

USING A HARRIS-STEPHENS ALGORITHM TO FIND “CORNERS” IN RAW DATA

WESLEY KETCHUM
LANL

OVERVIEW

Basic theory and what is currently in LArSoft

Implementation on RawDigits

First results

- ...on all of one event

Conclusions

- Putting it into LArSoft
- Items/methods to be tested
- Ideas on 3D

IMAGE PROCESSING: STRUCTURE TENSOR

Harris-Stephens is an image-processing technique based on locating large changes in “intensity”

- Take digitized 2D-image
- Create partial derivative images for both dimensions
 - I'll call the directions x and y
- Construct “structure tensor” over some local neighborhood of pixels

$$A = \sum_{u,v} \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

WITH THAT STRUCTURE TENSOR

Analyze the eigenvalues of A

- The eigenvalues tell you how fast things are changing, and in how many dimensions
- $\lambda_1 \sim \lambda_2 \sim 0$
 - No big changes – things look similar in every direction
- $\lambda_1 \gg \lambda_2 \sim 0$
 - See large directions along one direction (given by the eigenvector...), and not so much along the other
 - This would be an edge
- $\lambda_1 \sim \lambda_2 \gg 0$
 - Big changes in intensity along all directions – a corner!
 - Or an endpoint, or a singularity

HOW TO ASSIGN A SCORE TO EACH PIXEL

Harris-Stephens

- $\det(\mathbf{A}) - \kappa \cdot \text{tr}(\mathbf{A})^2 = (\lambda_1 \lambda_2) - \kappa (\lambda_1 + \lambda_2)^2$
 - κ is just empirically determined; typically ~ 0.05
 - For noise-space, score ~ 0 ; for edges, score < 0 , and for corners, score > 0

Noble (?)

- $\det(\mathbf{A}) / \text{tr}(\mathbf{A}) = (\lambda_1 \lambda_2) / (\lambda_1 + \lambda_2)$
 - For noise, score ~ 0 ; for edges, score is smallest eigenvalue; and, for corners, score \sim eigenvalue magnitude

Shi-Tomasi

- $\min(\lambda_1, \lambda_2)$
 - More computationally intensive than the others...

DOES IT WORK?

Sure it does!

- Widely-used in image processing

Of course, there are other methods too

- Second-derivative approaches
- Template-matching



<http://glowingpython.blogspot.com/2011/10/corner-detection-with-opencv.html>

COULD WE USE IT IN LAR TPC RECONSTRUCTION?

Sure we can!

In fact, already exists as a vertex-finder

- My understanding of `HarrisVertexFinder_module`
 - Takes in hits from clusters
 - Fills 2D images with Gaussian centered around hit center
 - Uses Noble score as measure of cornerness
 - Matches every found corner to a hit
 - Sends out list of corners up to specified maximum

So, can we do better?

- I don't know...but, this feels unnatural to me

TREATING OUR DETECTOR LIKE A CAMERA

- We take NPlanes 2D pictures
 - x = wire direction, y = time direction

Why not apply algorithm to raw or calibrated data?

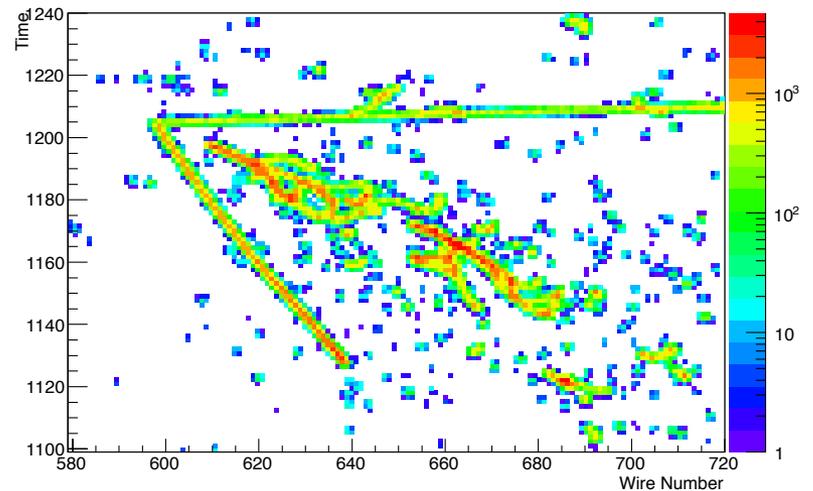
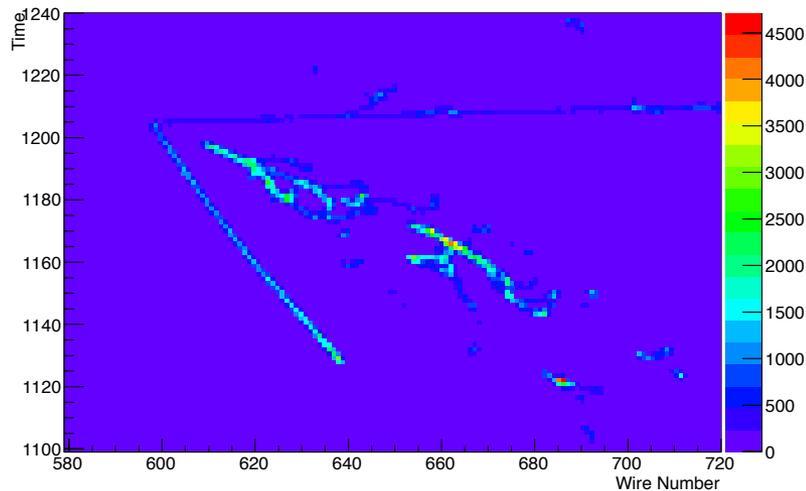
- Benefits:
 - No discontinuities that aren't real properties of the event
 - Use all of the data possibly available to you
 - More time-efficient
 - data \rightarrow hits \rightarrow clusters \rightarrow hits \rightarrow (less) data \rightarrow corners ?
 - Let's you start to dream about low-latency feature finding
- Drawbacks
 - None that I've thought of (*i.e.* I haven't tried to think of any)
 - Well, matching to hits could be an issue

TRIAL RUN ON RAWDIGITS

Got (1) CCQE event from Jonathan Asaadi

Looking at collection plane

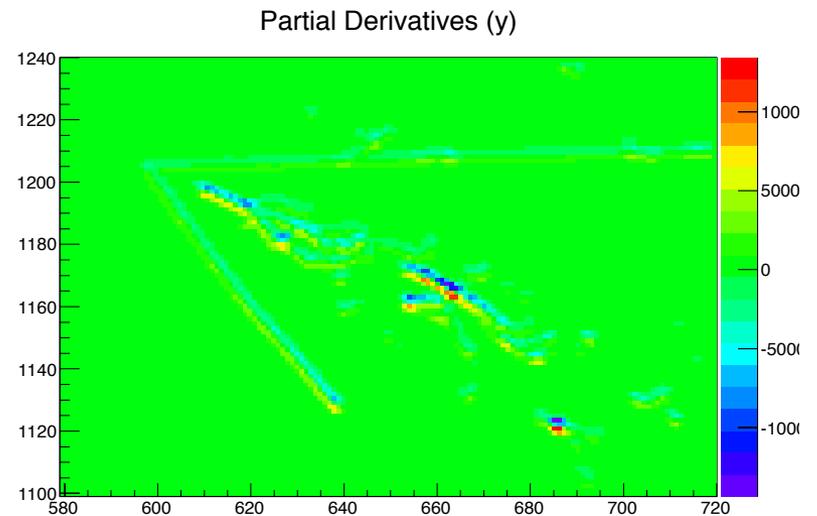
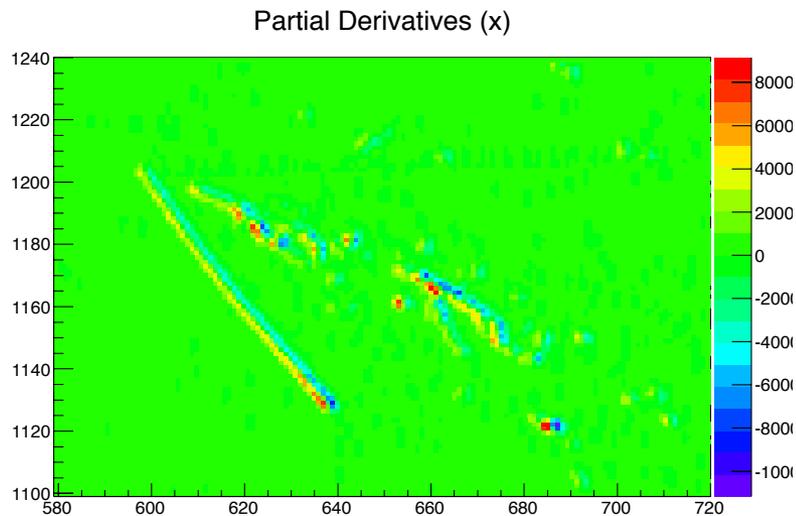
- Collapsed time bins to match spatial range of wire pitch
 - And zoomed in on the action



NEXT, TAKE THE DERIVATIVE

There are a number of ways one can do this

- I am using a “Sobel” mask, which looks at 3x3 pixel region, to get somewhat smoothed derivative
 - This is standard, but it’s not clear if it’s the best for *us*



RESULTS OF ANALYZING EIGENVALUES

Figure of Merit (Harris)

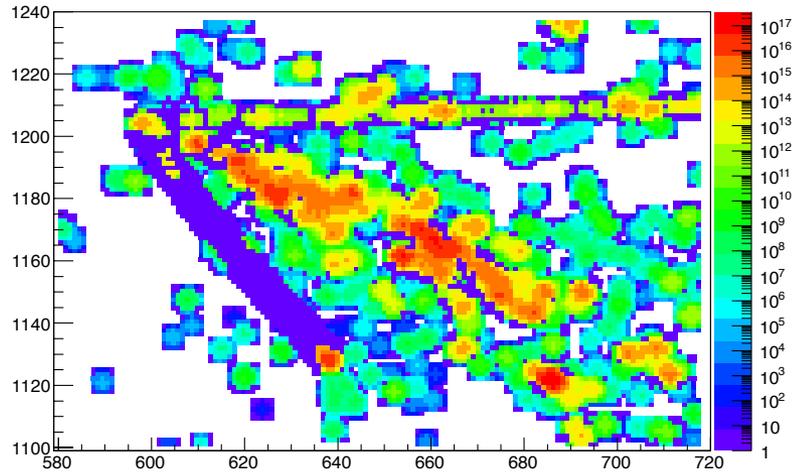


Figure of Merit (Noble)

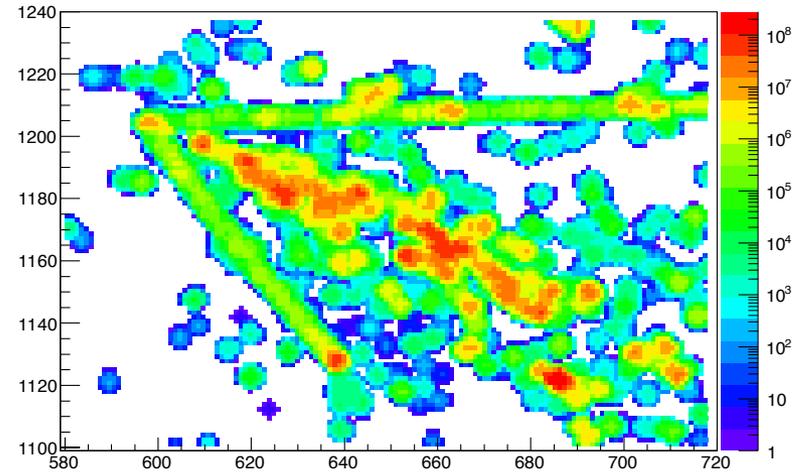
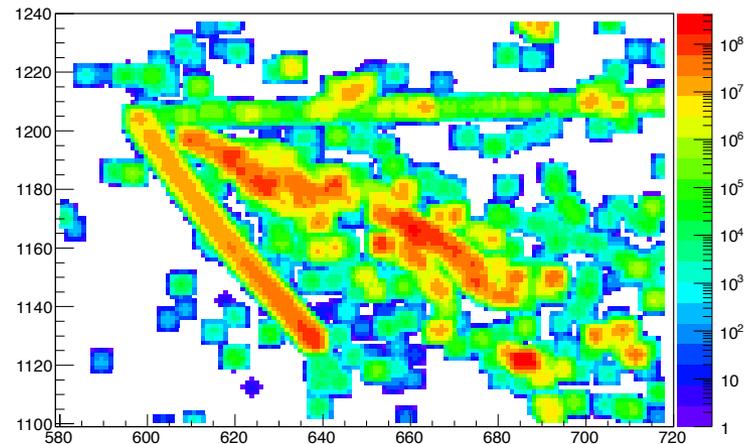


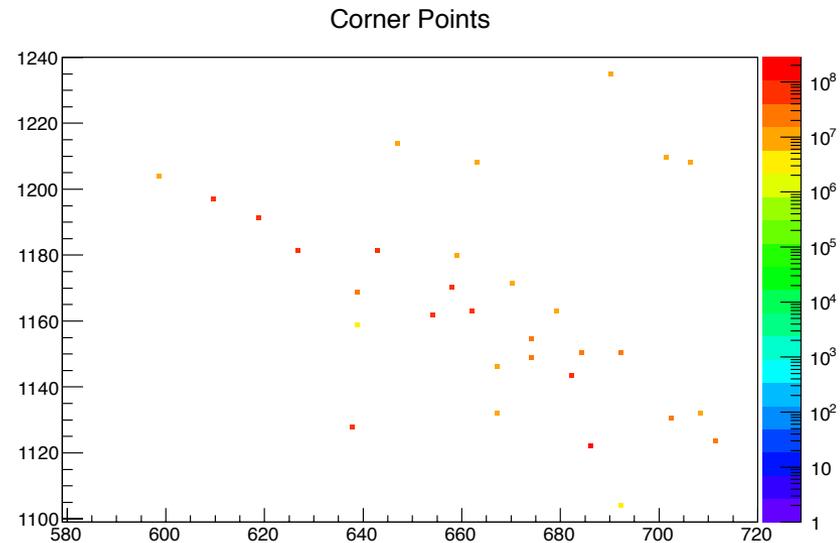
Figure of Merit (Shi-Tomasi)



REDUCE TO FIND LOCAL MAXIMA

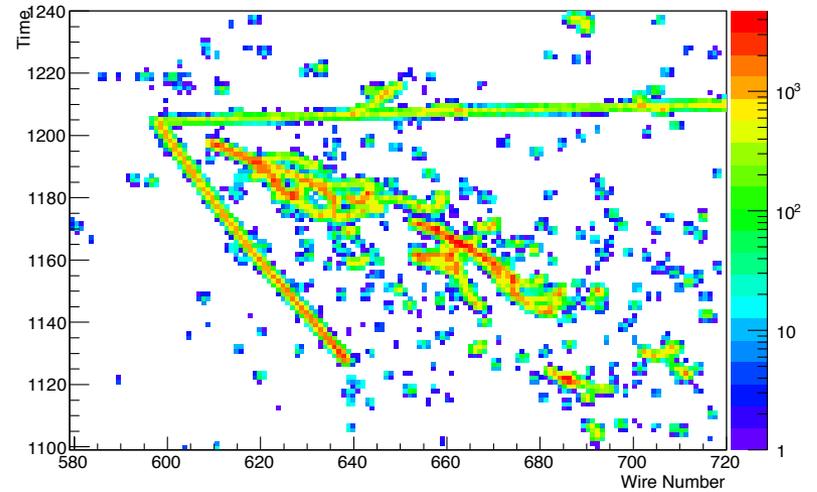
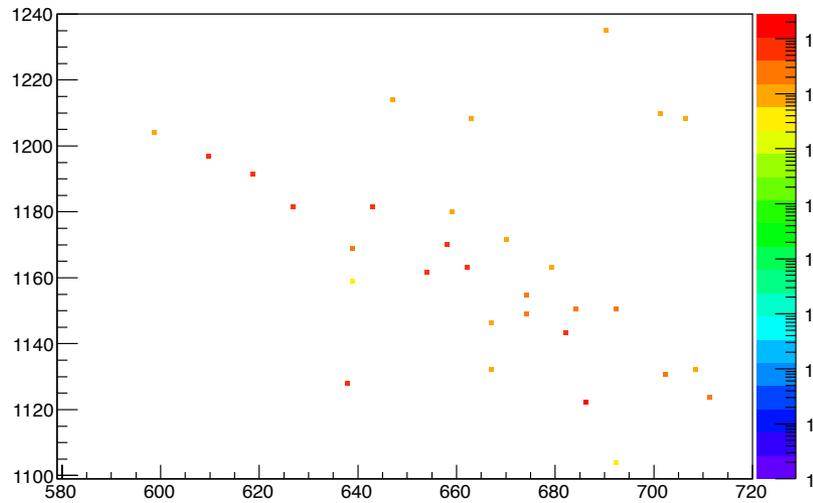
- Set a minimum threshold of interest, and a neighborhood around which to select only local maxima

- Shown here
 - Noble score
 - threshold $> 5e6$
 - neighborhood of 3 pixels
 - Total of 31 feature points



SIDE-BY-SIDE WITH RAW DATA

Corner Points



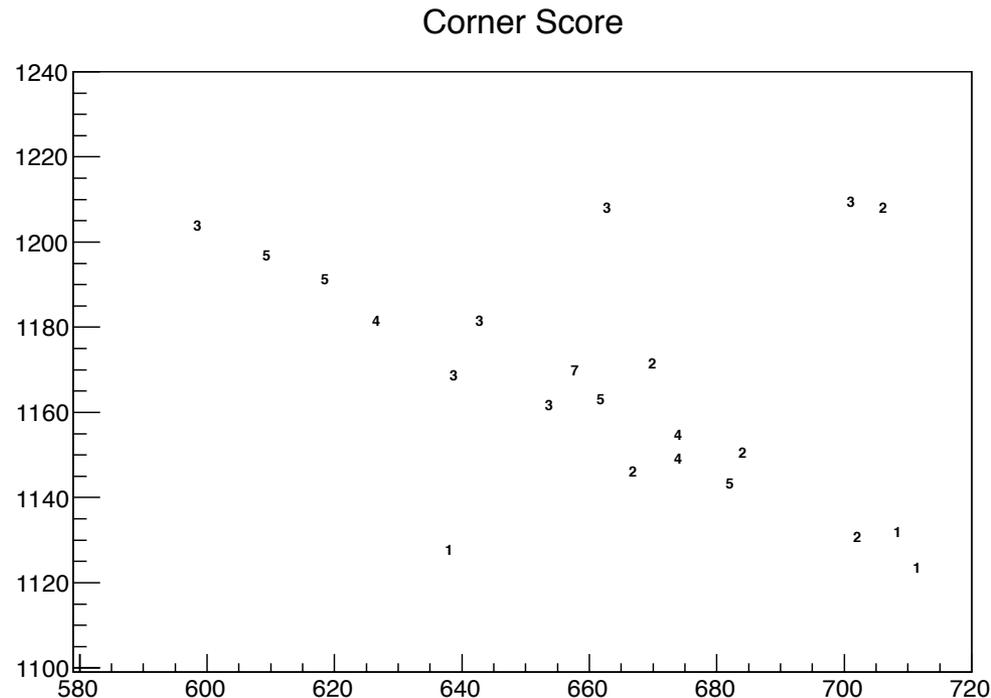
OK, NOW WHAT?

Up to everyone else

Here's what I've started playing with: finding primary vertex by performing "path integrals"

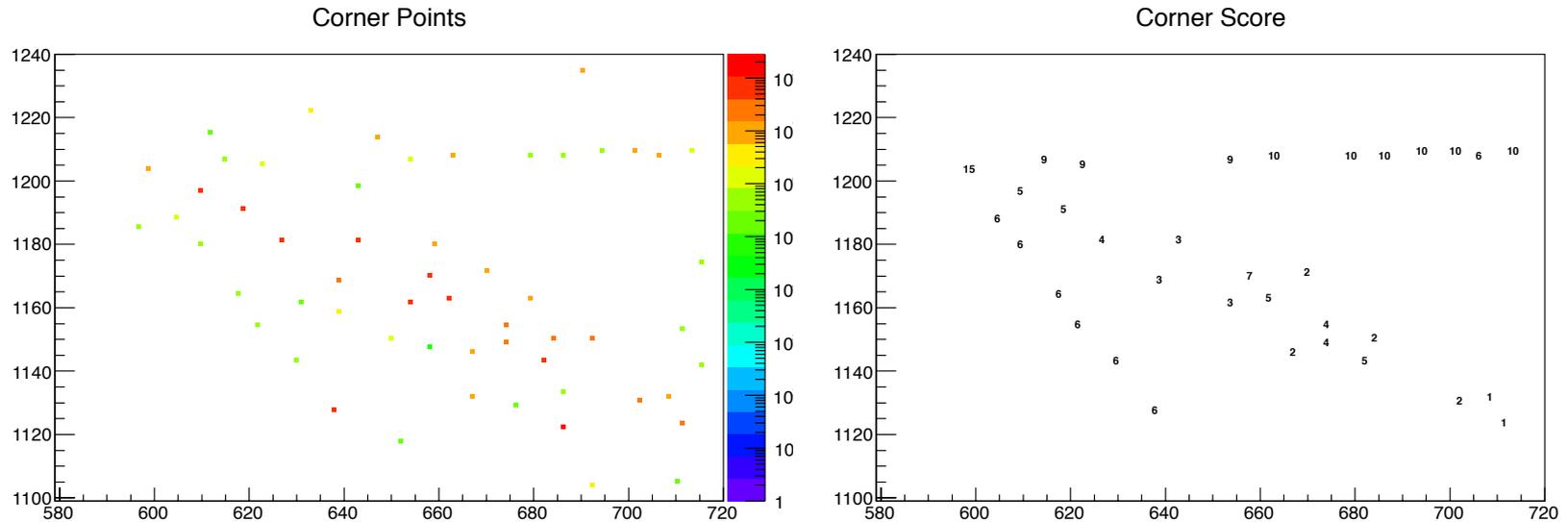
- Idea: lines between feature points should be populated consistently by charge
- So...draw a line between all pairs of feature points
 - Keep count of how many lines have most intersected bins (>90%) with some activity (>5 ADC counts for now)
- Vertices should have multiple lines coming out

FOR THE NOBLE SCORE HISTO...



- Getting a lot of stuff in the shower
 - Not surprising
- What happens if we lower the threshold from creating the max suppressed plot?

LOWERING THAT THRESHOLD



- We get more feature points along each track → picking out where the long tracks intersect rather well
 - Likely needs a lot of tuning, but this may be promising

CONCLUSIONS FOR NOW

Working on putting this into LArSoft

- Brian suggested implementing as algorithm
- Ben C., Jonathan, and I put in the bulk of the code last night
 - Needs fixes to make it work

Needs lots of testing/tuning

- Derivatives, corner score, neighborhood sizes, thresholds, etc.

Should be easily extendable to 3D

- Working on framework for that