Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

RECURSIVE MIDPOINT SEARCH FOR LINE OF SIGHT

Paul F. Evangelista

Office of Data and Analytics
Taylor Hall
United States Military Academy
West Point, NY 10996, USA

Vikram Mittal

Department of Systems Engineering
Mahan Hall
United States Military Academy
West Point, NY 10996, USA

ABSTRACT

Line of sight (LoS) calculation and LoS analytics support a wide variety of applications, particularly simulations that involve interactions of entities across simulated terrain. This research proposes a LoS algorithm that recursively searches the midpoints of array segments and shows significant efficiency over a naive linear search. The algorithm casts every LoS query into an array of length 2^n, enabling precise and complete indexing of an array ordered by the recursive midpoints of the array. This method samples the array with broadness and symmetry. Experimental results demonstrate significant efficiencies when compared to linear search approaches for LoS. This search algorithm has the potential to apply more broadly to any situations that benefit from the binary search of unknown data that may be autocorrelated.

1 INTRODUCTION

Agent-based simulations and games rely on line of sight (LoS) calculations to support a nearly continuous flow of simulated decisions and actions (Seixas, Mediano, and Gattass 1999). In particular, in combat simulations, LoS calculations impact a wide variety of decisions and actions, with the most typical adjudications involving the outcome of multiple engagements across multiple entities (e.g., soldiers, tanks, military units). A first step in modeling an engagement is determining if the different entities have LoS of a potential target. The constant stream of LoS calculations quickly become a costly component of processing requirements for simulation models (Floriani and Magillo 1993). Salomon et al. (2004) report that LoS calculations can exceed 40% of computing costs in military simulations. While pre-processing LoS can limit some run-time processing requirements, customized terrain representations and the need to model ever-changing urban terrain create endless requirements for LoS calculations in simulations and other computer models.

In addition to adjudicating visual contact between two entities, LoS calculations can also support decision making. LoS influences a wide number of tactical military decisions such as route planning, site selection, and target search (Evangelista et al. 2011; Evangelista et al. 2013). Using LoS for these types of decisions remains an active area of research.

This paper introduces a new algorithm that has the potential to substantially reduce the number of calculations required to determine LoS. The proposed algorithms leverage the inherent autocorrelation of terrain elevation and systematically distribute data sampling, or search, across the range between the observer and target. Searching the values of an array efficiently often requires direct manipulation of the array as a data structure, and this costs processing time. The algorithms in this paper leverage the structure of arrays of length 2^n to compute precise midpoint indexing that searches arrays with broadness and symmetry. Recursive binary search and midpoint search algorithms have been researched in computer science for decades, however most applications involve sorted arrays and indexing of associative arrays (Bentley 1975; Williams 1976). The novelty of the proposed algorithms involves searching every array as an array of length 2^n in order to leverage fast array search through precise indexing of midpoints to support LoS adjudication.

2 OVERVIEW OF LINE-OF-SIGHT CALCULATIONS

LoS algorithms commonly represent an imaginary ray between a target and an observer. LoS algorithms systematically check whether or not terrain intersects this ray, which would result in broken LoS between the target and observer. The frequency of interrogating this ray and discrete representation of the ray vary as a function of the terrain representation and the granularity of the algorithm. Regardless, it is necessary for the algorithm to check multiple points along the ray. Depending on the distance between the target and observer, this could involve hundreds of LoS checks. In the event that LoS exists, every point will be checked and the sequence of checking is irrelevant. In the event that LoS does not exist, it is ideal for the algorithm to discover this by examining the fewest number of points possible.

Bartie and Mackaness (2017) extensively investigated LoS computing costs and algorithmic approaches to more efficiently perform LoS caculations. Bartie and Mackaness explain the need for efficient LoS search and propose several algorithms, to include a simple linear search, checking every n points, and a 'divide and conquer' approach with strong similarity to the algorithm proposed in this paper. Bartie and Mackaness directly manipulate arrays to support a complete 'divide and conquer' search, which they admit creates costly processing as the arrays grow in size (Bartie and Mackaness 2017, p. 822). The algorithms in this paper rely on precise indexing created by arrays of length 2^n, capitalizing on the benefits of 'divide and conquer' without incurring the costs of array manipulation.

The algorithms proposed by Bartie and Mackaness (2017) and within this paper create a more efficient search by leveraging the spatial autocorrelation inherent in terrain elevation data. Nearby points are commonly similar in height, and for this reason it does not make sense to check points consecutively—one should expect runs of similar results with this approach, which can avoid finding intersections that will efficiently end the search early. It is more efficient to check points in a much more distributed manner that minimizes the effect of the autocorrelation in terrain elevation.

Figure 1 depicts the scenario where an observer has LoS to their target and when they do not. In these depictions, the observer and target are separated by 200 points. When an observer has LoS to the target, every intermediate point between the observer and target must be tested to ensure that none of them occlude the view. However, since a single point can block LoS, once that point is detected, the algorithm can stop testing. As such, in the example shown in Figure 1 (right), a traditional LoS algorithm would need to test approximately 80 points before determining the LoS is broken.

The terrain depicted in Figure 1 was developed through integrating a sequence of uniform random numbers between -0.5 and 0.5. With this technique, the terrain elevation at point n is simply the sum of all the random numbers between 0 and $n-1$. This technique captures the natural autocorrelation commonly found in terrain elevation (Laib 1977). The changes are gradual as terrain elevation typically does not have significant deviations unless there is a dramatic terrain feature present.

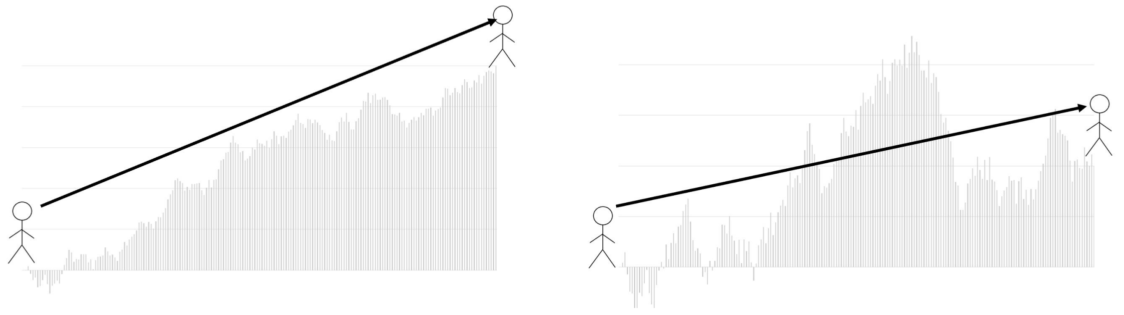


Figure 1: Observer/Target combination with unbroken LoS (left) and with broken LoS (right).

3 MIDPOINT SEARCH ALGORITHM

Given that terrain is typically autocorrelated, if a point does not break LoS, the points immediately adjacent to it will have a low probability of breaking LoS. As such, the testing of adjacent points in sequence is inefficient. Meanwhile, random sampling of points does not guarantee that every point will be tested to ensure LoS. This paper suggests a new algorithm where the sampling points are much more spread out, using a ”divide-and-conquer” approach commonly used in search algorithms (Abbiw-Jackson, Golden, Raghavan, and Wasil 2006).

This style of algorithm performs a fractional cascading, where a data set is continuously divided into smaller sections for analysis similar to methods found to ‘divide-and-conquer’ in parallel computing systems (Atallah, Cole, and Goodrich 1989). A ”divide-and-conquer” algorithm has been shown to reduce the number of searches required to find a non-random point (Zhang, Yu, Qin, and Shang 2015). Although these algorithms are commonly used in searching and processing large data sets, they have not been applied to terrain analysis, especially LoS calculations.

This paper presents two algorithms for determining LoS. Both algorithms are based on the same logic, where the terrain data is broken into sections, and the midpoint for that section is tested for breaking LoS. If that point does not break LoS, the data is further divided, until all points have been tested.

3.1 Basic Midpoint Search Algorithm

Given an array of length l and a desire to conduct recursive midpoint search, this algorithm selects unique positions across l possible positions by imagining our vector is the first l elements of a vector of length 2^u where $u = \texttt{roundup}(\texttt{log}(l)/\texttt{log}(2))$. The algorithm samples at the following quantiles: (1/2, 1/4, 3/4, 1/8, 3/8, 5/8, ... i/j) but never let i/j exceed $l/2^u$.

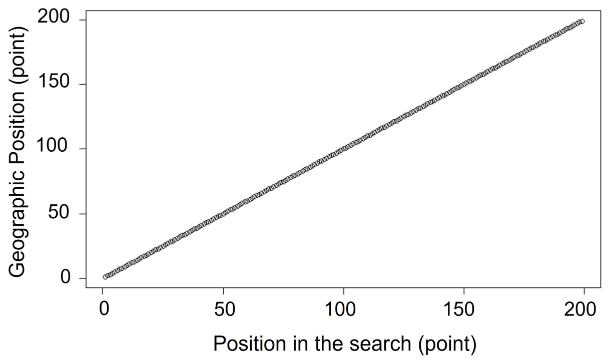
Algorithm 1 Midpoint search algorithm

```
1: $l$ = length of array to sample
2: $u$ = roundup(log($l$)/log(2))
3: for $k = 1$ to $u$ do
4:     $j = 2^k$
5:     $i = 1$
6:     while $i < lj/2^u$ do
7:         $f = i/j$
8:         $x = (2^u)f$
9:         Test LOS at index = $x$. If LOS test fails, end the process.
10:        $i = i+2$
11:    end while
12: end for
13: Note: There is no test for LOS at index = $l$, because we assume this is the target.
```

With this algorithm, if n=10, the order of search points would be 8, 4, 2, 6, 10, 1, 3, 5, 7, 9. Hence, no two consecutive points are being tested. Additionally at the lower values of k, the spread between the sample points is large to identify large-scale changes in the terrain.

Figure 2 compares the indexing sequence of linear search for determining LoS to the basic midpoint search algorithm. The horizontal axis shows where in the search algorithm a point is evaluated to determine if it breaks the LoS plane. The traditional linear approach starts at the observer and increments by one step until either it reaches the target or LoS is broken. Since terrain has a natural autocorrelation to it, if a point does not break LoS, there is a high likelihood that the adjacent points would not either.

The midpoint search algorithm interrogates the array with a broad and symmetric pattern, allowing for an initial coarse sampling of the terrain. The algorithm recursively searches midpoints with decreased



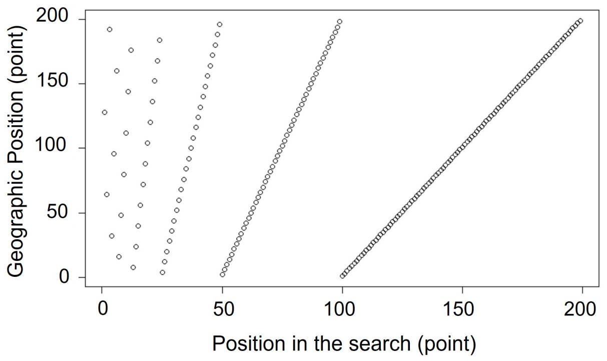


Figure 2: Linear LoS calculation (left) versus midpoint search algorithm (right).

separation, continuing the search until all the points have been tested or LoS is broken. If the observer has LoS to the target, then all points must be tested regardless of the algorithm. However, if the observer does not have LoS to the target, there is a higher likelihood that the midpoint search algorithm will find a point that breaks LoS in fewer comparisons than the linear approach.

3.2 Improved Midpoint Search Algorithm

As the number of points between the observer and target increases, the sampling pattern approaches linear behavior during the latter part of a sampled sequence (i.e. at high values of k). Therefore, a second algorithm is presented that allows for a continued spread of sample points, even for later passes, shown as algorithm 2.

Algorithm 2 Improved midpoint algorithm

```
1:  $l$ = length of array to sample
2:  $u$ = roundup(log($l$)/log(2))
3:  **for** $k = 1$ to $u$ **do**
4:      $j = 2^k$
5:      $i = 1$
6:      **for** $m = 1$ to $k$ **do**
7:          $i = 2(m-1)$
8:          **while** $i < lj/2^u$ **do**
9:              $f = i/j$
10:             $x = (2^u)f$
11:             Test LOS at index = $x$. If LOS test fails, end the process.
12:             $i = (i+2)k$
13:         **end while**
14:     **end for**
15: **end for**
16: Note: There is no test for LOS at index = $l$, because we assume this is the target.
```

Figure 3 compares the indexing sequence of the basic midpoint search algorithm to the improved midpoint search algorithm. The improved midpoint search algorithm clearly creates a broader distribution of points, especially for larger values of k in the algorithm.

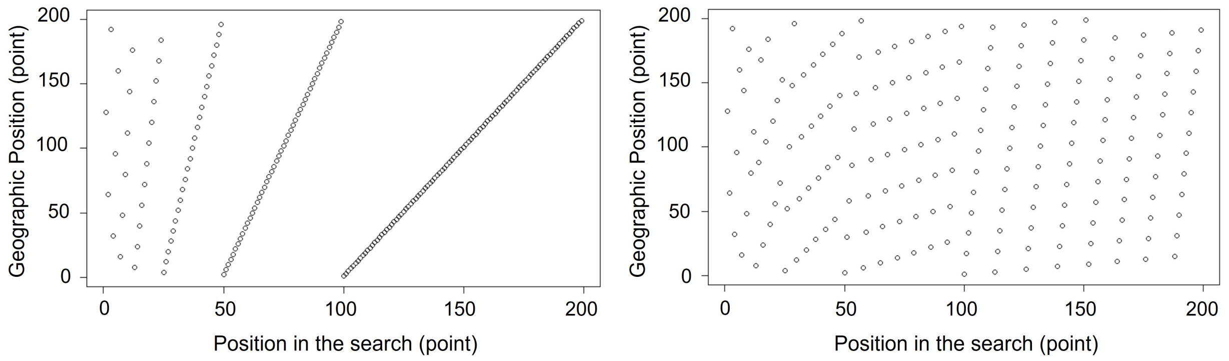


Figure 3: Linear LoS calculation (left) versus improved midpoint search algorithm (right).

4 ANALYSIS OF LOS ALGORITHMS WITH SIMULATED TERRAIN ELEVATION

The statistical programming language `R` provides an environment that supports coding and testing the algorithms in this paper with simulated terrain elevation (R Core Team 2021a). The `diffinv()` function from the `stats` package quickly creates an array of autocorrelated data similar to terrain data (R Core Team 2021b). As an example, `diffinv(rnorm(99))` will create 100 points where each adjacent point has a difference in elevation distributed with a mean of 0 and variance of 1. This approach is nearly identical with the approach used to create Figure 1.

With trial and error, it is possible to generate simulated terrain with a desired probability of LoS (P(LoS)) based upon the variance of the simulated data. The P(LoS) in natural terrain varies greatly depending on the variance of local elevation data. LoS across the ocean, as an example, would only be interrupted by the curvature of the earth and have a P(LoS) of 1 for nearby points; LoS in the Rocky Mountain region of the United States would have a P(LoS) that is much lower due to rough and undulating terrain, and this P(LoS) would approach 0 as the distance between points increases.

Each row in Table 1 represents an experimental design point with 1000 replications, each replication representing an array of *n* points created with the `R` command `diffinv(rnorm(mean=0,sd=`*s*`))`, with *s* equating to a standard deviation that created the desired P(LoS). Table 2 shows the standard deviation values used to build the experimental results in Table 1. A sample of these design points and algorithm behavior is shown in Figure 4.

Table 1 clearly shows the efficiency of midpoint search over linear search. The statistics shown in Table 1 reflect only cases with broken LoS; cases with unbroken LoS require checking every point regardless of the algorithm, making the comparison of the algorithms for these cases uninformative. As the P(LoS) increases, one could imagine that this represents less undulating terrain with gradual changes. The observer for each simulated case stood 10 units tall. Broken LoS typically did not occur close to the observer because of this height. It should also be noted that terrain where P(LoS) = 0.15 reflects relatively flat terrain, and if the LoS was broken this typically occurred at points near the end of the array.

Table 1 indicates that for every test, the ”improved” midpoint search algorithm showed worse performance than the standard technique. This trend occurred in part by how the notional terrain data was generated. Typically, a region of points will break LoS. The larger sampling gaps between consecutive samples for the ”improved” algorithm tended to miss these clusters; meanwhile, the smaller sampling gaps tended to find these clusters slightly faster. Note that the average number of interrogated points increased by less than 1 point across all the samples and was consistently lower than the number of points interrogated by the linear search algorithm. Moreover, this difference was statistically insignificant.

There is an obvious bias in this experiment against linear search because points closer to the observer are unlikely to break the LoS due to the observer height. Regardless, the experiment shows the merit of the midpoint search algorithm and has been included largely for illustrative purposes. An analysis of midpoint

Table 1: Average points interrogated before detecting broken LoS using simulated terrain elevation.

n	P(LoS)	linear	midpoint search	improved midpoint search
200	0.05	111.82	13.42	14.01
200	0.10	128.99	14.69	15.17
200	0.15	154.71	22.19	22.72
400	0.05	241.19	18.62	19.62
400	0.10	290.57	23.50	24.06
400	0.15	345.53	41.38	42.06
800	0.05	499.93	23.28	23.83
800	0.10	646.25	45.49	46.15
800	0.15	725.60	66.55	67.02
1600	0.05	1221.38	55.73	56.79
1600	0.10	1462.81	104.05	104.80
1600	0.15	1516.33	133.03	133.59
3200	0.05	2687.47	75.42	76.09
3200	0.10	2991.48	191.53	192.68
3200	0.15	3121.87	259.70	260.27

Table 2: Standard deviation values used to generate simulated terrain.

	n				
P(LoS)	200	400	800	1600	3200
0.05	0.99	0.58	0.39	0.17	0.09
0.10	0.71	0.39	0.21	0.08	0.05
0.15	0.46	0.22	0.12	0.06	0.03

search with real terrain data is presented in section 5. For a deeper investigation of alternative algorithms, see Bartie and Mackaness (2017).

5 APPLYING MIDPOINT SEARCH ALGORITHM TO REAL TERRAIN LOS CALCULATION

5.1 Experimental Design

A linear search LOS algorithm was compared to the midpoint search algorithm to compare the number of interrogations required to determine LoS integrity across natural terrain. The expectation is that the midpoint search algorithm will result in a reduction in number of comparisons.

This experiment compared the performance of the midpoint search algorithm and linear search across various terrain and distances. To ensure a representative data set, this analysis used terrain elevation data from a geotif file available from the United States Geological Survey, which provided digital elevation model (DEM) data at 7.5-arc-seconds (225 meters) (see USGS DEM viewer). For a fixed distance, a random location was selected for the location of the observer, along with a random direction between the observer and their target. The linear search algorithm and the midpoint search algorithm were coded into R to determine if the observer has LoS to their target. The number of comparisons required for each algorithm was averaged over 100 random locations and directions for distances ranging from 1 km to 400 km. The curvature of the earth was ignored. The results are shown in Table 3.

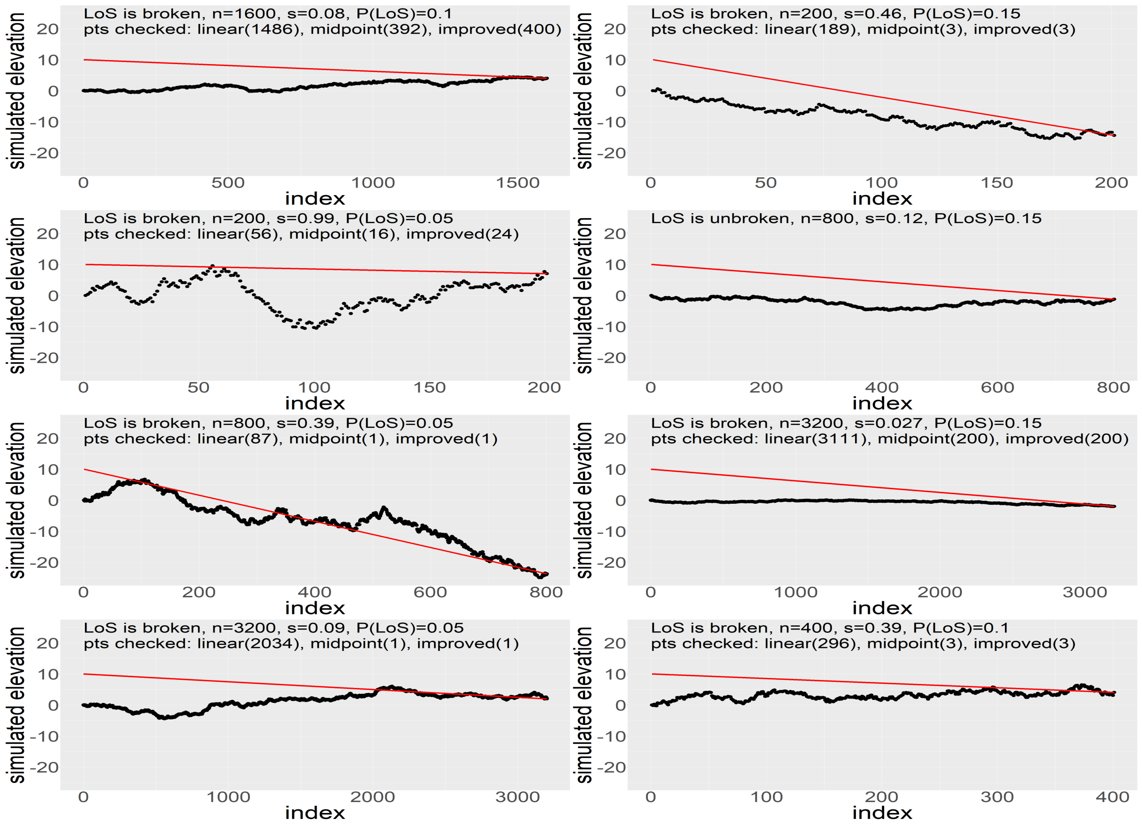


Figure 4: Sample of design points used to support simulated terrain analysis.

Table 3: Comparison of the average number of interrogations needed to detect broken LoS.

distance (km)	*n*	empirical P(LoS)	linear search	midpoint search	improved midpoint search
1	5	0.36	2.1	1.8	1.8
2	10	0.27	3.4	2.8	2.7
4	20	0.25	5.1	3.8	3.7
10	50	0.23	12.4	8.2	8.1
20	100	0.22	23.7	7.1	7.0
40	200	0.14	45.9	10.2	9.2
100	500	0.09	97.6	14.0	12.9
200	1000	0.02	160.4	7.1	6.9
400	2000	0.00	204.0	4.6	3.8

5.2 Results

The results in the last two columns of Table 3 clearly indicate a reduction in the number of interrogations between the linear search and midpoint search algorithms. In particular, as the distances increase, the reduction in the number of interrogations also increases, as further illustrated in Figure 5. Additionally, since the spacing between each sample point is approximately 200 meters, there were only 5 sample points between the observer and target for the 1 km calculation, as compared to 2000 points at 400 km. Meanwhile, only a small percentage of observer-target pairs at 400 km had LoS, resulting in 2000 point comparisons.

Figure 6 compares the basic midpoint search algorithm with the improved midpoint algorithm for the same data set. Although Table 3 indicates that the improved midpoint search resulted in a reduction in number of interrogations, the confidence intervals in Figure 6 indicated no significant change in the number of sample points compared. Typically, both algorithms found a break in LoS at low values of *k* before the two algorithms deviated significantly at higher values of *k*. The improved midpoint search algorithm did not show a significant improvement or difference over the basic midpoint search algorithm.

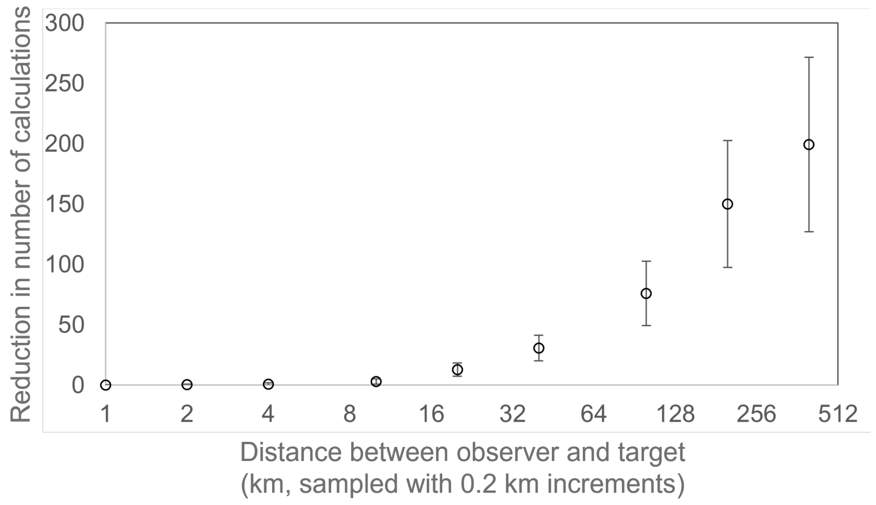


Figure 5: Reduction in number of calculations when using a midpoint approach over a linear approach.

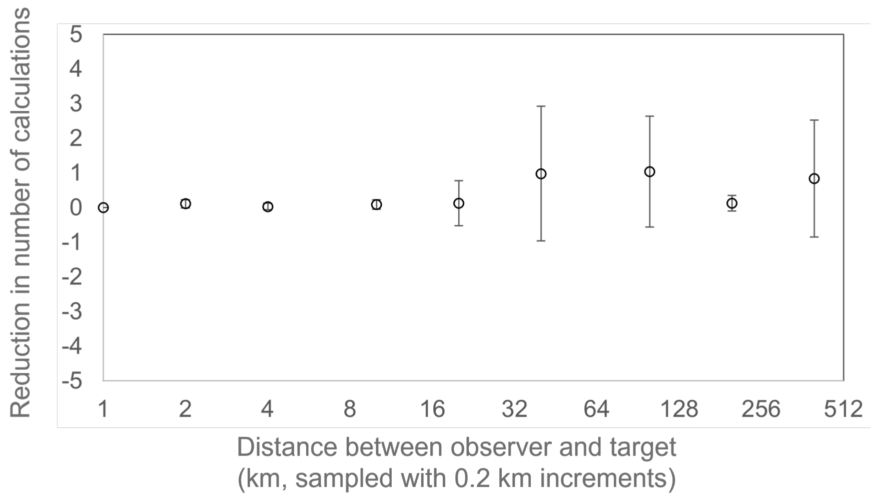


Figure 6: Reduction in number of calculations when using the improved midpoint search compared to the basic midpoint search.

6 CONCLUSIONS

LoS calculations are important across an array of applications. In particular, they play a critical role in modeling engagements in combat simulations; however, LoS calculations and analytics are cumbersome in any simulation that operates in a dynamic terrain. This paper suggests two algorithms for determining LoS. Both algorithms cast every LoS query into an array of length 2^n, enabling precise and complete indexing of an array ordered by the recursive midpoints of the array. They allow for sampling the points in an array with broadness and symmetry.

Experimental results with simulated and real terrain demonstrate significant efficiencies when compared to a naive linear search. However, these experimental results found no significant change between the two presented algorithms. This search algorithm has the potential to apply more broadly to any situations that benefit from the binary search of unknown data that may be autocorrelated.

REFERENCES

Abbiw-Jackson, R., B. Golden, S. Raghavan, and E. Wasil. 2006. "A divide-and-conquer local search heuristic for data visualization". *Computers and Operations Research* 33:3070–3087.

Atallah, M. J., R. Cole, and M. T. Goodrich. 1989. "Cascading divide-and-conquer: A technique for designing parallel algorithms". *SIAM Journal on Computing* 18:499–532.

Bartie, P., and W. Mackaness. 2017. "Improving the sampling strategy for point-to-point line-of-sight modelling in urban environments". *International Journal of Geographical Information Science* 31:805–824.

Bentley, J. L. 1975, September. "Multidimensional Binary Search Trees Used for Associative Searching". *Commun. ACM* 18(9):509–517.

Evangelista, P., I. Balogh, C. J. Darken, and J. Ruck. 2011. "Visual awareness in combat models". In *Proceedings of the 20th Behavior Representation in Modeling and Simulation (BRIMS) Conference*, edited by T. S. Jastrzembski, B. J. Best, W. G. Kennedy, and F. E. Ritter, 170–176. Sundance, Utah.

Evangelista, P., C. J. Darken, and P. Jungkunz. 2013. "Modeling and integration of situational awareness and soldier target search". *Journal of Defense Modeling and Simulation* 10(1):3–21.

Floriani, L. D., and P. Magillo. 1993. "Algorithms for visibility computation on digital terrain models". In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice*, Volume 1, 380–387.

Laib, L. 1977. "Measurement and mathematical analysis of agricultural terrain and road profiles". *Journal of Terramechanics* 14:83–97.

R Core Team 2021a. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

R Core Team 2021b. *stats: the* `R` *stats package*. R package version 4.1.

Salomon, B., N. Govindaraju, A. Sud, R. Gayle, M. Lin, and D. Manocha. 2004. "Accelerating line of sight computation using graphics processing units". In *Proceedings of the 24th Army Science Conference*, edited by H. Kwon, N. M. Nasrabadi, W.-H. Lee, P. D. Gader, and J. N. Wilson. Orlando, Florida: United States Army.

Seixas, R., M. Mediano, and M. Gattass. 1999. "Efficient line-of-sight algorithms for real terrain data". In *Proceedings of XII SIBGRAPI*, Volume 3, 1–9.

Williams, L. F. 1976. "A Modification to the Half-Interval Search (Binary Search) Method". In *Proceedings of the 14th Annual Southeast Regional Conference*, ACM-SE 14, 95–101. New York, NY, USA: Association for Computing Machinery.

Zhang, Z., J. X. Yu, L. Qin, and Z. Shang. 2015. "Divide and conquer: I/O efficient depth-first search". In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, Volume 1, 445–458.

AUTHOR BIOGRAPHIES

PAUL EVANGELISTA is a Colonel in the United States Army and an associate and academy professor at the United States Military Academy (USMA). He is currently serving as the USMA chief data officer and teaches in the Department of Systems Engineering. His email is paul.evangelista@westpoint.edu. His website is http://paul-evangelista.com.

VIKRAM MITTAL is an Assistant Professor in the Department of Systems Engineering at the United States Military Academy. His e-mail is vikram.mittal@westpoint.edu.