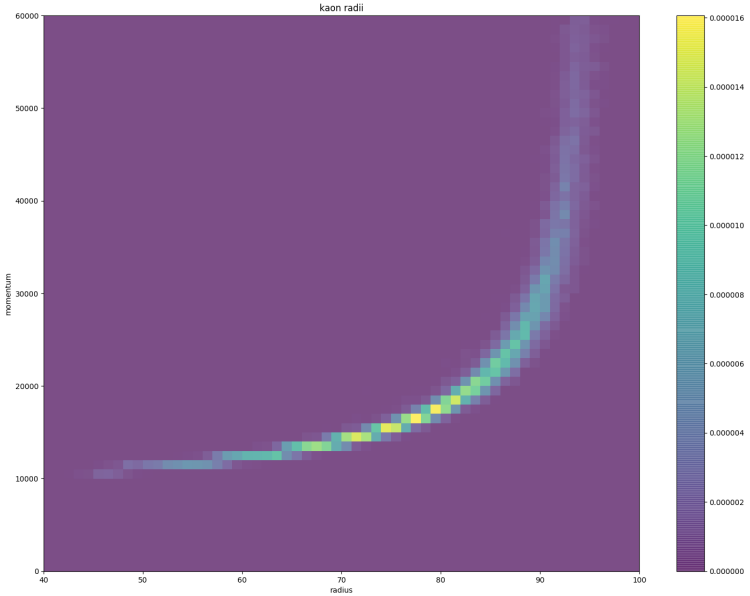
¿Quién es quién?

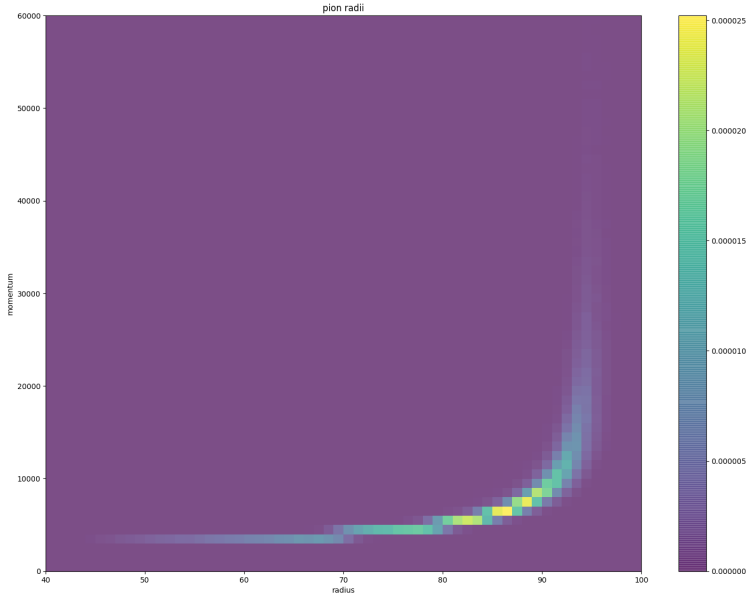
Si intentamos identificar una partícula únicamente por el radio del círculo, fallaremos estrepitosamente



Kaon y pion

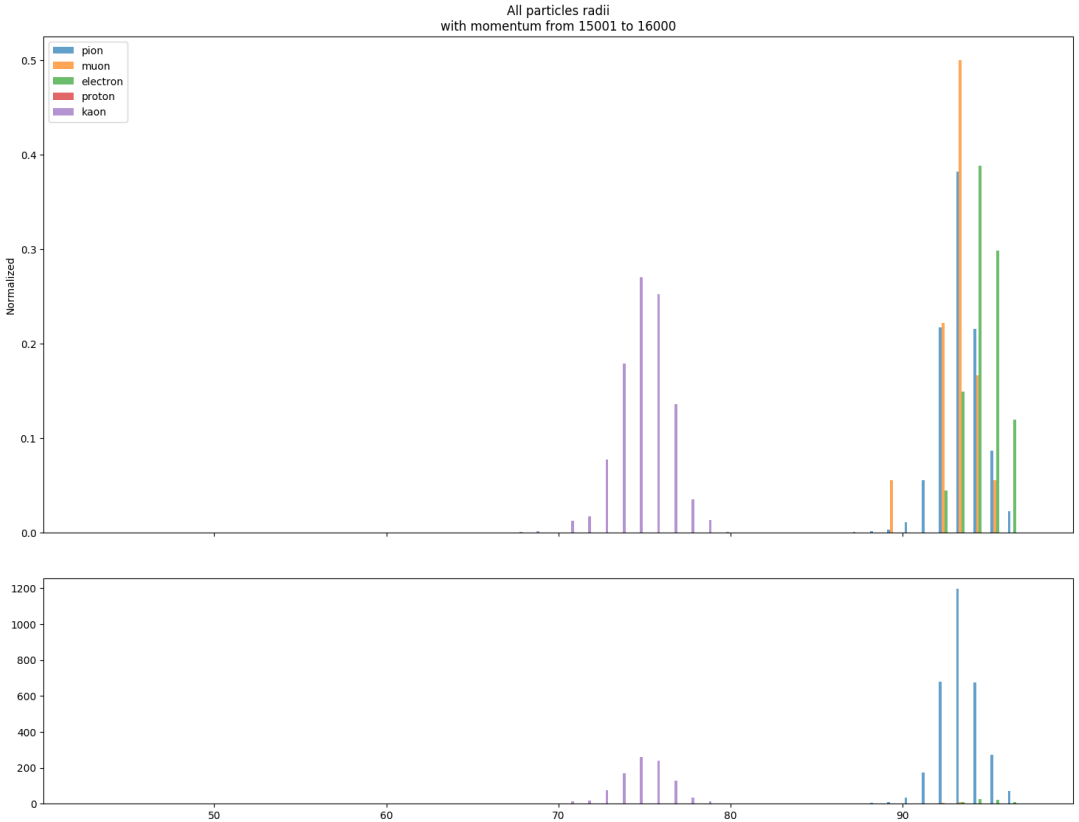
El rango de posibles radios varía en función del momento





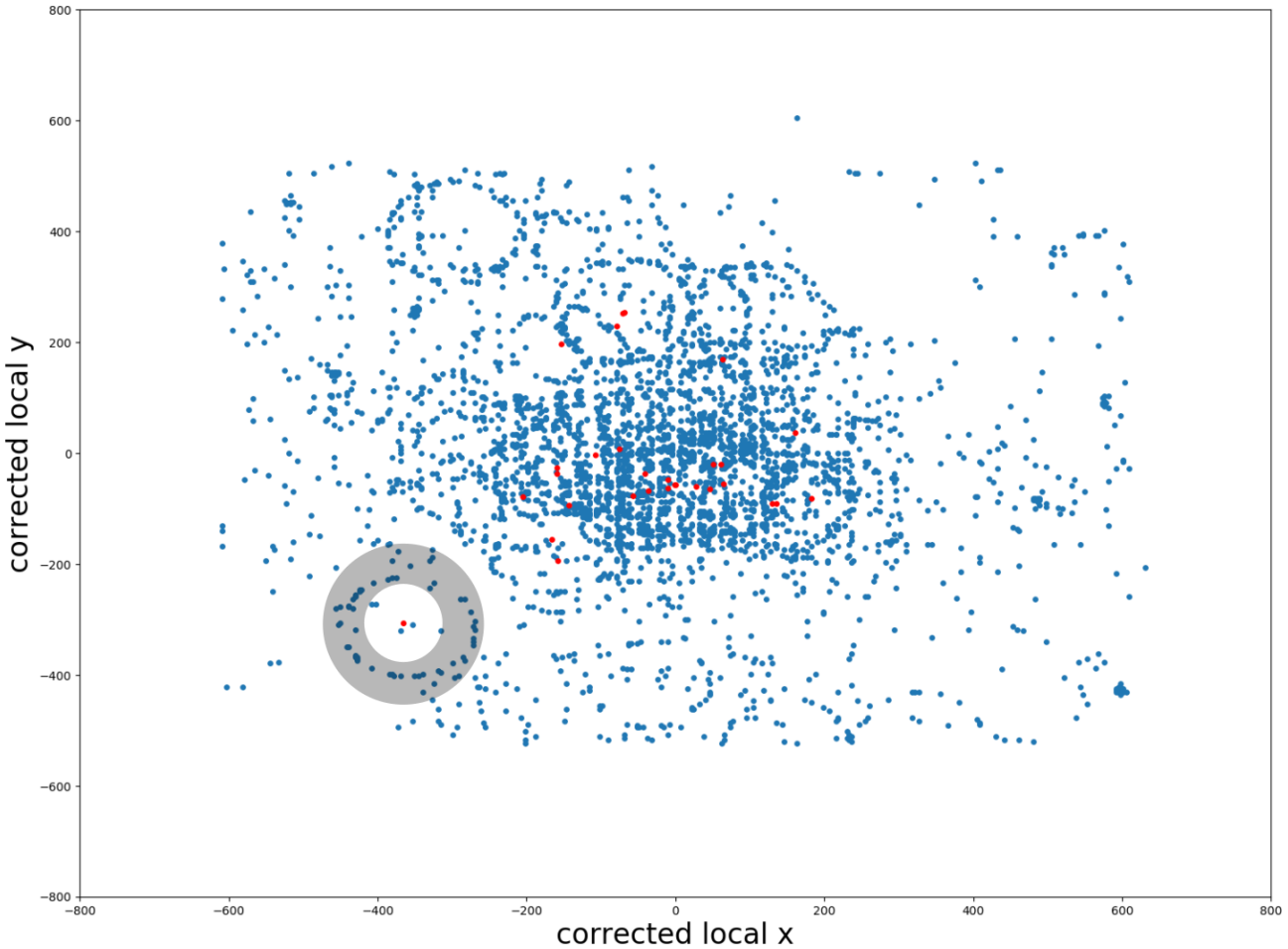
Rangos de momentos

Si nos centramos en un rango de momento concreto, el radio es suficiente para identificar a las partículas



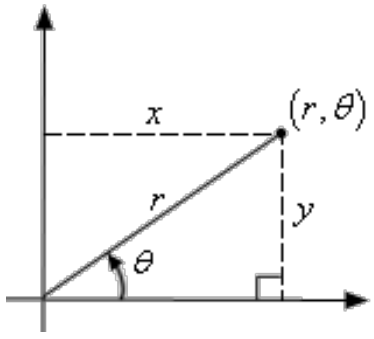
Dividiendo el problema: Una partícula cada vez

Podemos centrarnos en partículas individuales, y tomar la corona del rango de radio que nos interese



Polares

Recordando la transformación a polares



podemos entonces convertir esa imagen en algo más fácil de tratar, y normalizado:

Un problema de clasificación

Finalmente, tenemos un problema de clasificación, de una imagen binaria de 32x32

Figure 1: Izquierda: Pion. Derecha: Electron.

en seis posibles partículas.

Probando, probando...

Este problema se parece relativamente bastante al MNIST. Recordemos que el MNIST consiste en clasificar imágenes de números escritos a mano por carteros del 0 al 9.

Las imágenes en este caso son también binarias, de 28x28 píxeles

Ongoing research

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_28 (Conv2D)           (None, 30, 30, 32)        320
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_23 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
flatten_13 (Flatten)         (None, 12544)             0
_________________________________________________________________
dense_24 (Dense)             (None, 128)               1605760
_________________________________________________________________
dropout_24 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_25 (Dense)             (None, 6)                 774
=================================================================
Total params: 1,625,350
Trainable params: 1,625,350
Non-trainable params: 0
_________________________________________________________________
```

Looking good

Particle type	# particles	Accuracy	Loss
electron	76	0.013158	5.977299
kaon	540	0.557407	1.147596
muon	12	0.000000	12.524119
pion	2058	0.965015	0.140477
proton	299	0.304348	2.457402

Predictions (%tot)	electron	kaon	muon	pion	proton	Purity (%)
electron	0.034	0.034	0.000	2.412	0.067	1.316
kaon	0.000	10.084	0.000	5.829	2.178	55.741
muon	0.000	0.000	0.000	0.369	0.034	0.000
pion	0.034	1.374	0.000	66.533	1.005	96.501
proton	0.000	2.647	0.000	4.322	3.049	30.435
Efficiency (%)	50.000	71.327	0.000	83.727	48.148	

ID eff (%)	K->K,Pr,D :	90.047	pi->e,m,pi :	87.226
MisID eff (%)	K->e,m,pi :	9.953	pi->K,Pr,D :	12.774

Tracking

Una imagen vale más que mil palabras

Concurso de Kaggle

kaggle Search kaggle Competitions Datasets Kernels Discussion Learn ••• Sign In

Featured Prediction Competition

TrackML Particle Tracking Challenge

High Energy Physics particle tracking in CERN detectors

$25,000
Prize Money

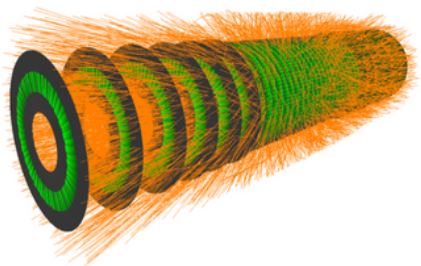
CERN · 381 teams · 2 months to go (2 months to go until merger deadline)

Overview Data Kernels Discussion Leaderboard Rules

Overview

Description
Evaluation
Prizes
About The Sponsors
Timeline

To explore what our universe is made of, scientists at CERN are colliding protons, essentially recreating mini big bangs, and meticulously observing these collisions with intricate silicon detectors.

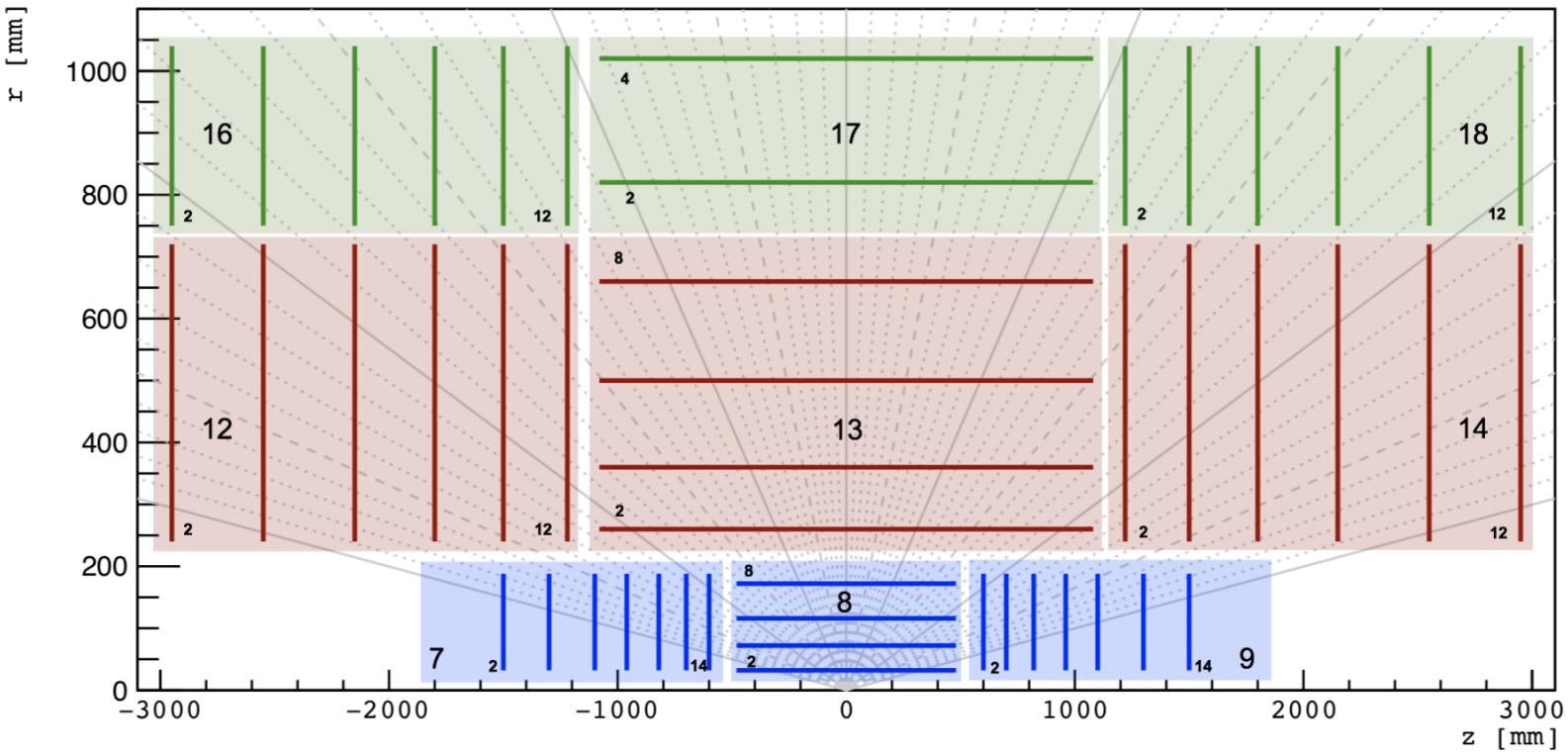


While orchestrating the collisions and observations is already a massive scientific accomplishment, analyzing the enormous amounts of data produced from the experiments is becoming an overwhelming challenge.

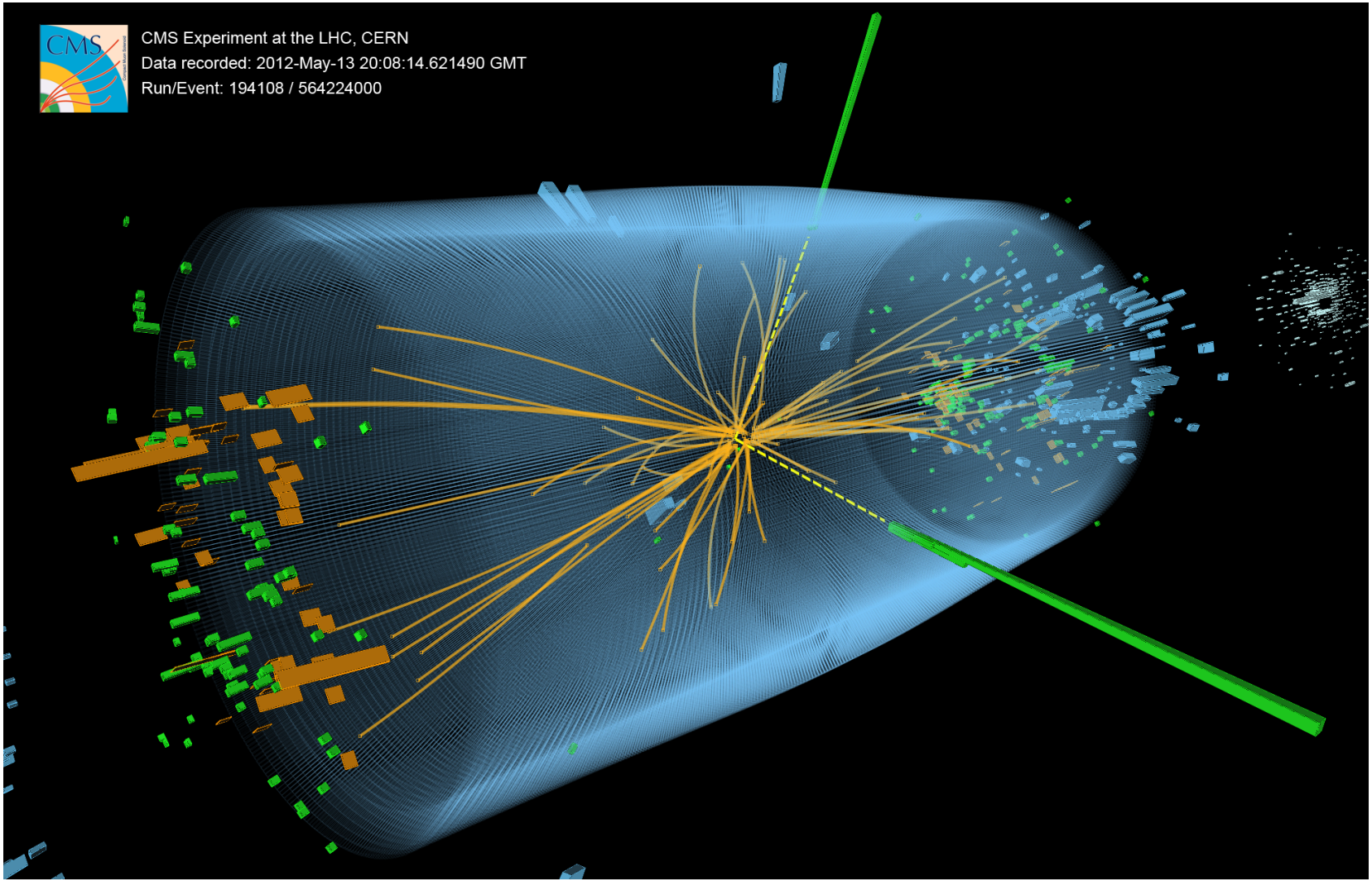
Event rates have already reached hundreds of millions of collisions per second, meaning physicists must sift through tens of petabytes of data per year. And, as the resolution of detectors improve, ever better software is needed for real-time pre-processing and filtering of the most promising events, producing even more data.

To help address this problem, a team of Machine Learning experts and physics scientists working at CERN (the world largest high energy physics laboratory), has partnered with Kaggle and prestigious sponsors to answer the question: can machine learning assist high energy physics in discovering and characterizing new particles?

Reconstrucción de trazas



Ejemplo de colisión



Tracking: Lo básico

- Eficiencia de reconstrucción:

$$\frac{\text{Número de tracks correctamente reconstruidos}}{\text{Número total de tracks reconstructibles}}$$

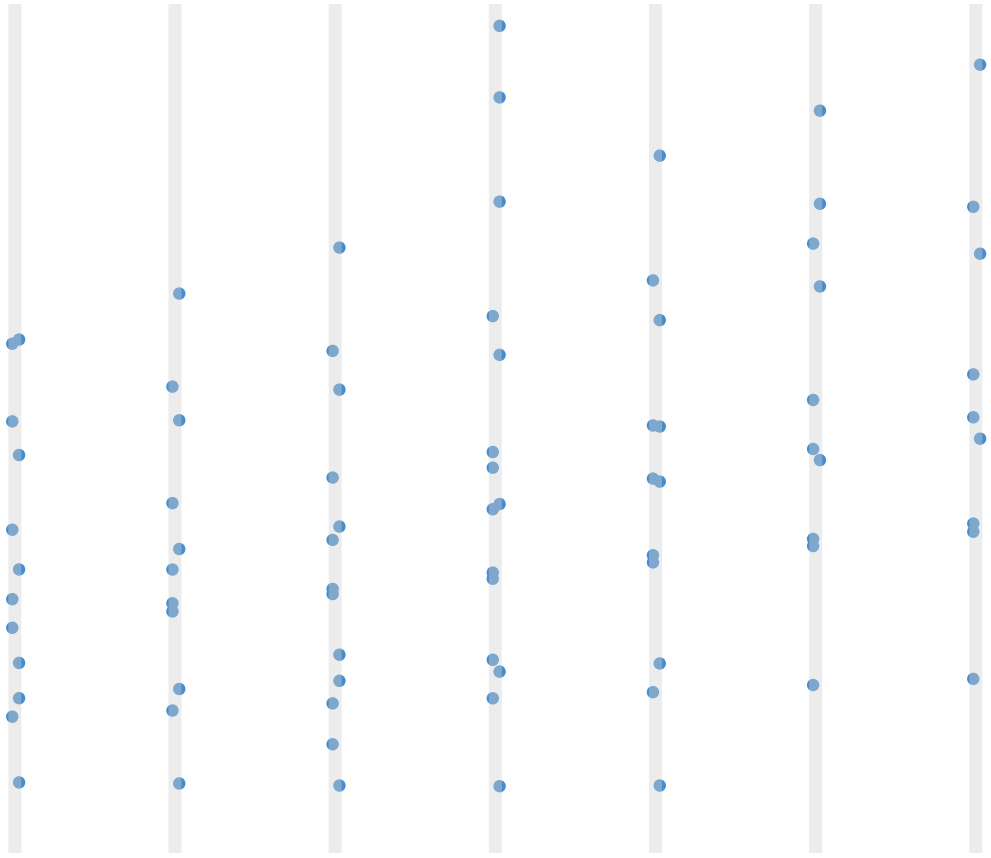
- Fracción de tracks clones:

$$\frac{\text{Número de tracks clones}}{\text{Número de tracks correctamente reconstruidos}}$$

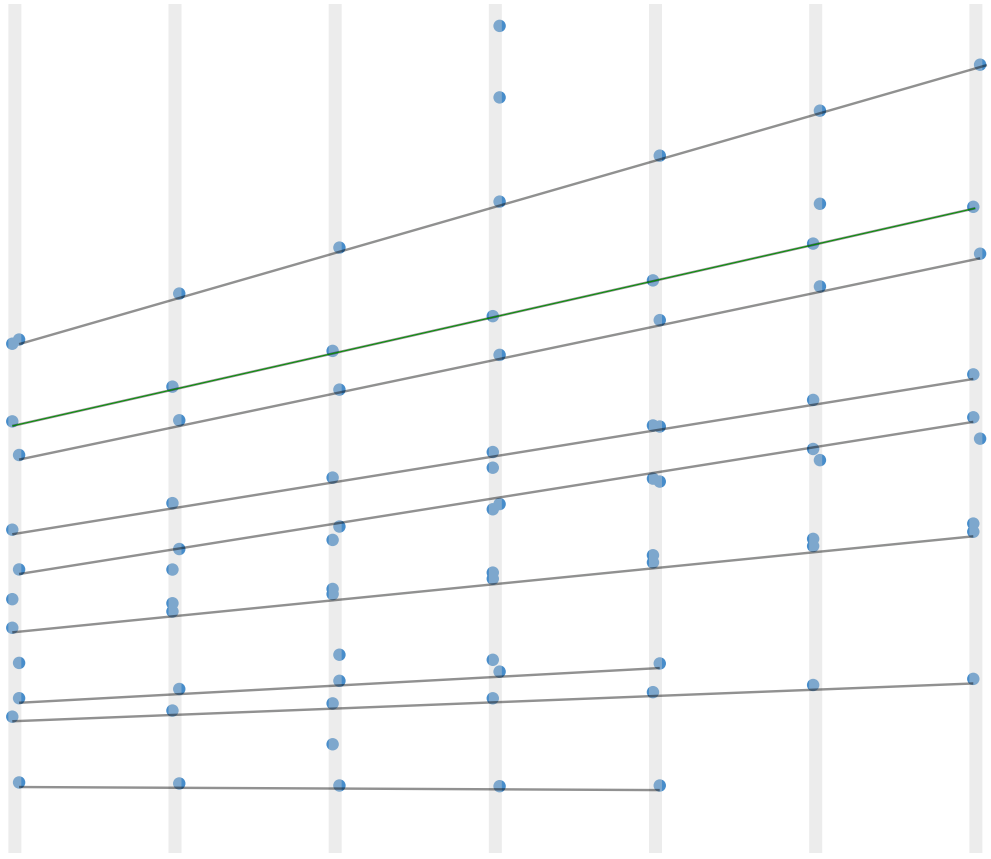
- Fracción de tracks fantasma:

$$\frac{\text{Número de tracks fantasmas}}{\text{Número de tracks reconstruidos}}$$

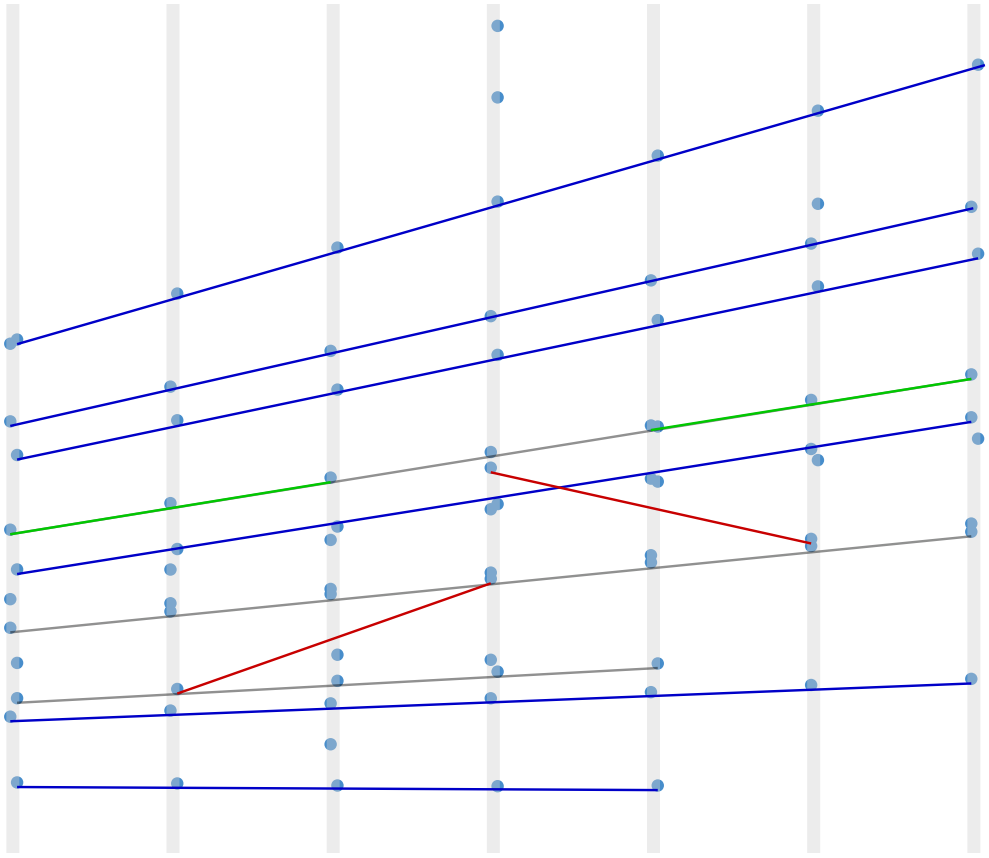
Un ejemplo (1)



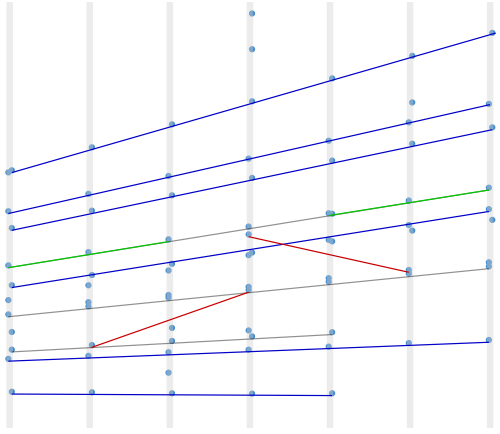
Un ejemplo (2)



Un ejemplo (3)



Un ejemplo (4)



- Eficiencia de reconstrucción: $\frac{7}{9} = 78\%$
- Fracción de tracks clones: $\frac{1}{7} = 14\%$
- Fracción de tracks fantasma: $\frac{2}{10} = 20\%$

La letra pequeña de Kaggle

Custom metric

The evaluation metric for this competition is a custom metric. In one line : *it is the intersection between the reconstructed tracks and the ground truth particles, normalized to one for each event, and averaged on the events of the test set.*

First, each hit is assigned a weight:

- the few first (starting from the center of the detector) and last hits have a larger weight
- hits from the more straight tracks (more rare, but more interesting) have a larger weight
- random hits or hits from very short tracks have weight zero
- the sum of the weights of all the hits of one event is 1 by construction
- the hit weights are available in the truth file. They are not revealed for the test dataset

Then, the score is constructed as follows:

- tracks are uniquely matched to particles by the double majority rule:
  - for a given track, the matching particle is the one to which the absolute majority (strictly more that 50%) of the track points belong.
  - the track should have the absolute majority of the points of the matching particle. If any of these constraints is not met, the score for this track is zero
- the score of a surviving track is the sum of the weights of the points of the intersection between the track and the matching particle.
- the score of an event is the sum of the score of all its tracks.
- the final score is the average on the events of the public and private leaderboard test respectively.

Cómo ganar la competición

Técnica local: Buscar semillas en una pequeña parte del detector y extrapolar, marcando los hits como *usados*:

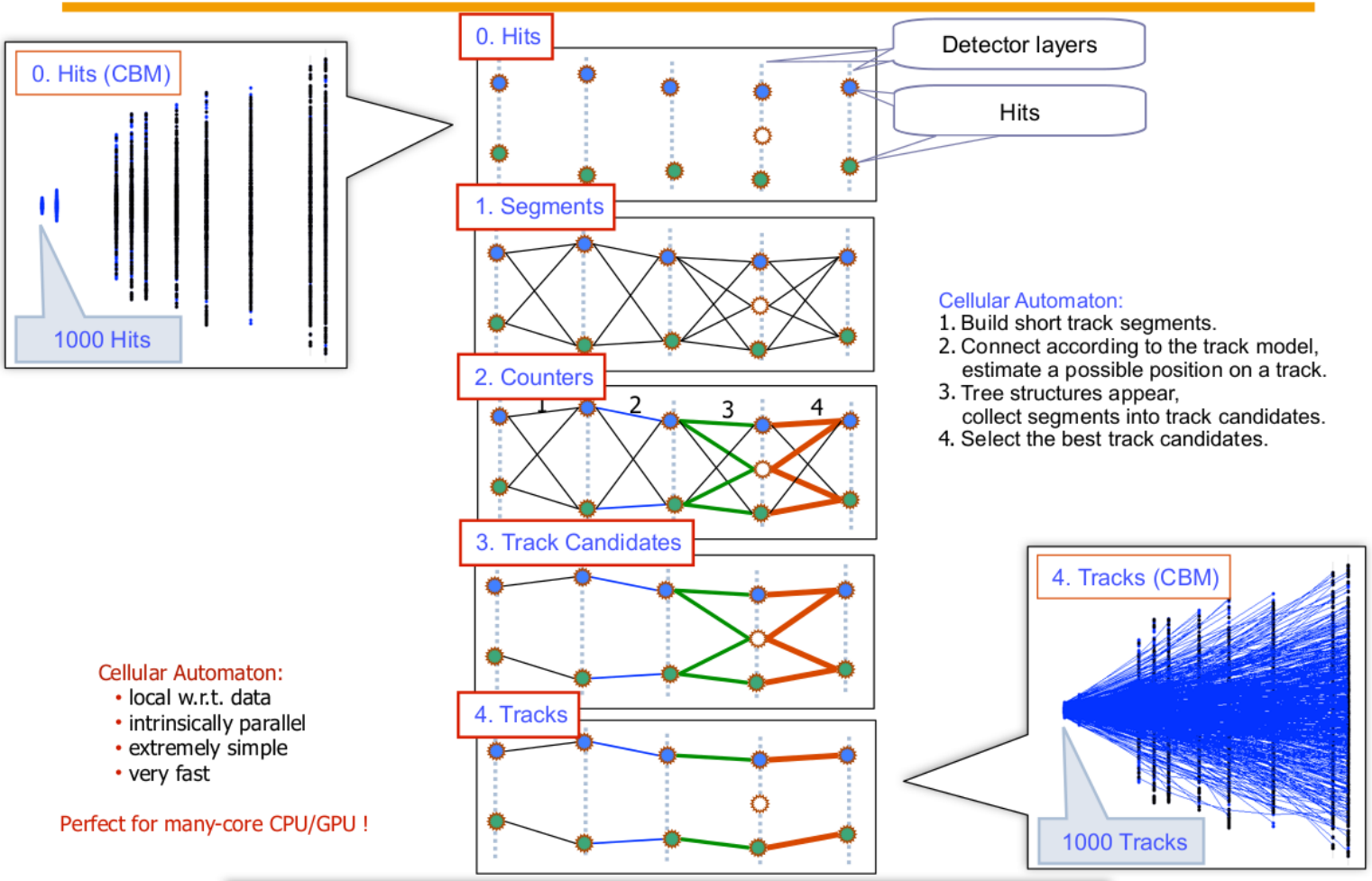
- Track forwarding

Técnicas globales: Resolver el problema de una tacada

- Transformada de Hough
- Clustering
- Machine Learning
- Automata
- Retina
- R-trees

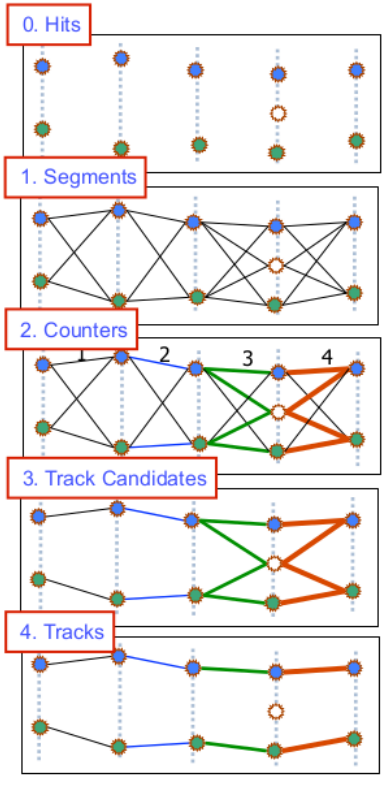
Automata tracking

Cellular Automaton (CA) Track Finder



Automata tracking - Implementación de Kisel

CA Track Finder: Pseudocode



Pseudocode for CBM CA Track Finder

```
1  Sort_Input_Hits_According_to_Grid();
2
3  for track_set (high_p_primary, low_p_primary, secondary, broken)
4
5  switch (track_set)
6    case high_p_primary:
7      Build_Triplets (min_momentum_for_fast_tracks,
       primary_track_parameter_initilisation, triplets_wo_gaps);
8
9    case low_p_primary:
10     Build_Triplets (min_momentum_for_slow_tracks,
       primary_track_parameter_initilisation, triplets_wo_gaps);
11
12   case secondary:
13     Build_Triplets (min_momentum_for_slow_tracks,
       secondary_track_parameter_initilisation, triplets_wo_gaps);
14
15   case broken:
16     Build_Triplets (min_momentum_for_slow_tracks,
       secondary_track_parameter_initilisation, triplets_with/
       wo_gaps)
17
18 Find_Neighbours();
19
20
21
22
23   for track_length := NStation to 3 do
24     for station := FirstStation to NStation do
25       for triplets := First_Triplet_Station to
         Last_Triplet_Station do
26         track_candidate = Build_Best_Candidate (triplet);
27
28
29   Save_Candidates(all_track_candidates);
30
31   Delete_Used_Hits();
```

```
void function Build_Triplets (min_momentum,
prim/sec_track_parameter_initilisation,
triplets_with/wo_gaps)
{
  for station := (NStation-2) to FirstStation do
    for hits_portion := First_Portion_Station to
Last_Portion_Station do

      Find_Singlets(hits_portion);
      Find_Doublets(singlets_in_portion);
      Find_Triplets(doublets_in_portion);
}
```

```
void function Find_Neighbours (All_Triplets)

{
  for triplet := First_Triplet to Last_Triplet
do

      Find_Save_Neighbours(triplet);
      Calculate_Level(triplet);

}
```

```
void function
Save_Candidates(All_Track_Candidate)
{
   Sort_Candidates();
     for candidate := First_Candidate to
Last_Candidate do
       if (used_hits) discard candidate
       else save candidate;

}
```

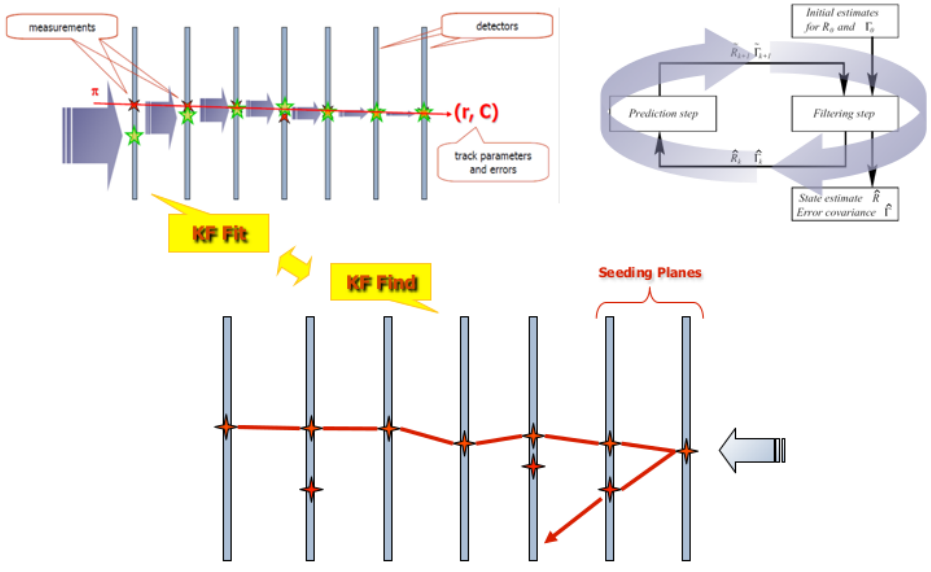
Staged track finding

Implementación sencilla en Velopix

```
# 1. Fill candidates
candidates = self.fill_candidates(event_copy)

# 2. Create all segments, indexed by outer hit number
(segments, outer_hit_segment_list, compatible_segments, \
  populated_compatible_segments) = \
  self.populate_segments(event_copy, candidates)

# 3. Assign weights and get roots
self.assign_weights_and_populate_roots(segments, compatible_segments, \
  populated_compatible_segments)

# 4. Depth first search
tracks = []
for segment_id in root_segments:
  root_segment = segments[segment_id]
  tracks += [track([root_segment.h0] + dfs_segments) \
    for dfs_segments in self.dfs(root_segment, segments, compatible_segments)]

# 5. Clone and ghost killing
tracks = self.prune_short_tracks(tracks)
```

https://github.com/dcampora/velopix_tracking/blob/master/run_graph_dfs.py

Local Methods: Kalman Filter for Track Following



Features:

- Psychologically easy to accept hit by hit track finding
- Combined track finder and fitter based on KF
- Development of a new experiment starts with an ideal MC track finder and a realistic KF track fitter, therefore the next step to a realistic track finder is obvious – KF
- ...

Weak points:

- Based on a single track approach
- Needs seeding (starting short track segments)
- Final efficiency is always limited by seeding efficiency
- It is limited also by the efficiency of the seeding chambers
- Works at the hits level, searching for hits within a region
- Repeats calculations, when discarding track candidates
- Therefore needs a lot of seeds -> even larger combinatorics
- How many inefficient detectors can be tolerated in general ?
- Too early competition between track candidates
- ...

Useful for relatively simple event topologies and as a second after the ideal track finder

Kalman Filter based Track Fit

The Kalman filter is a recursive estimator – only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state.

mean value over n measurements

n

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

n+1

$$\mu_{n+1} = \frac{1}{n+1}\sum_{i=1}^{n+1} x_i = \frac{n}{n+1}\left(\frac{1}{n}\sum_{i=1}^{n} x_i + \frac{1}{n}x_{n+1}\right) = \frac{n}{n+1}\mu_n + \frac{1}{n+1}x_{n+1} = \mu_n + \frac{1}{n+1}(x_{n+1} - \mu_n)$$

mean value over n+1 measurements

previous estimation

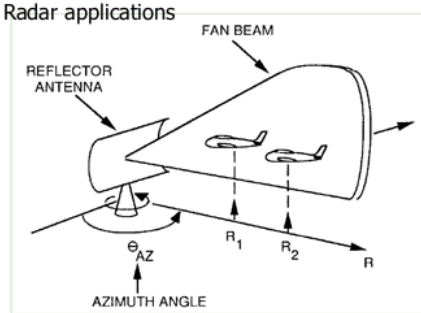
new measurement

weight

correction

For this work, U.S. President Barack Obama rewarded Kálmán with the National Medal of Science on October 7, 2009.

Radar applications



state vector:

$\mathbf{r} = \{ x, y, z, v_x, v_y, v_z \}$

covariance matrix:

$$C = \begin{Bmatrix} \sigma^2_x & & & & & \\ & \sigma^2_y & & \cdots & & \\ & & \sigma^2_z & & & \\ & & & \sigma^2_{vx} & & \\ & \cdots & & & \sigma^2_{vy} & \\ & & & & & \sigma^2_{vz} \end{Bmatrix}$$

Apollo Flight Journal

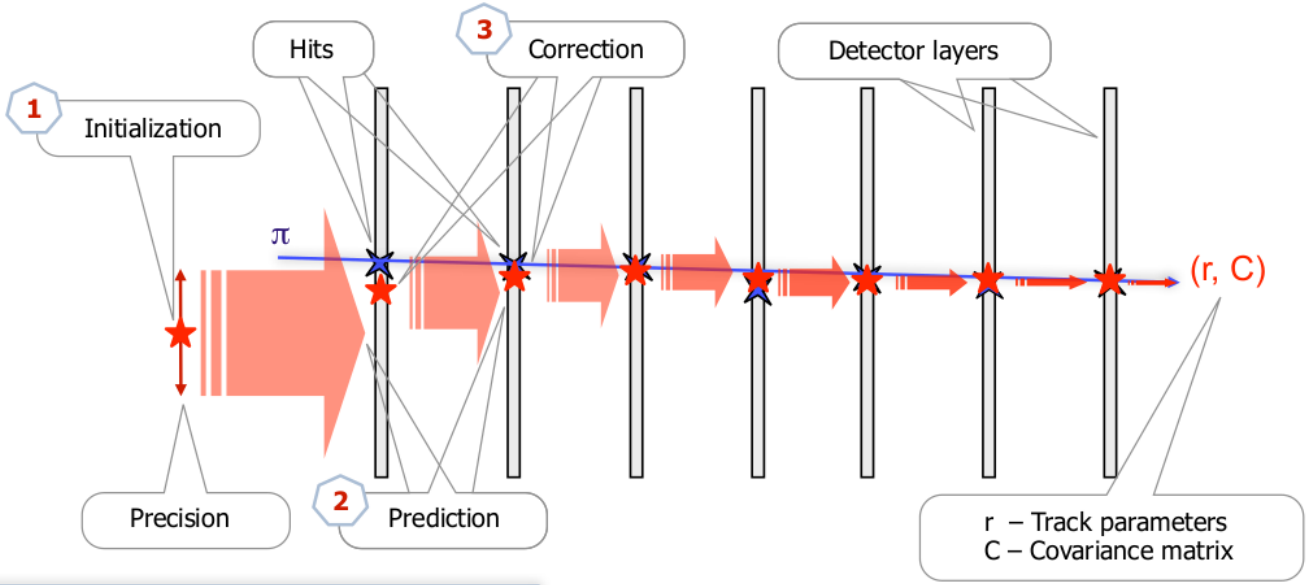
December 21, 1968. The Apollo 8 spacecraft has just been sent on its way to the Moon.

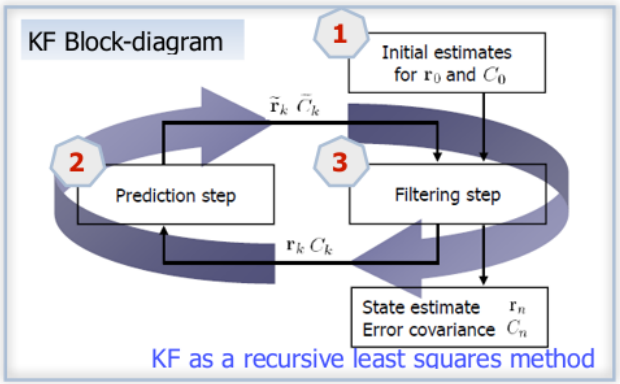
003:46:31 Collins: Roger. At your convenience, would you please go P00 and Accept? We're going to update to your W-matrix.

Track forwarding - Filtro de Kalman (2)

Kalman Filter based Track Fit

Estimation of the track parameters at one or more hits along the track – Kalman Filter (KF)





State vector — Position, direction and momentum

$r = \{ x, y, z, p_x, p_y, p_z \}$

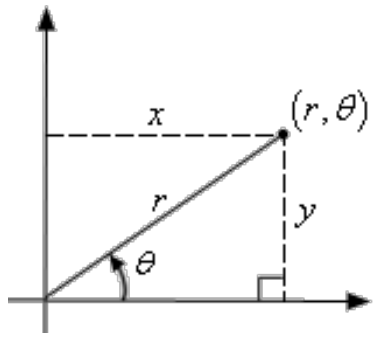
Kalman Filter:

1. Start with an arbitrary initialization.
2. Add one hit after another.
3. Improve the state vector.
4. Get the optimal parameters after the last hit.

Nowadays the Kalman Filter is used in almost all HEP experiments

Transformada de Hough

Volviendo a la transformación a coordenadas polares:



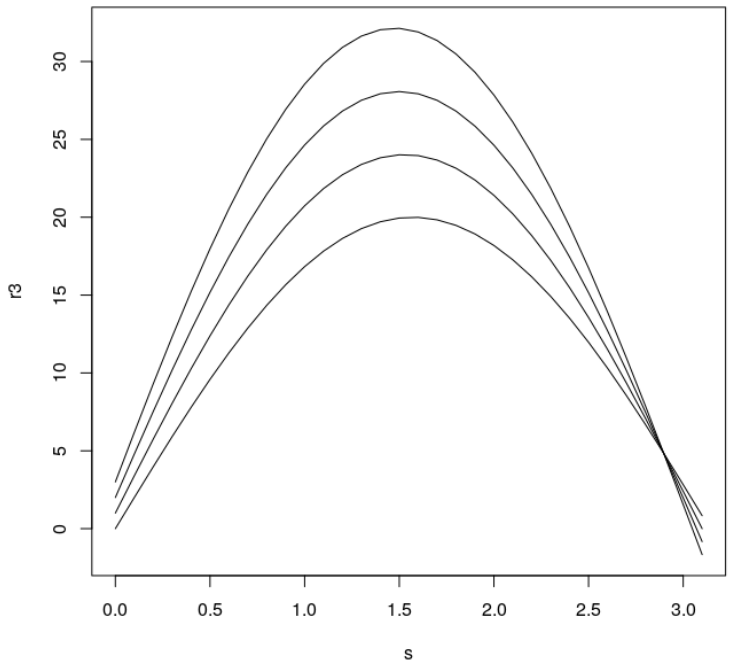
Todas las líneas que pueden pasar por un punto x, y pueden expresarse con la función

$$f(\rho) = x * cos(\sigma) + y * sen(\sigma)$$

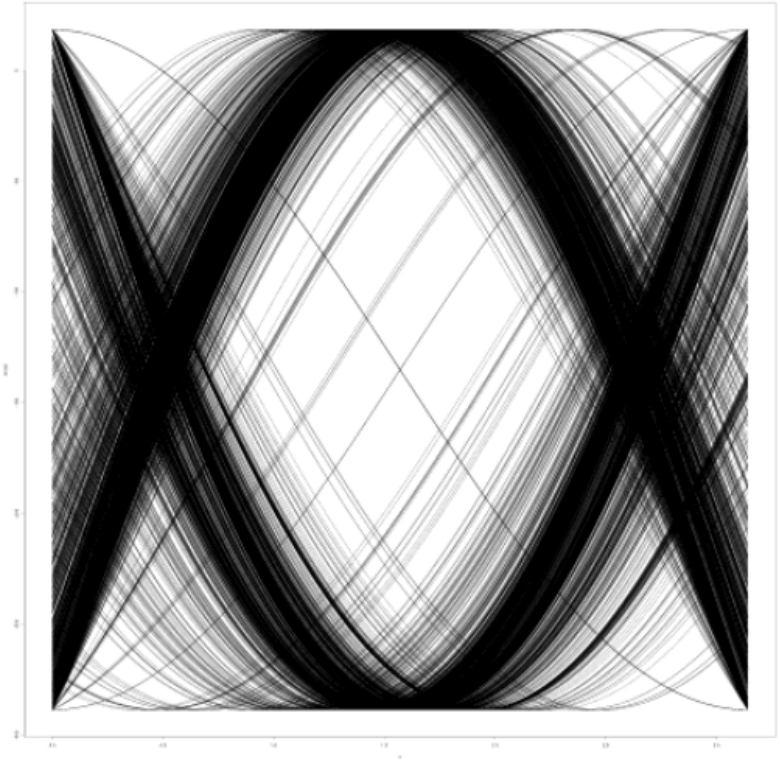
Transformada de Hough (2)

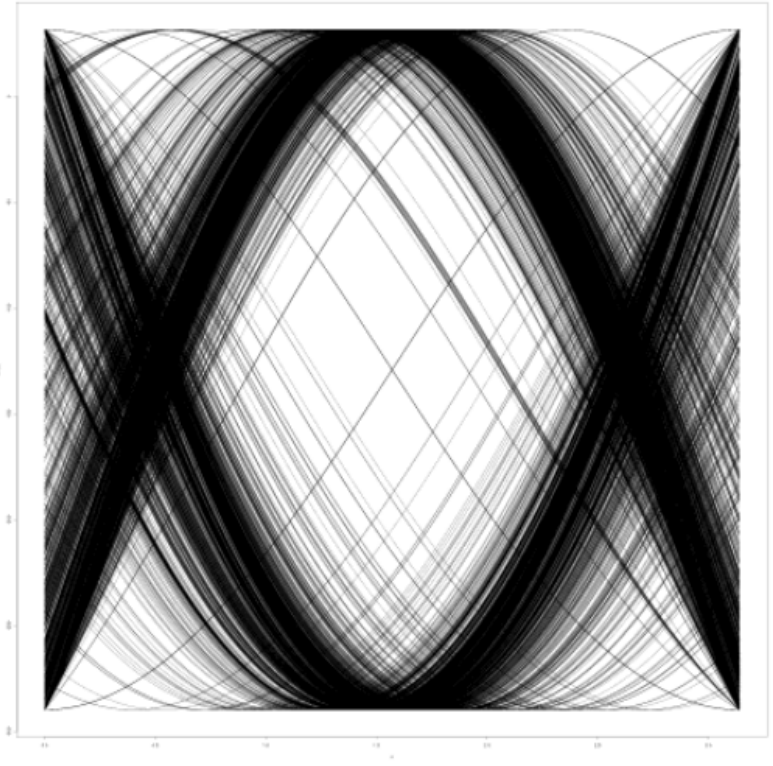
Por ejemplo, una sesión en R...

```
> s = seq(0,3.14,0.1)
> #y = 4x + 20
> r0 = 0*cos(s) + 20*sin(s)
> r1 = 1*cos(s) + 24*sin(s)
> r2 = 2*cos(s) + 28*sin(s)
> r3 = 3*cos(s) + 32*sin(s)
> plot(s, r3, type="l")
> lines(s, r2, type="l")
> lines(s, r1, type="l")
> lines(s, r0, type="l")
```



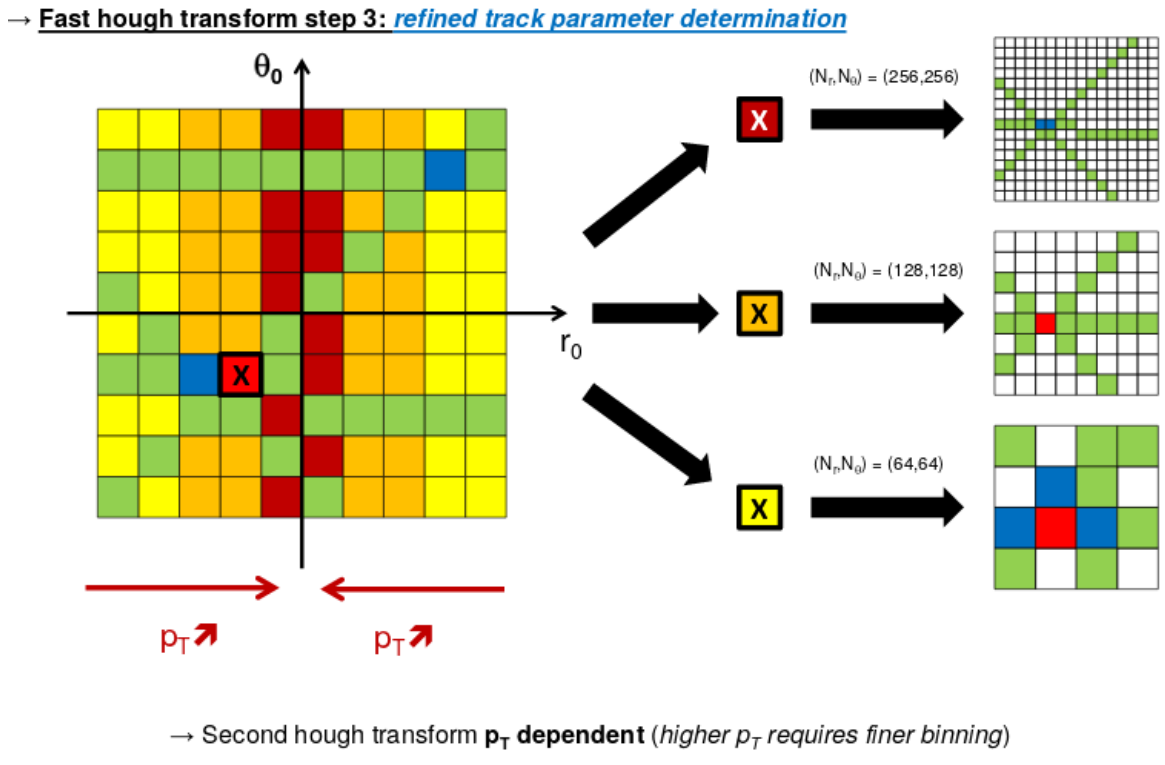
Pero la realidad no es tan fácil





Ideas para la transformada de Hough

- Hacer un análisis más fino donde sea necesario
- Elegir un threshold y un binning correcto
- Paralelizar



Materiales de LHCb

Repositorio con ejemplos reales de reconstrucción de Velopix en Python 3:

- `https://github.com/dcampora/velopix_tracking`

Repositorio con reconstrucción en GPU (optimizada) de tracking en C++:

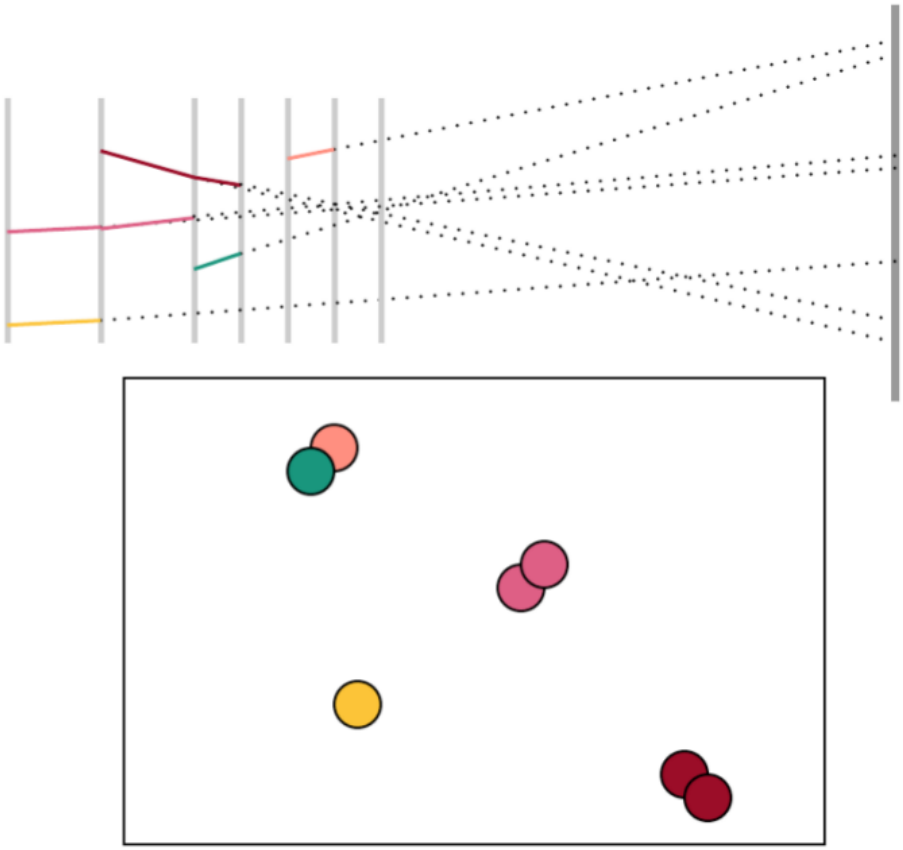
- `https://gitlab.cern.ch/lhcb-parallelization/cuda_hlt`

¡Gracias! ¿Preguntas?

Backup

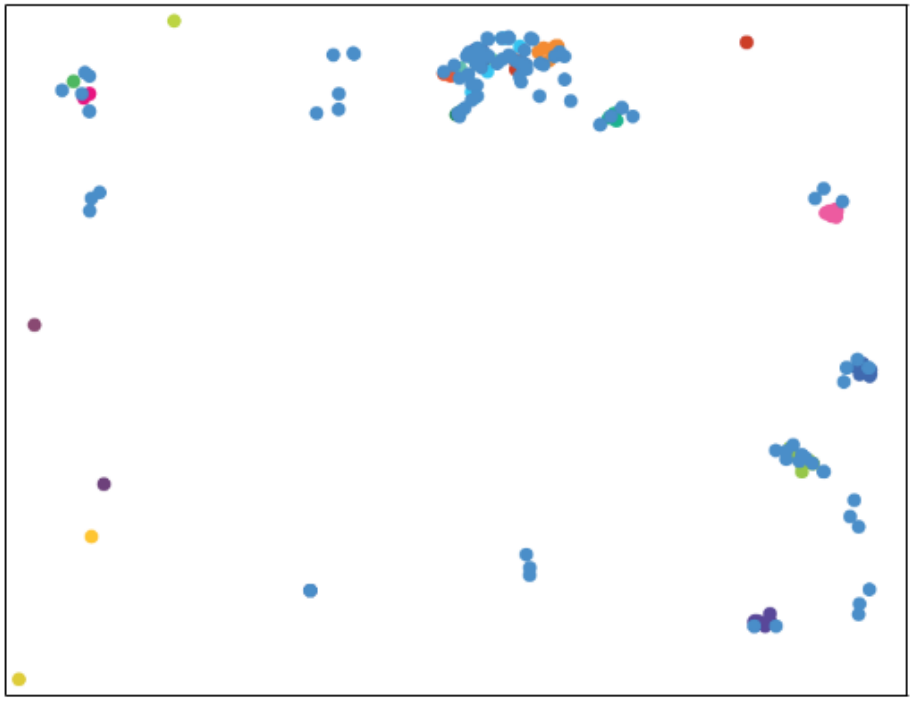
Clustering

Podemos transformar el problema en un problema de clustering

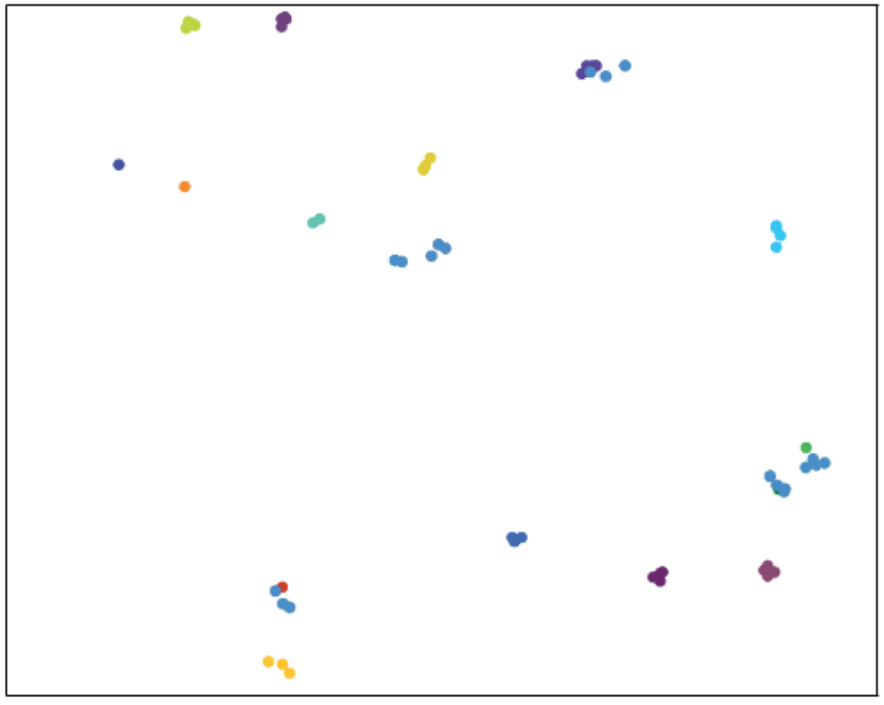


Clustering (2)

Esa idea no funcionó bien: *Mucho ruido y pocas nueces*



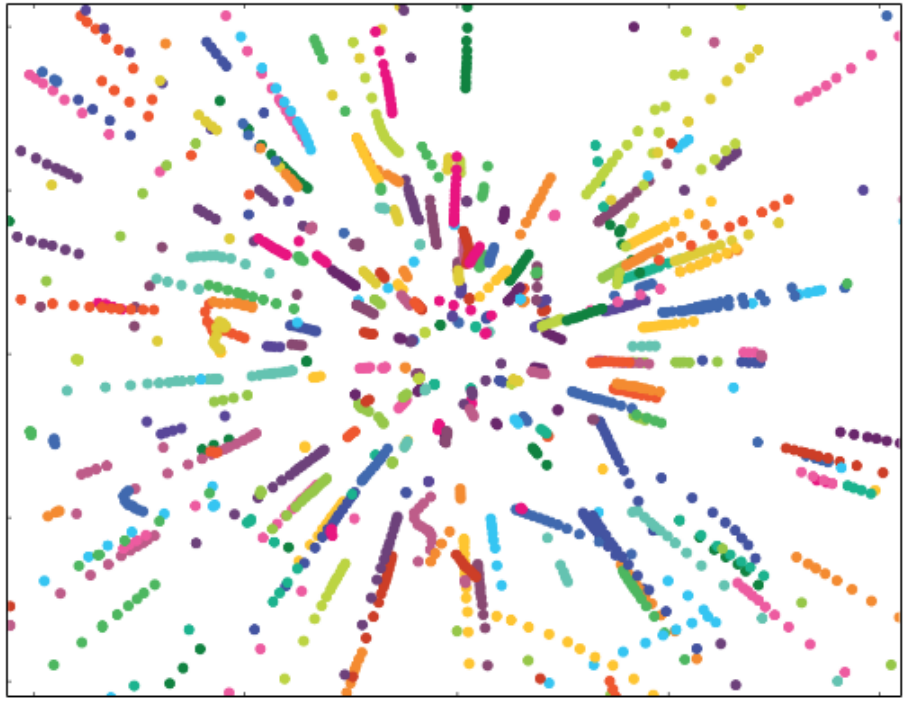
(a) Extrapolating pairs of hits(segments)



(b) Extrapolating triplets of hits

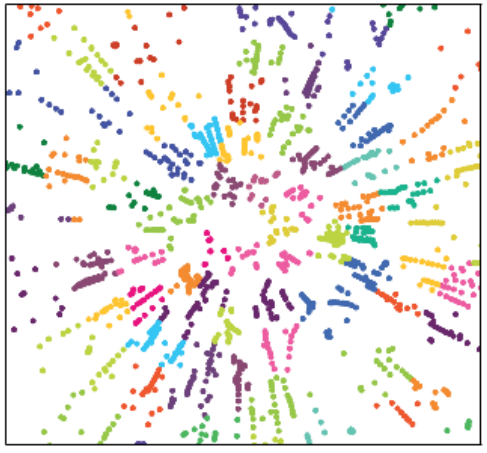
Clustering (3)

No obstante, algo mucho más sencillo da mejores resultados

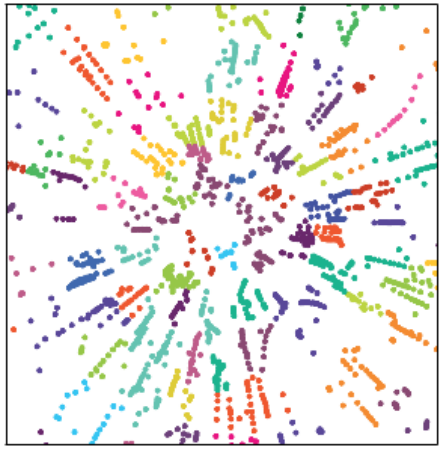


(c) Extrapolating segments with a virtual point in the origin. We can notice the lack of noise(dark blue) in this final picture, since the procedure inherently creates only one point in the plane for every hit.

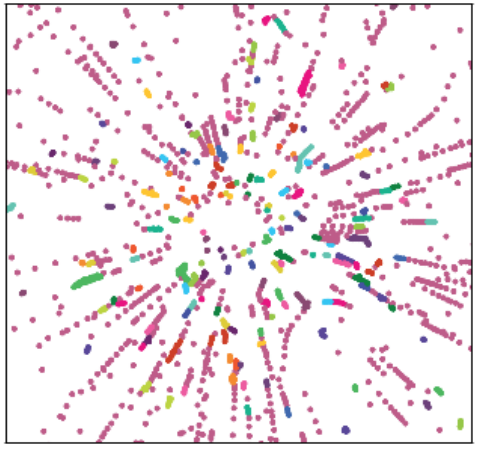
Clustering (4)



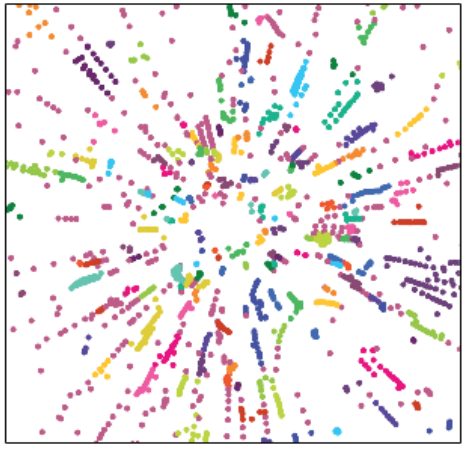
(a) KMeans



(b) Agglomerative Clustering



(c) DBSCAN



(d) HDBSCAN

Clustering (5)

Hasta ahora, HDBSCAN con extrapolación desde el origen es lo que mejor resultados nos ha dado, usando una técnica no convencional. ¡Pero los resultados están lejos de ser utilizables!

```
16621 tracks including     8784 ghosts ( 52.8%). Event average  50.7%
              velo :     7623 from    28019 ( 27.2%,  28.3%)       55 clones (  0.72%), purity: ( 90.65%,  90.65%),  hitEff: ( 88.87%,  88.85%)
              long :     4171 from     9063 ( 46.0%,  47.7%)       37 clones (  0.89%), purity: ( 91.73%,  91.76%),  hitEff: ( 89.01%,  88.94%)
        long>5GeV :     3147 from     5950 ( 52.9%,  54.2%)       26 clones (  0.83%), purity: ( 92.38%,  92.47%),  hitEff: ( 90.40%,  90.43%)
     long_strange :       90 from      357 ( 25.2%,  22.9%)        0 clones (  0.00%), purity: ( 90.57%,  90.01%),  hitEff: ( 84.32%,  83.87%)
long_strange>5GeV :       47 from      169 ( 27.8%,  23.7%)        0 clones (  0.00%), purity: ( 91.17%,  91.30%),  hitEff: ( 92.30%,  90.36%)
       long_fromb :      239 from      553 ( 43.2%,  44.7%)        2 clones (  0.84%), purity: ( 91.80%,  91.20%),  hitEff: ( 84.87%,  88.10%)
  long_fromb>5GeV :      197 from      441 ( 44.7%,  45.7%)        1 clones (  0.51%), purity: ( 92.04%,  91.29%),  hitEff: ( 86.42%,  88.39%)
```

(g) HDBSCAN

```
8569 tracks including       49 ghosts (  0.6%). Event average   0.5%
              velo :     8287 from    28019 ( 29.6%,  30.8%)       89 clones (  1.07%), purity: ( 98.36%,  98.35%),  hitEff: ( 86.67%,  86.35%)
              long :     3792 from     9063 ( 41.8%,  43.4%)       67 clones (  1.77%), purity: ( 98.13%,  98.17%),  hitEff: ( 86.62%,  86.19%)
        long>5GeV :     2826 from     5950 ( 47.5%,  48.7%)       50 clones (  1.77%), purity: ( 98.31%,  98.40%),  hitEff: ( 88.21%,  87.91%)
     long_strange :       89 from      357 ( 24.9%,  23.9%)        0 clones (  0.00%), purity: ( 97.33%,  97.14%),  hitEff: ( 79.48%,  78.60%)
long_strange>5GeV :       44 from      169 ( 26.0%,  22.9%)        0 clones (  0.00%), purity: ( 97.87%,  97.82%),  hitEff: ( 83.85%,  82.62%)
       long_fromb :      216 from      553 ( 39.1%,  40.1%)        3 clones (  1.39%), purity: ( 98.01%,  97.92%),  hitEff: ( 84.52%,  86.70%)
  long_fromb>5GeV :      176 from      441 ( 39.9%,  40.8%)        3 clones (  1.70%), purity: ( 98.07%,  97.93%),  hitEff: ( 85.85%,  87.04%)
```

(h) HDBSCAN with postprocessing