Reconstructing Photodot Positions

in MarlinTPC

Patrick Conley
University of Victoria

Terminology

- Photodot (photoline): refers to aluminum dots (lines) on the patterned cathode.
- Expected image: location on the readout electronics opposite a photodot.
- Image: location where an electron cloud from a photodot is absorbed by the readout electronics

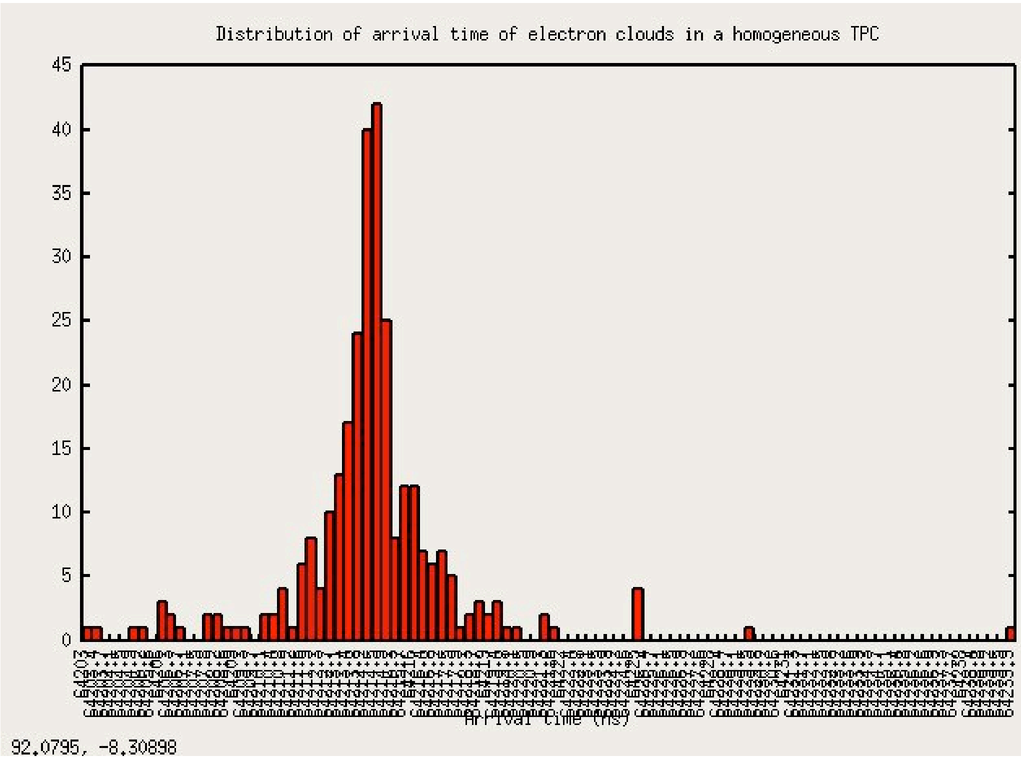
Overview of the processor

1. Determines which pulses fall in a time range appropriate to be considered part of a photohit
2. Matches images with corresponding photodots
3. Finds the approximate centre of each image and divides the charge in a hit into quadrants
4. Uses an inverse error function to calculate the centre of each image for each coordinate
5. Outputs the displacement between the centre of the image and the expected centre

Finding appropriate pulses

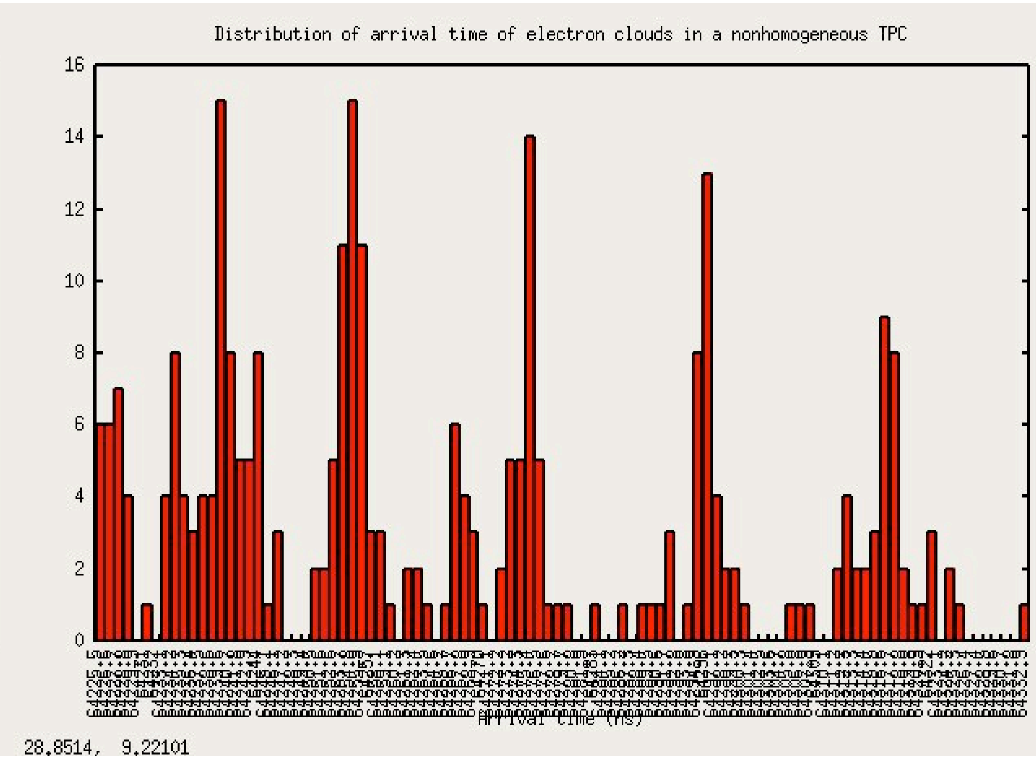
- The time of each pulse should be checked to ensure it is part of a photoelectric event, not a particle track
- Ideally, a histogram would be constructed of all pulses, and the most prevalent time would be assumed to be the time of the event
- This step spectacularly in strongly inhomogeneous fields:

Distribution of arrival times in a homogeneous field



- Pulses are spread over ~20ns
- Strongly peaked distribution

Distribution of arrival times in an inhomogeneous field



- Pulses are spread over ~100ns
- No clear distribution

Which distribution best shows the field in the prototype?

Finding appropriate pulses

- Stopgap solution relies on simulating the drift of a single electron from each photodot
- In the absence of transverse or longitudinal diffusion, and in the known field of the TPC, this electron gives an approximate drift time

✳ The current method assumes we know the exact time the laser pulse hit the cathode, and the gas parameterization and drift velocity of the TPC

- These are known for simulated data, but not for test beam data

Finding corresponding hits

- Calculates the distance from a given photodot to each pad with a pulse
- The nearest charged pad is assumed to be part of the hit corresponding to the photodot

✳ Processor does not currently check that this nearest pulse is part of any hit, ie. that it is not just noise

Finding approximate image centres

- Using the nearest pulse found previously, the processor iterates through adjacent pads to the centre pad in the bottom row of the hit:
  - Processor checks pads on either side of the current pad to find pad with the highest charge
  - If adjacent pads both have less charge, processor checks for the existence of charge on the pad below it and selects that one if possible
- If neither of these steps results in a change in the current pad, the current pad is taken as the centre of the bottom row

✳ An improvement would find the bottom row from the geometric centre of a hit, not the charge centre

Calculating image centres

‘Row’ centre (y- or r- coordinate):

- Relies on the assumption a hit will have at least two rows
- A dividing line is drawn between the approximate top and bottom halves of the charge, and the fraction of the charge below this line is calculated
- Root’s TMath::ErfInverse is used to calculate the row centre of the hit based on this fraction

Calculating image centres

'Column' centre (x- or phi-coordinate):

- In each row, the pad selected (in slide 8) is used to divide the charge into approximate halves
- TMath::ErfInverse is used to calculate the centre of column-centre of the row
- Taking the weighted mean of the location calculated corrects for the offset in pad position between rows

Coordinates of each photodot, expected image (will be eliminated when a better method of calculating time is developed), and image are now output to a text file.

Calculating image centres

- The method described will not give accurate results for test beam data.
  - Currently relies on the value of transverse diffusion used to generate simulated data
  - Proper value for diffusion will be calculated from the spread in the width of photolines, but code to do this does not yet exist

Using the processor

- PhotodotReconstructionProcessor can be called from a steering file in the normal way
- Available parameters are:

```xml
<processor name="MyPhotodotReconstructionProcessor" type="PhotodotReconstructionProcessor">
  <parameter name="InputTrackerPulses" type="string">TPCPulsesAsElectrons</parameter>
  <parameter name="GeometryFilename" type="string">photo_geometry.txt</parameter>
  <parameter name="ComparedPhotodotsFilename" type="string">compared_photodots.txt</parameter>
  <parameter name="TransverseDiffusion" type="double">40</parameter>
  <parameter name="OmegaTau" type="double">10</parameter>
  <parameter name="DriftVelocity" type="double">35</parameter>
  <parameter name="TPCOuterRadius" type="double">1800</parameter>
  <parameter name="TPCInnerRadius" type="double">305</parameter>
  <parameter name="InnerZ" type="double">2.5</parameter>
  <parameter name="OuterZ" type="double">2250</parameter>
</processor>
```

- All but the first three will be eliminated when a better method of calculating time is found

File layouts

```
CARTESIAN
photodots
   0 400 5
   0 500 5
   0 600 5
   0 700 5
   0 800 5
-400 400 5
-500 500 5
-600 600 5
-700 700 5
-800 800 5
 400 400 5
 500 500 5
 600 600 5
 700 700 5
 800 800 5
photolines
0 0 10 10 1
```

Format of the input file

(x position, y position, diameter)

```
CARTESIAN
 Photodot position    Expected image        Image position
(        x,        y) (        x,        y) (        x,        y)
         0       400   -1.48049   414.53   -1.44063  414.302
         0       500   -1.84096  518.064   -1.86243  517.911
         0       600    -2.2068  621.643   -2.19475  621.525
         0       700   -2.59105  725.398   -2.54635  724.865
         0       800   -2.91816  828.611   -2.95096  828.642
      -400       400   -415.982  413.025   -415.688  412.773
      -500       500    -519.95  516.256   -519.707  516.018
      -600       600   -623.494  619.145   -623.841  619.389
      -700       700   -727.943  722.757   -727.743  722.582
      -800       800   -831.825  825.913   -831.466  825.606
       400       400    413.025  415.982      412.8  415.669
       500       500    516.256   519.95    516.024  519.713
       600       600    619.145  623.494     619.41   623.84
       700       700    722.757  727.943    722.558  727.735
       800       800    825.913  831.825    825.646  831.477
```

Format of the output file. Subsequent runs are appended without repeating headers