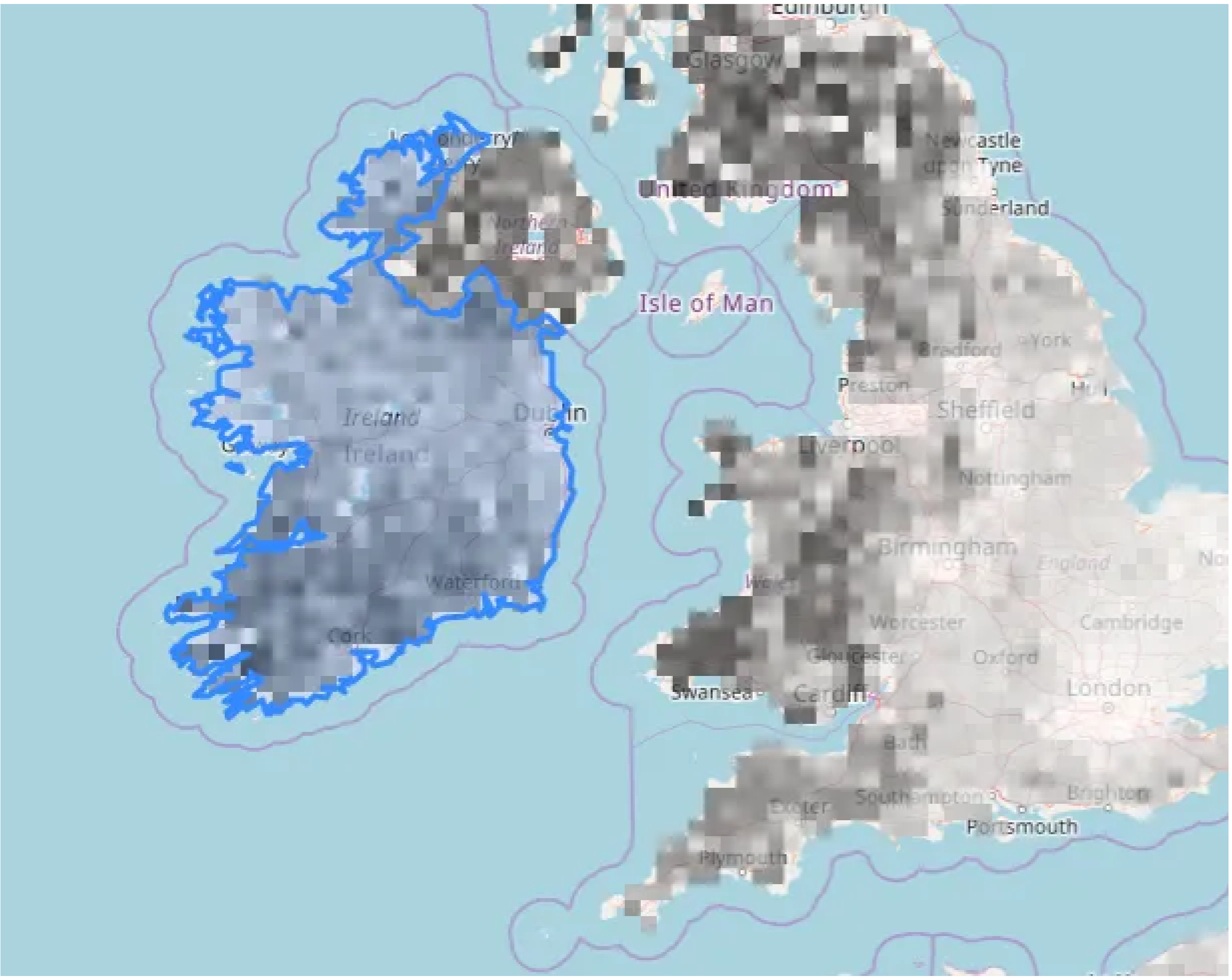
Calculating Intersection of Polygon with a Raster

Daniel J. Dufour · Follow

3 min read · Mar 8, 2018

Stephen Peyton and I recently developed an algorithm for calculating which pixels of a raster intersect an arbitrary polygon. The following is a high-level overview of the algorithm.



A raster image of pastureland with a polygon of Ireland's borders. Raster data is from the Center for International Earth Science Information Network (CIESIN). Basemap is from © OpenStreetMap.

Basic Overview of Algorithm

1. Create a horizontal line for each row of pixels in the raster image
2. Calculate intersections of the horizontal lines and the polygon's edges
3. Group edges by intersection
4. Determine whether edges pass completely through their horizontal lines
5. Take pairs of intersections where edges completely cross a horizontal line
6. **Create Horizontal Lines**

Rasters can be thought of as as a table of pixel values. This two-dimensional organization makes iterating over the rows easy. We convert each row in the table of pixels to a horizontal line. Here's what an image of all the rows converted to horizontal lines looks like.

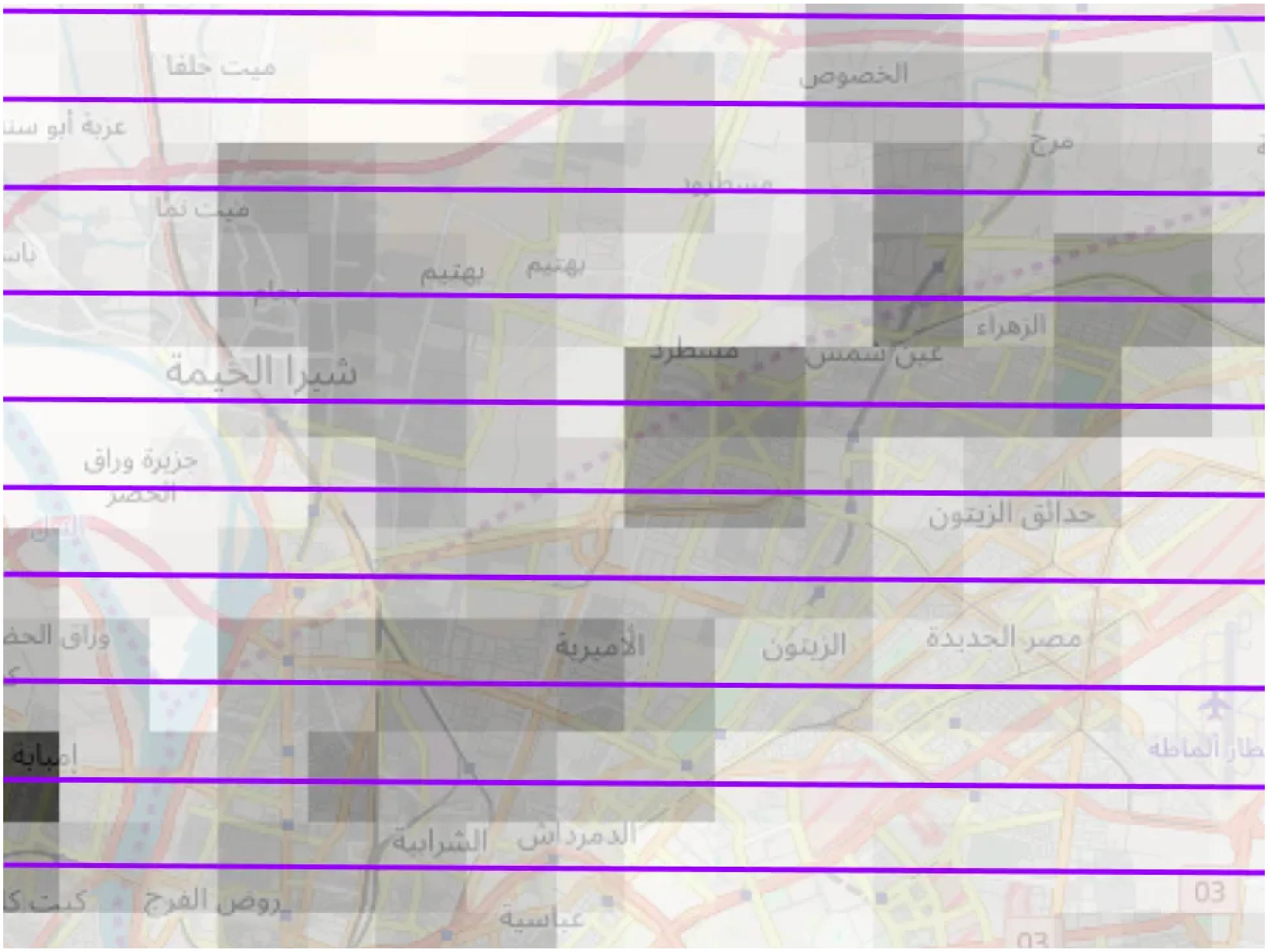
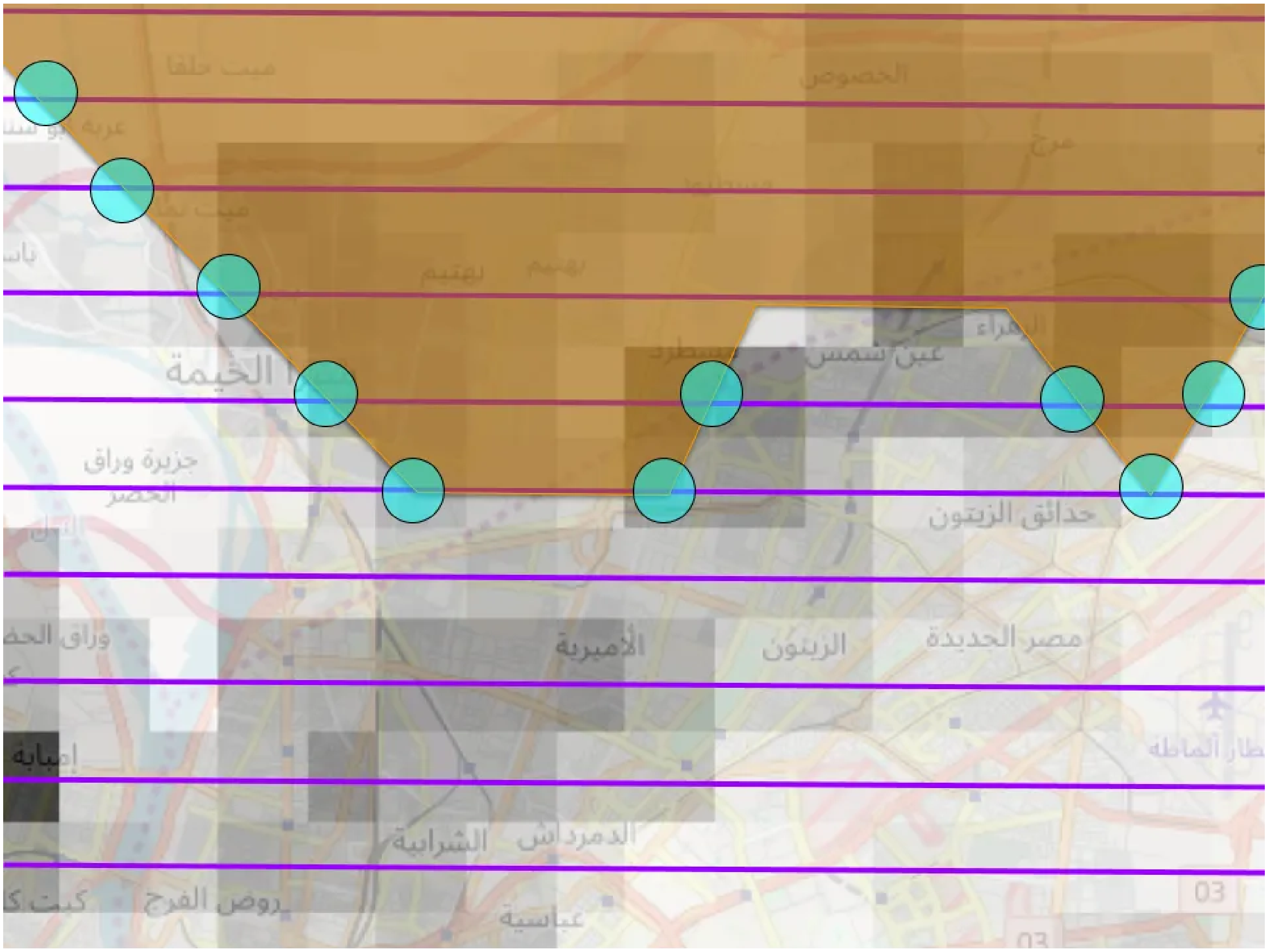


Image of Raster with Purple Horizontal Lines. Raster data is from Global Human Settlement Layer (GHSL). Basemap is from © OpenStreetMap Contributors.

2. **Calculate Intersections of Horizontal Lines and Polygon Edges**

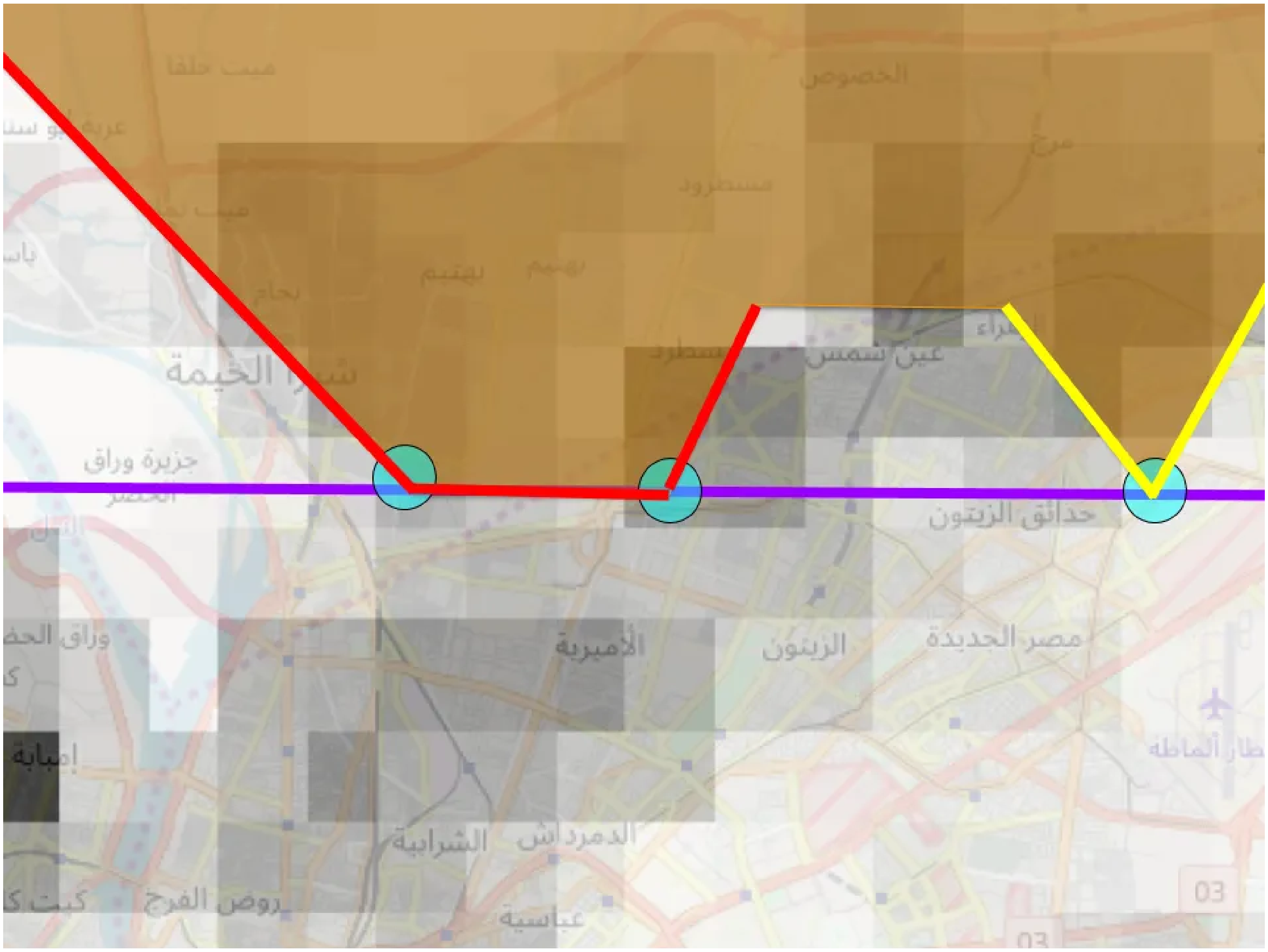
Our next step is to calculate where the horizontal lines intersect with the line segments that make up the polygon. We use your basic linear algebra for these calculations.



Horizontal lines in purple with intersections in light blue. Raster data is from Global Human Settlement Layer (GHSL). Basemap is from © OpenStreetMap Contributors.

3. Group Edges by Intersection

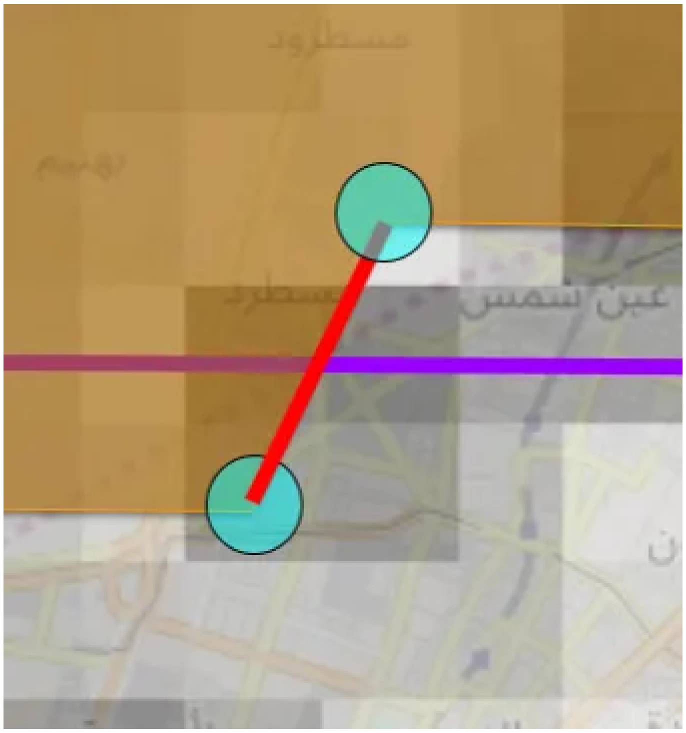
Our third step is to group line segments around consecutive intersections. A new group begins whenever a line segment enters a horizontal line and ends when a line segment exits a horizontal line.



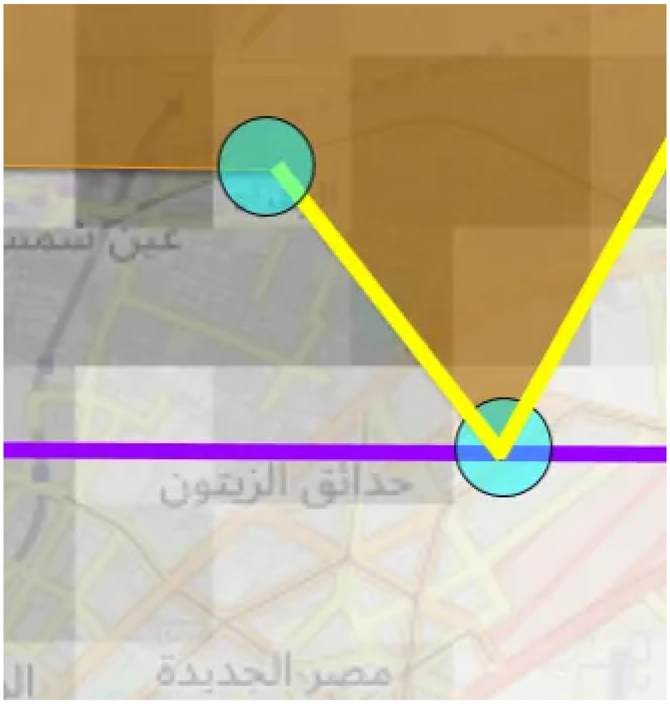
An illustration of line segments grouped by intersection. The first group of line segments is in red. The second group is in yellow. Raster data is from Global Human Settlement Layer (GHSL). Basemap is from © OpenStreetMap Contributors.

4. **Determine Throughness**

The next step is to iterate through the groups and determine whether the line segments of the polygon completely pass through the horizontal line or not.



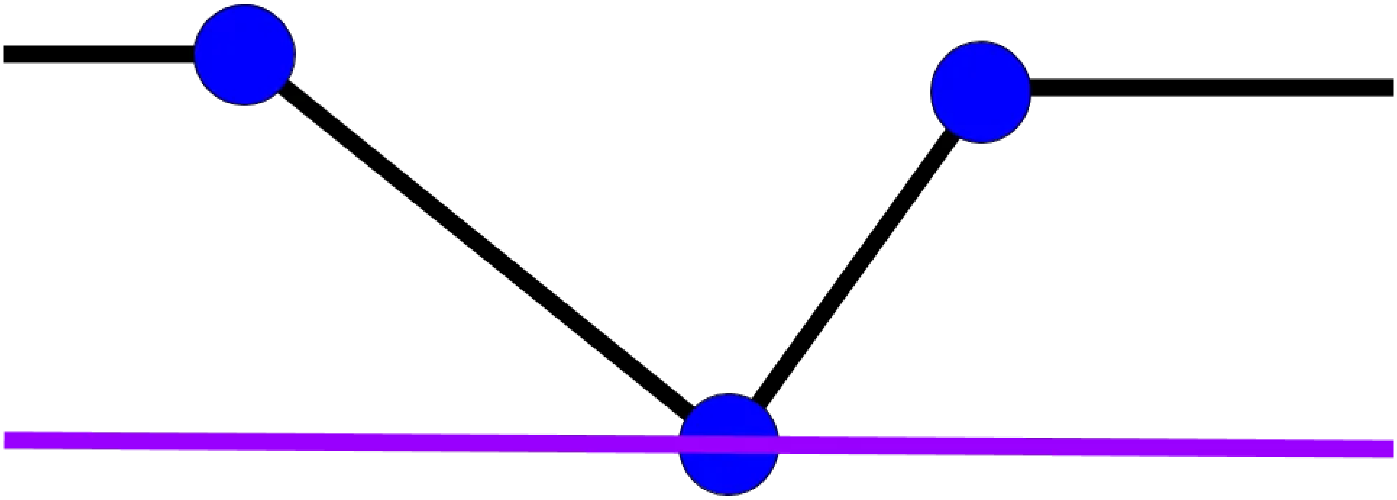
Raster data is from Global Human Settlement Layer (GHSL). Basemap is from © OpenStreetMap Contributors.



Raster data is from Global Human Settlement Layer (GHSL). Basemap is from © OpenStreetMap Contributors.

Example of a Line Segment in Red Running Completely Through a Horizontal Line in Purple.

Example of a group of yellow line segments not making it through a purple horizontal line.

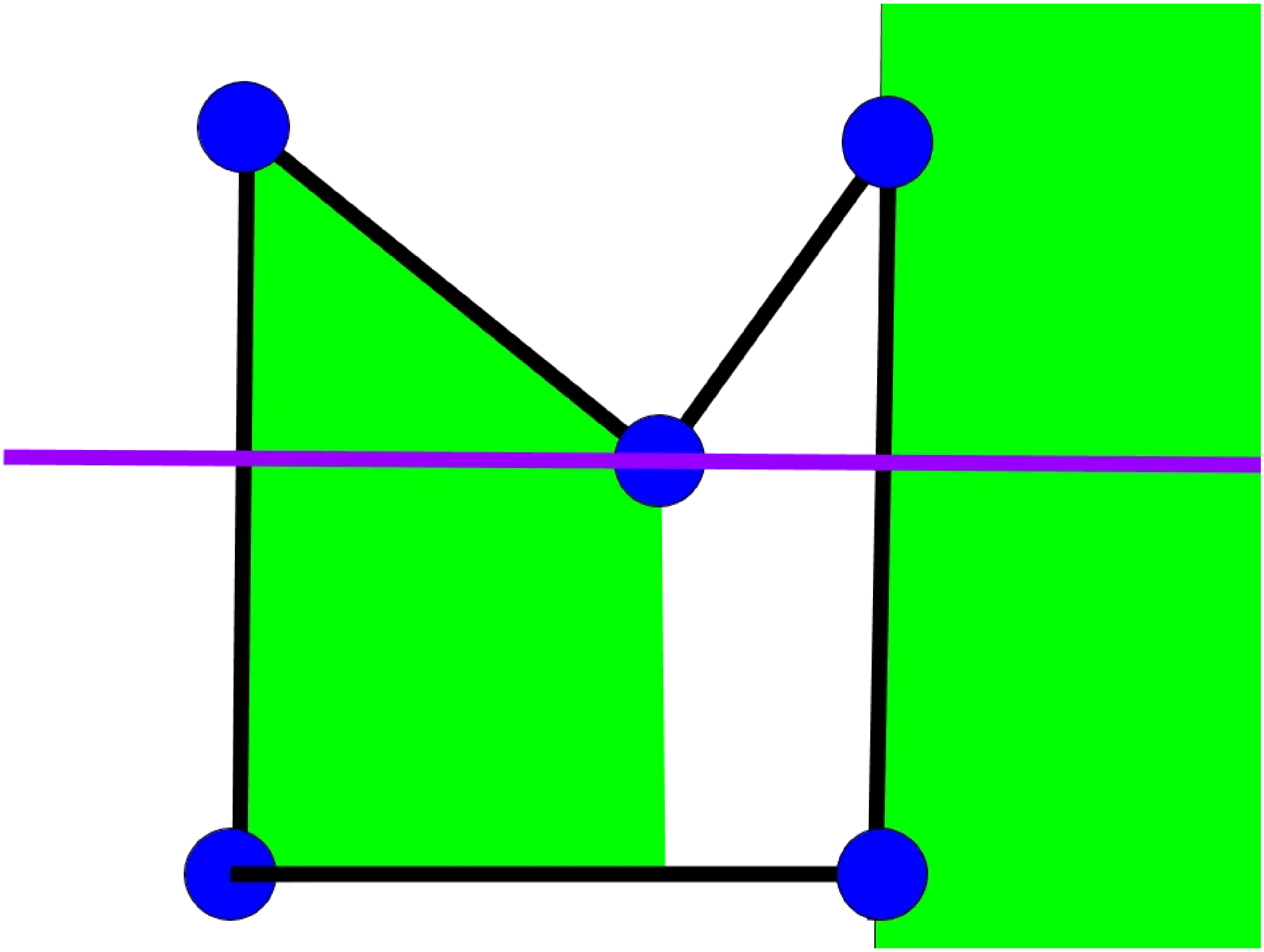


Another example of a “rebounding” intersection that doesn’t go completely through the purple horizontal line.

5. **Calculate Insides of a Polygon**

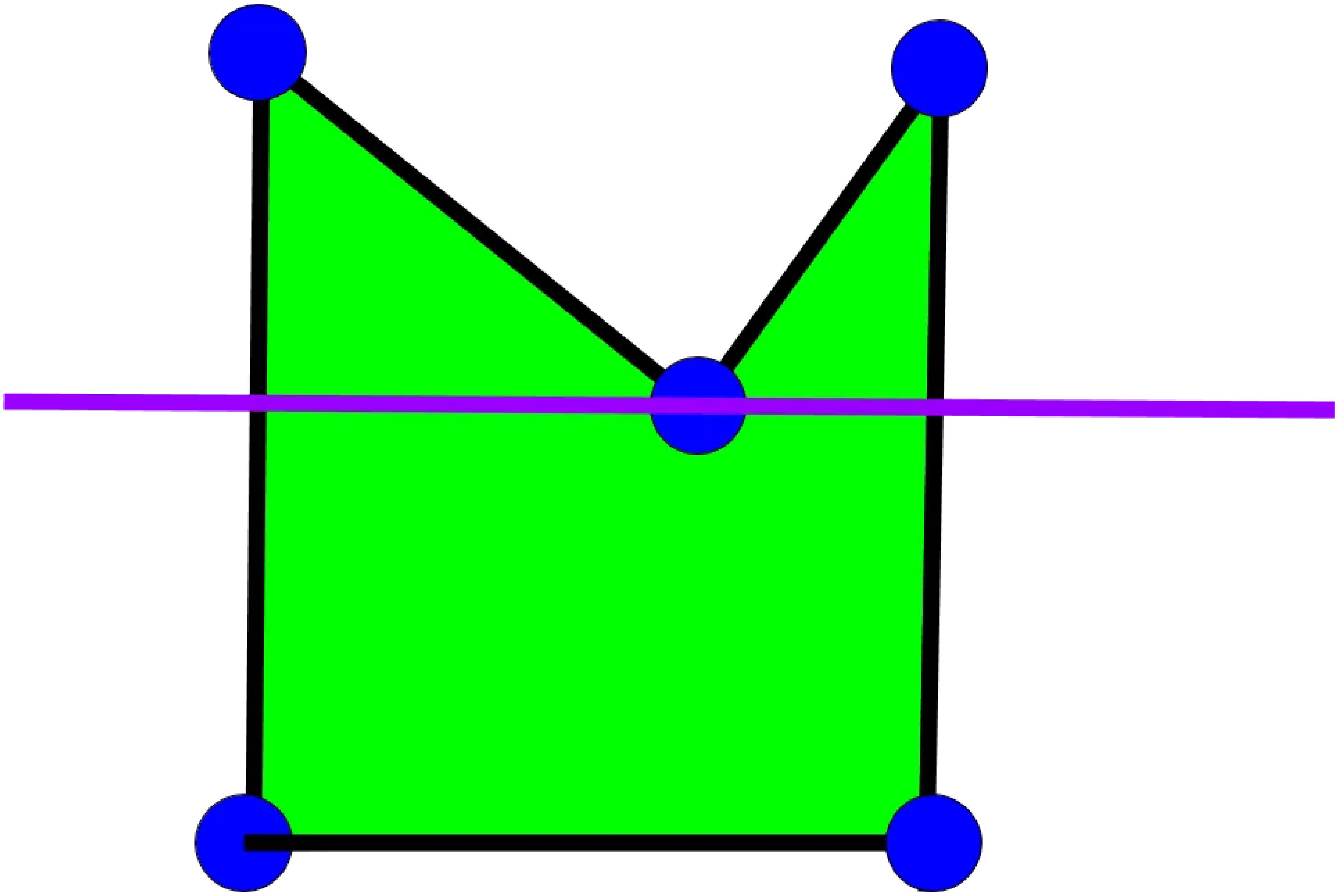
Our fifth step is to calculate the insides of a polygon. We use an algorithm derived from W. Randolph Franklin’s “PNPOLY — Point Inclusion in Polygon Test”. We assume that every group of line segments that goes completely through the row line represent a boundary between inside and outside the polygon. For each row line, we take each consecutive pair of intersections and assume that the polygon lies between them.

The following is an example of what happens when the calculation is made without respect to whether the line segments completely cross the horizontal line.



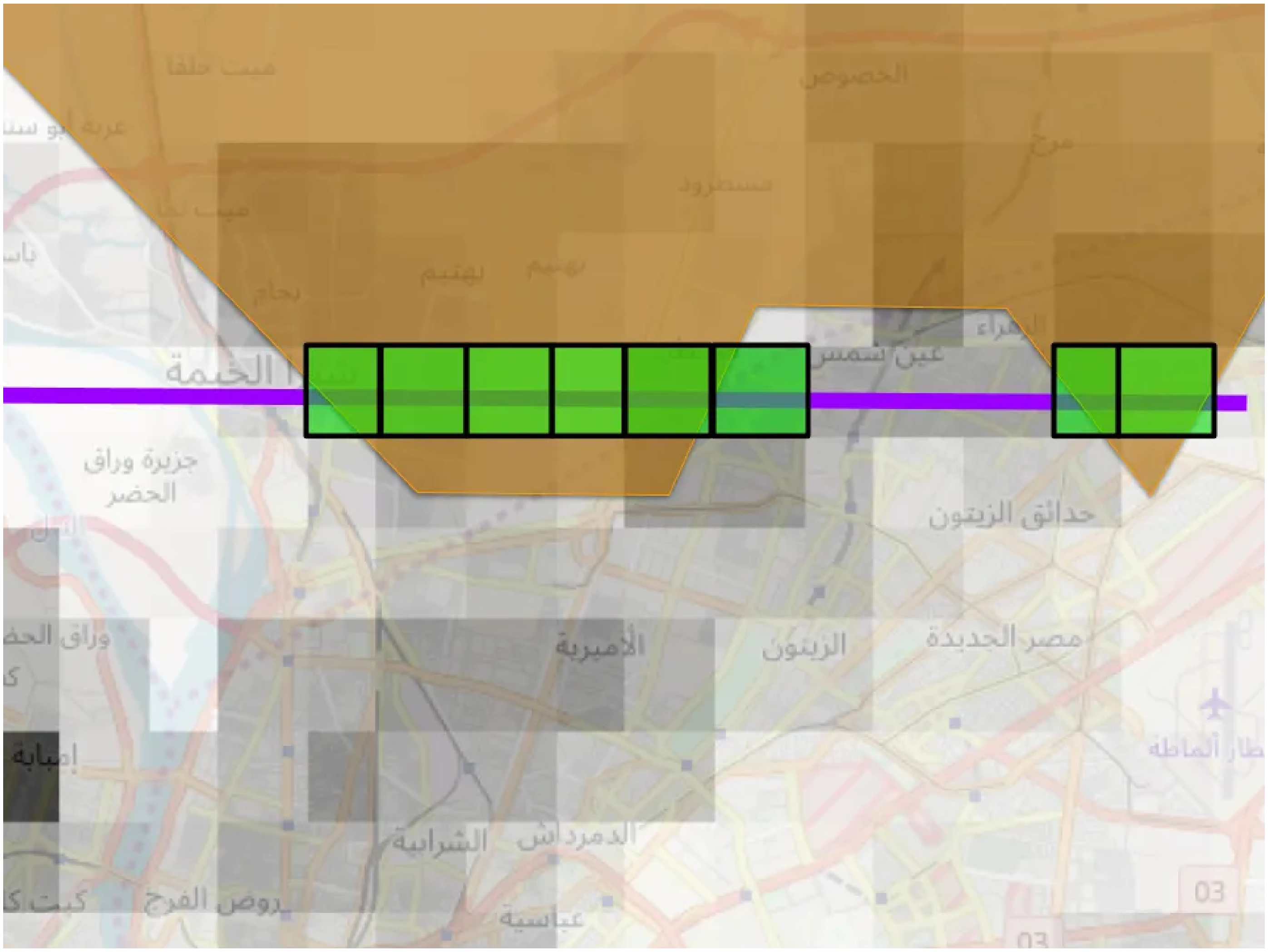
Insides (green) computed without first determining type of intersection

If we properly skip over intersections that don’t go completely through the horizontal line, we get correct results that look like the following.



Insides (in green) computed with respect to throughness of intersection

A more realistic image is shown before. The green squares represent pixels that have been determined to be intersecting the polygon



Raster data is from Global Human Settlement Layer (GHSL). Basemap is from © OpenStreetMap Contributors.

Next Steps

We’re publishing this research in part to receive feedback. We’re interested in edge cases that might confuse this algorithm. Let us know. Thanks!

GIS Remote Sensing Maps Algorithms Satellite Imagery

Follow

Written by Daniel J. Dufour

48 Followers

CEO @GeoSurge. Builds geotiff.io & geoblaze.io. Co-organizes @geodcmeetup. Contributes to @codedotgov. Led development of GADAS.

More from Daniel J. Dufour