GRAPHICS HARDWARE ACCELERATED FIELD STRENGTH PREDICTION FOR RURAL AND URBAN ENVIRONMENTS

M. Reyer, T. Rick, R. Mathar

Institute for Theoretical Information Technology,
RWTH Aachen University
D-52062 Aachen, Germany
E-mail: {reyer,rick,mathar}@ti.rwth-aachen.de

Keywords: Field Strength Prediction, Urban, Rural, Graphics Hardware.

Abstract

Acceleration techniques on graphics hardware for field strength prediction algorithms are presented. We identify some basic building blocks of common propagation algorithms and propose methods which can be very efficiently implemented on graphics processing units. We show that the use of such acceleration techniques leads to huge speedups in the evaluation time of propagation algorithms.

1 Introduction

Field strength prediction is of tremendous importance for planning, analysis, control and optimization of radio networks. For instance, coverage analysis, interference estimation as well as channel and power allocation are based on field strength prediction. These predictions have to be both accurate and fast in order to cope with the vast amount of different configurations to select the best candidate from.

An overview of common radio wave propagation models can be found in [4] or [16]. Models proposed in literature can be basically divided into (semi) empirical and ray optical (*ray tracing*) models. Empirical models estimate the received power predominantly on the basis of frequency and distance to the transmitter. Ray optical approaches identify ray paths through the scene, based on wave guiding effects like reflection, diffraction or scattering. Empirical models usually offer fast computation times but suffer from inherent low prediction quality if the influence of wave guiding effects is high. Ray optical algorithms are known to achieve very accurate predictions at the cost of higher computation times. Depending on the complexity of the propagation environment, the computation of ray paths is known to be the most time consuming task in ray optical algorithms.

In this paper we focus on acceleration techniques for wave propagation predictions. A very promising approach is the use of ordinary graphics cards, nowadays available in every personal computer. Modern graphics hardware combines extremely huge computing power, which is achieved by a strict parallel architecture, with a high level of programmability. The key challenge of programming graphics hardware is the unusual programming paradigm coming from the primary areas of application which is predominantly interactive computer graphics. Growing support of the so called *General Purpose Computation on Graphics Hardware* (GPGPU) led to recent changes in this architecture [11], providing more common ways of programming.

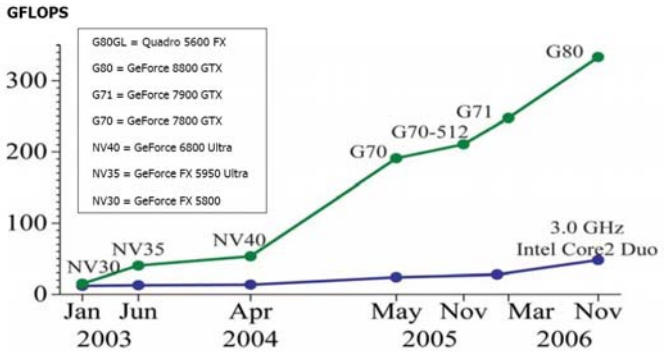

Figure 1: Floating-Point Operations per Second for CPU and GPU, cf. [11].

The computational power offered by graphics cards is already exploited for problems that go beyond graphical applications, like sorting or physical simulations. Implementations on the *Graphics Processing Unit* (GPU) often accelerate algorithms by orders of magnitude compared to standard CPU implementations. An overview on some ideas of GPGPU is presented in [7]. Recent work comprises real-time ray tracing for image synthesis [3], [10], which is of particular interest with regard to wave propagation algorithms based on ray tracing.

With the implementation of some basic building blocks of wave propagation algorithms on graphics hardware, the evaluation time of a wide range of propagation models can be accelerated significantly. Our ultimate goal is to provide wave propagation predictions for automatic planning or control algorithms on demand, thus eliminating the need for time consuming pre-processing and storing predictions in a database.

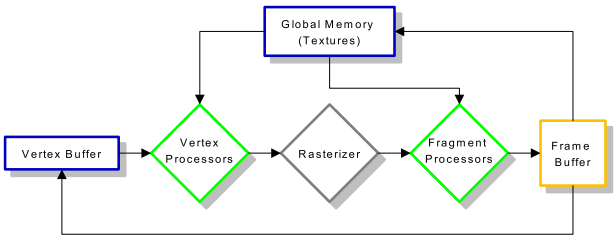

Figure 2: The Graphics Rendering Pipeline

This paper is organized as follows. In Section 2 we present the basic concepts of graphics hardware. Acceleration techniques that exploit the special architecture of graphics hardware for different propagation models are discussed in Section 3. We conclude this work with a short summary in Section 4.

2 Graphics Hardware

In the last few years, the programmable graphics processing units have evolved into an extremely powerful computing device, as illustrated by Figure 1. The main reason for the high throughput is that the Graphics Programming Unit (GPU) is specialized for computational intensive, highly parallel calculations. That is, the GPU is especially designed to support data processing, rather than data caching and flow control as is the CPU. Thus, the architecture of graphics cards is a *Single Instruction Multiple Data* (SIMD) architecture, i.e., many parallel processors simultaneously execute the same instructions at a time on different parts of data. For the computation of scientific problems as radio wave propagation prediction, the key challenge of programming graphics hardware to contribute in is to correctly map the problem related tasks to the graphics rendering context.

The programming paradigm of today's graphics hardware is best described by the stages of the *Graphics Rendering Pipeline* (Figure 2).

The input of the pipeline consists of planar geometric objects (triangles or quadrangles) which are described by their coordinates (*vertices*) and additional arbitrary numerical information (*textures*). In the first processing step of the rendering pipeline, multiple vertex processors execute in parallel the instructions from a user-written program on the vertices. Commonly, geometric transformations like translations and rotations are applied here.

In the subsequent step, the processed geometry is sampled (*rasterized*) into discrete points (*fragments*). Each fragment has a pixel position on the screen, a *depth value* and additional data.

Analogous to the vertex processors, multiple fragment processors execute a user-written program on each fragment in parallel, producing the final result of the GPU computation. Usually, the output consists of a three-dimensional vector which is commonly interpreted as color information.

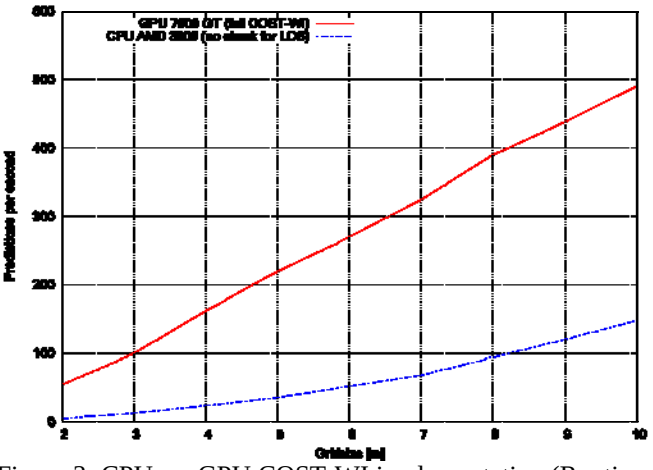

Figure 3: CPU vs. GPU COST-WI implementation (Runtimes from the COST-Munich scenario [8]).

Finally, all fragments are collected and recorded in the *frame buffer*. If multiple fragments are mapped to the same pixel position, the *depth test* specifies which one is written into the frame buffer by evaluating the fragments' depth values.

For more details on the programming of today's graphics hardware see [7].

3 Propagation Models and Acceleration Techniques

In this section we review main building blocks of common prediction algorithms and their counterpart on the GPU. A basic requirement for radio wave propagation models is to determine whether transmitter and receiver are in line of sight (LOS) or not (NLOS). Transmitter and receiver are in line of sight whenever the direct line between them is unobstructed. For instance, the well-known COST-Walfisch-Ikegami (COST-WI) model [4] gives different path loss formulas for the LOS and NLOS case.

In the graphics community the line of sight problem is very well studied and is called the *visibility problem*. Furthermore, efficient algorithms [9] have been developed which are explicitly designed to benefit from the special architecture of the GPU. A common task in graphics application is to determine the shadow of an object which is equivalent to finding all areas which are not visible by a certain light source.

The propagation environment is often described by terrain, building and vegetation data. To decide between LOS and NLOS the geometry of the propagation environment is uploaded to the graphics hardware and shadow algorithms are applied to mark the NLOS region of a certain source point. When executed on the GPU, special algorithms feature run times in the order of milliseconds, even for complex scenarios consisting of millions of triangles.

Building data for urban field strength prediction is usually represented by a polygonal outline and one height value, i.e., buildings have flat roofs. Data given in this format is called

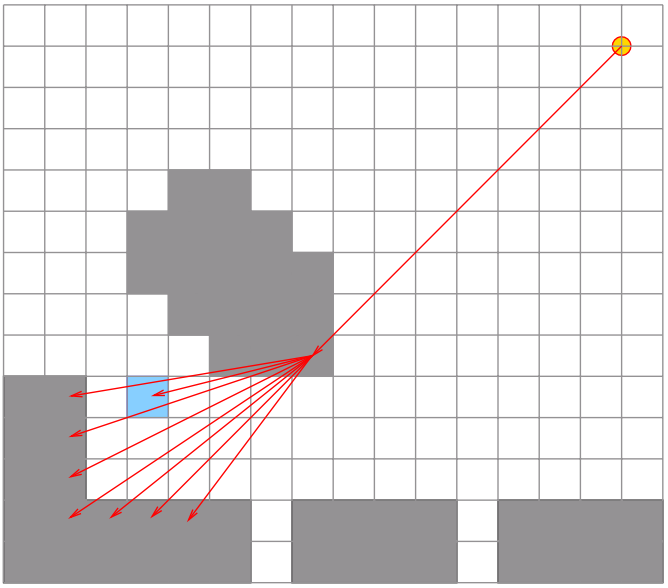

Figure 4: Discretization of propagation environment and ray launching principle.

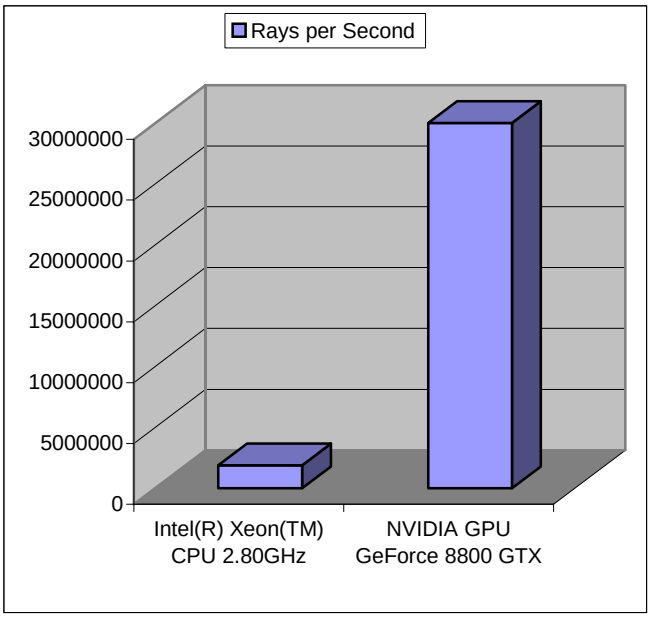

Figure 5: Rays per Second on CPU and on GPU.

2.5D. This structure can be utilized to develop an efficient shadow algorithm especially for graphics hardware, cf. [1]. Using the COST-Walfisch-Ikegami model, this implementation allows predicting up to 200 different transmitter sites in one second in the well-known COST Munich scenario [8]. This includes the distinction between LOS/NLOS for every single transmitter location and a successive path loss evaluation. For comparison we implemented just the COST-WI evaluation for line of sight on the CPU. In this particular implementation we do not check whether a point is in LOS on the CPU, which would presumably be the most expensive part of a full COST-WI implementation. Figure 3 illustrates the run times of a CPU and a GPU COST-WI implementation at different resolutions of the supplying area. It can be seen that our GPU implementation of the full model (including LOS/NLOS) outperforms even the simplified CPU implementation by orders of magnitude.

3.1 Acceleration Techniques for Ray Tracing Algorithms

Ray tracing propagation algorithms are a means to model complex indoor and outdoor propagation environments deterministically [16]. The basic task is to identify ray paths through the scene which may be deflected due to wave guiding effects like reflection, diffraction and scattering.
Ray optical algorithms are known to allow extremely accurate predictions [13], [12]. However, because of the multitude of ray paths due to complex interactions of radio waves with the propagation environment (e.g. buildings) ray tracing algorithms are computational intensive and show rather large runtimes. The most time consuming part in ray tracing algorithms is to solve the problem of visibility between objects, i.e., identify all possible sources of deflection. In standard ray tracing algorithms rays are intersected with the geometry of the scene to determine visibility. Even with efficient data structures, a lot of objects have to be tested for intersections which leads to the high computational complexity.
Existing approaches to reduce the computational complexity often rely on a (pre-) processing step, in which the building database is processed once and the possible visibility relations between objects are determined and stored for future access.
In [12], a so-called ray launching approach has been proposed. The key idea is to sample the propagation environment into a set of discrete cubes. A cube is filled, if it intersects with an object that describes the scene and is empty otherwise. Figure 4 illustrates that concept. The discrete structure allows a different approach of determining visibility of objects. Two cubes (objects) are visible to each other if all cubes on the direct line between the centers of those cubes are unfilled.
Carrying this principle over to the graphics hardware leads to two different approaches.
The first one is a backward mapping (ray tracing) algorithm, where similar to [12] rays are sampled. Starting from the transmitter, a bunch of rays is sent into all directions. As soon as a filled cube is reached on the sampled ray we stop and continue with the processing of secondary rays due to diffraction or reflection, if necessary. The sampling of a ray into a set of cubes can be done by line rasterization algorithms, well-known in computer graphics. However, line sampling algorithms are easily implemented on the graphics processing unit.
For comparison of the runtimes of line rasterization on CPU and GPU hardware we have implemented the rasterization for both. The reference scenario consisted of a 1000x600 grid where we have sampled rays from the center to every point in the grid. At each sampling step we performed a memory access and a summation of four floating point values. Total

runtime on the CPU (Intel(R) Xeon(TM) CPU 2.80GHz) was about 3.2 seconds. The GPU (NVIDIA GPU GeForce 8800 GTX) implementation exhibits a run time of only 0.02 seconds for the same computations. Hence, as Figure 5 illustrates, the CPU with roughly 187500 rays per second is clearly outperformed by orders of magnitude by the GPU which was able to process about 30 million rays per second, approximately 160 times faster than the CPU.

The second technique is a so-called forward mapping algorithm. The geometry of the scene is projected (e.g. orthogonal or perspective) onto a two-dimensional plane. This corresponds directly to drawing a scene on the screen. The geometry of the propagation environment (buildings, terrain, etc.) is uploaded to the graphics hardware and then rendered as a pixel image. Thus, each pixel corresponds to a geometric object. The rendering process is completely executed on the GPU. First, each object undergoes a certain geometric transformation and projection onto the so-called image plane, as described in Section 2. Hence, if we let the graphics card render the scene from the transmitters perspective, we can conclude which objects are visible, i.e., in line of sight, based on the numerical information in the frame buffer. Thus, the result of this computation (rendering) done on the graphics hardware provides us with a list of potential visible objects. Ray intersection computations can concentrate on these objects for finding deflection sources.

In [6] and [14] it has been observed that a detailed representation of a rural propagation environment can significantly improve the prediction accuracy. This modeling should include topographical as well as morphological information in an adequate resolution. The dominant propagation effects that have been identified in rural environment are diffractions in hilly terrain and scattering due to terrain irregularities. Again, usually ray tracing techniques are applied to approximate those effects.

For the calculation of ray paths, the terrain information has to be converted into geometric objects. This task is a well-studied object in computer graphics, and [5] propose a method for rendering height profiles on the GPU. This technique has two major benefits. First, it can efficiently deal with large and detailed terrain databases, and second, it features a built-in adaptive resolution of the geometry which is extracted from the terrain data. The main motivation for this algorithm has been to display large terrain profiles. We benefit from this for solving the visibility problem in hilly environments as only rays to visible surfaces have to be emitted.

4 Summary

In this paper, we have presented graphics hardware as a very attractive platform for the acceleration of common radio wave propagation algorithms. The graphics cards feature a large number of floating-point operations per second, due to a highly parallel architecture which, in particular is interesting for computational intensive scientific tasks like ray tracing in propagation predictions.

We identified mainly three techniques for accelerating propagation algorithms which can be entirely implemented on the GPU. These are, shadow computations for marking non line of sight regions, line sampling as a backward mapping technique to speed up ray-object intersections and direct rendering of geometry (forward mapping) to identify possible deflection sources where wave guiding effects like reflection and diffraction can be further processed. Hence, all presented techniques solve an approximation to the introduced visibility problem and thereby greatly reduce the number of objects that contribute to propagation effects.

We compared standard CPU implementations with some of the proposed GPU methods. The speedups that we achieved by an implementation on graphics hardware ranged from a factor of 10 for the COST-WI model to a factor of 160 when considering the line sampling on the CPU und GPU.

The main benefits, are on one hand a greatly reduced time for simulation and optimization algorithms where usually a vast amount of field strength prediction is required. On the other hand, faster algorithms allow higher resolutions of the supplying area and consideration of more detailed description of the propagation environment.

Acknowledgements

The authors would like to thank Valeria Gracheva for her valuable contributions.

References

[1] D. Catrein, M. Reyer, T. Rick. “Accelerating Radio Wave Propagation Predictions by Implementation on Graphics Hardware”, Proceedings: IEEE VTC Spring, (2007).

[2] D. Catrein, V. Gracheva, T. Rick, R. Mathar. “Land Cover Field Stength Prediction for Suburban Scenarios”, Technical Report, (2007).

[3] D. Weiskopf, T. Schafhitzel, T. Ertl. “GPU-Based Nonlinear Ray Tracing, Computer graphics forum, **23**, pp. 625-633, (2004).

[4] E. Damosso, Ed., COST Action 231. “Digital Mobile Radio Towards Future Generation Systems, Final Report”. Luxembourg: Office for Official Publications of the European Communities, (1999).

[5] H. Hoppe, F. Lasasso. “Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids”, ACM Siggraph, (2004).

[6] I. Schneider, F. Lambrecht, A. Baier. “Enhancement of the Okumara-Hata Propagation Model using detailed Morphological and Building Data”, In Proceedings of Personal, Indoor and Mobile Radio Communications, (1996).

[7] J. D. Owens, D. Luebke , N. Govindaraju , M. Harris,

J.Krueger , A. E. Lefohn ,T. J. Purcell. “A Survey of General-Purpose Computation on Graphics Hardware”, Eurographics 2005, State of the Art Reports, pp. 21-51, (August 2005).

[8] Mannesmann Mobilfunk GmbH, Germany. “COST 231 - Urban Micro Cell Measurements and Building Data”, [Online]. Available : http://www.ihe.uni-karlsruhe.de/forschung/cost231/cost231.en.html

[9] M. McGuire. GPU Gems, Addison Welsey, ch. “Effective Shadow Volume Rendering”, pp. 137-166, (2004).

[10] N. A. Carr, J. Hoberock, K. Crane, J. C. Hart. “Fast GPU Ray Tracing of Dynamic Meshes using Geometry Images”, Proceedings of Graphics Interface, pp. 203-209, (June 2006).

[11] NVIDIA, “CUDA Compute Unified Device Architecture”, [Online]. Available: http://www.nvidia.com

[12] R. Mathar, M. Reyer, M. Schmeink. “A Cube Oriented Ray Launching Algorithm for 3D Urban Field Strength Prediction”, Proceedings: IEEE ICC 2007, Glasgow, (June 2007).

[13] R. Wahl, G. Wölfle, P. Wertz, P. Wildbolz, F. Landstorfer. “Dominant Path Prediction Model for Urban Scenarios” 14th IST Mobile and Wireless Communications Summit, Dresden, (June 2005).

[14] T. Kuerner, D.J. Cichon, W. Wiesbeck. “Verification and Deterministic Wave Propagation Models for Rural and Urban Areas”, in Proceedings of the IEEE AP-S Int. Symposium and URSI Radio Science Meeting, pp. 1376-1379, (July 1992).

[15] T. Rick, R. Mathar. “Fast Edge-Diffraction-Based Radio Wave Propagation Model for Graphics Hardware”. ITG INICA Munich, (March 2007).

[16] T.S. Rappaport, Ed. “Wireless Communications: Principles and Practice”. Upper Saddle River, NJ, USA: Prentice Hall, (1995).