Using An Implicit Min/Max KD-Tree for Doing Efficient Terrain Line of Sight Calculations

Bernardt Duvenhage*
The Council for Scientific and Industrial Research

Abstract

The generation of accurate Line of Sight (LOS) visibility information consumes significant resources in large scale synthetic environments such as many-on-many serious games and battlefield simulators. Due to the importance of optimum utilisation of computing resources, a number of LOS algorithms are reported in the literature to either efficiently compute LOS information or reduce the impact of LOS queries on the run-time performance of synthetic environments. From the literature it is known that a k-dimensional tree (kd-tree) based raytracing approach, to calculating LOS information, is efficient.

A new implicit *min/max* kd-tree algorithm is discussed for evaluating LOS queries on large scale spherical terrain. In particular the value of low resolution *boundary* information, in quickly evaluating the LOS query, is emphasised. The min/max algorithm is empirically compared to other LOS approaches that have either implicitly or explicitly used kd-trees to optimise LOS query evaluation. The min/max algorithm is shown to have comparable performance to these existing LOS algorithms for flat earth, but improved performance when the application domain is extended to spherical earth. An average of a factor 3.0 performance increase is experienced over that of the existing implicit and explicit max kd-tree algorithms on spherical earth. This is achieved by combining the existing kd-tree algorithm with the classic *smooth-earth* LOS obscuration test and from there the *min* in min/max kd-tree.

CR Categories: I.6 [Simulation and Modelling]: General;

Keywords: line of sight, implicit kd-tree, spherical earth

1 Introduction and Overview

Line of Sight (LOS) information is used extensively in synthetic environments such as shown in Figure 1. *Accurate* LOS information is, according to Mlaker [2004], one of the most important inter-system interactions for modelling a combat environment effectively. Seixas et al. [1999] defines LOS information between the positions A and B—of two systems within a synthetic environment—as an indication of whether or not the line segment AB intersects the terrain or some other environmental structure. LOS information, in other words indicates whether the system at A could potentially *see* the system at B across terrain, buildings, etc. and vice versa. A *LOS query* in turn looks up, or triggers the calculation of, the LOS information/result between A and B.

The first and most obvious use case of a LOS algorithm in such an environment is by a virtual individual to simulate which of its opponents it can *see or detect*. Section 2 presents three use cases in more detail and shows the approximate frequency of LOS queries typically required in large and many-on-many scenarios.

*e-mail: bduvenhage@csir.co.za

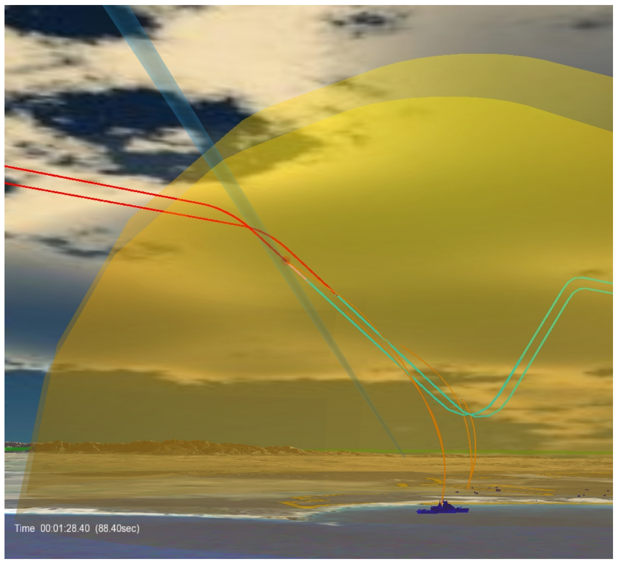

Figure 1: *A Visual Representation of a Synthetic Environment*

1.1 Background

For the purposes of this paper, terrain representations are divided into two basic groups viz. *regular* grid-post—a.k.a. height-map—and triangulated *irregular* network (TIN) representations. Section 3 gives more details on these and the advantages/disadvantages of regularised versus irregular terrain representations.

According to Mlaker [2004], and Fought and Kull [1993], a large portion of current LOS algorithms are designed for grid-post terrain and use a common basis to determine whether a sensor at position A has LOS to a target at position B. This common basis may be phrased as, *stepping* in small increments along the line AB and applying a line-of-sight-to-terrain hit test at each step. A classic and *efficient* line stepping algorithm for grid-post terrain is the Bresenham [1998] algorithm. Bangay [1993] has however shown that an optimised Digital Differential Analyser (DDA) line stepping approach can be almost twice as efficient as a naive Bresenham algorithm implementation. Further, Boyer and Bourdin [2000] have shown that N-step—a.k.a. multi-step—and their *Auto Adaptive* extensions to the Bresenham algorithm can be up to four times as efficient as the naive Bresenham algorithm. A disadvantage of these algorithms is however that they all have $O(n)$ performance, where n is some measure of the distance between the sensor and target.

Fought and Kull [1993] mention that, for TIN terrain, an *intersection of line segments* algorithm can outperform a line stepping algorithm when high frequency terrain features are absent and the terrain has a low number of triangles. The TIN algorithm's performance is potentially linearly related to the number of triangle edges in the TIN. The TIN LOS algorithm then approximately has $O(n^2)$ performance, where n is a measure of maximum LOS query length.

Section 4 explains the particulars of the above two classic LOS algorithms and the advantages of each over the other. However, for large scale virtual environments the $O(n)$ or worse performance of these algorithms, where n may be generalised to a measure of the length of LOS queries, becomes prohibitively slow. More efficient LOS algorithms and terrain representations are therefore required.

1.2 The Problem Statement and Previous Work

Calculating LOS information does indeed consume significant amounts of processing time. Salomon et al. [2004] quotes synthetic environment studies indicating as much as 27 % of processing time typically being consumed by LOS queries. Further, as the number of virtual participants (observers and observees) increase, the number of LOS queries and therefore the processing time required may potentially increase exponentially. As stated by Fought and Kull [1993], LOS algorithms are however a trade-off between storage requirements, computational time and *accuracy*. Due to the importance of optimum utilisation of computing resources a number of LOS algorithms are reported in the literature to therefore *either* more efficiently compute LOS information *or* alternatively reduce the performance impact of LOS queries by pre-computing or caching LOS information.

The first group of existing algorithms, *which attempt to compute LOS information efficiently*, seem to make use of either: the Graphics Processing Unit (GPU) and computer graphics techniques, or a CPU ray tracing approach and related accelerated quad- or kd-tree terrain geometry. The second group of existing algorithms, which *focusses on reducing the performance impact of LOS queries* on the simulation, typically makes use of memory- and lookup-efficient approaches and data structures to store pre-computed LOS information. Section 5 discusses examples of the above mentioned two approaches and the details, advantages and disadvantages of each algorithm in terms of performance and accuracy. In summary:

- The current GPU approaches potentially have better than $O(n)$ performance. An average of 4 μs per-query has been measured by Salomon et al. [2004] even on the hardware of a few years ago. GPU approaches are however only efficient for large batches of LOS queries. GPUs are relatively inefficient for small batches due to high CPU to GPU communication overhead.

- The existing ray tracing—with quad- or kd-tree terrain geometry—approaches have better than $O(n)$ performance at 3-4 μs per-query as measured by Funfzig et al. [2007], but existing implementations seem to have focussed on flat earth and does not directly support *spherical* earth scenarios.

- The pre-computed LOS approaches potentially have $O(1)$ performance, but the accuracy is roughly proportional to the cache size and inversely proportional to the scenario size for a fixed cache size.

The proposed min/max LOS algorithm was developed for a Ground Based Air Defence (GBAD) synthetic environment, as discussed by Duvenhage and Nel [2008]. The terrain data procured for use is 100 m grid-post Digital Terrain Elevation Data (DTED-level 1) mapped to a spherical earth. An algorithm for the *grid-post* terrain is considered. There is however scope for future research in using a TIN algorithm as, according to Chamberlain et al. [2003], there are well studied algorithms to efficiently convert between the terrain representations. It is *important to note* that man-made objects such as bridges that can not be represented by terrain augmentation is not directly included in the LOS algorithms discussed.

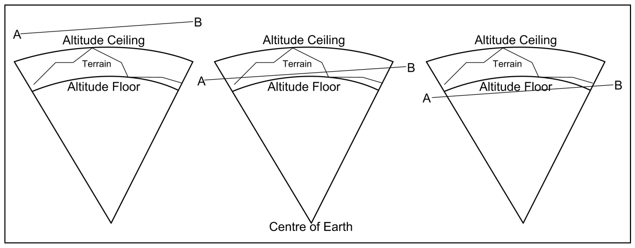

Figure 2: *Min/Max LOS Test Evaluates to (Left to Right): Certain True, Uncertain, and Certain False.*

For the GBADS application it was also decided to experiment with the first group of *dynamic* LOS algorithms that guarantee *per-query-accurate* results. Although future research may look into this decision, the pre-computed LOS cache size that would be required for a pre-computed cache based approach to LOS—especially for the high accuracy and large scenario sizes—seemed impractical in terms of time required for pre-computation and the run-time memory requirement. This article further focuses on a *CPU* LOS algorithm which has better than $O(n)$ performance for spherical earth. The value of implementing the implicit min/max kd-tree or similar algorithm on a GPU should be investigated in future to possibly further improve on the performance and/or explore algorithm variations.

The proposed min/max algorithm, in particular, exploits *the value of low resolution terrain data* to efficiently calculate LOS information. The lower resolution max tests have been implemented by both Chamberlain et al. [2003] and Funfzig et al. [2007] for their flat earth LOS algorithms as described in Section 5. The proposed *min/max* approach however combines the lower resolution max tests with lower resolution min tests—a variation of the classic smooth earth LOS obscuration test—to further optimise LOS for spherical earth.

1.3 The Basic Min/Max Algorithm

Figure 2 visually demonstrates the min/max algorithm and the value of low resolution *boundary* information. The figure shows a pie-slice of a side view of a spherical earth with some mountainous terrain. The left-hand-side diagram in the figure is an example of the LOS query—line AB—being above the altitude ceiling. In this case LOS between A and B is *certain* to be true *irrespective of the terrain detail*. The right-hand-side diagram is an example of the LOS query being below or intersecting the altitude floor. In this case LOS between A and B is certain to be false *irrespective of the terrain detail*. The diagram in the middle is an example of where part of AB is between the altitude ceiling and floor. In this case LOS is *uncertain* when the terrain detail is not known. Therefore, if the terrain's altitude ceiling/floor—the altitude min/max—is known, a very efficient constant time min/max *smooth-earth* test may be applied to first test whether the LOS query may be labelled as certain true or certain false.

The implicit min/max kd-tree algorithm, described in this article, is principally based on the above min/max test. A LOS query is firstly evaluated at a low resolution. If the result of this query is labelled as *uncertain*, then the sub-segments of the LOS query are evaluated at recursively higher resolutions until the query is resolved. The existing max algorithms by Chamberlain et al. [2003] and Funfzig et al. [2007] only does the max test and therefore can cull the tree traversal only on *LOS certain true* or *LOS uncertain* and *not on LOS certain false*.

A mip-map [Fernando and Kilgard 2003] (Latin phrase *multum in parvo* - much in a small space) is used to represent the implicit kd-tree. An example mip-map is shown later in the article in Figure 7. However, instead of an averaging function—as is usual in computer graphics—a min and a max function is applied to generate two sets of the lower levels of detail. Section 6 gives a brief introduction to kd-trees. Section 7 then discusses the details of the min/max kd-tree and LOS algorithm. Section 7 also contains details of what the cells of a grid-post terrain look like for a spherical earth.

Due to its recursive nature, the min/max algorithm's performance does not linearly depend on the query distance. The algorithm may, however, have a best case constant—$O(1)$—performance for high altitude and well-beyond horizon LOS queries, and potentially a worst case $O(n)$ performance for *short and long range* terrain-hugging LOS queries. Optimised implementations of both—max and min/max—algorithms were implemented and are evaluated over 6x6 degree flat and spherical real-world terrains. Experimental performance results and analysis are given in Section 8. In summary: For flat earth, the measured performance indicates a 2.5-7.4 % improvement above that of the implicit and explicit max kd-tree algorithms of Chamberlain et al. [2003] and Funfzig et al. [2007]. For spherical earth the min/max algorithm finds a solution in 24 - 33% of the time—a 200-300% performance increase—compared to the max kd-tree algorithm.

2 The Line of Sight Use Cases

The LOS algorithm use cases are presented in relatively large synthetic environment scenarios containing—for demonstration—100 virtual inhabitants. The ground-based simulated observers and sensors are placed at altitudes between 1.8 and 12 meter above ground while the airborne observers and sensors may potentially be at any altitude.

The first use case of the LOS algorithm is to simulate a virtual inhabitant's sense of *sight*. The observer might want to know which other inhabitants it can *potentially* see or not see due to terrain obscuration. A LOS query is generated between each observer-inhabitant pair, which results in $99 + ... + 1 = 4950 = \frac{n}{2} * (n-1)$ queries when taking into account $n = 100$ observers. Considering that some or all of the inhabitants may be dynamically moving at a simulation update rate of between 10 and 100 Hz, the LOS queries might have to be calculated as many as 100 times a second. This results in roughly *half a million LOS queries per-second.*

Taking the same scenario as above, the LOS algorithm may also be used to simulate line of sight for radio communication or radar and optical sensor detections. The LOS information as generated in the previous use case is potentially only partially reusable. This is due to radio communication including the antenna height in the LOS queries and sense of sight typically being simulated to the centre of an object. For example a soldier carrying a radio has sense of sight from eye level, but his radio might have a long antenna—or operate on a lower than microwave frequency—which increases the radio horizon. Up to an additional half a million LOS queries per-second may therefore be required to simulate radio communication and sensor detections.

Again taking the same scenario as above; There might be sensors or antennas—or cellphone towers—for which the sensor or antenna coverage must be calculated for evaluation and deployment. An antenna coverage pattern indicates the area of the responsibility or effectiveness and is hugely affected by the local terrain topography. The LOS algorithms typically only support point to point LOS queries. To calculate an *area* LOS query a large set of point-to-point queries are executed. It might, for example, be required to sample at 1 degree azimuth intervals and range intervals of 500 m. For an antenna with a typical range of 50km, in the order of 360*100=36 000 LOS queries is required each time a coverage pattern is generated or updated.

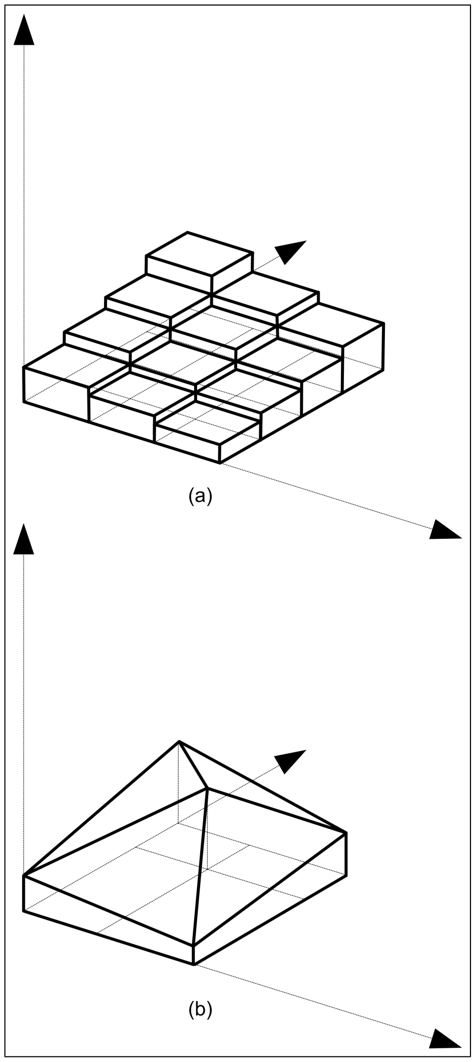

Figure 3: *Regular Grid-Post (a) and Irregular TIN (b) Terrain*

3 Regular and Irregular Terrain Representations

For the purposes of this paper, terrain representations are divided into two basic groups viz. *regular* grid-post—a.k.a. altitude-map—and triangulated *irregular* network (TIN) representations as shown in Figure 3. Grid-post terrain is named for the fact that terrain altitude *posts* are virtually placed at *regular* latitude and longitude intervals. An example of grid-post terrain is DTED. DTED level 1 represents each 1-by-1 deg of the earth as a 1 200-by-1 200 element height-above-sea-level map, placing the grid posts approximately 100 m apart.

A TIN terrain representation, on the other hand, constructs the terrain out of an *irregular* network of triangular polygons. Examples

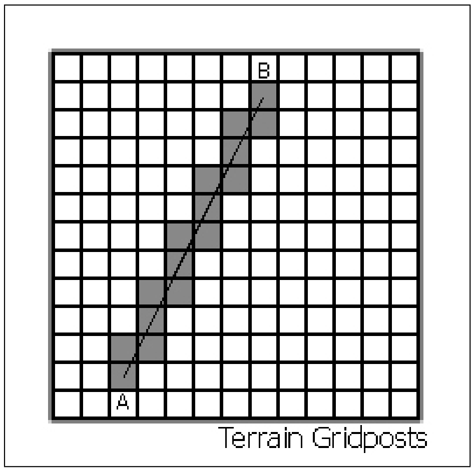

Figure 4: *Top View of the Line Stepping Algorithm*

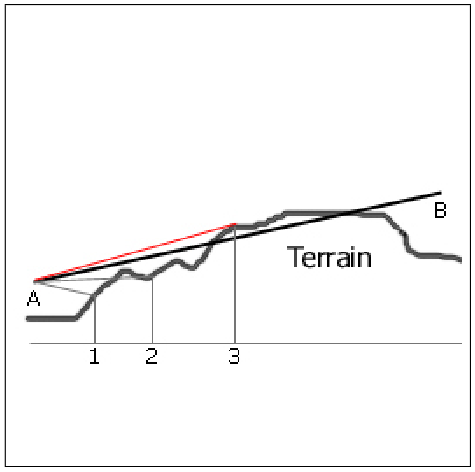

Figure 5: *Side View of the Line Step* Elevation *Hit Test*

are shown in Figure 3 (b) and Figure 6. The triangle/polygon vertices are placed at the correct terrain altitude with the polygon geometry being dependent on the terrain shape and detail. Chamberlain et al. [2003] states that there are well studied algorithms to efficiently and optimally convert from a grid-post to a TIN terrain representation if required. The reverse may be done within certain limits.

4 The Classic Grid-Post and TIN Line of Sight Algorithms

Grid-post and TIN terrain have each their own basic flat-earth LOS algorithm and approach. The line step DDA algorithm may be used to efficiently stepwise visit the required integer terrain grid positions of grid-post terrain along the surface projection of AB, as shown in Figure 4. At each step the altitude of the terrain may be found with a simple lookup into the terrain data. The line step LOS algorithm applies a line-of-sight-to-terrain hit test between the terrain altitude and AB at each step. The LOS hit test returns false at each successive step along AB until it is found that LOS between A and B is obstructed by something at the step position. A possible method, stated by Mlaker [2004], to test LOS obstruction is to compare the elevation of B to the elevation of the terrain surface at the step position. If the elevation of B relative to A is smaller than the elevation of the terrain relative to A, see step position 3 in Figure 5, then the hit test returns true and LOS is obstructed.

A line stepping approach also inherently does not sample the terrain in between steps. This leads to small features such as corners and edges of terrain features being potentially missed. The accuracy of the LOS information is therefore dependent on the accuracy of the grid-post terrain representation *and* how accurate the line steps sample the grid-post data. The influence of the step size on the accuracy of the DDA LOS algorithms are discussed in Section 8.

Modern terrain and synthetic environment visualisation systems often use TIN terrain representations to render the environment's terrain. Although the line-step LOS algorithm may be used, Fought and Kull [1993] mention that, LOS may be directly and efficiently computed over a TIN terrain by intersection of line segments. The first step in the algorithm is to find all the intersections between the 3D LOS line and the TIN's polygon edge line segments as projected on the zero-altitude surface plane. The intersections are indicated with dots in Figure 6. As these projected intersections are found, the intersection heights on the LOS line and line segments respec-

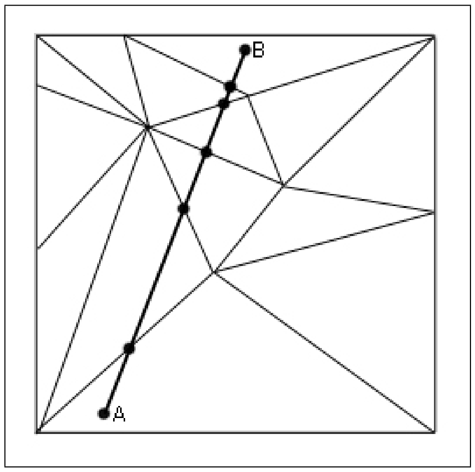

Figure 6: *Top View of the TIN LOS Algorithm*

tively are computed through substitution of the intersection coordinates into the respective line equations. If a LOS line intersection is found that is below its corresponding line segment intersection, or if A or B are under the terrain, then the LOS line intersects the terrain and LOS is blocked. According to Chamberlain et al. [2003] LOS over a flat terrain, with relatively few triangles, using a well optimized TIN can outperform a line step LOS calculation. For highly detailed terrain the number of edges can however quickly become the limiting factor in computing LOS queries efficiently as the TIN algorithm's average performance is linearly dependent on the number of polygon edges. The number of polygon edges in turn increase approximately exponentially with the terrain block's diameter.

The intersection of line segments does however not spatially discretise the search domain into steps. The LOS accuracy is therefore only dependent on the accuracy of the triangle terrain representation.

Both algorithms also support spherical earth terrain, by replacing the line stepping algorithm with an appropriate latitude/longitude curve stepping algorithm and mapping the TIN to a spherical earth. The line-of-sight-to-terrain hit test must then be applied in the spherical coordinate system and not, as before, in the flat earth coordinate system.

5 The Existing Optimised Algorithms

Existing Computer Graphics (CG) techniques may be used to efficiently process LOS queries. This is generally referred to as image based LOS due to an image being drawn that has the LOS query results encoded in it. Image based LOS is decoupled from the terrain representation and geometry as long as there is an algorithm to draw the terrain. A GPU is a specialised processor that uses a highly parallelised implementation of a stream programming model [Fernando and Kilgard 2003; Pharr and Fernando 2005]. The GPU may be used as a co-processor to accelerate the image based LOS calculations. GPU based LOS algorithms are discussed by Salomon et al. [2004] and Verdesca et al. [2006] which achieves an average of 4 μs per-query. The Top view image based LOS, for example, is algorithmically similar to the line step LOS algorithm, but with a z-buffer LOS hit test instead of an elevation condition. The communication overhead between the CPU and GPU, however, typically limits the frequency of GPU/CPU interaction—such as submitting a batch of LOS queries—to below 10 000 Hz. The hardware accelerated image analysis is therefore not efficient for small batches of LOS queries. As mentioned before, line stepping algorithms that compute LOS information suffer from sampling inaccuracies. The influence of the step size on the accuracy of the result for DDA algorithms are discussed in Section 8. Similarly, because image based LOS algorithms are pixel based, image based algorithms potentially suffer from sampling inaccuracies.

Seixas et al. [1999] describes an algorithm and R3-Tree data structure for optimised TIN LOS. They then compare the performance of the TIN LOS to that of Bresenham LOS and find that the Bresenham LOS still outperforms the optimised TIN LOS by a factor of 17.

Fought and Kull [1993] experimentally compare various line step DDA and a TIN LOS algorithm for determining line of sight in real-time simulations at the NASA-Ames Vertical Motion Simulator (VMS) facility. The LOS approach they found to be most efficient is a *grid-post terrain format in conjunction with a Digital Differential Analyser algorithm* which is similar to the Bresenham LOS algorithm.

Chamberlain et al. [2003] presents a LOS algorithm developed for the Jet Propulsion Laboratory at the California Institute of Technology. The algorithm is based on a quad-tree optimised terrain representation and ray tracing. They use the max terrain elevation within each quad-tree node to cull large pieces of terrain that are lower than the LOS query. According to the authors, the algorithm has a $O(\log n)$ performance and allows spherical earth and atmosphere to be approximated. They also compared it to an optimised TIN algorithm. The ray tracing algorithm was found to be more efficient. Chamberlain, et al.'s algorithm may also be described as an implicit max kd-tree ray tracing approach.

Similarly, Funfzig et al. [2007] have implemented a kd-tree which keeps track of the max terrain elevations within each node. They too have found that a kd-tree ray tracing approach that makes use of the low resolution elevation ceiling information has better than $O(n)$ performance and significantly outperforms the simpler line stepping and TIN LOS algorithms. Funfzig, et al. seem to have a well optimised implementation. They have achieved an average of 3-4 μs per-query. Their results also indicate a $log(n)$ relationship between the terrain size—therefore the LOS query lengths—and the average query execution time.

Mlaker [2004] gives an example of pre-computed LOS. The Probability of Line of Sight (PLOS) algorithm classifies terrain into templates according to roughness or type. Each terrain template is a mock-up of a typical block of terrain of the specific roughness and type. Every terrain template is then assigned a probability of LOS against range curve for every elevation in the chosen set of sensor elevations. A PLOS curve is calculated from pre-computing a batch of random LOS queries over the terrain template. When the system needs to determine LOS during a simulation run, it uses the sensor-to-target range, sensor elevation, and terrain type to simply determine the probability of LOS from the lookup table. The number of terrain templates, elevation PLOS curves per-template and the range resolution is finite which implies a finite set of unique cached LOS queries. A general LOS query is therefore *approximated* by choosing the closest terrain template and sensor elevation to the query and executes in a constant time.

6 An Overview of Explicit and Implicit KD-Trees

A kd-tree is a space partitioning data structure for organising points in a k-dimensional space [Bentley 1975; Groß et al. 2007]. Kd-trees are a special case of Binary Space Partition (BSP) trees. Kd-trees have for example been used to represent large synthetic scenes which then allow efficient ray traversal when doing ray tracing. In the case of the LOS algorithms, a 2 dimensional kd-tree is used to organise and store the spherical or flat earth terrain.

An implicit—as opposed to explicit—kd-tree is a kd-tree defined implicitly above a rectangular grid. The space partitioning planes are therefore not explicitly defined. A slim kd-tree is defined as an implicit kd-tree with the restriction that they can only be built over integer hyper-rectangles with side lengths that are powers of two. The slim implicit representation of a balanced kd-tree is used for its memory and time efficiency [Bentley 1975; Groß et al. 2007]. There is no overhead for keeping track of the parent child relationships as the tree is regularised.

In the case of the proposed min/max LOS algorithm the DTED grid-post map represents the leaf nodes of the tree. The successively lower resolution min/max layers of the mip-map, shown in Figure 7, then represent the intermediate to root layers of the implicit kd-tree. Before building the mip-map, the DTED data is always re-sampled to be a square map of dimension a power of two.

A min, max or min/max kd-tree is a kd-tree that, in each node, respectively stores the min, max, or min *and* max values of the node's children. The balanced min/max kd-tree for the LOS algorithm is built from the leaves to root, starting with the DTED as the leaves and systematically finding the min/max pair for each of the non-leaf nodes in the tree. The kd-tree is implicitly defined in flat latitude and longitude space for both the flat and spherical earth applications domains. Each non-leaf layer in the tree represents consecutively lower resolution min/max terrain cells and therefore the consecutively lower resolution mip-map images.

7 The Min/Max KD-Tree Algorithm Details

As mentioned, the mip-map implementation of the proposed LOS algorithm is an implicit 2D kd-tree and more specifically a slim kd-tree. This allows for an efficient time and memory footprint.

The basic building block for the LOS algorithm is the computation of line of sight between two points A and B over an $m \times n$ degree piece of flat or spherical earth, called a cell. Note that for spherical earth the latitude (North and South) boundaries are non-planar, but conical as can be seen in Figure 8. The min/max LOS test, as detailed in the introduction, is applied recursively from the tree root—lowest resolution cell—to the leaf nodes—the actual DTED pixel sized cells.

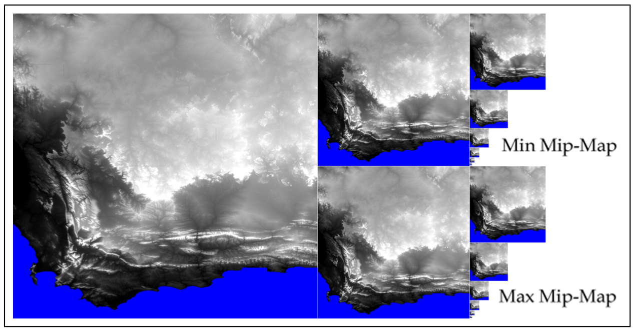

Figure 7: *An Example Mip-Map of the Western Cape, South Africa*

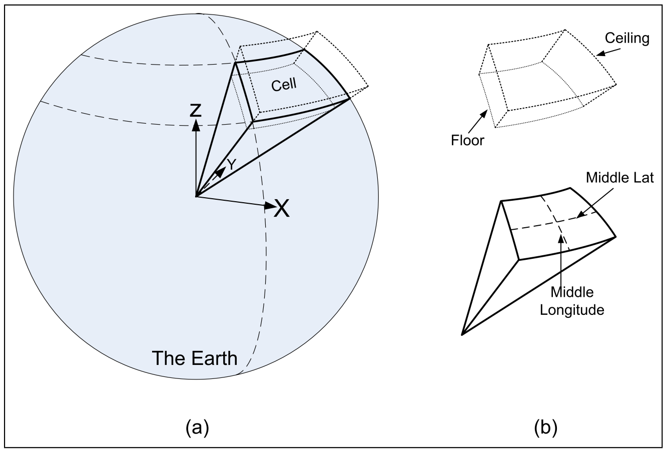

Figure 8: *The Shape of a Cell*

For each recursively evaluated tree branch, the LOS test may terminate early due to a *certain true* or a *certain false* min/max result in one of the following ways:

- A *certain true* result in general allows early culling of the current branch without reaching a result—the LOS result depends on the remaining branches.
- A *certain true* result at the root allows constant time evaluation of the query to true.
- A *certain false* result, at any point, allows early evaluation of the query to false.

The query also evaluates to true if all branches have been culled away without reaching a result. The $floor$ of AB is used in the min/max test against a cell's ceiling and floor. For flat earth, $floor(AB)$ is defined as the minimum altitude of A and B. For spherical earth $floor(AB)$ is defined as the minimum distance between AB and sea level of the spherical earth.

If, on the other hand, the min/max LOS result over a cell is *uncertain* then the LOS query is not terminated early, but propagated to the child cells pending a ray trace intersection. To intersect a line/ray AB with a cell C it is assumed that AB is contained within C. This need only be enforced at the root level by clipping AB to the root cell. Once it is known that AB is contained within C, the only line-to-cell intersections that need be calculated are the intersections of AB with the middle latitude cone and middle longitude plane that divides C into the four higher resolution child cells.

The middle longitude plane equation is found to be:
$x * n_x + y * n_y = 0.0$ for n the plane normal, and x and y - axes as in Figure 8. $n = (-sin(\theta), cos(\theta), 0.0)$ for θ = longitude.

The middle latitude cone equation is found to be:
$x^2 + y^2 = \frac{z^2}{\tan(\phi)}$ for ϕ = latitude and the x,y and z - axes as indicated in Figure 8.

The parametric form of AB, $x = t{\cdot}l + x_0$, $y = t{\cdot}m + y_0$ and $z = t{\cdot}n + z_0$, for $(l, m, n) = B - A$ and t in $[0..1]$, is then substituted into the plane and cone equations respectively to arrive at the intersection distances, viz.:
$t = -(y_0{\cdot}n_y + x_0{\cdot}n_x)/(l{\cdot}n_x + m{\cdot}n_y)$ for the plane, and
$t = \frac{-b \pm \sqrt{b_2 - 4ac}}{2a}$ for the cone where $a = l^2 + m^2 - \frac{n^2}{k}$, $b = 2x_0{\cdot}l + 2y_0{\cdot}m - \frac{2z_0{\cdot}n}{k}$, $c = x_0^2 + y_0^2 - \frac{z_0^2}{k}$ and $k = \tan^2(\theta)$.

The intersections outside AB are culled away. The resulting line of sight sub-segments that are created between the neighbours in the sorted list of coordinates A, the sub-cell intersections and B are then contained within their respective sub-cells. The centre of each sub-segment is used to calculate which sub-cell—viz. 0, 1, 2 or 3—the LOS sub-query should be submitted to. The sub-segment LOS queries are then recursively submitted to sub-cell min/max LOS tests and further subdivided as required. This recursive process is in fact a kd-tree hit-or-not ray trace algorithm.

When measuring the relative performance of the existing max algorithm the min/max algorithm is used, but the min test is simply not done until the leaf nodes—the DTED data elements—are reached. This simulates the behaviour of the max algorithm and allows the two algorithms to be accurately compared by using the same code base.

Due to the natural mapping of the grid-post terrain to cell grids, a grid-post terrain representation is preferred to easily compute the mip-map and easily identify which terrain elements belong to a certain cell. Nothing prevents a TIN terrain representation from being used though, as long as the polygons that are wholly or partly inside a cell may be found efficiently.

The *min/max kd-tree* algorithm was initially developed in 2004, independent from the max kd-tree algorithms by Chamberlain et al. [2003] and Funfzig et al. [2007]. Although the proposed algorithm is very similar in its operation to the existing max kd-tree algorithms, the min/max algorithm however includes the min test to optimise the execution for a spherical earth application domain. The implicit kd-tree representation also allows a much smaller memory and execution time footprint than possible when using an explicit non-slim tree structure.

The memory footprint of the min/max algorithm is the gridpost terrain itself, the max mip-map and the min mip-map. Each mip-map starts at *half the resolution of the initial grid-post* terrain and occupies less than half—$\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + ... < \frac{1}{2}$—of the memory of the grid-post terrain. Figure 7 demonstrates that the terrain and the two mip-maps—therefore the entire implicit min/max kd-tree—occupies *less than twice the memory of the DTED data itself.*

8 The Experimental Results

8.1 The Setup

As mentioned, the min/max algorithm is, in effect, an extension to the existing max kd-tree algorithms. To compare the performance of the new algorithm to that of the existing max algorithms an optimised implementation of a *slim* max kd-tree LOS algorithm was done. The performance of this algorithm is then experimentally determined with and without the min/max early termination extension. As mentioned, a 6x6 degree flat and spherical terrain, using

Western Cape DTED level 1, was used.

The results were generated on a laptop with an Intel Core 2 Duo 2GHz (T7200) CPU and 2 GByte of RAM. The tests were run within a single thread and therefore utilised only one of the two processor cores.

It is also argued that the regularised slim kd-tree implementation would be as fast or faster than an explicit kd-tree implementation [Bentley 1975; Groß et al. 2007]. Under this assumption the implicit kd-tree implementation may be used to represent both Chamberlain et al.'s [2003] and Funfzig et al.'s [2007] max kd-tree algorithms to, at least, the same level of optimisation as the proposed min/max algorithm.

It takes 0.37 seconds to build all the levels of detail of the min/max mip-map and in effect the implicit representation of the min/max kd-tree. Each 1x1 deg DTED tile is 1 200 x 1 200 pixels which results in a 6x6 deg terrain of 7 200 x 7 200 pixels. The terrain is then scaled up to have dimensions of a power of two viz. 8 192 x 8 192. The min-max mip-map—the implicit slim kd-tree—is created from the up-sized terrain map.

Optimised versions of a flat earth DDA and spherical earth DDA were also implemented to measure relative performance and accuracy of the line step algorithms against the kd-tree raytracing algorithms. A set of 5 000 random LOS queries were evaluated and the performance measured for each LOS algorithm. All queries were generated with A and B between 3.0 and 500.0 m above the terrain. The statistical results are shown and analysed below.

8.2 The DDA Results

For flat earth the DDA algorithm achieves an average of 5 984 queries per-second at 167.1 $\mu s/query$. An average 11 640 line step operations are executed per-LOS-query at 0.01435 $\mu s/operation$. The average number of DDA steps per-terrain-gridpost was empirically chosen to give the same LOS results as the max and min/max algorithms. For this specific experimental setup and the possibly naive *fixed*-step-size implementation, the minimum number of steps per-terrain-gridpost that is required is 10. Any lower number of samples results in LOS answers that differ from the results of the max and min/max algorithms due to the line steps that miss gridpost edges and corners.

For spherical earth the DDA algorithm achieves an average of 15 510 queries per-second at 64.48 $\mu s/query$. An average 215.0 line step operations are executed per-LOS-query at 0.2999 $\mu s/operation$. The effect of the spherical earth horizon is evident from the significantly smaller number of operations per-LOS-query compared to the flat earth DDA algorithm. This also results in the spherical DDA algorithm executing faster than the flat earth DDA algorithm, even though the spherical line step operations are more expensive than the flat earth line step operations at 0.2999 $\mu s/operation$ and 0.01435 $\mu s/operation$ respectively. The average number of DDA steps per-terrain-gridpost was empirically chosen to give the same LOS results as the max and min/max algorithms. For this specific experimental setup and the possibly naive fixed step size implementation the minimum number of steps per-terrain-gridpost that is required is 2. One hypothesis, on why the number of required DDA steps is lower for spherical DDA than flat earth DDA, is that in the spherical earth case a larger number of the LOS queries are beyond the horizon and an accurate DDA algorithm is required less often.

8.3 The Flat Earth Max and Min/Max Results

For flat earth the max kd-tree algorithm achieves an average of 329 290 queries per-second at 3.037 $\mu s/query$. The average number of max operations per-query is 38.84.

For flat earth the *min/max* kd-tree algorithm achieves an average of 337 500 queries per-second at 2.963 $\mu s/query$. The average number of min/max operations per-query is 38.03. The value of the low resolution min test of the min/max algorithm is already evident even on flat earth from the slightly lower number of operations per-query—38.03 versus 38.84.

For A and B between 3.0 and 500.0 m above the terrain, the proposed flat earth min/max algorithm executes a LOS query on average in 97.56% of the time of the max algorithm. For flat earth and at these heights above terrain, the min/max algorithms therefore does not offer a real benefit over the max algorithm. The max and min/max algorithms are however approximately 56 times faster than the flat earth DDA.

8.4 The Spherical Earth Max and Min/Max Results

For spherical earth the max kd-tree algorithm achieves an average of 60 080 queries per-second at 16.64 $\mu s/query$. The average number of max operations per-query is 28.52 at 0.5836 $\mu s/op$. The slower per-operation execution speed, compared to flat earth, is due to the terrain cells now having the more complex geometry shown in Figure 8 instead of a simple flat grid. Spherical earth does however create a horizon which allows even the max recursive algorithm to return earlier—28.52 operations versus 38.84 operations per-query for flat earth.

For spherical earth the *min/max* kd-tree algorithm achieves an average of 181 760 queries per-second at 5.502 $\mu s/query$. The average number of min/max operations per-query is however now 9.603 at 0.5729 $\mu s/op$. The added value of the min test of the min/max algorithm is evident from the much lower number of operations per-query—9.603 versus 28.52. Additionally the lower per-operation time, of 0.5729 μs versus 0.5836 μs of the max algorithm, confirms that the inexpensive low resolution min test has replaced some of the expensive sub-cell intersection tests.

For A and B between 3.0 and 500.0 m above the terrain, the proposed spherical earth min/max algorithm executes a LOS query on average in 33.05 % of the time of the max algorithm. For spherical earth the min/max algorithms therefore does indeed offer a benefit over the max algorithm. Even though there is a high cost coupled to the geometrically complex cell intersections which is not required for the spherical DDA algorithm, the min/max algorithm are approximately 12 times faster than the spherical earth DDA.

8.5 The Initial Analysis

Graphing the per-query results for the 8 192 x 8 192 spherical terrain is shown in Figure 9. The figure shows the measured per-query time against the distance between the A and B of each query. The worst case spherical DDA results that lie on the diagonal of the graph are due to LOS queries that step all the way from A to B and therefore return true. The better performing spherical DDA results are due to LOS obstructions between A and B being hit. At larger query distances, the terrain horizon would ensure early termination and better than $O(n)$ performance for n a measure of the LOS query distance.

From the figure it is clear that the spherical max and min/max algorithms do not display the $O(n)$ performance behaviour of the line

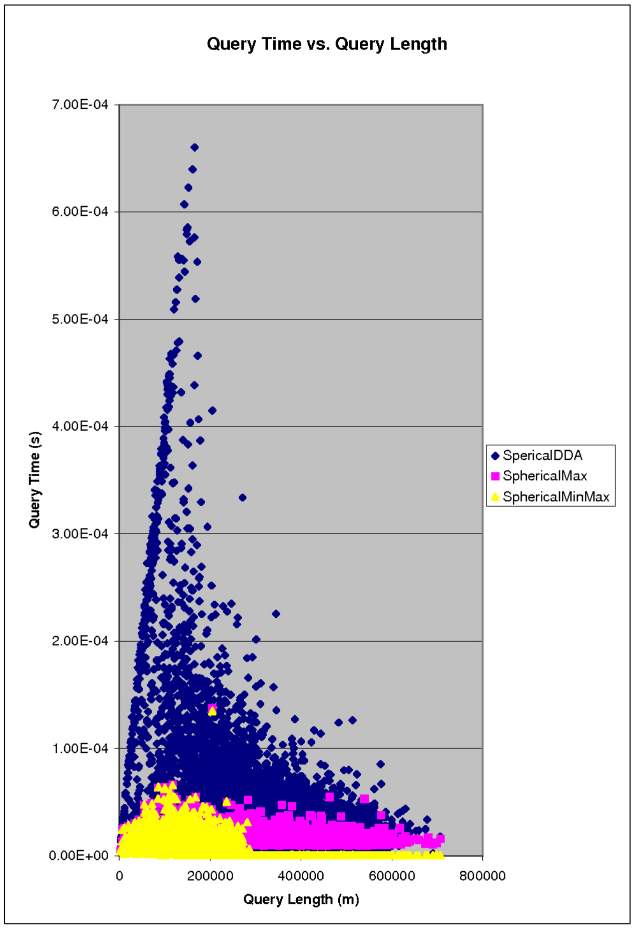

Figure 9: *LOS Query Time vs. Query Length for Real Spherical Terrain*

stepping algorithm. The performance of the min/max algorithm is also clearly superior to the max algorithm for longer ranges.

8.6 More Tests: Low Altitude

It was also found that for LOS queries where A and B are both fixed at a height of 1.0 m above the terrain, that the spherical earth min/max algorithm offers an average query time of 24.24% of that of the max algorithm—365 900 queries per-second versus 88 680 queries per-second for the max algorithm. The flat earth min/max algorithm's performance is also increased slightly to a LOS query now taking only 93.10 % of that of the max algorithm query—435 600 queries per-second versus 405 500 queries per-second for the max algorithm. For flat earth the DDA algorithm achieves an average of 39 330 queries per-second. For spherical earth the DDA algorithm achieves an average of 261 800 queries per-second.

At lower altitudes the average LOS-is-true fraction decreases—as does the average operations per-DDA-query—due to the terrain geometry and spherical earth horizon playing a larger and larger role. For the low height LOS queries the operations per-DDA-query has decreased significantly to 10.34 and 1 777 for spherical and flat earth respectively. The min/max algorithm however still outperforms these DDA algorithms for low level queries at 365 900 queries per-second versus 261 800 queries per-second for spherical earth *and* 435 600 queries per-second versus 39 330 queries per-second for flat earth. It should however be noted that the simplicity of an efficient implementation of a line step—and even N-step—algorithm does make it very attractive for applications requiring short range queries or where algorithmic simplicity is important.

8.7 More Tests: High Altitude

As may be expected, for high altitude queries—above 4 000 m in these experiments—when the terrain and horizon play only small roles in the outcome of the LOS queries, the value of the min/max algorithm over that of the max algorithm disappears. At these altitudes, however, the max and min/max algorithms are at their most efficient performance point while the line stepping algorithms are as inefficient as possible resulting in an average min/max query time that is approximately one 1000'th of that of the line stepping algorithms.

8.8 More Tests: Simplified Algorithm

Another interesting result is that if a single simple LOS test against the spherical earth's smooth sea level—or some other minimum altitude reference—is done before each LOS query, some of the benefit of the min/max algorithm is already achieved. The once-per-query check to see if line AB is below sea level potentially gives an early certain false result. In other words the low resolution min test may be approximated to the first order if each LOS query is preceded by a simple below sea level check. The min/max algorithm then, on average, only executes in 74.13 %, compared to 33.05 %, of the time of a max algorithm query for A and B between 3.0 m and 500.0 m above the terrain. The min/max algorithm also, on average, then only executes in 61.72 %, compared to 24.24 %, of the time of a max algorithm query for A and B 1.0 m above the terrain. The min/max algorithm therefore allows a 35 - 62 % performance increase beyond what a simple once-per-query below sea level check gives. The advantage of the min/max algorithm would however be lost for queries at higher altitudes when each query is preceded by a simple below sea level check.

Figure 10 shows the min/max LOS algorithm performance versus the performance of a max LOS algorithm preceded by a simple below sea level check for A and B between 3.0 m and 500.0 m above the terrain. Notice the performance advantage of the min/max algorithm between 150 000 m and 300 000 m. From the figure it is also clear that the spherical earth horizon is hit for all queries beyond approximately 300 000 m. Figure 11 similarly shows the performance advantage of the flat earth min/max algorithm over the max algorithm. Notice the good performers across all ranges at the bottom of the figure.

8.9 More Tests: Various Terrain Resolutions

Artificially lowering the resolution of the input DTED data—and in effect shortening the average LOS query without removing the curvature of the earth—results in the performance behaviour shown in Table 1.

The results in Table 1—also graphically shown in Figure 12—indicate that the DDA algorithm has, on average, $O(n)$ performance while the min/max algorithm has better than $O(n)$ performance for n a measure of the LOS query distance. From the figure the performance of the min/max algorithm seems to be $O(\log n)$ which agrees with Chamberlain et al.'s [2003] analysis of their max algorithm.

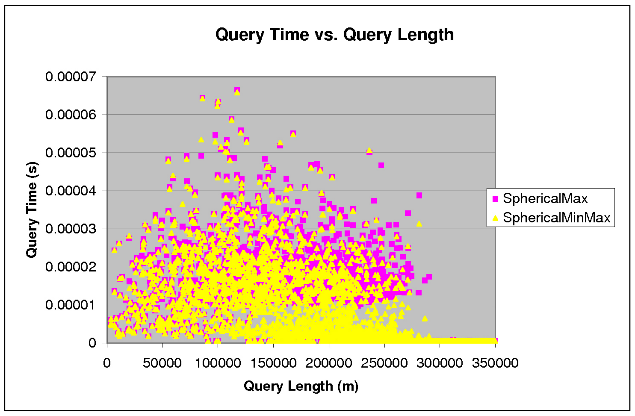

Figure 10: *LOS Query Time vs. Query Length for Real Spherical Terrain when each query is preceded by a simple below sea level check*

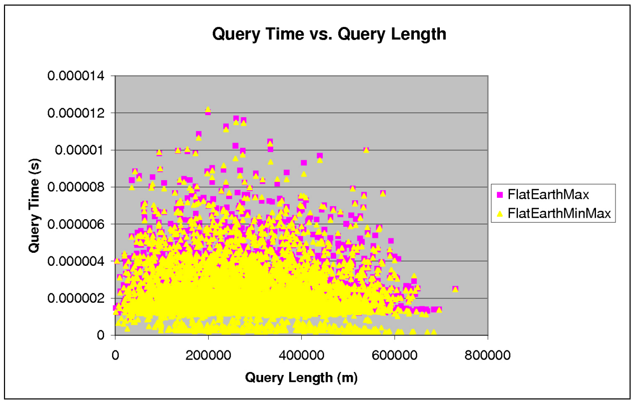

Figure 11: *LOS Query Time vs. Query Length for Real Flat Terrain when each query is preceded by a simple below sea level check*

9 Conclusion and Future Work

9.1 Results Analysis and Conclusion

For *spherical earth* LOS, a once-per-query smooth earth LOS test against some minimum altitude reference results in a factor of 2.55 performance increase compared to the max algorithm without such a pre-query. Specially considering the horizon within the spherical LOS algorithm therefore already creates a significant performance enhancement. Similarly significant performance improvement would be gained by extending a spherical earth DDA algorithm with such a simple pre-query.

The complete min/max algorithm is responsible for a further 2.5 - 7.4 % and 35 - 62 % performance increase for flat earth and spherical earth respectively, for A and B placed 1.0 - 500.0 m above the terrain. The min/max algorithm is, in general, most valuable for surface to surface—such as ground-to-ground or ocean-to-coast—type LOS queries on flat *and* spherical earth.

The complete spherical min/max algorithm includes the adaptive smooth earth early termination condition and exhibits a factor of 3.03 performance increase compared to the max LOS query. The min/max algorithm was also shown to have better than $O(n)$—and indeed on average $O(\log n)$—performance for n a measure of the LOS query distance. To demonstrate the value of $O(\log n)$: For spherical earth the min/max algorithm did, on average, 9.603 operations per-LOS-query while the non-recursive DDA did 215.0 operations per-query. For flat earth—not impeded by the horizon—the min/max algorithm did 38.03 operations per-LOS-query while the non-recursive DDA did a much higher 11 640 operations per-query.

Map Dim	Min/Max Query Time (μs)	DDA Query Time (μs)
8192	5.485	64.32
4096	4.868	32.57
2048	4.386	16.86
1024	3.843	9.050
512	3.386	5.170
256	2.900	3.242
128	2.493	2.241
64	2.139	1.745

Table 1: *Query Time vs. Map Dimension for Min/Max and DDA*

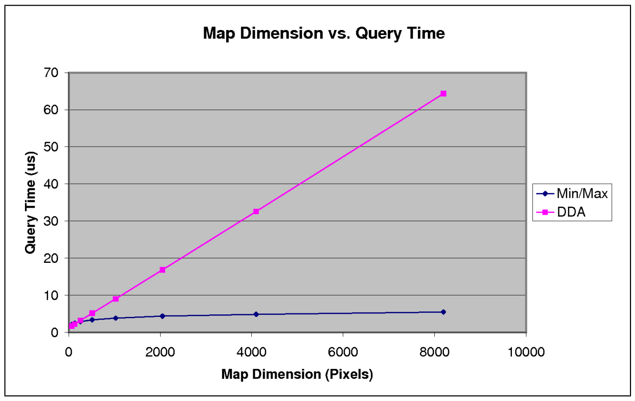

Figure 12: *Query Time vs. Map Dimension for Min/Max and DDA*

At 337 500 and 181 800 queries per-second for the flat and spherical earth respectively, the min/max LOS algorithm does not quite meet the demands of the use cases in Section 2. Reaching the 500 000 - 1 000 000 queries per-second required in the sketched use cases can be accomplished by distributing the computational load of the synthetic environment across multiple PCs as discussed by Duvenhage and Kourie [2007; 2008]. Further research should however be done on creating even more efficient, possibly fine-grained parallelised, LOS algorithms as discussed below.

9.2 Future Work

It was consciously decided to investigate a CPU LOS algorithm and not a parallelised GPU algorithm. Technological advances since early 2003 have however ensured that GPUs are now general purpose programmable. This makes the GPU a very powerful computing platform for new LOS algorithms. The stream programming model is ideally suited to algorithms that require a high ratio of arithmetic intensity to communication intensity. NVidia's new Compute Unified Device Architecture (CUDA) technology allows the GPU to be programmed in standard C. This allows easier migration of CPU LOS algorithms to the GPU, but it also opens the door to new LOS algorithms tailored to the GPU's stream programming architecture. The value of implementing the min/max algorithm on the GPU should be investigated to possibly further improve on the performance.

It has further been found that, for at least these battlefield simulations, a large fraction of LOS queries are spent in calculating area coverage of sensors or such things as antennae patterns (use case 3 in Section 2). Finding an efficient solution to the area/volume LOS query is therefore a hard, but potentially valuable problem to solve.

Acknowledgments

The author would like to acknowledge and give thanks for the funding provided by the Armaments Corporation (ARMSCOR) of South-Africa, the South-African DoD, the CSIR—where the LOS algorithm was initially developed [Duvenhage and Nel 2008]—, and the National Research Foundation without which this research would not have been possible.

References

BANGAY, S. 1993. *Parallel Implementation of a Virtual Reality System on a Transputer Architecture*. Master's thesis, Rhodes University, Grahamstown, South-Africa.

BENTLEY, J. L. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM 18*, 9, 509–517.

BOYER, V., AND BOURDIN, J. 2000. Auto-adaptive step straight-line algorithm. *IEEE Computer Graphics and Applications 20*, 5, 67–69.

BRESENHAM, J. E. 1998. Algorithm for computer control of a digital plotter. *Seminal Graphics: Poineering efforts that shaped the field*, 1–6.

CHAMBERLAIN, R., GONZALEZ, J., GUTT, G., AND TAILOR, E. 2003. New line of sight algorithm renders superlative tins superfluous. In *Proceedings of INFORMS Annual Meeting*.

DE BEAUCLAIR SEIXAS, R., MEDIANO, M., AND GATTASS, M. 1999. Efficient line-of-sight algorithms for real terrain data. In *Proceedings of SPOLM 1999*.

DUVENHAGE, B., AND KOURIE, D. 2007. Migrating to a real-time distributed parallel simulator architecture. In *SCSC: Proceedings of the 2007 summer computer simulation conference*, Society for Computer Simulation International, San Diego, California, USA, 575–582.

DUVENHAGE, B., AND NEL, J. J. 2008. A line of sight algorithm for a system of systems air defence simulation, DPSS-SM-MSADS-006 Rev 3. Tech. rep., South African Council for Scientific and Industrial Research.

DUVENHAGE, B. 2008. *Migrating to a Real-Time Distributed Parallel Simulator Architecture*. Master's thesis, University of Pretoria, Pretoria, South-Africa.

FERNANDO, R., AND KILGARD, M. 2003. *The Cg Tutorial: The Definitive Guide to Programming Real-Time Graphics*. Addison-Wesley, New York, USA.

FOLEY, J., VAN DAM, A., FEINER, S., AND HUGHES, J. 1997. *Computer Graphics: Principles and Practice, Second ed.* Addison-Wesley, New York, USA.

FOUGHT, D., AND KULL, F. 1993. Line-of-sight determination in real-time simulations. Tech. rep., American Institute of Aeronautics and Astronautics (AIAA).

FUNFZIG, C., ULLRICH, T., FELLNER, D., AND BACHELDER, E. 2007. Empirical comparison of data structures for line-of-sight computation. In *Proceedings of Intelligent Signal Processing 2007*, IEEE Computer Society Press, Alcala de Henares, 291–296.

GROSS, M., LOJEWSKI, C., BERTRAM, M., AND HAGEN, H. 2007. Fast implicit kd-trees: Accelerated isosurface ray tracing and maximum intensity projection for large scalar fields. In *Proceedings of Computer Graphics and Imaging - 2007*.

MLAKER, J. 2004. *Aggregate Models for Target Acquisition in Urban Environments*. Master's thesis, Naval Post Graduate School, Monterey, California, USA.

PHARR, M., AND FERNANDO, R., Eds. 2005. *GPU Gems 2*. Addison-Wesley, New York, USA.

SALOMON, B., GOVINDARAJU, N., AND SUD, A. 2004. Accelerating line of sight computation using graphics processing units. In *Proceedings of the 24th Army Science Conference*.

VERDESCA, M., MUNRO, J., AND HOFFMAN, M. 2006. Using graphics processor units to accelerate onesaf: A case study in technology transition. *Journal of Defence Modelling and Simulation 3*, 3, 177–187.