USING NEAREST EDGE DISTANCE TO POLYGONS TO MINIMIZE THE IMPACT OF HIGHER POINT DENSITIES ON POINT IN POLYGON ALGORITHMS

June 2012

Kyle Martin
Geospatial computer scientist
kyle.martin@mosaicsgis.com

Keywords: laser scanning, point clouds, point in polygon algorithm, nearest edge distance

Abstract:

Laser scanning has seen a significant increase in collection frequencies over the past several years. Increased collection frequencies, yield higher point densities, larger point cloud datasets and has significant impacts to the growth of algorithms. Algorithms that are marginally practical now, will likely suffer impractical running times with point densities from newer or more advanced sensor technologies. The point in polygon operation in particular plays a significant role in point cloud applications and is impacted when point densities increase. In general, the running time of a point in polygon algorithm will increase linearly with point counts. We propose a nearest edge distance to polygon heuristic to minimize the increase in point tests as point densities increase, and will show that the heuristic produces logarithmic growth in point tests as point densities increase. Use of the nearest edge distance heuristic can be a very useful tool for existing point in polygon algorithms to adapt more practically to higher point density datasets that are common among newer laser scanning technologies.

Introduction

Laser scanning has seen a significant increase in collection frequencies over the past several years. Increased collection frequencies yield higher point densities, and for some laser scanning technologies such as vehicular (VLS) or terrestrial (TLS) laser scanning, the point densities can be significantly higher than airborne (ALS) collections. VLS and TLS is generally <= 5 cm point spacing and ALS is generally >= 0.5 m (Gong *et al*., 2012; Fowler *et al*., 2007; author observation). Synthetically derived point clouds using highly overlapping high resolution imagery (e.g., 3D vision) can produce even higher point densities (e.g., as low as 1 cm point spacing) than TLS or VLS (Leberl *et al*., 2010). With these increased densities comes the need to handle much larger point cloud datasets, and it is not inconceivable for these larger point cloud datasets to render existing algorithms inadequate or impractical when the nominal point spacing is cut in half or by a quarter for example (i.e., roughly a 4 and 16 fold increase in total point counts respectively).

The point in polygon operation in particular plays a significant role in point cloud applications and for example, is used to classify points within or out of hydrological areas, or buildings (author observation). The invention of better point in polygon algorithms is of significant interest in computational geometry, and is a basic operation in computer graphics. Various strategies and solutions that range from ray tracing, counting edge intersections, to angular summations have been invented that solves the point in polygon problem (Haines, 1994; Preparata, 1985; de Berg, 2008; Devadoss, 2011). Generally the solutions apply to convex only or to both concave and convex polygons and the analysis of the algorithms range from $O(\log n)$ time (Preparata, 1985), to $O(n)$ time (Haines, 1994; Preparata, 1985). In all cases, n represented the number of vertices in the polygon.

The implementation of a point in polygon algorithm is a foundational component to any point cloud software system. And in order for the system to be practical, a quality point in polygon algorithm must be implemented. It is not uncommon for the number of point in polygon tests to be in the millions for large polygons that for example may exist in low lying hydrological areas. These polygons are not only large in area but will have thousands of vertices, multiple and non-contiguous parts, and likely a number of islands (i.e., donuts, or out areas).

Previous studies on point in polygon problems have focused primarily on the increasing complexity of the polygon itself in terms of the numbers of vertices. The complexity of the polygon ultimately defines the size of the problem, thus resulting in algorithms where performance is directly related to the complexity (Haines, 1994; Walker, 1999). However, no studies have been found by the author that focus on a constant polygon with increasingly higher numbers of point in polygon tests as would be seen if the density of points over an area significantly increased due to more advanced laser sensor technologies.

In this study, we assume that the complexity of the polygon would not be required or expected to change as would be the case for many laser scanning update projects (author observation), and that existing algorithms would incur a linear increase in the number of point in polygon tests to handle the increase in point density. Consider the following statement that describes the total point in polygon test running time for any given algorithm:

$$nP * T \qquad (1)$$

where nP is the number of points or in this case also the number of point in polygon tests, and T is the running time for the point in polygon algorithm for a single test.

If nP were quadrupled, the total number of tests would increase linearly

$$[f(nP)] * T \qquad (2)$$

where *f* is the factor representing the increase in point counts.

The linear growth of the number of point tests can be reduced or the shape of the growth can be changed if the number of point tests can be decreased or minimized and not allowed to grow at the same rate as the point density. A reduction in the slope of the linear growth of point tests is shown by

$$O(nPt*T) + (nP-nPt)[O(1)] \qquad (3)$$

where *nPt* equals the number of points tested. So as *nPt* approaches zero, a higher number of points (i.e., approaching *nP*, total number of points) takes constant time (i.e., *nP-nPt*), to process the point in polygon operation, thus reducing the total running time. The magnitude of *nP - nPt* serves as a measure of how well the minimization of the number of point tests is performing. Magnitudes closer to *nP* have better minimization functions because more constant time operations are performed.

Here we propose a near distance to edge heuristic that minimizes the increase in *nPt* very effectively and can be adopted by any algorithm that can be modified to provide a nearest distance from any edge in the polygon to the test point.

Nearest Distance to Edge Heuristic

When performing a point in polygon test, the nearest edge distance (*nD*) from the polygon to a test point (*Tp*) is compiled. *nD* and *Tp* can then be used to eliminate future tests and conclude that a candidate point (*Cp*) has the same state or spatial relationship (*q*) to the polygon as the previous test (i.e., *Tp*). Therefore we postulate the following proposition

$$(cTpD < nD) \rightarrow q \qquad (4)$$

where *cTpD* is the euclidean distance between *Tp* and Cp, and *nD* is the nearest edge distance from *Tp* to the polygon boundary. If *cTpD* < *nD* then *q* is the same state or spatial relationship as the one found for *Tp*.

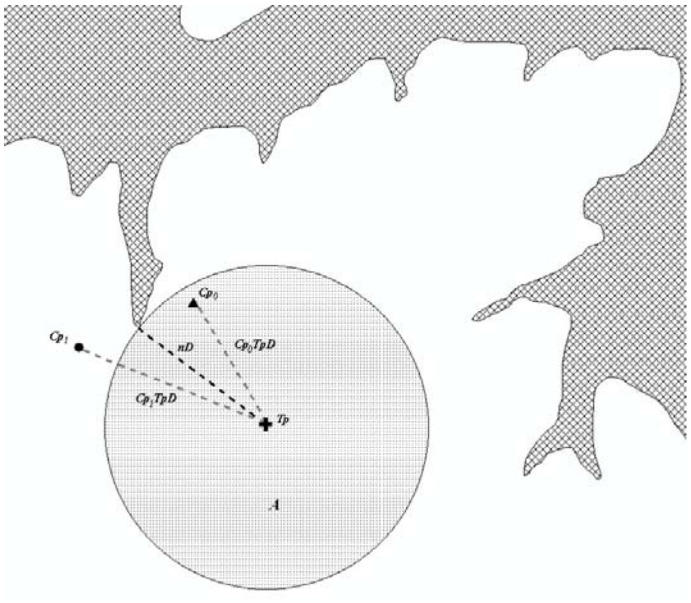


Figure 1: Schematic of the nearest distance to edge heuristic proposition. No point within region A can have a different topological relationship to the polygon than the relationship that *Tp* has with the polygon.

Figure 1 shows a graphical demonstration of the proposition. The minimum edge distance of *nD* says that no line segments of the polygon can be *within* region **A**. And if there are no polygonal line segments *within* region **A**, the line segment *TpCp* cannot cross or intersect any line segment of the polygon. Since no crossings can exist and using logic from the ray crossings point in polygon test (O'Rourke 1998), any point *within* region **A** (i.e., *cTpD* < *nD*), cannot be a different topological relationship than *Tp*. Thus no candidate points require testing *within* region ***A***. Once the distance between *Tp* and *Cp* is greater than or equal to *nD*, a new test is performed and *Tp* and *nD* is reset.

The programmatic execution of figure 1 is illustrated in the prototype code

```
double nearDist = -1;
Point testPoint;
bool currentState;

for each (Point p in points) {
    if (nearDist < 0 || p.GetDistance(testPoint) >= nearDist) {
        currentState = polygon.PointIn(p,&nearDist);
        testPoint = p;
    }
    //do something with the state
}
```

A simple scenario can show the increase in *nPt* (Eq. 3) as the point density increases is logarithmic when using the nearest distance to edge heuristic. Consider the scenario in figure 2 with a single edge where each point is considered consecutively from the left to right. Using the nearest distance to edge heuristic, the point tests are displayed with a black dot and a grey dot is a point that is skipped

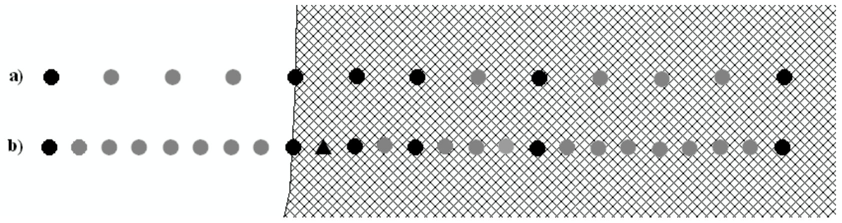


Figure 2: Moving from left to right, points that are tested (black circles), and skipped (grey circles) using the heuristic. Point spacing is cut in half in figure 2b and results in one additional point test displayed as a black triangle.

It is observed in figure 2b that each time the point spacing is cut in half one additional point test is performed for each edge encountered. As *nP* increases to infinity (i.e., point spacing is halved infinitely), the derivative of the function that determines *nPt* approaches zero. The simple case in figure 2 suggests that a similar logarithmic growth curve for *nPt* would be observed using real laser scanning and polygonal data.

Testing the nearest Distance To Edge Heuristic

To test the heuristic a real world polygon was chosen (NCFMP, 2012), and synthetically derived points were created in the polygon's area of interest. The polygon physiographically represents a winding wide river/lake polygon and serves as a practical and realistic challenge to the nearest distance heuristic and is shown in figure 3.

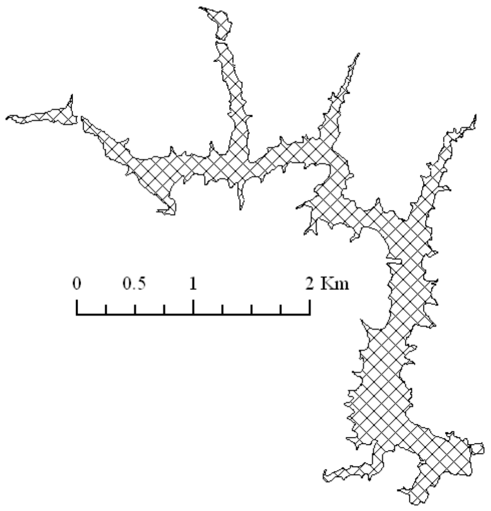
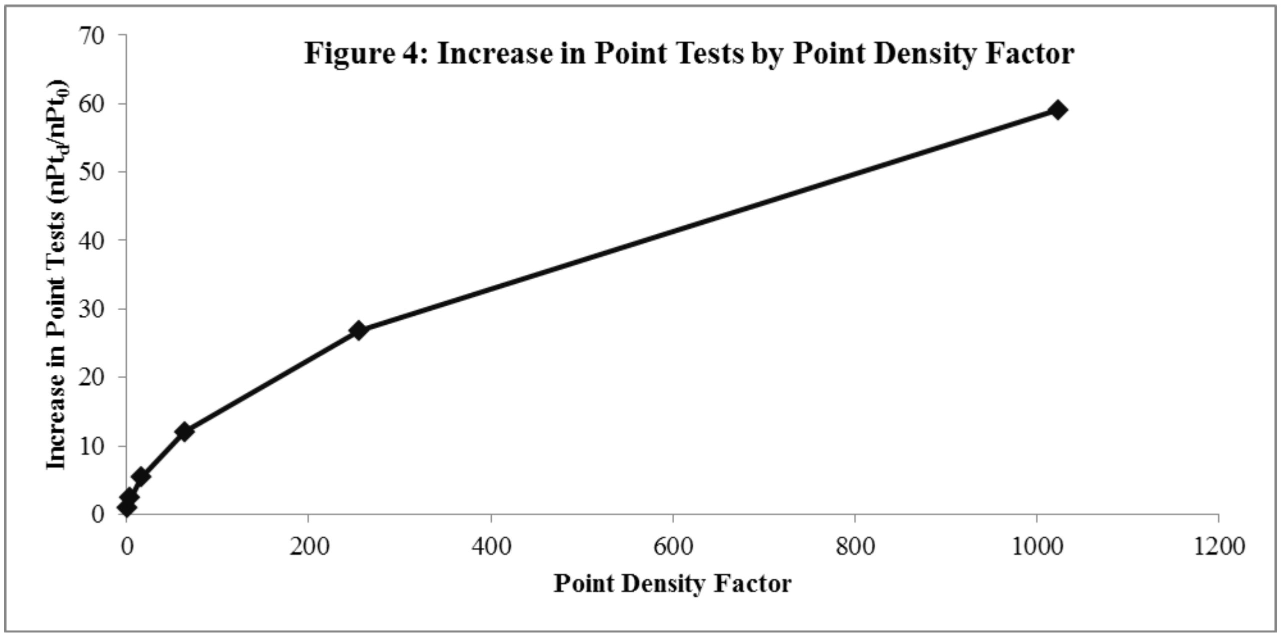


Figure 3: The polygon used to test the nearest distance heuristic.

An adequate series of point cloud datasets with increasing point densities was not available to the author. Instead, synthetically generated points were used in the test. If a series of actual point cloud datasets with increasing point densities were to be studied, it would be anticipated that on average there would be a constant number more points between any two points of the lesser density (e.g., higher frequency sensor captures an additional point(s) between points of a lower frequency sensor), thus a similar pattern to figure 2 would result. Because of this similarity, the results produced by synthetic or actual laser points would not differ significantly with respect to any findings of the nearest distance to edge heuristic.

The synthetically generated points were generated on regularly spaced intervals and in a sinuous scan line pattern to closely simulate the point ordering that would be encountered in actual point cloud datasets (Fowler, 2007). Six datasets were synthetically generated with each consecutive dataset representing quadruple the amount of data as the one before (i.e., point spacing is cut in half). A program was built to iterate the points in the same fashion as the prototype code and the number of point in polygon tests were tabulated as a ratio of *nPt* (Eq. 3) for dataset *d* (nPt_d) to *nPt* in the least dense dataset (nPt_0). The ratio of *nPt* (nPt_d/nPt_0) was plotted against its respective point density factor and is presented in figure 4.



To facilitate the examination of the growth of *nPt* as point density increased, nPt_d was normalized to the least dense dataset (i.e., nPt_0). Figure 4 graphically shows that *nPt* was approximately 1, 2, 5, 12, 27, and 59 for each dataset of increasing point density (1, 4x, 16x, 64x, 256x, 1024x). From these results we see that the slope of growth curve in *nPt* flattens as the point density increases (i.e., the derivative of the function that determines *nPt* with respect to point density approaches zero). Thus a logarithmic growth of *nPt* results as *nP* increase (Figure 4).

Discussion

The nearest distance heuristic is significantly sensitive to the ordering of the points. If the sequential order of the points is such that the distance from one point to another (*cTpD, Eq. 4*) is greater than or equal to *nD (Eq. 4)*, the proposition is false resulting in a point test. Therefore the magnitude of the minimization of tests is dependent on the ordering of the points. If the points are perfectly spaced apart so that *cTpD* exceeds *nD* every time no benefit is gained, however the more clustered the sequential movement of points are the better the heuristic performs in minimizing *nPt (Eq. 3)*. This assumption may render this heuristic less valuable when used with technologies that do not produce a natural proximity sequence of the points. Laser scanning point clouds exhibit this natural proximity sequence (Fowler, 2007). And to verify that the point order of laser scanning data has no significant impact on the expected outcome of the heuristic, actual laser scanning points (NCFMP, 2012) were classified as in or out of the polygon using the nearest distance heuristic. Out of 540,268 points within the extent of the polygon, only 7.8% of the points were tested. If the data increased in density, one would expect to see the number of point in polygon tests to grow logarithmically in a similar fashion as figure 4.

Significant improvements to the heuristic can be made by employing spatial data partitioning techniques that allow for quick elimination of large numbers of points such as those that would exist in the nodes of a tree data structure (Rosen, 2007; Goodrich, 2004). Using the bounding box of the node and its furthest distance to the previous test point, the containing points in the nodes can be eliminated based on the near distance heuristic proposition.

Because the nearest distance heuristic requires only a nearest distance between test geometry and an opposing geometry (*nD, Eq. 4*), and a calculation of the furthest distance between the test geometry, and candidate geometry (*cTpD, Eq. 4*), an inherent flexibility exists that allows the heuristic to be used by other geometry combinations and in other spatial relationships. For example, one may want to classify the points within a distance of a polyline. By simply modifying the nearest distance metric, one could quickly identify upcoming points as outside the buffer, still within the buffer, or intermediate, requiring a test. The heuristic can also be extended to work in *3d* by returning *3d* distances of *3d* geometries rather than distances on the *xy* plane. A promising use of the heuristic in *3d* is for classifying points based on a topological relationship to a surface (i.e., above, below, within a distance to, or on the surface).

Conclusion

As sensor technologies continue to provide higher point densities, algorithms must have a reasonable growth potential. Algorithms that are marginally practical now, will likely suffer impractical running times with point densities from newer or more advanced sensor technologies. The point in polygon operation is a foundational computational geometry concept, and is impacted significantly when point densities increase. The nearest distance to edge heuristic provides a methodology to minimize the impact of increasing point densities on point in polygon algorithms. The overall impact or running time is reduced by minimizing the number of point tests performed using a nearest distance to edge heuristic. It is this nearest distance to any polygon edge that allows for quick classification of any subsequent points not exceeding the nearest edge distance. The number of point tests increase logarithmically as point density increases. The reduction in the number of point tests as point counts increase allows the point in polygon algorithm to adapt more reasonably to newer laser technologies and sensors that produce significantly higher point densities than older or less advanced sensors.

References

De Berg, Mark, Otfried Cheong, Marc van Kreveld, Mark Overmars, 2008. Point location, *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer-Verlag, Berlin, (121 – 144).

Devadoss, Satyan L., and Joseph O'Rourke, 2011. Polygons, *Discrete and Computational Geometry*. Princeton University Press, Princeton, New Jersey, (1 – 32).

Fowler, Robert A., A. Samberg, M. J. Flood, and T. J. Greaves, 2007. Topographic and Terrestrial Lidar, *Digital Elevation Model Technologies and Applications: The DEM Users Manual* (2nd ed.). Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing.

Goodrich, Micheal T., Roberto Tamassia, and David Mount, 2004. Trees and Search Trees, *Data Structures and Algorithms in C++*. John Wiley and Sons, Inc., Hoboken, New Jersey (252 – 301, 411 – 475).

Gong, J., Q. Zhu, R. Zhong, Y. Zhang, and X. Xie, 2012. An efficient point cloud management method based on a 3d R-tree, *Photogrammetric Engineering & Remote Sensing*, 78(4):373 - 383.

Haines, Eric, 1994. Point in Polygon Strategies, *Graphics Gems IV*, ed. Paul Heckbert, Academic Press, (24-46).

Leberl, F., A. Irschara, T. Pock, P. Meixner, M. Gruber, S. Scholz, and A. Wiechert, 2010. Point clouds: Lidar versus 3d vision, *Photogrammetric Engineering & Remote Sensing*, 76(10):1123 - 1134.

North Carolina Floodplain Mapping Program (NCFMP), 2012. Durham County, State of North Carolina, USA. May 2012 < http://www.ncfloodmaps.com>

O'Rourke, Joseph, 1998. Search and Intersection, *Computational Geometry in C* (2nd Ed.). Cambridge University Press, (239-245).

Preparata, F.P., and Shamos, M.I., 1985. Geometric searching, *Computational Geometry*. Springer-Verlag, New York, (41-67).

Rosen, Kenneth H., 2007. Trees, *Discrete Mathematics & Its Applications* (6th ed.). McGraw-Hill, New York (627 – 687).

Walker, Robert J., and Jack Snoeyink, 1999. Practical Point in Polygon Tests Using CSG Representations of Polygons. *Technical Report TR – 99 – 12*. Department of Computer Science, University of British Columbia.