Data Structure for Efficient Line of Sight Queries

Swapnil Gaikwad
Computer Science
San José State University
San José, California
swapnil.gaikwad@sjsu.edu

Melody Moh
Computer Science
San José State University
San José, California
melody.moh@sjsu.edu

David C. Anastasiu*
Computer Engineering
San José State University
San José, California
david.anastasiu@sjsu.edu

ABSTRACT

Given the great amounts of data being transmitted between devices in the 21^{st} century, existing channels of wireless communication are getting congested. In the wireless space, the focus up to now has been on the microwave frequency range. An alternative for high-speed medium- and long-range communication is the millimeter wave spectrum, which is most effectively used through point-to-point links. In this paper, we develop and compare methods for verifying the Line of Sight (LOS) constraint between two points in a city. To be useful for online wireless network planning systems, the methods must be able to process terabytes of 3D city geolocation data and provide answers in milliseconds. We evaluate our methods using data for the city of San José, a major metropolitan area in Silicon Valley, California. Our results indicate that our *Hierarchical Polygon Aggregation* (HPA) method is able to achieve millisecond-level query times with very little loss of precision.

CCS CONCEPTS

• **Information systems** → **Data structures**; *Specialized information retrieval*;

KEYWORDS

Wireless Emergency Network, Millimeter Waves, Line of Sight, Data Structures, Information Retrieval.

ACM Reference Format:
Swapnil Gaikwad, Melody Moh, and David C. Anastasiu. 2018. Data Structure for Efficient Line of Sight Queries. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269238

1 INTRODUCTION

Wireless communication primarily uses the 3 KHz to 6 GHz frequency spectrum. The spectrum includes all major communication applications in use today, including government and military communication, AM/FM radio, TV broadcasting, cellular transmissions, aviation and radar, wireless LAN, GPS, and many more [1]. With such a wide variety of applications, the microwave frequency range is rapidly getting congested. The data deluge from newly introduced smart devices and Internet of things (IoT) sensors are adding even more stress to this crowded spectrum.

One solution to the congestion problem is to develop novel communication strategies over the sparsely used millimeter wave spectrum, which utilizes the 6 GHz to 300 GHz frequency spectrum. Operating in such a wide spectrum will relieve the pressure currently placed on the microwave spectrum and improve our current congestion related problems. Advantages of millimeter wave communication include increased reliability, ultra-low latency and high security. This advantage is more pronounced when using directional antennae, which are capable of transmitting gigabytes of data per second between two points without the need to broadcast the signal in 360 degrees. This leads to reduced energy consumption, but has the disadvantage of requiring direct line-of-sight (LOS) between the communicating antennae.

Millimeter wave networks are often deployed in concert with microwave communication infrastructure. Point-to-point segments provide fast aggregate signal transmissions over long distances, while local transmissions happen over WiFi or WiMax connections. Designing such a heterogeneous wireless communication network at scale, however, is a difficult task, complicated by the LOS requirement of the millimeter wave network. Given the 3D layout of a city and a set of requirements, such as the desired coverage, number of long-range nodes, minimum LOS cardinality for each node, and a set of preferred and unaccessible structures in the city, an optimization algorithm may need to execute hundreds of thousands of LOS queries to find the optimal placement of antennae for the desired network.

In this paper, we develop and compare novel data structures and methods for efficiently verifying the LOS constraint between two points in a city. To be useful for online wireless network planning systems, the methods must be able to process terabytes of 3D city geolocation data and provide answers in milliseconds. As a test case for our method, we evaluate tens of thousands of LOS queries with points located in the city of San José, a major metropolitan area in Silicon Valley, California, housing more than 1 million people. This city has an odd elongated shape and is nestled against hills that cut off LOS access to some areas. Our experiments show that our *Hierarchical Polygon Aggregation* (`HPA`) method is able to achieve millisecond-level query times with very little loss of precision.

The remainder of the paper is organized as follows. Section 2 introduces existing solutions, giving an overview of their implementations and limitations. Section 3 presents two methods for estimating LOS between two locations in a city. We describe our evaluation methodology in Section 4, analyze our experimental results in Section 5, and Section 6 concludes the paper.

*Corresponding Author

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269238

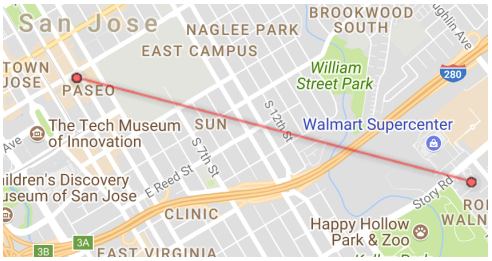

Figure 1: Line of Sight for call-sign 3428484

Figure 2: OSM polygons, San José

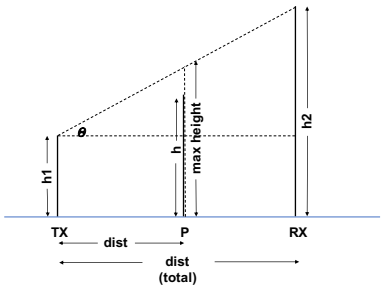

Figure 3: Line of Sight computation

2 RELATED WORK

Existing solutions for the LOS identification problem are developed by millimeter antennae manufacturers [3, 5, 7, 8]. For establishing clear LOS between just two locations, these solution are more than enough. However, tens or hundreds of thousands of such queries should be executed when solving an optimization problem for a city-wide wireless network planning. Given a pair of locations, these solutions check every point on the segment between the locations, attempting to locate possible obstructions.

Figure 1 shows a real world microwave network transmission segment for FCC call-sign 3428484. A typical GIS database stores data at every 1/3 arc seconds (10 meters) in a 3-parameter tuple, denoting latitude, longitude and height. For a relatively small distance of around 2 miles shown in Figure 1, it will take around 320 point checks to confirm clear LOS. Retrieving the correct points from the database takes a significant amount of time, even if the GIS data is properly indexed. Hence, it is necessary to reduce query execution time for LOS queries. In addition to this shortcoming, most of the existing solutions do not consider man-made objects or vegetation as possible LOS obstructions. The methods we devise should be extensible with diverse data that may improve effectiveness.

3 METHODS

In our proposed framework for classifying LOS, we significantly reduce the number of point height checks required to address a LOS query by preprocessing and aggregating GIS data. We develop two techniques and algorithms to solve this problem. The first technique utilizes natural polygon structures for buildings found in OpenStreetMaps as the basis for data aggregation. In the second technique, we use artificially drawn polygons to aggregate data. We discuss these techniques in the remainder of this section.

3.1 Natural Polygon Aggregation

In our Natural Polygon Aggregation LOS method (NPA), we utilized readily available OpenStreetMap polygons for data aggregation. OpenStreetMap stores 2 dimensional layout of buildings, roads, open spaces, etc., in the form of polygons. We imported these polygon data in a PostgreSQL database and overlapped elevation data from NED satellites on these polygons with the help of the PostGIS PostgreSQL extension, which allowed us to determine the maximum height in each polygon. We stored the maximum height information as an additional polygon parameter in the PostreSQL database and used it in NPA to answer LOS queries.

Given a pair of locations for transmitting antenna (TX) and receiving antenna (RX), we determine all intersecting polygons along the way with the help of readily available PostGIS methods. Based on the input heights h_1 and h_2 of two TX and RX points at a distance of $dist_{total}$ from each other, we first compute the angle of elevation, which is then used to determine the allowed height for each intersection polygon, namely $h_1 + dist \times \frac{h_2 - h_1}{dist_{total}}$, where $dist$ is the distance from the source to the checked point. Any violation of the allowed maximum height results in an obstructed result. Figure 3 demonstrates this process in graphical form. This approximation for classifying LOS greatly reduces the number of comparisons required to check a potential network segment. A disadvantage of the NPA methods is that all data must be stored in the PostgreSQL database and takes up a great deal of space. We further refine our NPA method in the Hierarchical Polygon Aggregation method.

3.2 Hierarchical Polygon Aggregation

In NPA, we noticed some discrepancies, such as uneven polygon sizes and missing polygon structures. If a certain polygon covers a very big area, storing a single height for that entire area reduces the accuracy of the LOS queries. In contrast, very small polygon structures increase the time required for the query.

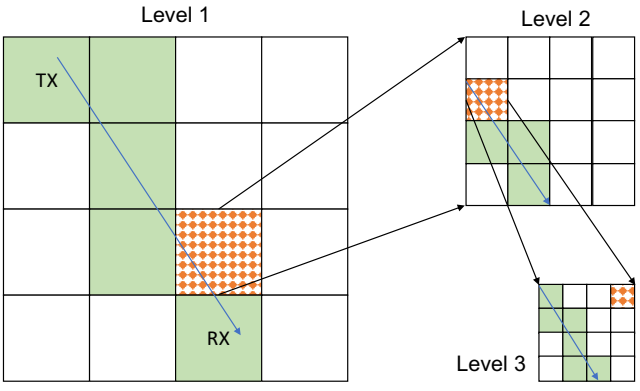

Figure 4: Surface area tiling. *Green tiles* have a maximum height lower than the minimum intersecting point of the transmission line slope, while *red tiles* exceed that height.

To address these issues, we created artificial polygons of even sizes, which we call *tiles*. Unlike OSM polygons, tiles are adjacent

to each another. This reduces the possibility for errors due to non-polygon structures, such as those immediately outside a building. Moreover, an LOS query can be answered by relying on aggregate statistics stored in each tile along the path from the source RX to the destination TX, without any need to query the database.

Algorithm 1 and Figure 4 describe our HPA LOS query procedure. The HPA data structure consists of a hierarchy of l tile levels with consistent $h \times w$ layout. Given a designated surface area of a city and parameters h, w, and l, we compute the total number of tiles required at the lowest level to cover the city and generate latitude and longitude bounds for each tile. The *height* of each tile is then calculated by finding the point with the maximum height in the specified tile. Tiles are then further aggregated hierarchically at higher levels of the data structure. Figure 4 shows an example tile representation and the pooling procedure, which we previously described in [10].

With the HPA data structure, we can use hierarchically stored tile statistics to further reduce the number of checks required to answer an LOS query. We store only the bottommost level of tiles on disk. When loading the data structure in memory, further levels are aggregated using convolution style max pooling. In order to answer an LOS query, we compute the angle of elevation in the same way as in our NPA method. We then start checking the intersecting tiles at the topmost level, which has the highest aggregation. Here, intersecting tiles are determined by computing the angle of direction along the transmission path. If a tile fails the maximum allowed height check, the method drills down the failed tile, checking tiles along the path in the next level. This hierarchical check is extended until the last available level is reached. If the tile in the last level fails the check, the method returns the query result as obstructed.

Algorithm 1: Hierarchical Polygon Aggregation LOS

Input: Points $TX(lat_1, long_1)$ and $RX(lat_2, long_2)$, level l
Output: Binary Line of Sight Result
t_1 = T[lat_1, $long_1$, l], t_2 = T[lat_2, $long_2$, l]
d = distance(t_1, t_2)
Δ = height(t_1) - height(t_2)
θ_1 = Δ / d
θ_2 = $long_1$ - $long_2$ / lat_1 - lat_2
$long$ = minLong(t_1), lat = minLat(t_1)
while $long < long_2$ **do**
  $\delta_{long} = long - long_1$
  $\delta_{lat} = \delta_{long}/\theta_2$
  lat = lat(t_1) + δ_{lat}
  t = T[lat, $long$, l]
  $\delta_h = \theta_1 \times$ distance(t_1, t)
  if *height(t)* - *height*(t_1) > δ_h **then**
    if l = *maxLevel* **then**
      return $False$
    HPA (l + 1)
Repeat while loop along lat
return $True$

4 EXPERIMENT SETUP

For testing our methods, we used a server with two 12-core 2.5 GHz Intel Xeon E52680 v3 (Haswell) processors and 384 GB RAM. We used PostgreSQL version 9.6.1 to host our GIS database. As a proof of concept, we chose a subset of California OSM and LiDAR data covering the city of San José.

For our first method, NPA, we retrieved natural 2-dimensional polygons from Mapzen [6], which internally uses OpenStreetMap (OSM) [11]. There were 153,935 OSM polygons present in the boundary of the city of San José. Beside building geometries, the dataset includes polygons for other structures, such as roads and open plots of land. Figure 2 shows a qGIS rendering of a section of the imported OSM data. We computed the heights of structures in our chosen area from highly accurate LiDAR data [9]. These data have a resolution of 1/3 arc second, i.e., a height reading exists approximately every 10 meters. We converted these data from their default Lambert Conformal Conical projection to standard WGS84 projection before using them in our experiments. The total number of LiDAR data points in the boundary of the city of San José was 2,668,443,461. Even though this is a lot of elevation data, certain areas had a very low coverage in the LiDAR data. For those areas, we used the Google Maps Elevation API [4] to fill in missing data, adding 98,000 more data points. The total 2,668,541,461 data points required 794 GB of storage (591 GB for data + 203 GB for indexes).

In our second method, HPA, we chose $l = 3$, $h = 70$ and $w = 7$, creating 70×7 tiles at each of the 3 levels, which covered an area of roughly 70×7 km. The rectangular shape of the top level corresponds to the general shape of the city of San José. The total number of generated lowest level tiles was 117,649,000, which is considerably higher than the number of polygons in the NPA method.

Both methods were implemented in Java, using standard Java thread pool mechanisms for shared memory parallel processing. We used OpenJDK 1.8.0_144 on Arch Linux and ran both methods with 24 threads in all experiments. The NPA method requires database connections to find the intersecting polygons and their heights. We used a connection pool of 24 connections for this purpose. Additional precaution was taken to ensure that no other programs were running during the execution of either method. We also compared our method against existing online services for estimating line of site. Before executing those experiments, we tested the Internet upload and download speed of our server and found them on average to be 80 Mbps and 40 Mbps, respectively.

5 RESULTS & DISCUSSION

We executed two sets of experiments, one measuring the effectiveness and the other the efficiency of our methods.

5.1 Effectiveness Experiments

In order to test of our methods' effectiveness to return accurate LOS information in response to queries, we constructed a test dataset consisting of 1506 point pairs located in San José. We chose clear LOS (positive) pairs based on existing FCC tower pair location data [2]. For negative samples, we made use of an external LOS testing utility [5] and generated obstructed line of sight location pairs. In total, we obtained 753 positive and 753 negative samples. We used *Accuracy* to measure the effectiveness of our methods, which is defined as the ratio of correctly classified samples over the total number of samples.

Both our methods performed really well with regards to effectiveness. NPA misclassified 47 queries, resulting in an accuracy of 96.87%,

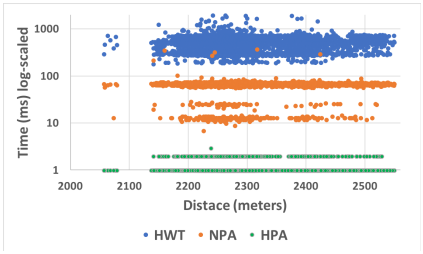

Figure 5: Distance vs. time for HWT, NPA and HPA

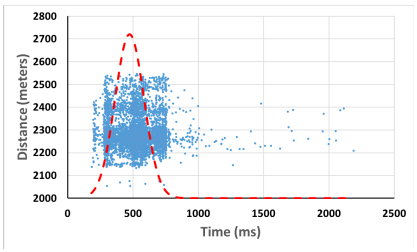

Figure 6: Query time distribution for HWT

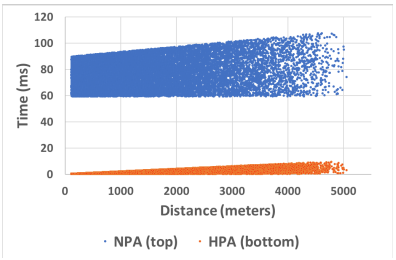

Figure 7: Query distance and time correlation

and HPA misclassified 18 queries, resulting in an accuracy of 98.80%. Six out of the 18 samples misclassified by our HPA method were obstructed LOS paths that HPA pronounced to be clear. Upon further investigation using the external LOS testing utility, we found existing structures along these paths that our system did not have data for, causing the false positive results. False negative results can be further reduced by more carefully tunning the tile size and number of layers parameters in our HPA method. By simply using additional data to update our GIS database (for the NPA method) or our hierarchical height index (for the HPA method), our methods can further improve their accuracy.

5.2 Efficiency Experiment

Our efficiency experiments were centered around comparison of our method against baselines with respect to query execution time, and analyzing the relationship between query point distance and execution time. In the first experiment, we executed 5,303 queries using our two methods and the HeyWhatsThat Path Profiler Web service [5] baseline, which we simply call HWT. Distances for queries in this experiment range between 2,000 and 2,600 meters, which were similar to links we found in existing microwave network installations in San José. Figure 5 shows the distribution of query times for the three methods. Note that query times are log-scaled to better portray the differences between the methods. The figure is best viewed in color. While the HWT method took on average 527.65 (168.30 stdev) ms to execute queries, our NPA and HPA methods answered the same queries in 66.60 (14.59 stdev) ms and 1.08 (0.27 stdev) ms on average, respectively. To get a better understanding of factors affecting query execution for HWT, we plotted execution time vs. distance in Figure 6. The red dashed line shows a Gaussian distribution fitted on the method's execution time data. Results show that HWT queries generally execute in 300–800 seconds irrespective of query execution distance. Queries times in HWT are likely dependent on Internet transmission routes and the current load of the Web server hosting the service.

In a second experiment, we compared our two methods, NPA and HPA, on a much larger set of 20,000 random queries with distances ranging between 0 and 5000 meters. Figure 7 plots the query distance vs. execution time for these queries. The average execution times were 77.82 (10.51 stdev) ms and 2.35 (1.48 stdev) ms for the NPA and HPA methods, respectively. While the NPA method takes a minimum of 58 ms to execute due to the need to execute database queries, HPA maintains an in-memory index data structure that facilitates very fast LOS queries. Both NPA and HPA show a positive correlation between query segment distance and execution time, yet the slope of the HPA increase is smaller than that of the NPA method, pointing to better scalability of the HPA method as distances increase. In general, microwave antennae are placed at distances of at most 10 miles (16 Km). Our HPA method should be able to produce results in less than 5 ms for most such queries.

6 CONCLUSIONS

In this paper, we presented two techniques for efficiently solving the problem of identifying whether a clear line of sight (LOS) exists between two points in the 3D geometry of a large city. Our first baseline algorithm, Natural Polygon Aggregation LOS, uses off-the-shelf GIS aware database systems and open-source data to effectively solve the problem, resulting in 96.87% accuracy and 78 ms average query execution time. Its long execution time may be prohibitive when trying to solve a multi-constrained optimization problem for city-wide wireless network planning. We thus designed a novel index data structure and the Hierarchical Polygon Aggregation LOS method to improve LOS query efficiency. Our HPA method works by drastically reducing the number of points whose height must be checked to ensure clear LOS between two locations and resulted in improved 98.80% accuracy and 2 ms average query execution time.

REFERENCES

[1] 2015. Introduction to RF and Wireless Communications Systems. (2015). Retrieved April 15, 2018 from http://www.ni.com/tutorial/3541/en/

[2] 2017. fcc67GHz_20170720. https://fusiontables.google.com/DataSource?docid=1-teDJHcmE21UMaaWnhCb2oy92eFD6m046UrbpGtQ#rows:id=1. (2017). Accessed: 2018-04-18.

[3] 2018. airLink - Outdoor Wireless Link Calculator. https://airlink.ubnt.com/#/. (2018). Accessed: 2018-04-18.

[4] 2018. Developer's Guide | Google Maps Elevation API. https://developers.google.com/maps/documentation/elevation/intro. (2018). Accessed: 2018-04-18.

[5] 2018. HeyWhatsThat Path Profiler. http://www.heywhatsthat.com/profiler.html. (2018). Accessed: 2018-04-18.

[6] 2018. Mapzen ? start where you are. http://mapzen.com/. (2018). Accessed: 2017-02-01.

[7] 2018. RF Line of Sight - SCADACore. http://www.scadacore.com/field-tools/rf-path/rf-line-of-sight/. (2018). Accessed: 2018-04-18.

[8] 2018. Solwise - Surface Elevation Tool. http://www.solwise.co.uk/wireless-elevationtool.html. (2018). Accessed: 2018-04-18.

[9] 2018. TNM Download. https://viewer.nationalmap.gov/basic/. (2018). Accessed: 2018-04-18.

[10] Swapnil Gaikwad and David C. Anastasiu. 2017. Optimal Constrained Wireless Emergency Network Antenna Placement. In *Proceedings of the IEEE Smart City Innovations 2017 Conference (IEEE SCI 2017)*.

[11] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org. (2017).