

Vertex Finding

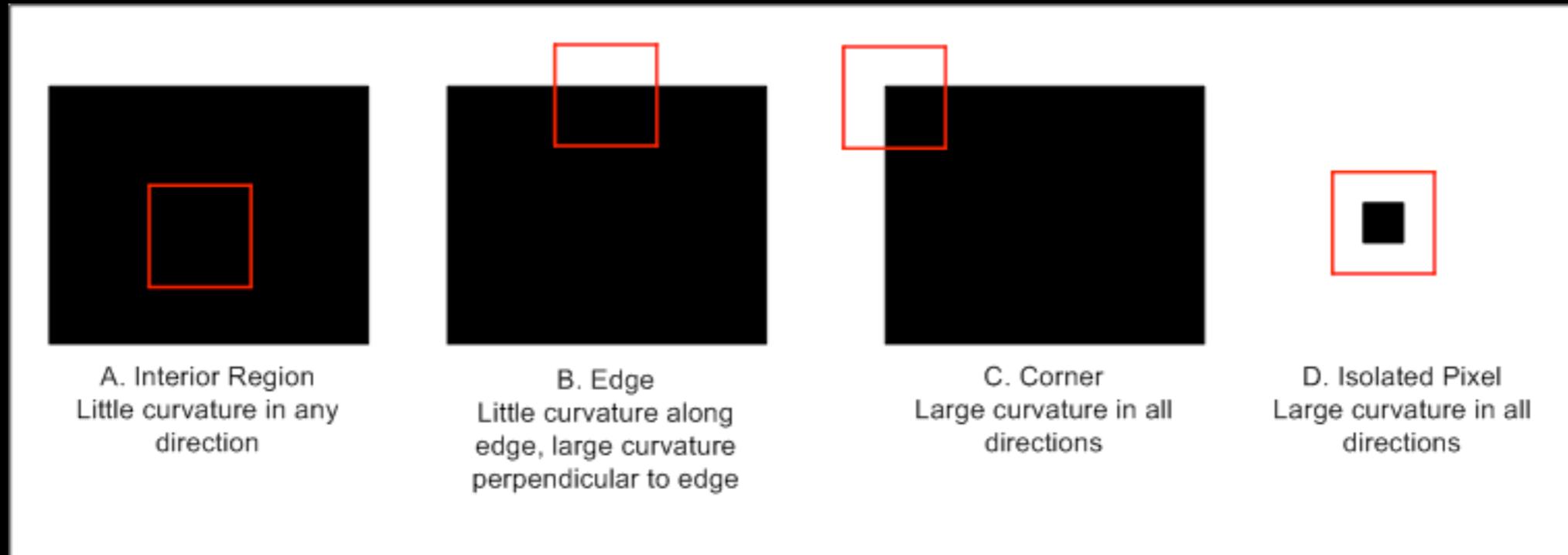
Joshua Spitz

5/13/2010

Motivation for vertex finding

- Determine if neutrino-candidate's vertex is inside fiducial volume.
- Assist in matching tracks in 3D.
- Seed cluster finding and track fitting algorithms.
- Seed final-state-interaction characterization algorithm.
- ...

The Harris corner detector*

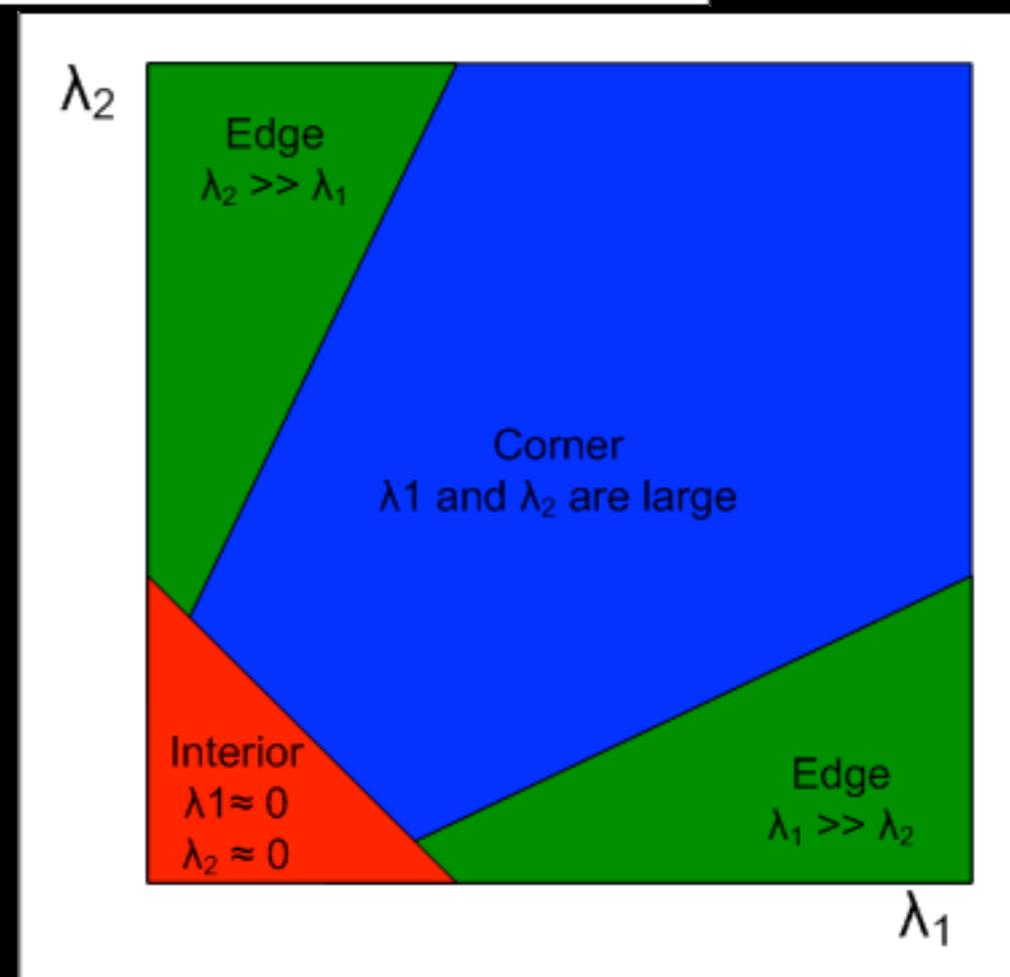


$$H(x, y) = (\Delta x \Delta y) A \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$A = w(x, y) \begin{pmatrix} (\partial_x)^2 & (\partial_x)(\partial_y) \\ (\partial_x)(\partial_y) & (\partial_y)^2 \end{pmatrix}$$

Δx = shift in x $w(x, y)$ = 2D Gaussian weight

∂_x = intensity variation in x direction λ = eigenvalues of A

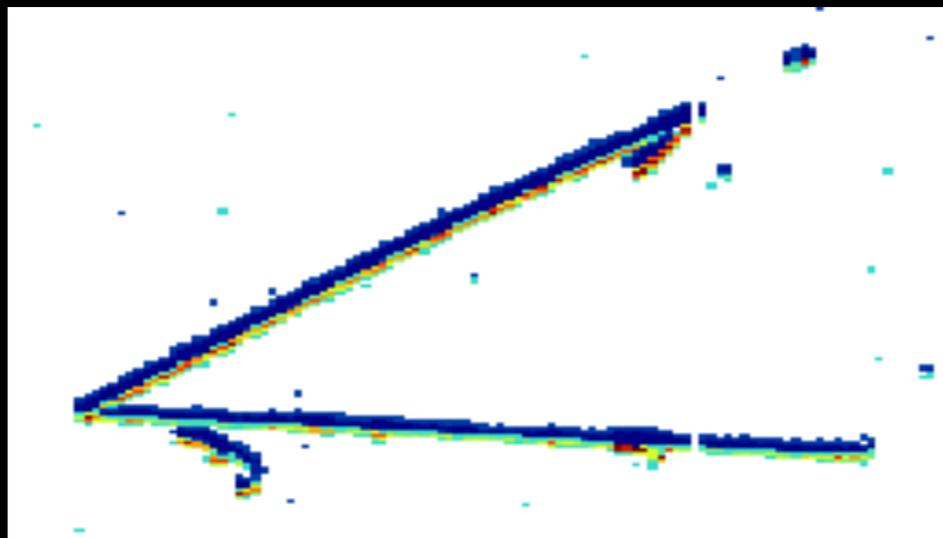


*Motivated by U. Warwick group

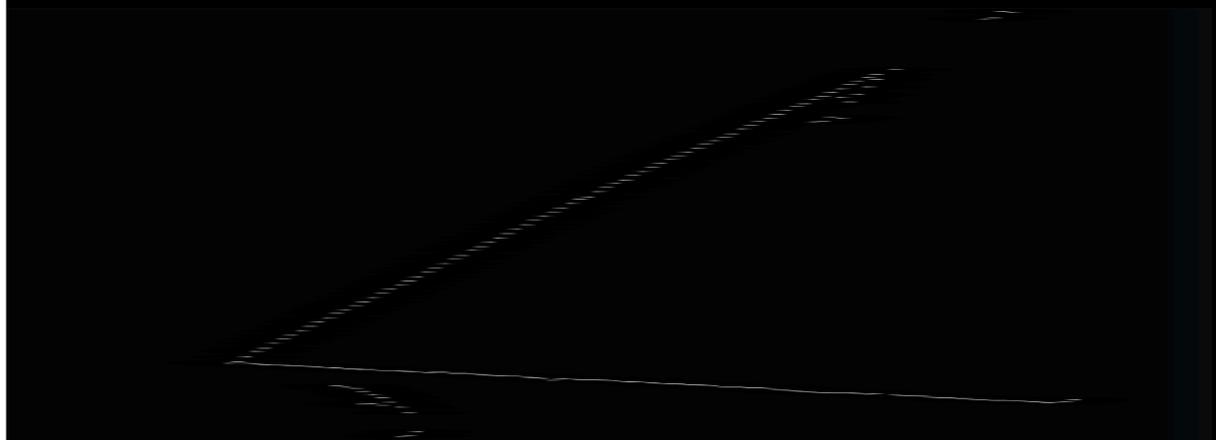
The corner detector input

- The Harris corner detector is inherently very sensitive to noise. An actual raw data image of the event will not work as an input to the algorithm.
- Instead of raw data, the vertex finder takes hits that have been associated with DBSCAN clusters (noise-free hits) as an input.

Raw data

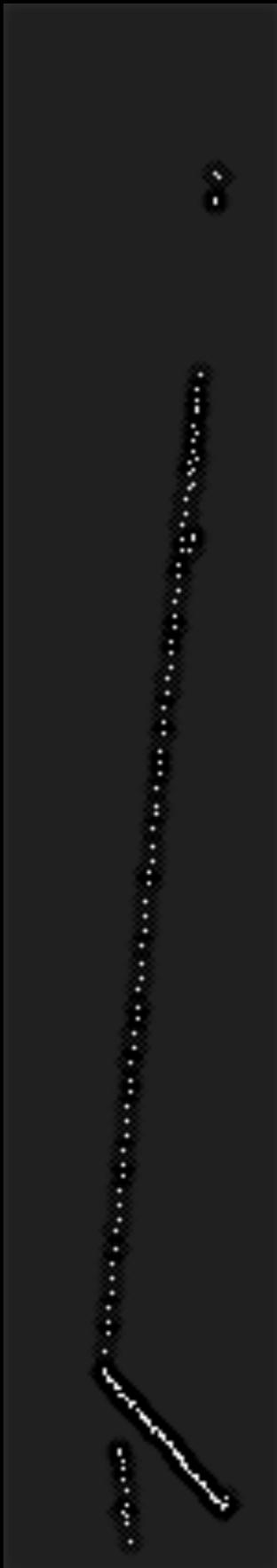


DBSCAN Hits

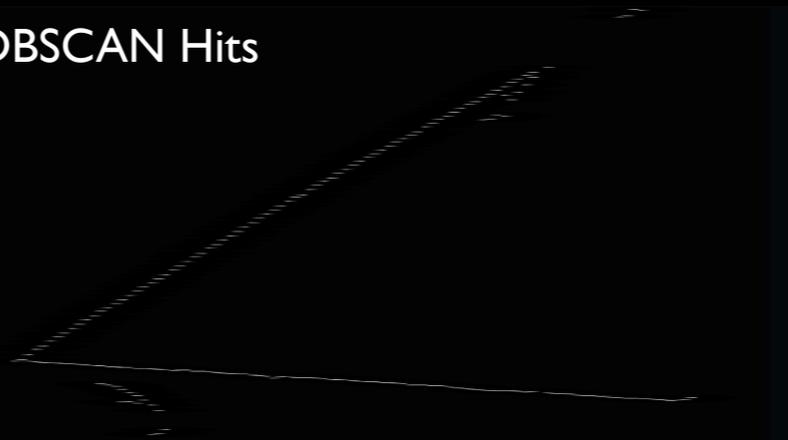


Pixelization

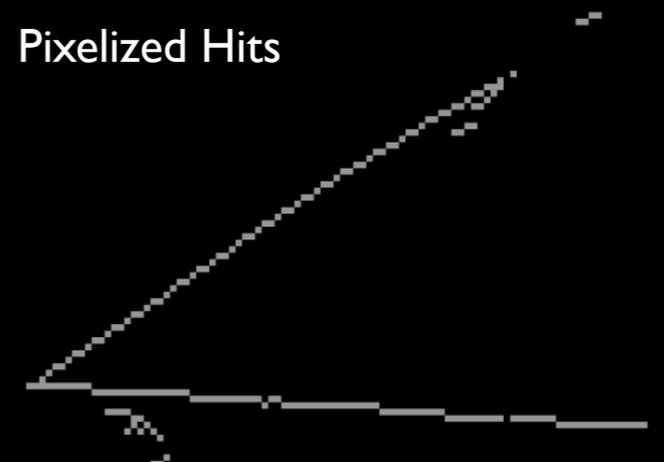
- The problem with inputting the DBSCAN hits (wire, crossing time) directly is that there is a lot of empty space in between hits. This empty space can confuse the corner finding algorithm.
- Pixelization is necessary to ensure equal hit sizes and minimal empty space in between adjacent hits on a track.
- Pixelization is done in the time direction only and is equivalent to root's ReBin(n) method. That is, it breaks up the 2048 time samples into $2048/n$ bins
- Perhaps a more sophisticated smoothing algorithm will work better.



DBSCAN Hits



Pixelized Hits

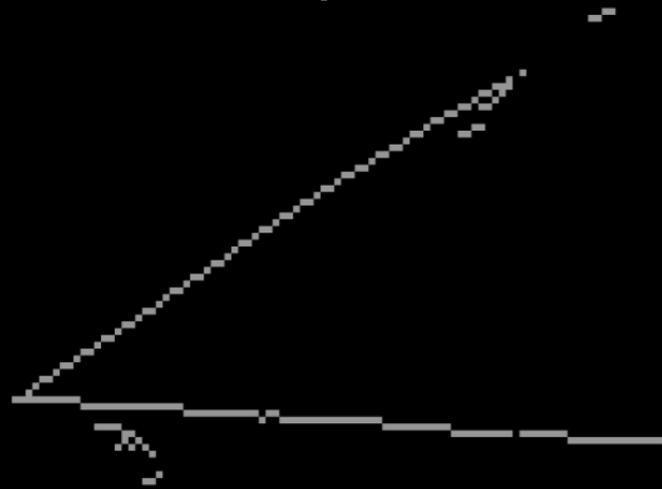


A stretched view of the event.

Remember that there are 2048 samples in time and 240 samples in wire.

Difficulties

Input

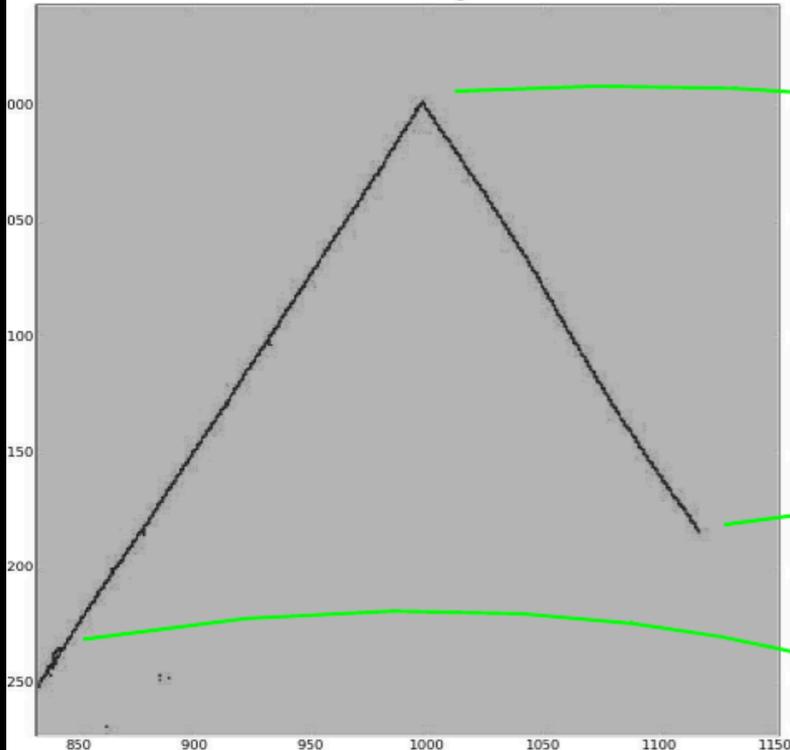


J. Spitz, Yale

Harris Response

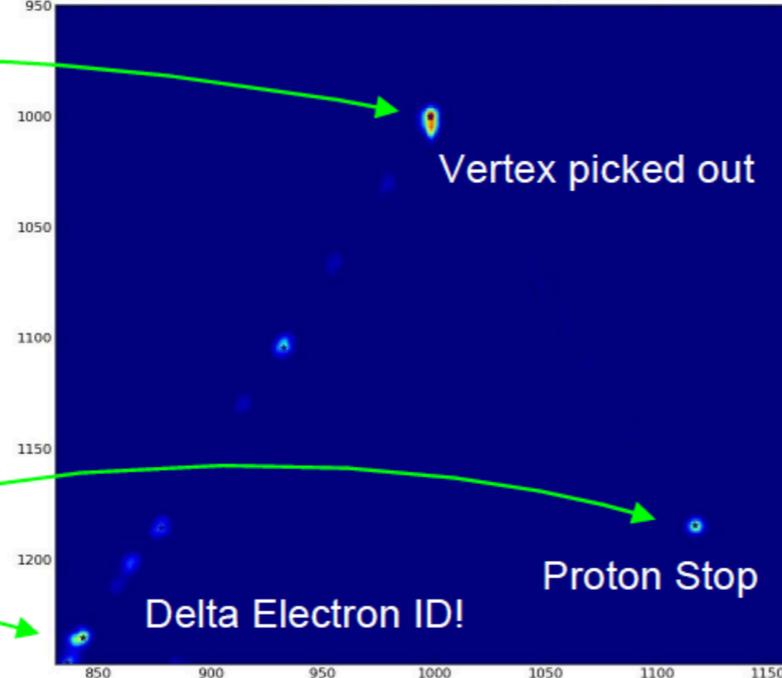


Flattened Image



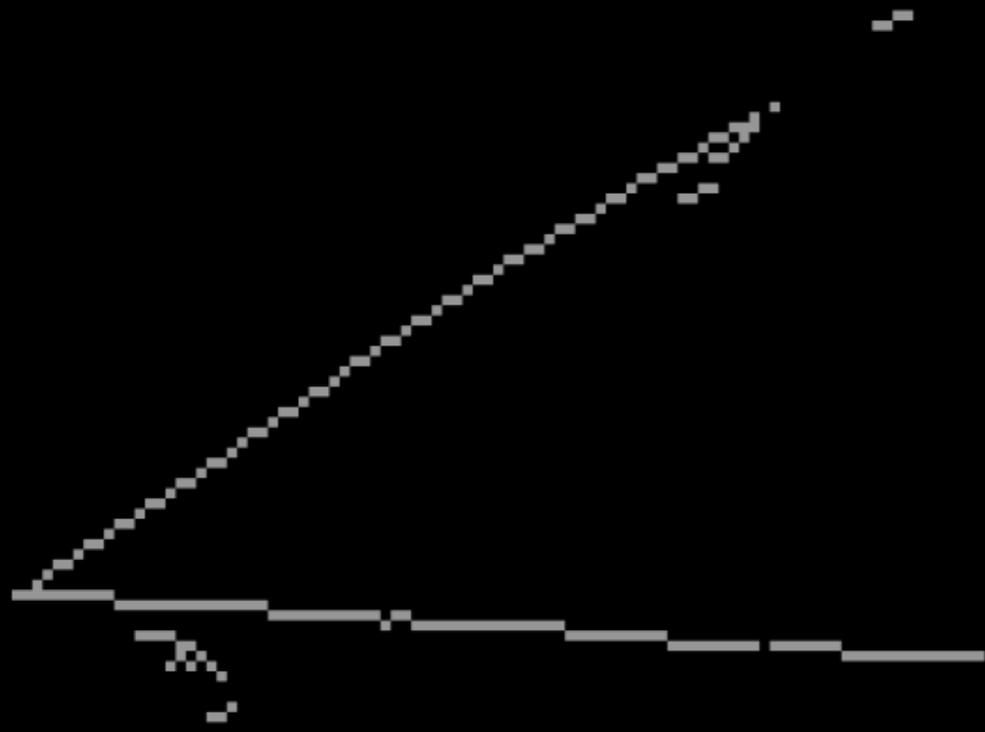
Ben Morgan, Warwick

Harris Response for Flattened Image

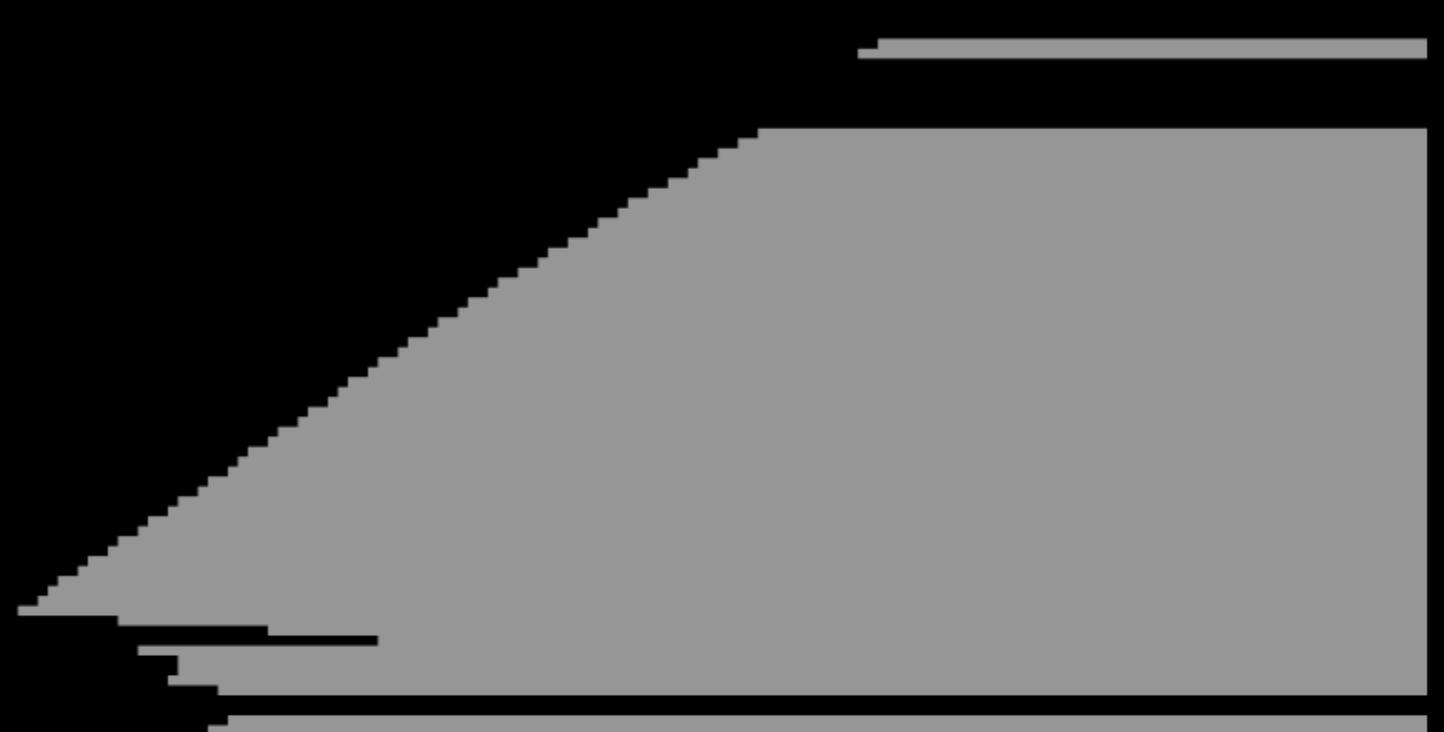


- I have found that the Harris algorithm is quite sensitive to diagonal (jagged) tracks and will often find spurious corners at these jagged edges.
- This may be a product of a poor parameter choice. The k-value, Gaussian window sigma, and pixelization factor are tunable parameters in the algorithm.
- An improved pixelization (smoothing) algorithm would reduce the jaggedness of the diagonal.
- Perhaps the Warwick group can comment?
- It seems to me that some sort of shading is required for the algorithm to work to its full potential.

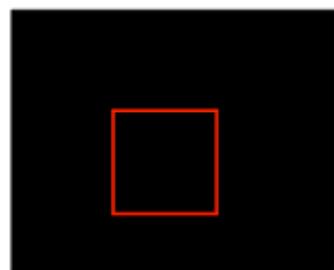
Making the corner finding algorithm's job easier with shading



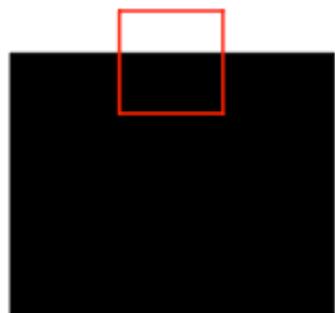
Pixelized, binary hits



Pixelized hits with shading in the positive wire direction



A. Interior Region
Little curvature in any direction



B. Edge
Little curvature along edge, large curvature perpendicular to edge



C. Corner
Large curvature in all directions

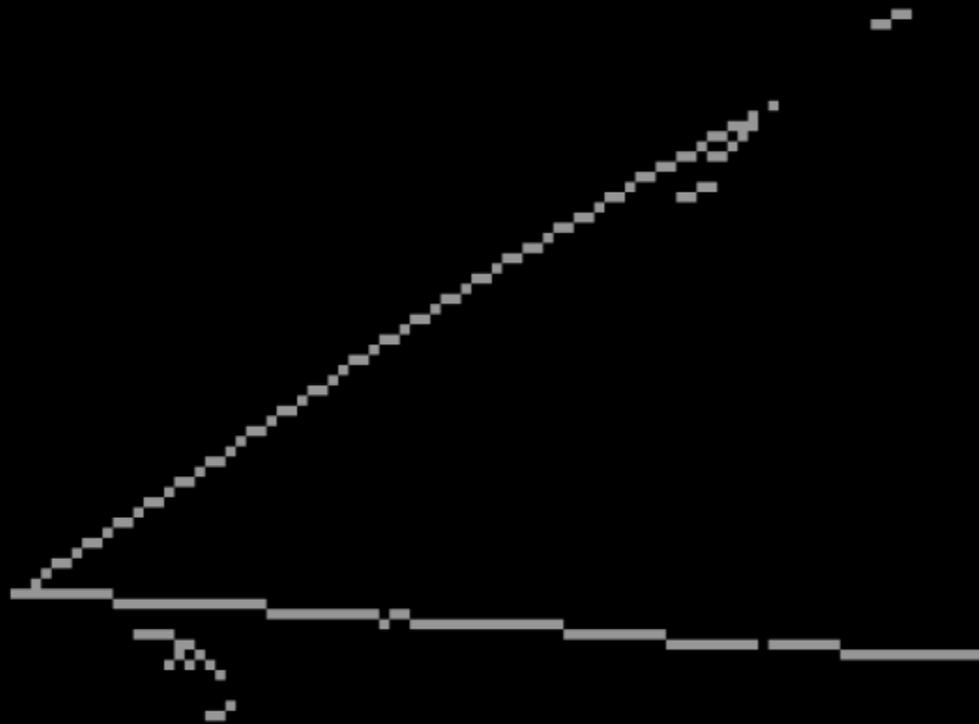


D. Isolated Pixel
Large curvature in all directions

↑
The corner finding algorithm works much better with well defined regions of high and low.

Shading cont'd

- One can imagine a scenario in which complete one-sided shading obfuscates the vertex. A one-sided Gaussian shading is employed instead in order to reduce the chance of this.

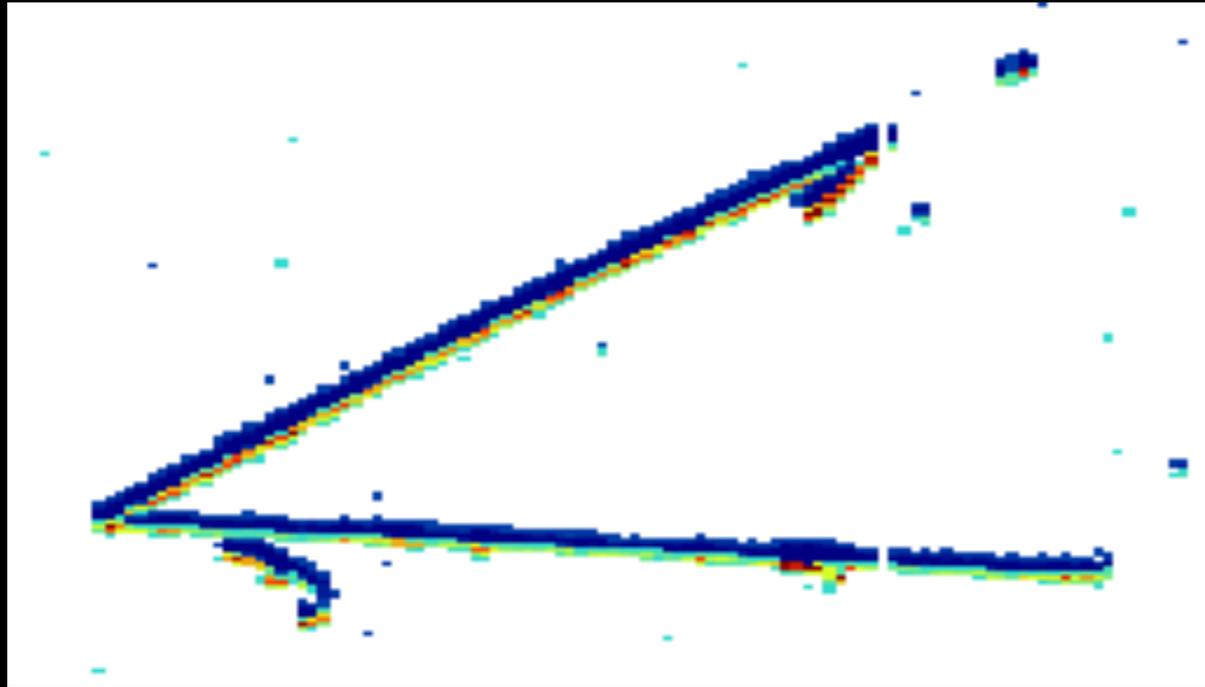


Pixelized hits

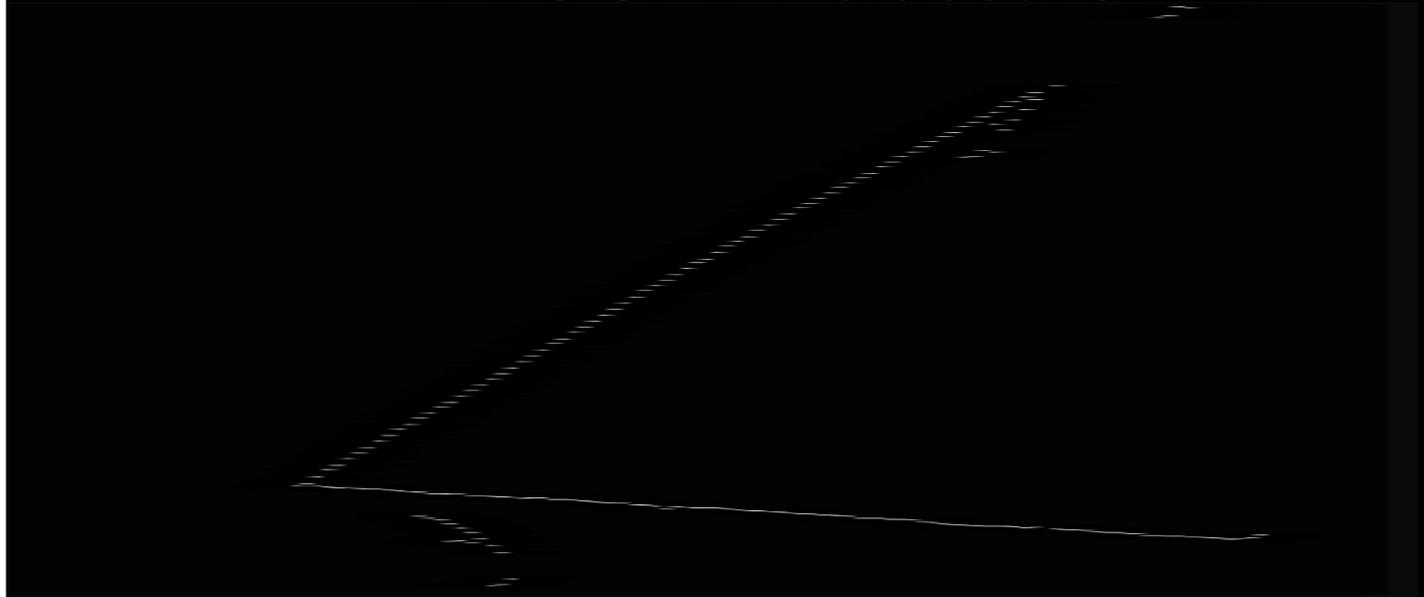


Pixelized hits with Gaussian shading in the positive wire direction

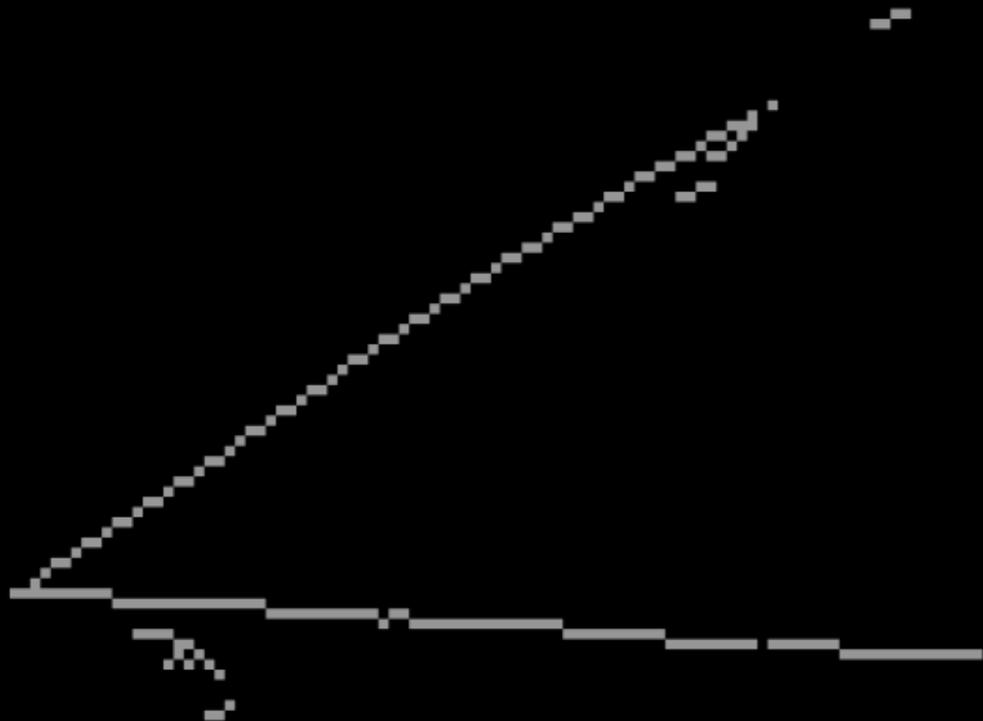
Step 1. Raw data



Step 2. Hits associated with DBSCAN clusters



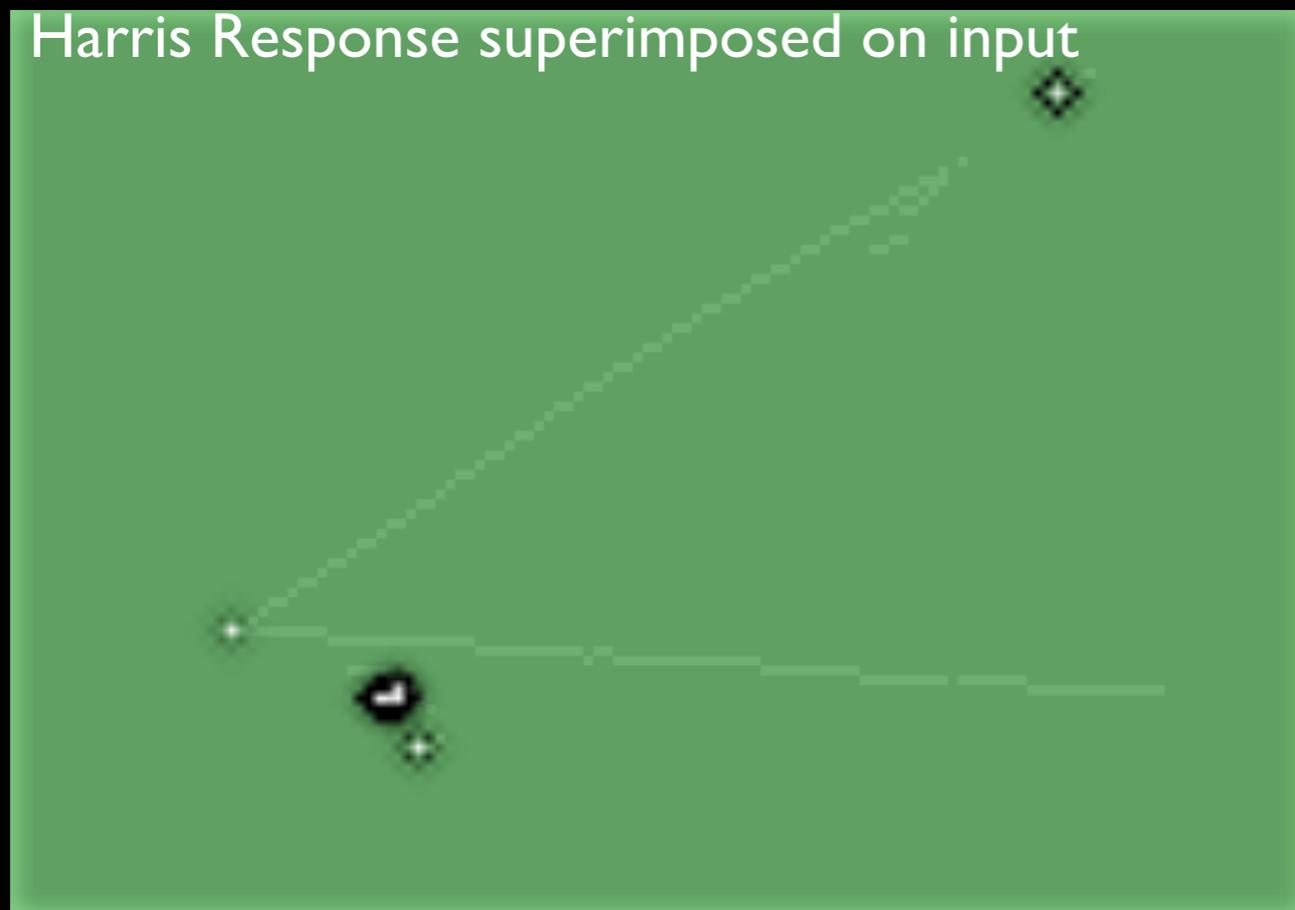
Step 3. Pixelized, binary hits



Step 4. Pixelized hits w/ one-sided Gaussian shading



The Harris response after shading



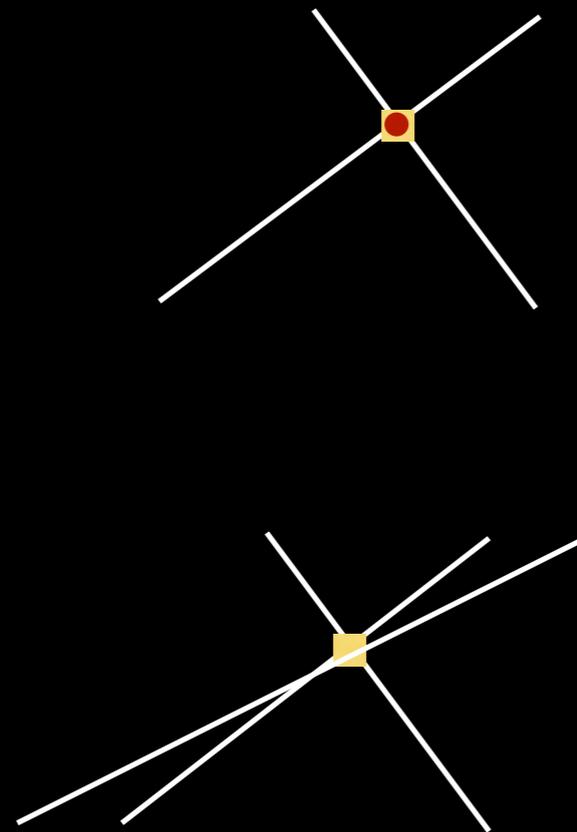
- After shading, the Harris response no longer includes those hits along the jagged diagonal.
 - The “true” neutrino vertex is found along with the other (neutron-induced? de-excitation γ ?) vertices.
-
- Note that all Harris vertices are required to fall within the time (endtime-starttime) of an existing hit.

Strong/weak vertex definition

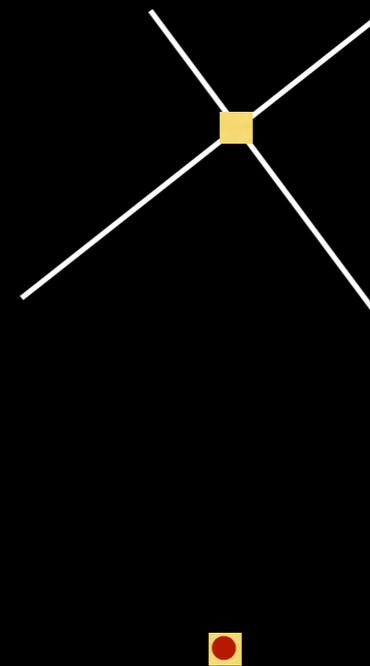
- A missed vertex seems much worse than a spurious vertex. For this reason, I have employed the concept of “weak” and “strong” vertices, with the Hough line finder working together with the Harris vertex finder to categorize the vertex in question.

- Window around a hit
- Harris-vertex
- Hough-line (endpoints at +/-inf)

Strong vertices

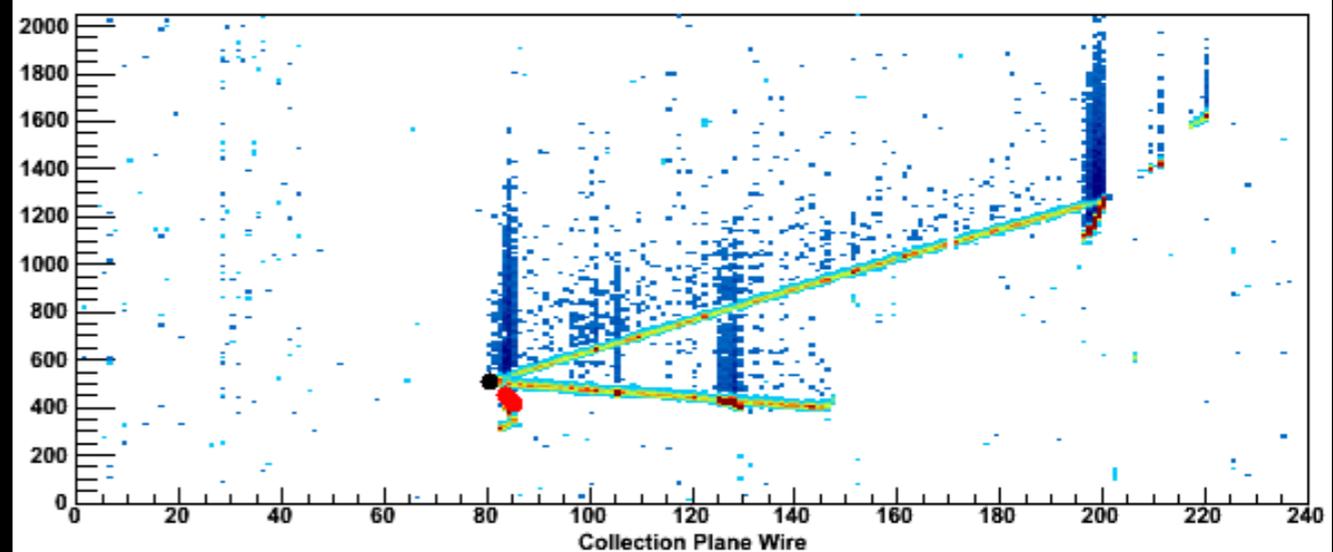
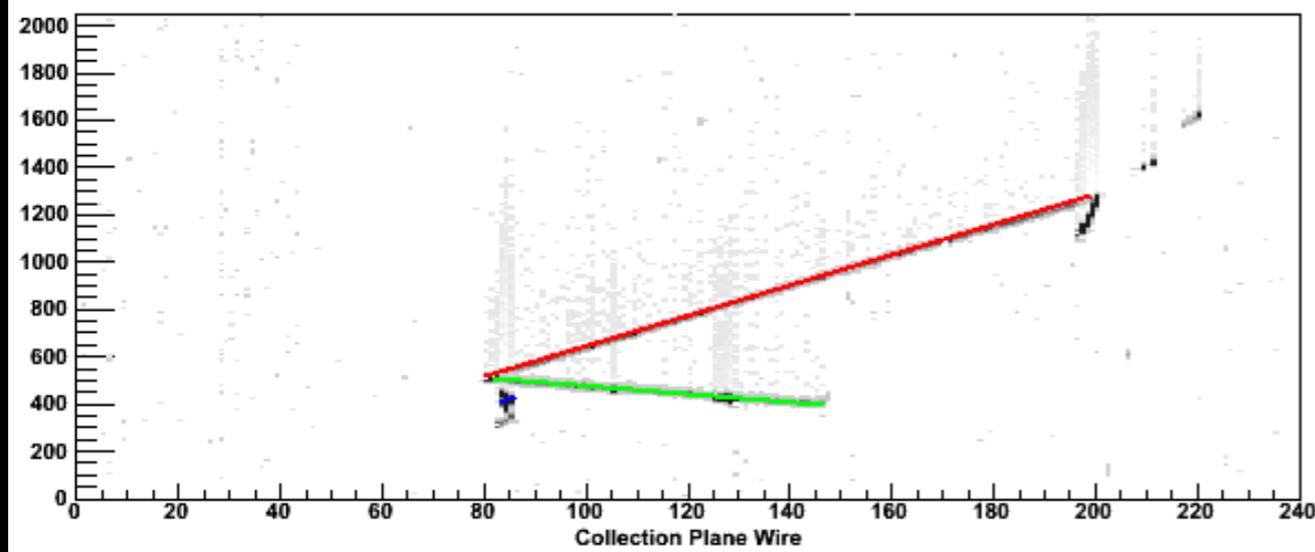
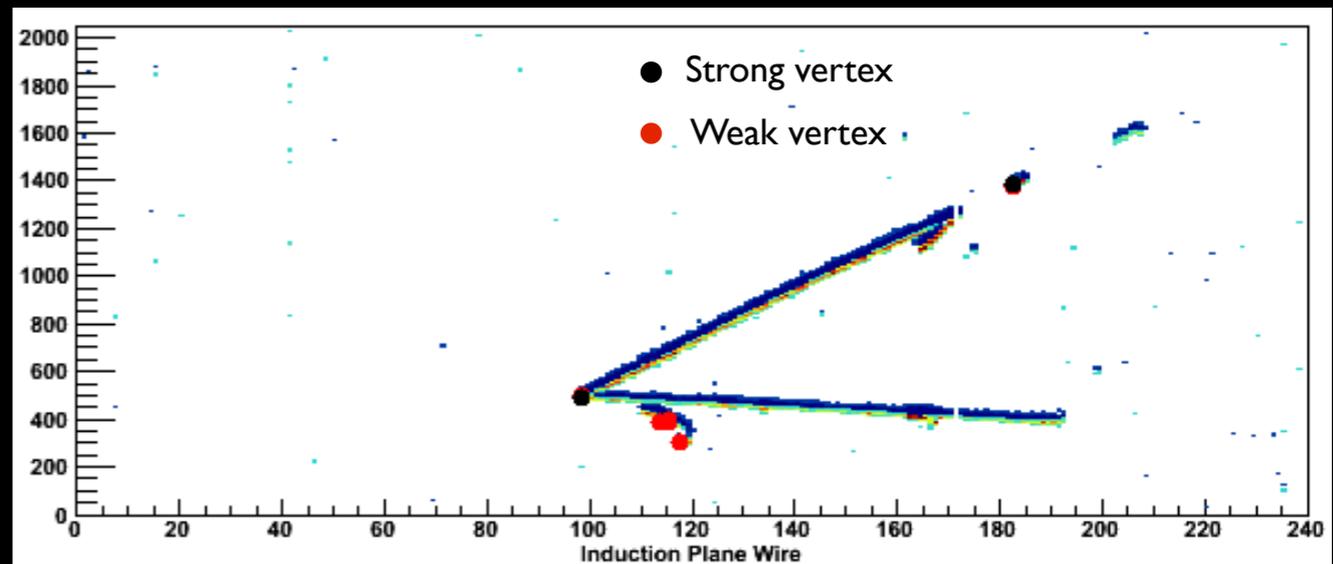
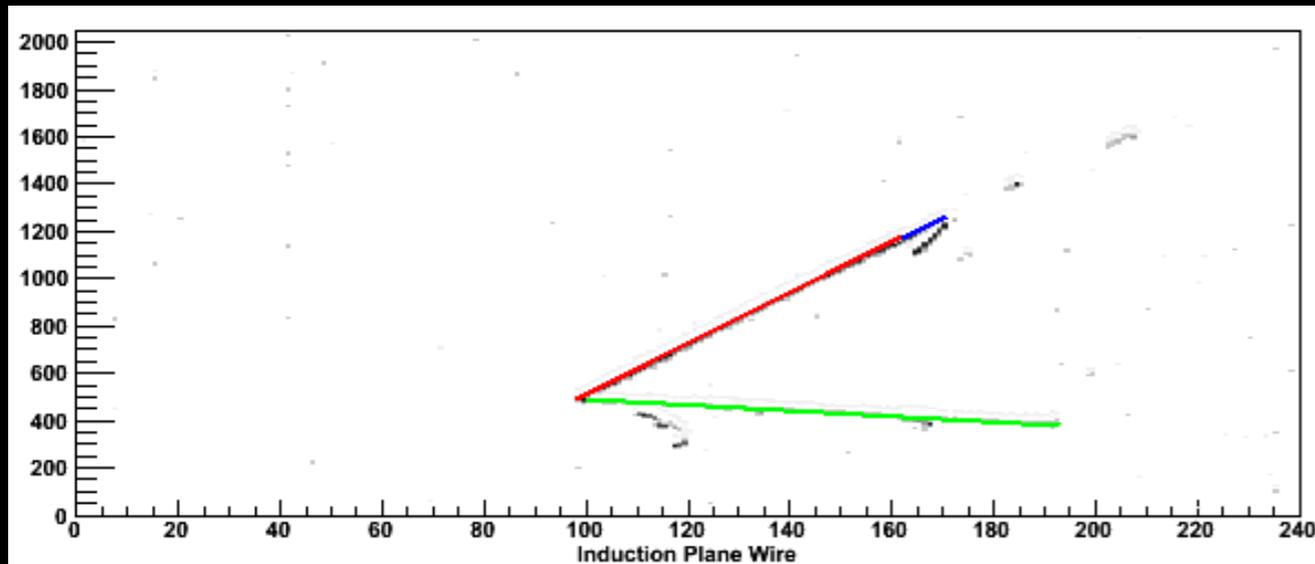


Weak vertices



Strong and weak vertices

- The VertexMatch module in VertexFinder matches hits, Harris vertices, and Hough line crossings, given a proximity requirement.



Hough lines

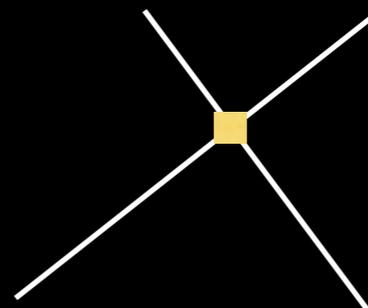
Vertices

HarrisVertexFinder

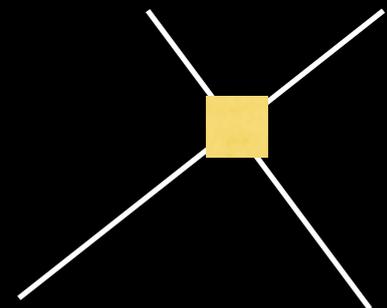
- A LArsoft module to find vertices.
- xml parameters:
 - kValue, the parameter that defines the contours of the eigenvalue space in determining corneriness.
 - TimeBins, the number of time bins to use in pixelization.
 - Gsigma, sigma of the Harris algorithm's Gaussian window.
 - Gsigmasmear, sigma of the one-sided Gaussian shade.
 - MaxCorners, maximum number of vertices to find.

xml parameters for VertexMatch

- A LArSoft module to match hits, Hough line crossings, and Harris vertices and define the strength of a vertex.
- xml parameter:
 - Window size for hit-line match



vs.



To do

- Make pixelization (smoothing) and/or shading more sophisticated.
- Match vertices in 3D.
 - strong vertex in one view + weak/strong vertex in another view = 3D vertex, if “close” in time.
- MC comparison (need matching in 3D first)