ML based RICH reconstruction

Daniel Hugo Cámpora Pérez
dcampora@cern.ch

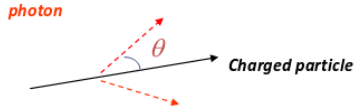
Computing Challenges, 8th May, 2018

Universidad de Sevilla
CERN

The RICH reconstruction

Problem formulation

Cherenkov radiation principle



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

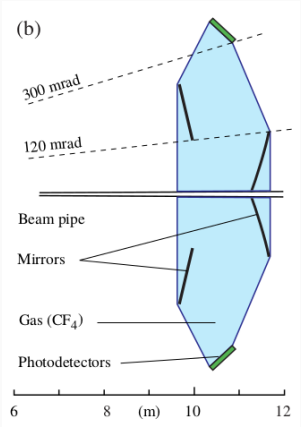
$$\beta = v/c = p/E = p/(p^2 + m^2)^{0.5} = 1/(1+(m/p)^2)^{0.5}$$

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)

RICH detectors in LHCb

In LHCb, we have two RICH detectors. Below is a schematic XZ view of RICH1:



Visually

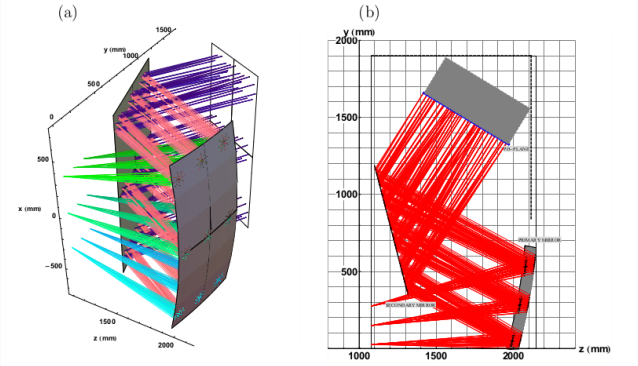


Figure 2.21: Simulated Cherenkov photons in the upper half of the upgraded RICH 1; (a) 3D view, (b) 2D view in the vertical plane.

Analytical solution

The analytical solution consists in *creating photons*, ie. associations of detected pixels in the HPDs / MaPMTs with their originating track segments through the Rich detector.

Once this association is found, a likelihood minimisation algorithm is run in order to find the most likely candidate for each particle.

- Photon creation, heavily involving ray tracing
- Likelihood minimisation

Analytical solution - Ray tracing

The creation of the photons involves a ray tracing algorithm from each segment to the candidate pixels. In turn, this means solving the quartic equation:

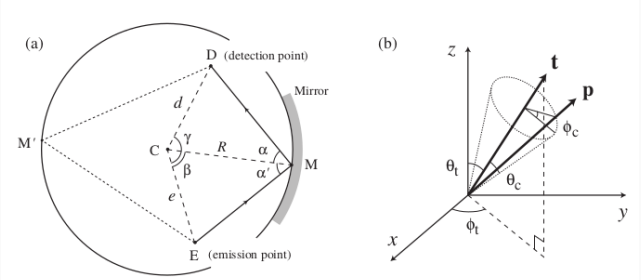


Figure 3: (a) Definition of parameters used in the reconstruction of the Cherenkov angles, for a photon emitted from point E, reflected in a spherical mirror at M and detected at D. (b) Definition of the angles describing the direction of a track $\mathbf{t}$ in the lab frame, and a photon $\mathbf{p}$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

Analytical solution - Ray tracing (2)

In turn, this means solving the quartic equation:

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

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.

Analytical solution - Likelihood minimisation

Afterwards, each particle segment could be associated with one of six possibilities (electron, pion, muon, kaon, proton, deuteron). A brute-force search would imply solving the intractable problem of testing 6^n possibilities!

Instead, a local search is done by sequentially finding the best local possibility for each particle. This reduces the search to only testing $6n$ possibilities, at the cost of possibly falling into a local minimum.

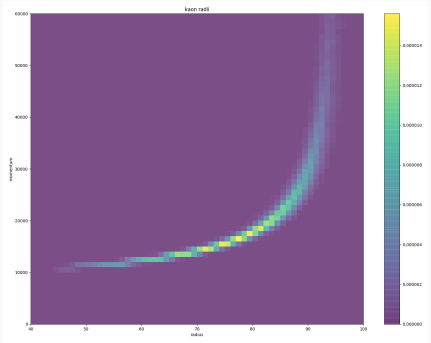
Rico

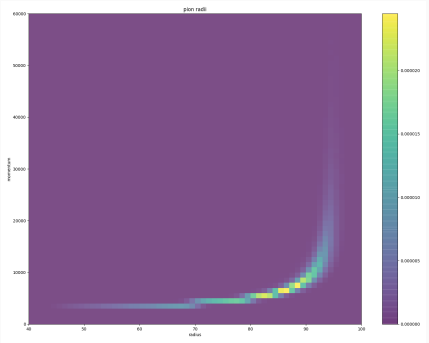
Overview of Rico

Rico is a small set of algorithms to attempt Machine Learning techniques for RICH reconstruction.

Motivation

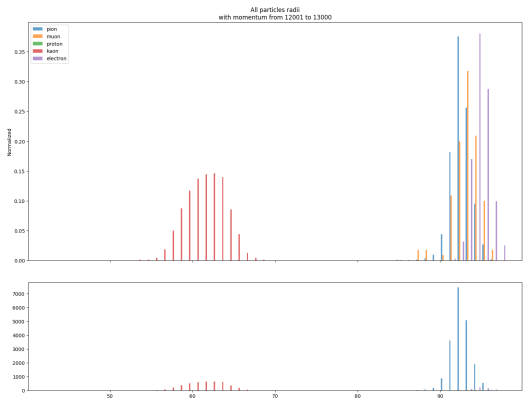
It is possible to detect the PID based on the momentum of the track (Y axis - known a priori from the tracking detectors reconstruction) and the radius of the detected circle / ellipsoid in the detectors (X axis):





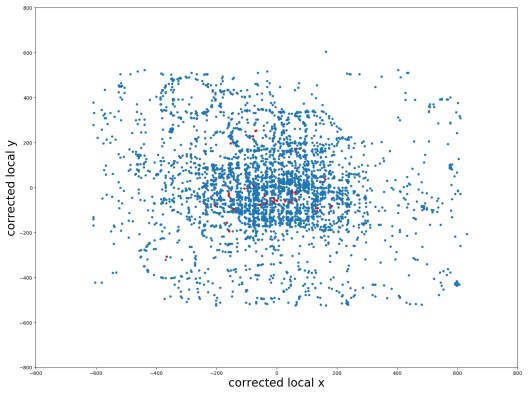
Motivation (2)

We can divide the input tracks into momenta ranges, and determine the correlation between radius and PID for that range:



Motivation (3)

This is a view of an event in RICH1. Red dots represent the extrapolated centroid for each reconstructed long track:



Motivation (4)

If we observe the surrounding area of a single centroid and convert it to polars, we end up with the figures below:

Figure 1: Left: Pion. Right: Electron.

A CNN problem

We have effectively transformed the RICH reconstruction into a *classification* problem. The input is a binary image, the output is a classification of the particle into one of six possibilities.

Now, we can define various Convolutional Neural Networks in order to solve the problem.

Exploring possibilities

Some CNNs are more amenable to solving the problem at hand. In fact, the problem looks very similar to the classical MNIST problem. One such CNN is:

```
_________________________________________________________________
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
```

Ongoing research

```
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
```

Implementation back in the LHCb framework

I am working on an inference done in C++ back in the framework based on preexisting models.

Hardware optimized for inference is now appearing and is becoming more and more like a viable option. These could also be tested and evaluated: ie. GPUs, Intel Nervana, FPGA, etc.

The ML *hype* doesn't seem to be going anytime soon, so it is conceivable that these technologies will make it into more affordable solutions, ie. gaming GPUs.

Software technologies at hand

I would very much encourage anybody interested in doing research on CNNs to look out for preexisting technologies:

- Keras
- Caffe
- Tensorflow
- Theano

Nvidia GPU-based ML is very accessible and many preexisting models exist in literature.

Ari ga tou!