Trilateration Index - Masters Thesis

Chip Lynch

6/24/2018

Using Trilateration Distances as Geospatial Coordinates, Indexes, and Geohashes

Abstract

We present an alternative method for pre-processing and storing geospatial point data to improve query performance for some distance related queries such as "find all points in a set P within distance d of query point q". This pre-processing can be used to improve existing algorithms for Nearest-Neighbor (NN) searches, and lend themselves to new algorithms which offer some performance improvements and tradeoffs vs. existing algorithms.

Our construction includes storing the distances from fixed points (typically three, as in trilateration) as an alternative to Latitude and Longitude. This effectively creates a coordinate system where the coordinates are the trilateration distances. We explore this alternative coordinate system and the theoretical, technical, and practical implications of using it. Trilateration is a common concept, widely used in GPS and closely related to Triangulation used in cartography, surveying, and orienteering, although algorithmic use of these concepts for NN-style problems are scarce.

Initial results are promising for some use cases. Nearest-neighbor logic is both simplified (compared to R/kd/ball-Tree style indexing) and performant. Trilateration (or, more generally, "n-point lateration") is applicable to 2D, 3D, and higher dimension systems with minimal adaptation. We see a roughly 10x performance improvement in SQL queries using this preparation

[Add high level results for python tests when complete]

Further, we discuss the problem of "Network Adequacy" common to medical and communications businesses, to analyze questions such as "are at least 90% of patients living within 50 miles of a covered emergency room". This is in fact the class of question that led to the creation of our pre-processing and algorithms, and is a generalization of a class of Nearest-Neighbor problems.

We construct implementations of the Trilateration Index in both python, for general algorithimc use, and in SQL for use in relational databases, showing flexibility that is not as readuly available with the more complicated space partitioning algorithms.

While we focus primarily on geospatial data, otential applications to this approach extend to any distance-measured n-dimensional metric space, and we touch on those (briefly, here, to constrain our scope). For example, we consider applying the technique to Levenshtein distance, MIST datasets via cosine-similarity, and even facial recognition or taxicab systems where distances do not follow the triangle inequality.

Trilateration Index – General Definition

Given an n-dimensional metric space (M, d) (the universe of points M in the space and a distance function d which respects the triangle inequality), a typical point X in the coordinate system will be described by coordinates $x_1, x_2, \ldots, x_n$, which, typically, represents the decomposition of a vector V from an "origin" point $O : 0, 0, \ldots, 0$ to X into orthogonal vectors $0, x1, 0, x2, \ldots, 0, xn$ along each of the n dimensional axes of the space.

The Trilateration of such a point requires $n + 1$ fixed points F_p (p from 1 to $n + 1$), no three of which occupy the same $(n - 1)$-dimensional hyperplane. The Trilateration Coordinate for the point X is then: $t_1, t_2, \ldots, t_{n+1}$ where t_i is the distance (according to d) from X to F_i (in units applicable to the system).

Network Adequacy

The trilateration index was originally designed to improve efficiency of the "network adequacy" problem for health care. Network adequacy is a common legal requirement for medicare or insurance companies with constraints such as:

- 90% of members must live within 50 miles of a covered emergency room
- 80% of female members over the age of 13 must live within 25 miles of a covered OB/GYN
- 80% of members under the age of 16 must live within 25 miles of a covered pediatrician
- etc.

Similar questions, if not legal requiremenst, show up in cellular network and satellite communication technology (numbers are illustrative):

- Maximize the number of people living within 10 miles of a 5G cell tower
- 100% of all major highways should be within 5 miles of a 4G cell tower
- There must be at least 2 satellites within 200 km of a point 450 km directly above every ground station for satellite network connectivity at any given time
- There must be at least 1 satellite with access to a ground station within 50 km of a point 450 km directly above as many households as possible at any given time

The nearest-neighbor problem was called the "Post-Office Problem", although the system of post offices lends itself to a similar construction: * Ensure that all US Postal addresses are within range of a post office

and so forth.

Note that these are all illustrative examples; the real "Medicare Advantage Network Adequacy Criteria Guidance" document for example, is a 75 page document.

Formalization of Network Adequacy

We formalize the concept of "Network Adequacy" mathematically:

Network Adequacy Definition

Given a non-empty set of points P and a non-empty set of query points Q in a metric space M (where $P \cap Q$ comprises the 'network'), the network is 'adequate' for a distance d and a distance function $D(a, b)$ describing the distance between points a and b for $a \in M$ and $b \in M$ if for every point q (where $q \in Q$) $\exists$ at least one point p ($p \in P$) $\ni D(p, q) <= d$. Otherwise the network is 'inadequate'.

Network Adequacy Threshold Definition

We can generalize this slightly more by describing a network as 'adequate with threshold T' by introducing a percent T $(0 <= T <= 1)$ such that the same network is adequate if for at least $T * |Q|$ (or T percent of points in Q) there exists at least one point $p \ni D(p, q) <= d$.

In this case, if $T == 1$ we have the original case. If $T == 0$ we have a trivial case where the network is always adequate (even if Q and/or P are empty, which is generally disallowed).

Existing solutions

We can find no literature where this topic is solved in a particular algorighmic way. There are numerous discussions in health care about satisfying network adequacy, but more as policy or health care topics.

In general, it appears that most practical solutions are done in SQL databases which are commonly the source of member and provider data for health care datasets. Still, there is little published here; this information is anecdotal based on the author's personal direct knowledge and informal research.

Satellite and cellular network discussions of this problem appear to be proprietary.

Where we can find references to actual applications, the implemented solutions tend to be iterative, exhaustive implementations of existing Nearest-Neighbor algorithms. That is, for each point q, find the nearest point p and if $D(p, q) < d$ count it as conforming, otherwise count it as non-conforming. We then calculate the ratio of $r = conforming/|Q|$ if $r >= T$

If we set $m = |Q|$ and $n - |P|$, and if we use a Nearest-Neighbor algorithm with $O(n\ log\ n)$ then the time complexity for Network Adequacy becomes $O(mn\ log\ n)$.

In the worst-case, we cannot improve on this mathematically, but we can introduce what we believe are novel algorithms, based on the Trilateration Index, which execute efficiently compared to this iterative approach, by deeply reducing the search space and number of times the distance functions must be called in typical real-world cases.

Experimental Results for Network Adequacy

Nearest Neighbor

2-D Bounded Example

Consider a 2-dimensional grid − a flattened map, a video game map, or any mathematical $x - y$ coordinate grid with boundaries. WOLOG in this example consider the two-dimensional Euclidean space $M = \mathbb{R}^{\not{\in}}$ and bounded by $x, y \ \epsilon \ \{0..100\}$. Also, let us use the standard Euclidean distance function for d. This is, trivially, a valid metric space.

Since the space has dimension $n = 2$, we need 3 fixed points F_p. While the Geospatial example on Earth has a specific prescription for the fixed points, an arbitrary space does not. We therefore prescribe the following construction for bounded spaces:

Construct a circle (hypersphere for other dimensions) with the largest area inscribable in the space. In this example, that will be the circle centered at $(50, 50)$ with radius $r = 50$.

Select the point at which the circle touches the boundary at the first dimension (for spaces with uneven boundary ratios, select the point at which the circle touches the earliest boundary x_i). Such a point is guaranteed to exist since the circle is largest (if it does not, then the circle can be expanded since there is space between every point on the circle and an axis, and it is not a largest possible circle).

From this point, create a regular $n+1$-gon (triangle here) which touches the circle at $n+1$ points. These are the points we will use as F_p. They are, by construction, not all co-linear (or in general do not all exist on the same n-dimensional hyperplane) satisfying our requirement [proof].

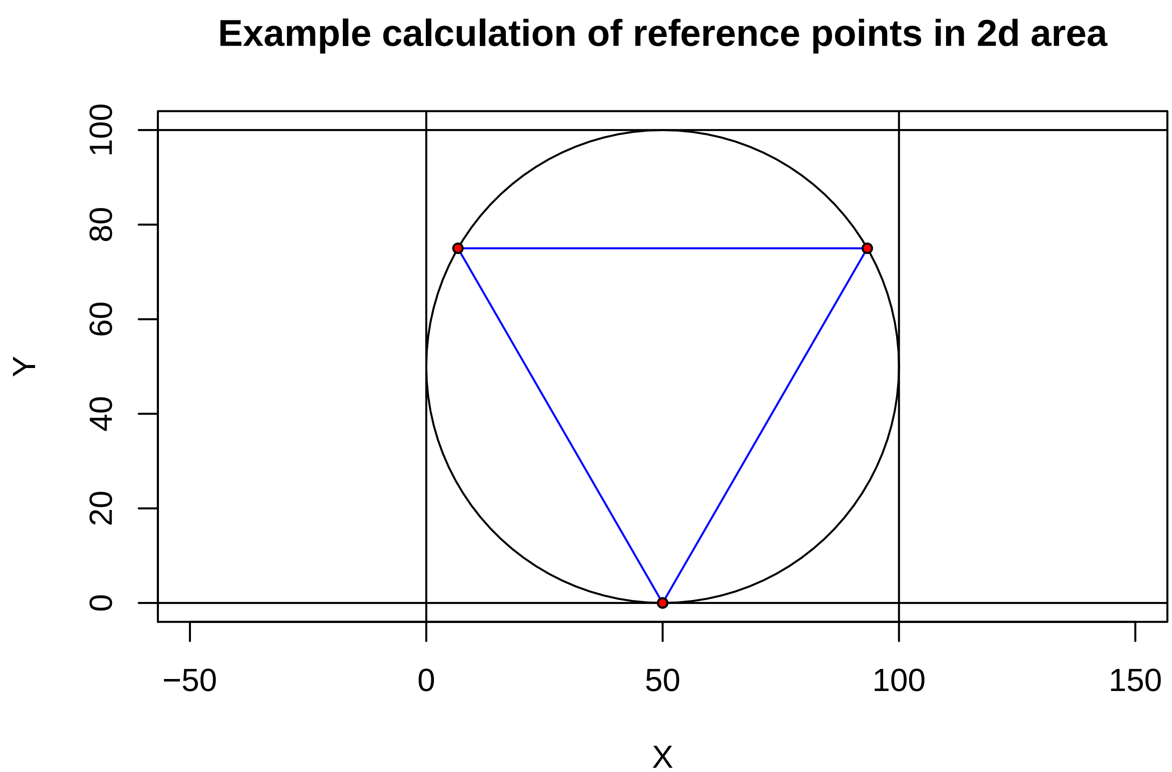
The point $y = 0$, $x = 50$ is the first point of the equilateral triangle. The slope of the triangle's line is $tan(\frac{pi}{3})$, so setting the equation of the circle:

$(x-50)^2 + (y-50)^2 = 50^2$ equal to the lines: $y = tan(\frac{\pi}{3})(x-50)$ gives $x = 25(2+\sqrt{3})$ on the right and $y = tan(\frac{-\pi}{3})(x-50)$ gives $x = -25(\sqrt{3}-2)$ on the left, and of course the original $(0, 50)$ point. Applying x to our earlier equations for y we get a final set of three points:

$$F_1 = (x = 50, y = 0)$$

$$F_2 = (x = 25(2+\sqrt{3}), y = tan(\frac{\pi}{3})((25(2+\sqrt{3})) - 50)$$

$$F_3 = (x = -25(\sqrt{3}-2), y = tan(\frac{-\pi}{3})((-25(\sqrt{3}-2)) - 50)$$



Remember, any three non-colinear points will do, but this construction spaces them fairly evenly throughout the space, which may be beneficial later* [Add section (reference) with discussions of precision and examples where reference points are very near one another].

The trilateration of any given point X in the space, now, is given by:

$$T(X) = d(F_1, X), d(F_2, X), d(F_3, X)$$

That is, the set of (three) distances d from X to F_1, F_2, and F_3 respectively.

10 Random Points

As a quick example of the trilateration calculations, we use a basic collection of 10 data points:

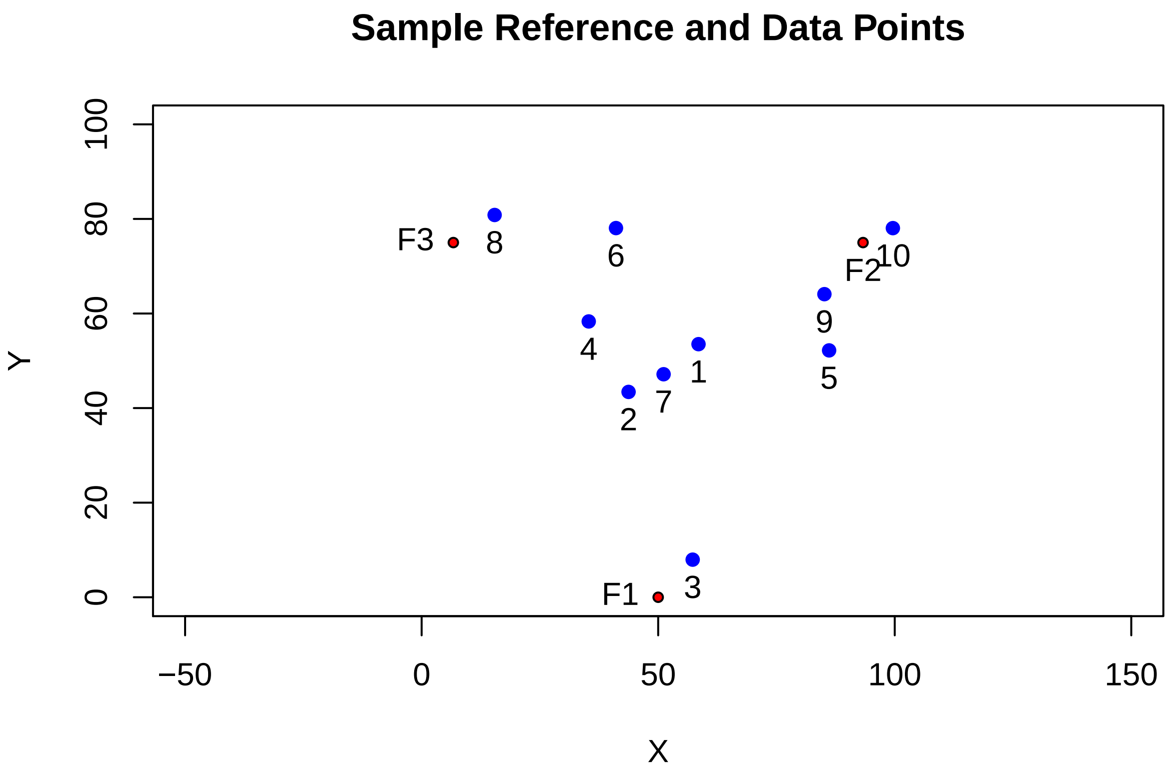
```
##           x         y
## 1  58.52396 53.516719
## 2  43.73940 43.418684
## 3  57.28944  7.950161
## 4  35.32139 58.321179
## 5  86.12714 52.201894
## 6  41.08036 78.065907
## 7  51.14533 47.157734
## 8  15.42852 80.836340
## 9  85.13531 64.090063
## 10 99.60833 78.055071
```

The trilateration of those points, that is, the three points $d_1, d_2, d_3 = d(F_1, X), d(F_2, X), d(F_3, X)$ are (next to the respective x_n):

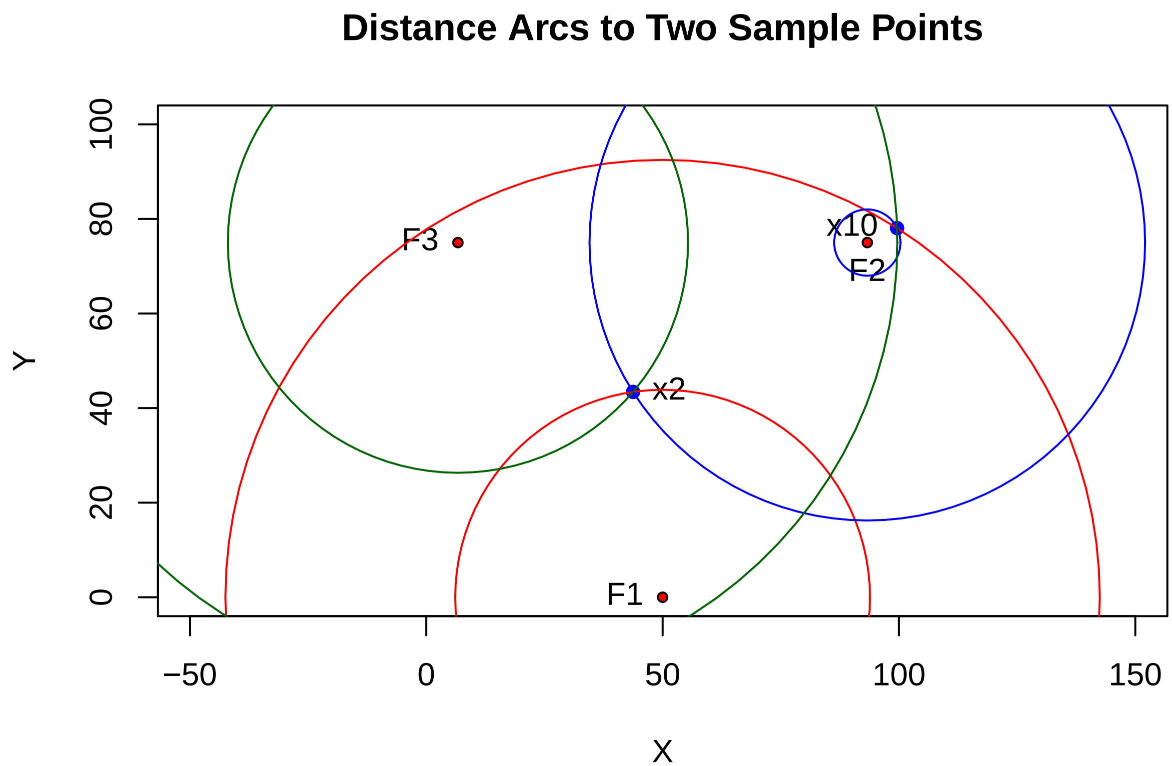
```
##           x         y       d1        d2       d3
## 1  58.52396 53.516719 54.19130 40.877779 56.10157
## 2  43.73940 43.418684 43.86772 58.768687 48.67639
## 3  57.28944  7.950161 10.78615 76.108693 83.99465
## 4  35.32139 58.321179 60.14002 60.331164 33.12763
## 5  86.12714 52.201894 63.48392 23.900246 82.63550
## 6  41.08036 78.065907 78.57382 52.310835 34.51806
## 7  51.14533 47.157734 47.17164 50.520446 52.44704
## 8  15.42852 80.836340 87.91872 78.091149 10.50105
## 9  85.13531 64.090063 73.08916 13.627535 79.19169
## 10 99.60833 78.055071 92.48557  7.008032 92.95982
```

Note that we do not need to continue to store the original latitude and longitude. We can convert the three d_n distances back to Latitude and Longitude within some ϵ based on the available precision. Geospatial coordinates in Latitude and Longitude with six digits of precision are accurate to within < 1 *meter*, and 8 digits is accuract to within < 1 *centimeter*, although this varies based on the latitude and longitude itself; latitudes closer to the equator are less accurate than those at the poles. The distance values d_x are more predictable, since they measure distances directly. While the units in this sample are arbitrary, $F(x)$ in a real geospatial example could be in kilometers, so three decimal digits would precisely relate to 1 *meter*, and so on. This is one reason that we will later examine using the trilateration values as an outright replacement for Longitude and Latitide, and this feature is important when considering storage requirements for this data in large real-world database applications.

For now, continuing with the example, those 10 points are shown here in blue with the three reference points F_1, F_2, F_3 in red:



To help understand the above values, the following chart shows the distances for points x_2 and x_10 above. Specifically, the distances d_1 from point F_1 are shown as arcs in red, the distances d_2 from point F_2 in blue, and d_3 from point F_3 in green.



Use of trilateration as an index for nearest-neighbor

One of our expected benefits of this approach is an improvement in algorithms like nearest-neighbor search.

Geospatial Example

Applying this to real sample points; let the following be the initial reference points on the globe:

Point 1: $90.000000, 0.000000$ (The geographic north pole)

Point 2: $38.260000, -85.760000$ (Louisville, KY on the Ohio River)

Point 3: $-19.22000, 159.93000$ (Sandy Island, New Caledonia)

Optional Point 4: $-9.42000, 46.33000$ (Aldabra)

Optional Point 5: $-48.87000, -123.39000$ (Point Nemo)

Note that the reference points are defined precisely, as exact latitude and longitude to stated decimals (all remaining decimal points are 0). This is to avoid confusion, and why the derivation of the points is immaterial (Point Nemo, for example is actually at a nearby location requiring more than two digits of precision).

Only three points are required for trilateration (literally; thus the "tri" prefix of the term), but we include 5 points to explore the pros and cons of n-fold geodistance indexing for higher values of n.

Theoretical Discussion

Theoretical benefits:

Precision: Queries are not constrained by precision choices dictated by the index, as can be the case in Grid Indexes and similar R-tree indexes. R-tree indexes improve upon naïve Grid Indexes in this area, by allowing the data to dictate the size of individual grid elements, and even Grid Indexes are normally tunable to specific data requirements. Still, this involves analysis of the data ahead of time for optimal sizing, and causes resistance to changes in the data.

Distributed Computing: Trilateration distances can be used as hash values, compatible with distributed computing (I.e. MongoDB shards or Teradata AMP Hashes).

Geohashing: Trilateration distances can be used as the basis for Geohashes, which improve somewhat on Latitude/Longitude geohashes in that distances between similar geohashes are more consistent in their proximity.

Bounding Bands: The intersection of Bounding Bands create effective metaphors to bounding boxes, without having to artificially nest or constrain them, nor build them in advance.

Readily Indexed (B-Tree compatible): Trilateration distances can be stored in traditional B-Tree indexes, rather than R-tree indexes, which can improve the sorting, merging, updating, and other functions performed on the data.

Fault Tolerant: This coordinate system is somewhat self-checking, in that many sets of coordinates that are individually within the correct bounds, cannot be real, and can therefore be identified as data quality issues. For example, a point cannot be 5 kilometers from the north pole (fixed point F1) and 5 kilometers from Louisville, KY (fixed point F2) at the same time. A point stored with those distances could be easily identified as invalid.

Theoretical shortcomings:

Index Build Cost: Up front calculation of each trilateration is expensive, when translating from standard coordinates. Each point requires three (at least) distance calculations from fixed points and the sorting of the resulting three lists of distances. This results in `O(n*logn)` just to set up the index.

*This could be mitigated by upgrading sensor devices and pushing the calculations back to the data acquisition step, in much the way that Latitude and Longitude are now trivial to calculate in practice by use of GPS devices. Also, we briefly discuss how GPS direct measurements (prior to converstion to Lat/Long) may be useful in constructing trilateration values.

Storage: The storing of three distances (32- or 64- bits per distance) is potentially a sizeable percent increase in storage requirement from storing only Latitude/Longitude and some R-Tree or similar index structure.

*Note that if the distances are stored instead of the Lat/Long, rather than in addition to them, storage need not increase.

Projection-Bound: The up-front distance calculations means that transforming from one spatial reference system (I.e. map projection – geodetic – get references to be specific) to another requires costly recalculations bearing no benefit from the calculation. For example a distance on a spherical projection of the earth between a given lat/long combination will be different than the distance calculated on the earth according to the standard WGS84 calculations).

*This said, we expect in most real-world situations, cross-geodetic comparisons are rare.

Difficult Bounding Band Intersection: Bounding Bands intersect in odd shapes, which, particularly on ellipsoids, but even on 2D grids, are difficult to describe mathematically. Bounding boxes on the other hand, while they distort on ellipsoids, are still easily understandable as rectangles.

Figure 1 - An example problem with radio towers R1, R2, and R3, and various receivers. Dashed lines represent the bounding bands with +/- a small distance from a given receiver (center black circle)

Figure 2 - A close look at the intersection of three bounding bands limiting an index search around a point with a search radius giving the circle in A. Note that the area B is an intersection of two of the three bands. Area C is the intersection of all three.

Appendixy stuff

Alternate Database Indexes

References (I need to finish reading and digesting these):

http://ieeexplore.ieee.org/document/5437947/

https://prezi.com/chnisgybkshy/gps-trilateration/

https://www.maa.org/sites/default/files/pdf/cms_upload/Thompson07734.pdf

https://www.researchgate.net/publication/2689264_The_X-tree_An_Index_Structure_for_High-Dimensional_Data

http://repository.cmu.edu/cgi/viewcontent.cgi?article=1577&context=compsci

https://link.springer.com/chapter/10.1007/10849171_83

http://ieeexplore.ieee.org.echo.louisville.edu/document/7830628/

https://link-springer-com.echo.louisville.edu/article/10.1007%2Fs11222-013-9422-4

https://en.wikipedia.org/wiki/R-tree

https://boundlessgeo.com/2012/07/making-geography-faster/

http://ieeexplore.ieee.org.echo.louisville.edu/document/6045057/

http://www.sciencedirect.com.echo.louisville.edu/science/article/pii/S002002550900499X?_rdoc=1&_fmt=high&_origin=gateway&_docanchor=&md5=b8429449ccfc9c30159a5f9aeaa92ffb&ccp=y

https://en.wikipedia.org/wiki/K-d_tree

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.219.7269&rep=rep1&type=pdf

http://www.scholarpedia.org/article/B-tree_and_UB-tree