A FAST 3D SPATIAL ANALYSIS TECHNIQUE USING GRAPHIC PROCESS UNITS

Ji Sang Park, Senior Engineer, ETRI
Jong Min Lee, Associate Professor, Dong-Eui University
Seung Yeob Lee, Researcher
Sung Woong Shin, Senior Engineer, ETRI
Electronics & Telecommunication Research Institute (ETRI)
Daejeon, Republic of Korea, 305-700
parkji@etri.re.kr, jongmin@deu.ac.kr, cppzeus@gmail.com, sshin@etri.re.kr

ABSTRACT

More and more 3D terrain information is converted and stored in a digital format, and they have been used as a prim ary data source for a variety of spatial analyses in various application domains. One unsolved issue is that 3D spatial analyses (e.g., 3D ray tracing) often require huge computing cost in terms of processing time and hardware. This pap er presents a new technique for the fast 3D visibility analysis to find relevant building surfaces for rapid wave propag ation analysis. At the begging of this paper, a set of algorithm for fast 3D spatial indexing algorithm is described. A g raphic process unit (GPU) that assists distributed computing process is also introduced in this paper as well. The perf ormance of the proposed algorithm was tested using a set of 3D building models in Daejeon, Korea. Preliminary test results shows that the proposed method can be used for complex 3D spatial analysis with datasets of large volumes a nd in wide area.

KEYWORDS: 3D building models; 3D spatial analysis; spatial indexing; graphic process unit; ray tracing

INTRODUCTION

In conventional geographical information system (GIS) databases, terrain features or object have 2-dimensional (2D) and/or 3-dimensional (3D) shape information including area and height as well as other attribute information (i. e., texture). Many GIS applications often require complex spatial analyses to obtain their own intended results, such as wave propagation analysis.

Ray tracing is a technique to analyze diverse propagation characteristics of radio waves by tracking individual ra ys. Because of detailed tracking of every single ray, it is well known that the estimation results are more accurate tha n other analysis techniques, especially in built area with complex urban structures. On the other hand, it often require s significant amount of computing time and memory space. Several studies have been conducted to attempt to reduce the processing time of the ray tracing technique. Spatial indexing techniques have been frequently used because of it s capacity for spatial partitioning and re-organizing.

Uniform rectangular grid and/or unstructured triangular grid (Yun et al. 2000) were used for area subdivision. Bi nary space partitioning (BSP) technique was a primary solution in several studies (Torres et al. 1999; Sebbani and D elisle 2005). Visibility trees were constructed through discretion of 3D building walls into several tiles, and viewing volume test was conducted to each tile considering other tiles (Rautiainen et al. 2002; Hoppe et al. 2003; Lim et al. 2 007). Son and Myung (1999) generated similar tree, called a ray tube tree, to illustrate visibility relationships among building walls. To combine data distribution and dimensionality, Ng (2004) produced BSP trees to determine intra-o bject visibility, and then a single quadtree was developed from the BSP trees. Unfortunately, those studies couln't pr ovide a decent solution for reducing processing time due to operational characteristics of ray tracing.

This paper presents a set of algorithm for fast and efficient visibility analysis among 3D building data for fast w ave propagation analysis. As a background of this study, the basic principles of ray tracing, spatial indexing, and gra phic process unit (GPU) are briefly discussed. Then, algorithmic operations of the proposed algorithm and implemen tations are described in detail. The implemented algorithm is applied to a test datasets in Daejeon, Korea for assessin g computing time. Preliminary test results shows that the proposed method can be used for complex 3D spatial analy sis with datasets of large volumes and in wide area.

BACKGROUND STUDIES

Ray Tracing for Wave Propagation Analysis

Ray tracing is a technique to analyze the propagation characteristics (i.e., signal strength, path loss, and delay spread) of radio waves by tracing radio signals one by one from transmitting antennae to receiving ones. Radio signals often travel in a sinusoidal format. As their frequency becomes busy, their sinusoidal format becomes in a straight line. By using ray tracing techniques, the movements of every single radio signal are computed and simulated in terms of reflections and diffractions from terrains and buildings. Due to the detailed calculations, the ray tracing techniques can provide further more accurate analysis results than other wave propagation methods based upon various statistical approaches.

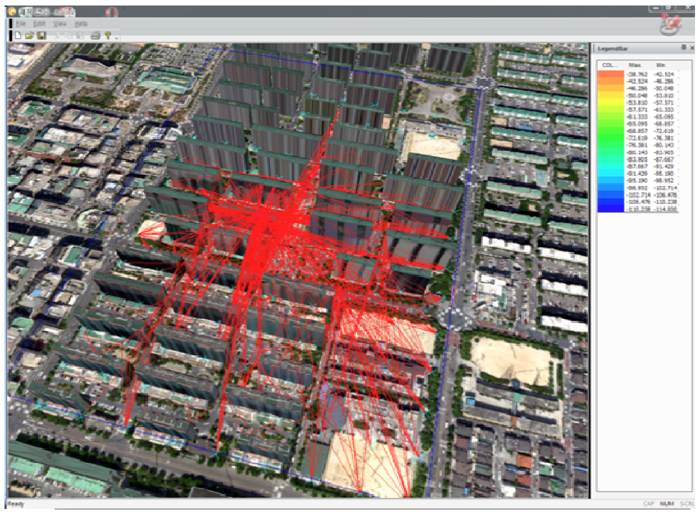


Figure 1. Screen Shots of Wave Propagation Results Using a Ray Tracing Method.

On the other hand, ray tracing techniques require significant amount of computing time and memory space. Their requirements are increased by applying higher level of reflections and diffractions. Therefore, the ray tracing technique is specifically useful to perform propagation analyses within small areas (e.g., pico- and micro-cells).

Spatial Indexing

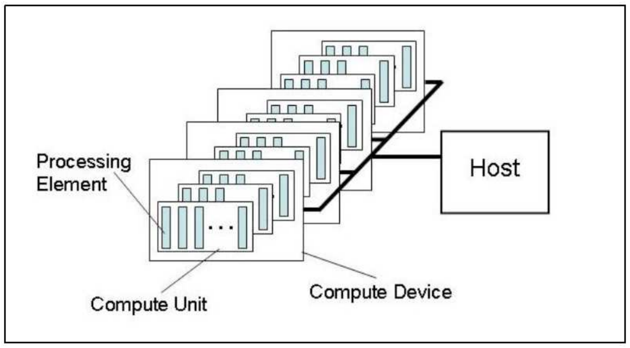
Spatial indexing is a technique to process large volume of datasets very fast by determining the spatial relationships among terrain features in an ordered sequence. Spatial indexing often starts with partitioning spaces into several subsections. As a result, the distribution of terrain objects can be identified. The, spatial relationships (e.g., vicinity, contactness, overlap, visibility) among terrain objects are determined based upon the operational demands. Meanwhile, the terrain objects are further separated into several spatial primitives (i.e., points, lines, and polygons), and those spatial primitives become the basis for determining spatial relationships. The spatial sequences among spatial primitives from terrain objects are determined and the orders results are stored into tree-type data structures. Since data can be found by searching some parts of the data structure not the entire tree structure, the less amount of tree traversing time is often required with the ordered data structures. Quadtrees, octrees, R-trees, R* trees, binary space partitioning (BSP), kd-trees are frequently used for fast data searching in many applications.

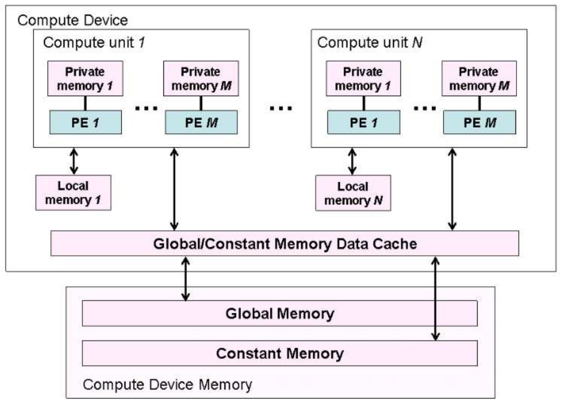
Graphic Process Unit (GPU)

A GPU's main goal is to accelerate graphics rendering and relieve the processing cost of the CPU. However, current GPU's outperforms CPU's in terms of the floating-point operation. For example, the GTX 580 shows more than 1.5 TFLOPS in single precision performance. In contrary, Intel CPU's are known to show the order of 100 MFLOPS. The fastest supercomputer in November 2010 is Tianhe-1A, which uses NVIDIA GPU (Top500 2010). The 4th fastest supercomputer, TSUBAME, is also known to use NVIDIA GPU to improve the overall performance.

As GPU's performance grows, the software environment also has been changed to develop software easily and efficiently, which uses the GPU architecture properly. OpenCL was standardized in December 2008 by the Khronos g roup . OpenCL is an open royalty-free standard for general purpose parallel programming across CPUs, GPUs and ot her processors (Munshi 2010). It is efficient to implement fine-grain parallelism by its parallel programming models and also it interoperates with OpenGL, OpenGL ES and other graphics APIs efficiently.

Figure 2(a) shows the general OpenCL platform model, which consists of a host computer and compute devices and Figure 2(b) is the conceptual OpenCL computer device architecture (Munshi 2010). The host computer should p rovide data to global memory through the bus between a host and a GPU device. For example, a compute device can be either NVIDIA GTX 580 or ATI HD 6970. Then a GPU device computes using compute units according to the k ernel program provided by the host computer. There are 512 compute units in NVIDIA GTX 580 and 1536 compute units in ATI HD 6970. After the computation, the host computer should also get the computation result in the global memory from the GPU device through the bus. The local memory in a compute device can be used for fast access to data in the global memory.





(a) Platform Model (b) Conceptual OpenCL Device Architecture

Figure 2. OpenCL Architecture.

PROPOSED ALGORITHM AND IMPLEMENTATION

Assumptions

In order to construct a simple test environment, several pre-conditions are setup for further implementation and test.

1) Terrain is for display only, line-of-sight analyses are conducted with 3D buildings only
2) For simple wave propagation analysis, reflections are considered during the analysis and deflections are exc luded.

Algorithm

Two key issues of this research are reducing unnecessary or redundant spatial searches and parallel processing o f independent spatial searches. Unnecessary searches often occur by tracking rays that are never received at antennae , and therefore they do not affect the strength of radio signals. Without proper handling, excessive processing time ca n be wasted to search the same building surfaces more than one time. Several types of spatial searches can be conduc ted at the same time at multiple places, since each building surface is located independently. A set of algorithm is des igned and implemented to reduce processing time of ray tracing-based wave propagation analysis by overcoming pro blems described above (See Figure 3).

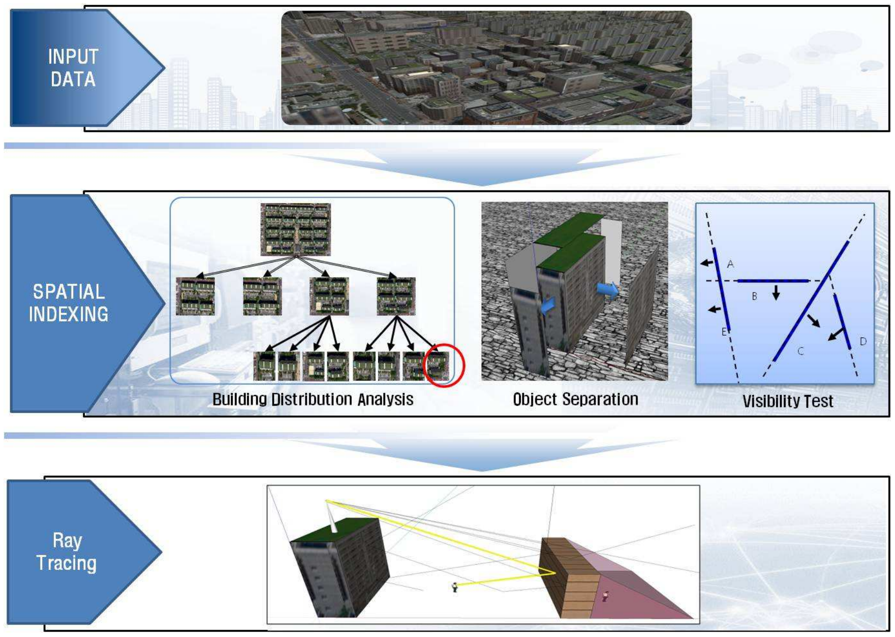


Figure 3. Research Procedure.

3D building models, DEMs, and several orthophotos are used in this study. 3D building models are mainly used for 3D spatial searches, whereas DEMs and orthophotos are used for display purpose only. First, the input data is pre pared in a VTSR format. As a XML-based generic data format, the VTST format normally contains information of b uilding footprints and heights. In this study, the VTST format is modified to store textures of building walls for realis tic 3D visualization. The VTST format is chosen because each building plane is subdivided into triangles, which bec omes the basis for defining 3D relationships among building surfaces. In addition the format can adjust building deta ils automatically based upon the level of detail (LOD) to speed up rendering process (VTP 2008).

Once input data is prepared, the spatial indexing process is conducted to arrange spatial features in order, and t o separate individual surfaces, and to determine visibility relationships among surfaces. A quadtree is constructed by recursively subdividing the extent into many sub-quadrants until when a sub-quadrant cannot be dividable (de Berg e t al .2000). The criterion of area subdivision is the distribution and density of 3D building models. The area with mor e 3D building models, the more sub-quadrants are generated, and the area with less 3D building models, the less sub-quadrants. The number of 3D building models at each leaf node is carefully controlled for prevent worst case scenari o that can happen with unbalanced tree structures. Through multiple tests, the number of 32 was selected, however, a further rigorous sensitivity analysis need to be conducted to determine the effective threshold value for the quadtree operation. Once a quadtree is generated, each building within each leaf node is further divided into individual surface s, and those individual surfaces become the basis for visibility tests among surfaces. As a "hidden surface removal" (de Berg et al. 2000) method, binary space partitioning (BSP) trees are constructed for inter-features visibility analyse s for each sub-quadrants. By recursively subdividing a pace into sub-regions by hyperplanes, a BSP tree is constructe d for back to front tree traversal from an arbitrary viewpoint, while implying planes in back node are often hidden by planes in front nodes from the viewpoint. As results, the visibility relationships among adjacent or relevant building surfaces are determined.

Finally, ray tracing operator can track the movement of multiple rays at the same time with the assistance of the GPU. From a radio transmitter, the ray tracing operator again searches potentially relevant surfaces based upon the gi ven number of reflections and/or deflections by following optic rules. Searching results are stored into a ray tube tree that contains information about which surfaces are associated to which surfaces in terms of reflections and deflectio ns. Once all propagation relevant surfaces are determined, then the ray tracing operator performs back-tracking of ra ys from a receiver to a transmitter. The back-tracking is for calculating the strength of radio signals that are finally re ceived at the receiver. These processes are often time consuming if they are executed ray by ray. On the other hand, t he processing time can be significantly reduced by computing multiple rays at the same time.

Implementations

The hardware and software environment to implement the proposed algorithm is as follows:

- Intel Core 2 Qual CPU @ 2.5GHz
- NVIDIA GTX 580 graphics card
- 4GB DDR2 memory
- Windows XP
- Microsoft Visual Studio 2005

In this paper, we use the NVIDIA GeForce GTX 580 graphics card to implement a parallel version of the propos ed algorithm developed in the single processor environment. The GTX 580 is not only a high-performance graphics card, but also it can be used for parallel processing since it shows 1.5GFLOPS peak performance in single precision operations. It is very appropriate for implementing a fine-grain parallelism.

EXPERIMENTAL RESULTS

For preliminary test, a simple test dataset with 12 3D building models in Daejeon, Korea, is prepared, and the pr oposed algorithm is applied to compare processing times among three different approaches; a) a ray tracing operatio n without spatial indexing and GPU processing, b) a ray tracing operation with spatial indexing only, and c) a ray tra cing operation with spatial indexing and GPU processing. Unfortunately, the process with GPU operation is not com pleted yet when this paper is still working on. Therefore, only two approaches (a, b) are compared at this point, but t he overall comparison will be presented during the presentation.

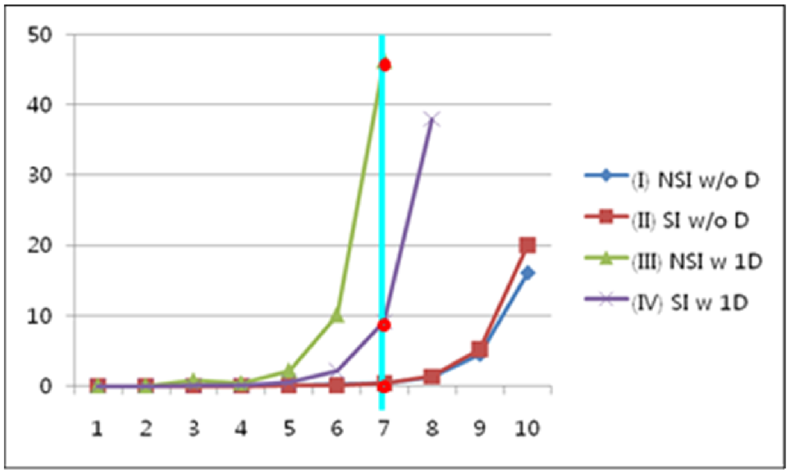


Figure 4. Processing Time Comparison.

Figure 4 shows the comparison results of different processing times occurred by the spatial indexing operation. With the cases of (I) and (II), it is shown that the it took less processing time with the ray tracing with spatial indexin g than that without spatial indexing. In addition, the advantages of spatial indexing become further prominent with th e ray tracing with spatial indexing while considering deflections from the cases of (III) and (IV). Since deflections al ways emit many relevant rays to arbitrary directions, spatial indexing routine has significant potential to effectively modeling the movement of emitted rays with less amount of processing time. In addition, the GPU routine will reduc e the overall processing time significantly further, and their results will be discussed later again.

SUMMARIES AND CONCLUSIONS

This paper proposes a set of algorithm for fast 3D spatial analysis that can process large volume of datasets with in relatively short time period, and its operational performance is evaluated by using actual datasets. The experiment al results show that the spatial indexing methods have huge potential for reducing processing time by efficiently re-ar ranging spatial objects in 3D space. Their performance will be augmented by applying distributed computing techniq ues in appropriate place. Since the demands for processing datasets with large volume and of large area are increasin g gradually, we will continue our studies to develop further advanced algorithms that will allow efficient and fast spa tial analysis.

ACKNOWLEDGEMENT

This paper is partially supported by the 2011 KT project “Development of HTML5-based 2D GIS Web App En gine.”

REFERENCES

de Berg, M. van Kreveld, M. Overmars, M. and Schwarzkopf, O. (2000), *Computational Geometry, Algorithms and Applications (2nd Edition)*, Springer, p.367.

Hoppe, R., Wolfe, G. Wertz, P., and Landstorfer, F.M. (2003), Advanced Ray-Optical Wave Propagation Modelling for Urban and Indoor Scenarios including Wideband Properties, *European Transactions on Telecommunica tions*, 14(1):61-69.

Lim, J.W., Kwon, S.W., Moon, H.W., Yoon, Y.J., and Sung, H.S., Lee, K.H., Choi, K.C., Lee, H.Y. (2007), A Study on an Advanced 3D Ray Tracing Method for Urban Propagation Environment with Terrain Geometry, *Pro ceeding of the Korean Institute of Electromagnetic Engineering & Science*, 29(3);695-698.

Munshi, A. (2010), The OpenCL Specification (Version 1.1), June 2010.

Ng, K.H., Tameh, E.K., and Nix, A.R. (2004), An Advanced Multi-Element Microcellular Ray Tracing Model, *Proc eeding of IEEE 2004 1st International Symposium on Wireless Communication Systems*, Mauritius, Sept. 20 -22, 2004;438-442.

Sebbani, Z. and Delisle, G.Y. (2005), Binary Space Partitioning Algorithm for EHF Channel Modeling, *Proceedings of IEEE International Symposium of Antennas and Propagation Society*, Washington D.C., U.S.A., Jul.3-8 ,2005,1B:665-668.

Son, H.W. and Myung, N.H. (1999), A Deterministic Ray Tube Method for Microcellular Wave Propagation Predict ion Model, *IEEE Transactions on Antennas and Propagation*, 47(8):1344-1350.

Top500 (2010), Top 500 Supercomputer, http://top500.org.

Torres, R.P. Valle, L. Domingo, M., and Loredo, S. (1999), An Efficient Ray-Tracing Method for Radiopropagation Based on Modified BSP Algorithm, Proceedings of IEEE 50th Vehicular Technology Conference, Amsterda m, the Netherlands, Sept. 19-22, 1999, 4:1967-1971.

Virtual Terrain Project (VTP) (2008), http://www.vterrain.org/

Yun, Z. Islander, M.F., and Zhang, Z. (2000), A Fast Indoor/Outdoor Ray Tracing Procedure Using Combines Unifo rm Rectangular Grid and Unstructured Triangular Grids, *Proceedings of IEEE International Symposium of Antennas and Propagation Society*, Salt Lake City, UT, U.S.A., Jul.16-21, 2000, 2:1134-1137.