Eurographics Symposium on Parallel Graphics and Visualization (2009)
J. Comba, K. Debattista, and D. Weiskopf (Editors)

Simulation of Radio Wave Propagation by Beam Tracing

Arne Schmitz[†*], Tobias Rick[+], Thomas Karolski[*], Thorsten Kuhlen[+] and Leif Kobbelt[*]

[*]Computer Graphics Group, [+]Virtual Reality Group, RWTH Aachen University

Abstract

Beam tracing can be used for solving global illumination problems. It is an efficient algorithm, and performs very well when implemented on the GPU. This allows us to apply the algorithm in a novel way to the problem of radio wave propagation. The simulation of radio waves is conceptually analogous to the problem of light transport. However, their wavelengths are of proportions similar to that of the environment. At such frequencies, waves that bend around corners due to diffraction are becoming an important propagation effect. In this paper we present a method which integrates diffraction, on top of the usual effects related to global illumination like reflection, into our beam tracing algorithm. We use a custom, parallel rasterization pipeline for creation and evaluation of the beams. Our algorithm can provide a detailed description of complex radio channel characteristics like propagation losses and the spread of arriving signals over time (delay spread). Those are essential for the planning of communication systems required by mobile network operators. For validation, we compare our simulation results with measurements from a real world network.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—: Raytracing

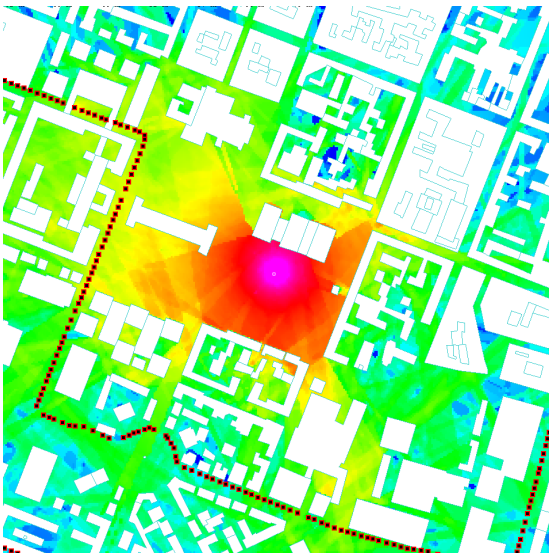

Figure 1: *An excerpt of the downtown Munich scenario, which shows the simulation result and markers of the measurement route that was taken. The raw data for the scenario is from the COST 231 project. [Dam99]*

1. Introduction

Global illumination deals with the propagation of light. It provides the basis for various image synthesis algorithms and leads to convincing, realistic results due to its physical correctness. Important propagation phenomena are reflection and refraction. Effective and efficient algorithms for solving the global illumination problem are long known [Kaj86]. Typical algorithms are often based on ray tracing techniques.

Visible light occupies only a small fraction of the electromagnetic spectrum. The propagation of waves in other frequency ranges like those of sound or radio waves behave similar to the propagation of light. Waves propagate as fronts and spread in different directions when interacting with obstacles. However, since wavelengths of sound and radio are of in size similar to the environment (up to several

[†] This work has been supported by the UMIC Research Centre, RWTH Aachen University.

meters), the effect of waves bending around corners (diffraction) is becoming important. Telecommunications systems for instance use carrier frequencies with wavelengths ranging from several meters to kilometers. Noticeable delays or echos are produced when multiple wavefronts with different travel times due to multiple propagation paths arrive at the same location. This effect is recorded in so called delay spread histograms.

In this paper we present two main contributions. First of all we developed a novel method for the efficient computation of complex radio channel characteristics by combining concepts of both electromagnetic principles and computer graphics. We apply and extend common principles of computer graphics, such as beam tracing, rasterization and general purpose GPU programming to simulate effects like diffraction that have a significant influence on the propagation behavior of radio waves. Second, we implemented a subset of a rasterization pipeline purely in software on the CUDA platform. We did this, since the OpenGL pipeline does not offer the flexibility needed to compute the 3D delay spread histograms.

We target at the propagation of radio waves of common mobile communication systems, which are in the range of several hundred MHz up to a few GHz. The knowledge of detailed radio channel characteristics is an essential requirement in the design of future communication systems. Multiple propagation paths give rise to signal fluctuations and additional propagation losses. A receiver has to deal with the arrival of signals coming from multiple directions which are spread over time. The so-called delay spread is therefore of great interest in order to provide lower and upper bounds for the lengths of incoming signals to avoid inter-symbol interference.

Both global illumination and radio wave propagation algorithms often rely on ray tracing techniques. However, high quality solutions often come at the cost of excessively high computation times. The prediction of the propagation behavior of radio waves in a common urban scenario can take from minutes up to hours. Excessive computation times still prevent the large scale deployment of exact propagation algorithms in the wireless communications community since a huge number of computations are required for the simulation of radio networks.

Therefore, we consider the reduction of computation time for radio propagation algorithms to be an important research challenge which we want to address in this paper. We show how to compute the delay spread histogram in an efficient and accurate manner, which was not possible before. We obtain a significant speedup by the following concept. Rather than tracing individual rays, the use of beams has the advantage that less undersampling occurs, since the beams represent a continuous bundle of propagation rays, which was also noted by Lehnert [Leh93]. Furthermore, we implemented the algorithm on the NVIDIA CUDA platform, since our algorithm is highly parallel in nature and as thus can utilize massively parallel compute platforms. There are a number of similar technologies emerging, apart from CUDA. For example the Larrabee platform is conceptually similar as shown in [SCS*08]. On that particular system the OpenGL pipeline is done entirely in software. Our approach uses a similar technique, implementing a subset of a standard rasterization pipeline.

The basic idea of our algorithm is to generate beams that emanate from a radiation source, and split those beams by a rasterization step, recursively creating new secondary beams. Beams are evaluated by a second rasterization step, which produces a 2D field strength map and a 3D delay spread map. All raster operations are implemented solely in CUDA, utilizing the parallel nature of the platform.

Common input data for radio wave propagation algorithms is a building database where a building is usually described by its polygonal outline and one height value for the roof. We refer to this description as 2.5 dimensional. An extension of the algorithm to full 3D is planned as future work, which will essentially enable the computation of indoor wave propagation as well.

2. Related Work

Classical ray tracing was introduced by Whitted [Whi80]. Since then it has been successfully applied and extended in numerous publications in order to compute global illumination effects based on geometrical optics. There are various publications that focus on mapping global illumination algorithms onto the GPU, which include, but are not limited to, Horn [HSHH07], Carr [CHCH06] and Dachsbacher [DSDD07]. Global illumination techniques have been used for different problems before, e.g., for sound rendering. Notable here are the works of Tsingos [TFNC01, TGD04] and Funkhouser [FMC99].

Global illumination and radio wave propagation are essentially the same problem statement, although they differ slightly in the kind of optical effects that are simulated. Diffraction and interference for example are usually left out of global illumination, due to the subtlety of the effect. But some works like Stam [Sta99] and Tsingos et al. [TFNC01] do incorporate these effects. However, the basic rendering equation, as formulated by Kajiya [Kaj86], still holds for radio wave propagation and can be used almost as is. In [ITY91] Ikegami showed that ray tracing is also an excellent technique for estimating radio propagation losses. Based on ray tracing algorithms, Schaubach [SIR92], Schmitz [SK06] and Kim [KGI*99] state that their predicted path loss values were generally within 4 to 8 dB of the measured path loss. Such predictions are considered to be of very high accuracy.

The idea of ray tracing can be extended to the concept of beams, which are a continuum of rays. Beam tracing was

introduced by Heckbert and Hanrahan [HH84]. It reduces intersection tests, as well as overcomes sampling problems, since ray samples tend to become too sparse or too dense. Many more works have been published in this area, which also concentrated on realtime rendering [ORM07], or non-graphical applications such as audio rendering [FMC99].

However, our approach takes the beam tracing idea to a different level of applications, not simulating light, but the radiation of different radio frequency bands. In combination with our novel data structures and an efficient implementation using general purpose GPU programming, this allows us to calculate field attenuation and delay spread at the same time with high accuracy and in a speed not possible before. Similar to our approach is the work by Rajkumar et al. [RNFR96], who also used a form of beam tracing for wave propagation, but determines visibility differently. Furthermore, Rick et.al. presented an GPU-based approach to radio wave propagation in Catrein [CRR07] and Rick [RM07]. They trace propagation paths in a discrete fashion by repeated rasterization of line-of-sight regions. By restricting computations to the strongest path only, propagation predictions are delivered at interactive rates. However, since only the mean received signal strength is computed, multipath effects, which are an essential requirement for delay spread estimations, are completely neglected. This is not the case with our algorithm. Besides basic propagation losses, advanced channel characteristics like the delay spread are computed at a considerably reduced run time.

3. Overview

Our approach allows to rapidly and accurately compute two important aspects of radio wave propagation: the field strength and a delay spread histogram at arbitrary points in the scene.

The algorithm consists of two parts. Building a beam hierarchy that describes the propagation of the electromagnetic radiation, and the evaluation of radio field properties based on this beam hierarchy.

The tracing algorithm relies on a small rendering pipeline similar to OpenGL, but implemented in CUDA, to determine the split positions inside of each beam. For efficiency, unnecessary geometry is clipped away by the use of a quadtree that is intersected with the beam. The pseudo code of the tracing algorithm is as follows:

```
1. Build scene geometry quadtree
2. Trace initial beams from source
  1 Clip scene against beam using quadtree
  2 Split beam according to visible geometry
  3 Generate reflected, refracted and
    diffracted beams
  4 Update signal time and attenuation
    for beam
  5 Trace recursively
```

The evaluation of the generated beams also uses a simplified rasterization pipeline, which accumulates the beam attenuation and the delay into 2D and 3D framebuffers. The pseudo code of the evaluation algorithm is thus:

```
1. Iterate over all beams
  1 Rasterize beam attenuation into 2D array
  2 Rasterize beam delay into 3D histogram
```

4. The Algorithm

The following sections describe the beam tracing algorithm, which heavily relies on our rasterization engine, which is a subset of a standard rendering pipeline. It is implemented in CUDA, and consists of transformation, clipping and rasterization steps. All steps are implemented in software only, and fully parallelized. The advantage is especially visible in arbitrarily sized framebuffers being supported, and also 3D framebuffers, which are needed for the computation of the delay spread histogram. As mentioned earlier, the implementation of such pipelines in software becomes ever more important with the advent of CPU and GPU architectures containing dozens of general purpose programming cores, instead of fixed function graphics pipelines.

4.1. Beam Tracing

A beam in 2D is defined as a quad. It represents a bundle of rays emanating from an edge, which might be degenerate in the case of a point radiation source, in which case the beam is equivalent to a triangle. Each beam carries information about signal travel time, for later evaluation of the path loss and delay spread. Our algorithm generates recursively a beam hierarchy, beginning at a radiation source. Beams are reflected, refracted and also diffracted at surface boundaries. A beam might be infinite at one end, if it does not intersect any geometry at all, or it might be finite, if it intersects the geometry of the scene.

The beam is constructed from four edges (compare Fig. 2). The two edges e_1 and e_2 are forming the beam-cone. If the beam was created due to a reflection on a building wall, it will have another edge i that coincides with the wall. We call this the image plane of the beam. This will actually be the image plane of the viewing frustum during the beam splitting step. The last edge lies virtually at infinity for beams that do not get split, or it coincides with a part of a wall for beams that have hit some part of the geometry.

4.1.1. Clipping Geometry Against the Beam

For our approach we mainly use two data structures. A quad tree for accelerating the intersection of beams with the scene geometry, and the beam hierarchy, which describes the propagation paths the electromagnetic radiation may take.

When a beam is formed, it has to be intersected with the scene geometry in order to recursively spawn new beams

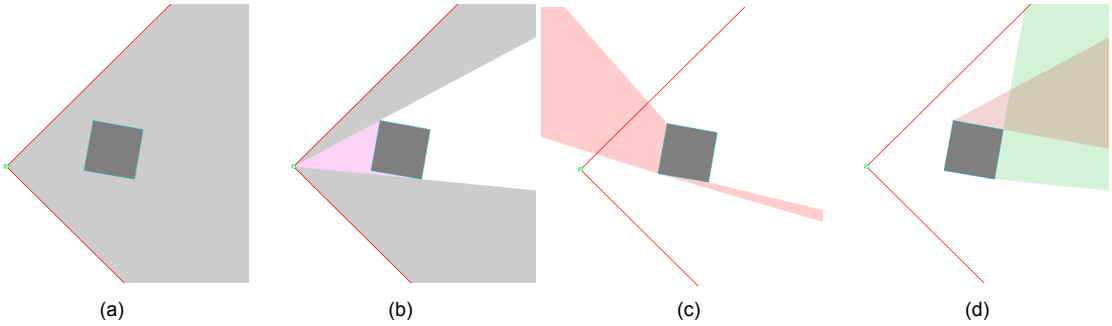

Figure 3: *(a) A beam is created and intersected with the geometry. (b) The beam is split into child-beams, according to the intersections. (c) Reflection edges are identified, and the old beam origin is reflected at those edges, constructing reflected beams. (d) In the same manner diffraction beams are generated at silhouette edges.*

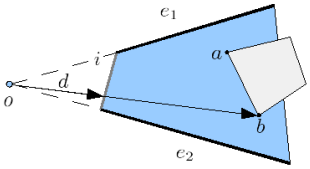

Figure 2: *A beam is defined by four edges. The two edges e_1 and e_2 form a quadrilateral whose baseline i we call the image plane of the beam. The point o is the (virtual) origin of the beam. A ray $r = o + \lambda d$ is constructed through the image plane and intersected with the face $\{a,b\}$ which was identified in the beam framebuffer.*

that in turn will form reflected and transmitted beams. The intersection is done by enumerating all quad-tree nodes that the beam overlaps with. This helps to cull away all unnecessary geometry from the computation.

4.1.2. Splitting the Beam

For fast and easy splitting of the beam into new sub-beams, we simply render the geometry contained in the beam into a 2D frame buffer, containing IDs for the geometric faces and an associated depth buffer. Since beams tend to become very thin after a small number of interactions, only few faces have to be rasterized.

When the faces that are hit have been identified, one has to compute exact intersection points of the beam with the wall. Given a beam as in Fig. 2, a ray $r = o + \lambda d$ from the beam origin o to the first rasterized intersection point of the beam with a face, and finally the face $f = \{a,b\}$, we can compute the intersection point $p = (x,y)^T$, by solving the two following equations:

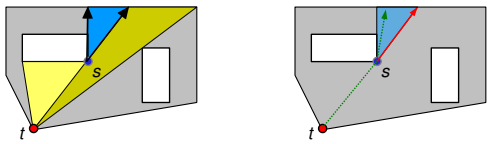

Figure 4: *(a) Diffraction beam in blue and (b) propagation path due to diffraction at building edge in green.*

$$\begin{vmatrix} x & y & 1 \\ x_o & y_o & 1 \\ x_o + x_d & y_o + y_d & 1 \end{vmatrix} = 0 \quad (1)$$

$$\begin{vmatrix} x & y & 1 \\ x_a & y_a & 1 \\ x_b & y_b & 1 \end{vmatrix} = 0 \quad (2)$$

We ensure that the resulting intersection point lies on the face being intersected, by clamping the newly created beam to the endpoints of the face being tested.

4.1.3. Generating Reflected and Diffracted Beams

If a certain recursion depth is not yet reached, we recursively generate reflected and diffracted beams, leading to a beam tree. The reflected beam is very easily constructed by mirroring the beam origin at the wall that the old beam has hit. This is shown in Fig. 3. The new beam originates from a virtual source that is constructed by simply mirroring the source of the parent beam at the reflecting face.

Diffraction beams are constructed at silhouette edges of the geometry for every parent beam that has been split at one such edge. The diffraction beam spans the area between the shadow boundary of the parent beam and the backfacing part of the silhouette, also see Fig. 4.

According to the *Geometrical Theory of Diffraction* [Kel62] diffracted rays are produced by incident rays which hit edges, corners or vertices of boundary surfaces. Diffracted wave fronts can enter the shadow regions of the objects that are the point of diffraction. The electromagnetic field is assigned to diffracted rays similar as for reflected or transmitted rays. The initial value of the field on a diffracted ray is obtained by multiplying the field of the incident ray by a diffraction coefficient. The actual values are determined by the incident direction and the diffraction direction, wavelength and physical properties (material) of the media at the point of diffraction.

Ray tracing algorithms for radio wave propagation commonly model diffraction effects by tracing a multitude of rays into the respective diffraction cones. This fits well into the concept of beams, see Fig. 4.

Note that the beam tree contains two different kinds of beams. At every even level in the tree (starting at level 0), it contains a beam that was spawned by means of reflection or diffraction. Those beams are only evaluated later if they are a leaf in the tree. If, on the other hand, they have children, they have been split. On the odd levels of the tree, there are only beams that were produced by the splitting algorithm. Those beams have to be evaluated always, since they have the sufficient visibility information to represent a valid part of the propagation path. This gives a simple rule for the second part of the algorithm, where the beams are evaluated and the path loss is computed, see Fig. 5 for an illustration.

4.2. Beam Evaluation

4.2.1. Computing the Attenuation

From global illumination we know the Bidirectional Reflectance Distribution Function (BRDF) denoted by the symbol f_r. It also exists in the context of radio wave propagation. However, measured BRDFs for wavelengths in the order of centimeters are not known. But many objects and materials in this frequency spectrum are of specular nature anyhow, in this frequency spectrum. [Rap95] So we just need to have a scene specific extinction coefficient, which can be estimated or guessed from sparse measurements.

In radio wave propagation, usually the path loss is computed, which is the attenuation of the signal. The most simple model for this is the freespace model, which expresses the path loss in dB:

$$L_{\text{fp}} = 10n\log_{10} d + C \tag{3}$$

For the path loss exponent $n = 2$ and the system loss constant $C = 0$ we get the same attenuation ($\Phi \sim \frac{1}{d^2}$), that is known from global illumination for the attenuation of the flux density with the distance to the light source. In section 4.2.2 we will further explain how to use this information to compute the final path loss at a specific point.

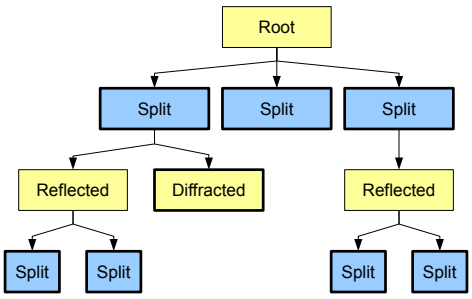

Figure 5: *The beam tree structure. Only the bold boxes will be actually rendered in the final rasterization step.*

4.2.2. Evaluating the Beam Hierarchy

After the beam tree has been generated, it has to be evaluated, to generate the final signal strength image, and to compute the delay spread histogram. This is done by traversing the beam tree and rasterizing each beam, accumulating the results in two buffers. See Fig. 5 for an illustration of the beam tree. Only beams that were created in a split step and those that are leafs of the tree will be rendered and contribute to the final field strength.

The first buffer is a 2D buffer that accumulates the signal strength, and the second buffer is a 3D buffer that contains the delay spread histogram. The algorithm rasterizes each beam, and computes the attenuation and the travel time inside of it. The attenuation is used for computing the final signal strength, and it propagates through the beam-hierarchy and through each beam with the accumulated distance from the radiation source.

Given the set B of all beams of the beam tree that lead to an individual beam b, we compute the path loss $L_p(x)$ of a point $x \in b$. We accumulate the BRDFs f_r for each reflection and also take the length of the propagation path into account. It is defined by the distance from x to the transmitter origin t_o (eq. to the $\frac{1}{d^2}$ from global illumination). Hence, the path loss is then defined as:

$$L_p(x) = \frac{\prod_{b_i \in B} f_r^i}{|x - t_o|^2} \tag{4}$$

This value can be written directly into a 2D array or framebuffer, using efficient polygon rasterization, implemented on the CUDA device. The accumulated path loss is computed by simply summing all beams into the 2D buffer.

4.2.3. Computing the Delay Spread

The path loss and the delay spread are computed at the same time. We use a 3D array for collecting the histogram. Every column in this array represents a discrete delay spread histogram for the given 2D position. When we evaluate (i.e. rasterize) a beam, we compute for each pixel the distance

$d = |x - t_o|$ the signal has travelled so far from its virtual source t_o. This can be mapped to a travelling time, which is approximately $t = d \cdot c$, where $c \approx 299 \cdot 10^6 \frac{m}{s}$, depending on the optical density of the material. This is then mapped to one of the bins of the histogram and the path loss is added to that bin. Accumulating over all beams gives an accurate estimation of the delay spread at this specific point of the scene.

In our implementation the user can specify all relevant simulation parameters, such as the maximum recursion depth, the transmitter and receiver position, and resolution of the resulting simulation. Feedback is given on the spot, and is often interactive, depending on the scene complexity and number of reflections and diffractions. The evaluation of the delay spread is obviously instantaneous for a static transmitter and a moving receiver, and allows the user to intuitively explore the temporal spreading characteristics of the signal.

5. Evaluation

In this section we will take a look at the two most important properties of our algorithm. First we analyze the space and time complexity, and second we evaluate the accuracy, compared to real-world measurements and other works.

5.1. Complexity and Performance

The performance of the algorithm depends on several parameters. First of all the complexity of the scene geometry influences the beam splitting algorithm. It is vital to our algorithm that this visibility computation is done on the GPU, since it takes up most of the time in our algorithm. An important aspect to this is to find out the optimum configuration for the chosen platform. In our case, the dimensions of the computing grid on the CUDA device had to be optimized. Experiments showed that for this algorithm 128 threads are the optimal grid size on current devices.

However, the number of recursive reflection and diffraction steps also influences the complexity of the algorithm. In the beginning, we create only one beam in a certain direction, but with every split and reflection, it will spawn a number of additional beams, thus letting the number of beams grow exponentially in the worst case. However, beams tend to get very thin after a few reflections, so that on levels near the bottom of the beam tree there will be only a few children per beam.

Because of this the evaluation of the beam hierarchy becomes linear in the number of beams to be evaluated, as can be seen in Fig. 6, and also depends on the spatial resolution of the framebuffer, which can be defined by the user in m/pixel. The evaluation itself is a simple polygon rasterization, and has been implemented using CUDA. We chose not to use OpenGL here, because it does not allow to render easily into a 3D texture, which is necessary for the delay spread histogram computation.

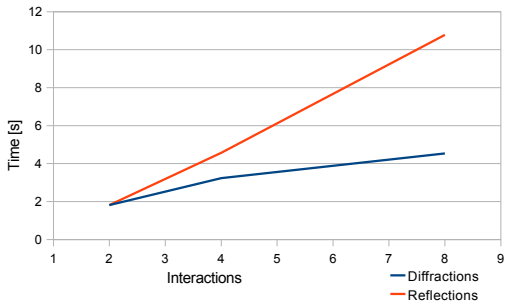

Figure 6: *Performance of our algorithm for varying levels of interactions. Computation complexity grows approximately linearly in the number of reflections.*

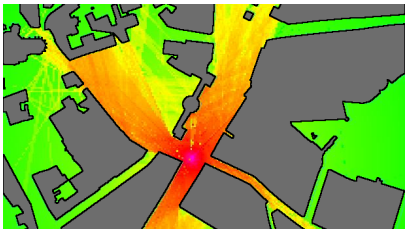

Figure 7: *An excerpt rendering from the Aachen scenario, which contains more than 90k vertices.*

In the Munich scenario (see Fig. 1) there are approximately 80.000 vertices for the buildings. This is a reasonably complex scene, which is computed in several minutes on normal ray tracers for radio wave propagation. Our algorithm computes the scene in less than 3 seconds, even with complex recursive interactions. The machine used in this case was a Core2Duo with 2.4 GHz and a NVIDIA GeForce 8800 Ultra.

On the other hand the diffracted beams are usually much broader, often with an opening angle $> 90^\circ$. This leads to much more recursively spawned rays. However it is often not necessary to do more than one or two diffraction steps, because the signal strength is attenuated very fast by diffraction. Thus it is more important to create a propagation path that diffracts only once into a street, where reflection will carry on to propagate the signal.

5.2. Accuracy

For the evaluation with measured data, we use the widely known Munich dataset from the COST 231 project, which contains a 3D model of Munich downtown and measurement data of a sample GSM base station. The measurements can be taken as a reference solution for our algorithm and were taken on three different routes on street level. We compare the measured data with our simulation to estimate the quality of the simulation. It has to be noted that there are many more aspects to radio wave propagation than only interaction with static geometry. Especially moving objects,

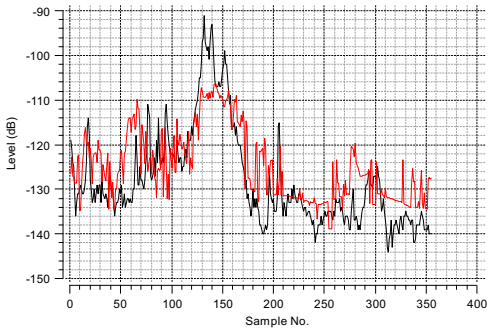

Figure 8: *A plot showing the measured results (black) versus our simulation (red) for the downtown Munich scenario.*

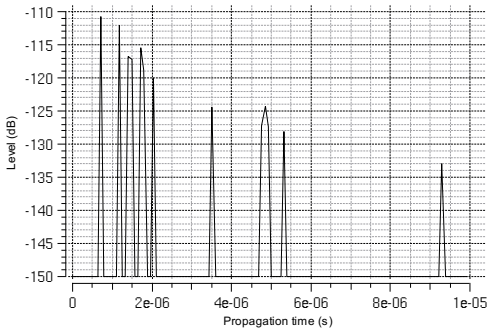

Figure 9: *A simulated delay spread histogram for one receiver position in the Munich scenario. The multiple peaks represent different propagations paths, which arise due to reflection and diffraction.*

like people or cars, or even the weather can add significant, time based noise to measurements. This will influence measurements drastically, so that most other works leave these aspects out of the simulation, and use stochastic methods to model these effects on top of the ray- or beam traced simulation. As a second scenario we use a model of the city of Aachen, containing about 90.000 vertices, covering the inner part of the city with more than one square kilometer in size, which shows the scalability of the algorithm (Fig. 7).

In Fig. 8 we show one of the route plots in black compared to our simulation in red. One can see that the overall shape of the curves match, which indicates that effects based on the scene geometry and its interaction with the radio waves are matched well by our algorithm. The higher level fluctuations in the measured signal most probably stem from dynamic effects which are not included in our algorithm. An example delay spread plot is shown in Fig. 9. Note the different spikes that are due to multiple reflections off of building walls. This will influence how well the receiver can decipher the transmitted signal, due to the resulting echos. A method to optimize the simulation by taking measured data into account is discussed by Schmitz et al. [SRK*09].

The most common form of comparing radio wave propagation algorithms, is to compute the standard deviation of the errors between the simulation and the measurements. In Table 1 we compare our method to three other state of the art works. Besides the timings and the accuracy it is also most important to compare the features of the simulation. Although Rick [RM07] is able to run the Munich scenario at interactive rates, their method does not allow for the computation of reflections, and in turn the computation of an effective delay spread.

6. Conclusion

We have shown a fast and accurate algorithm that can simulate radio wave propagation in urban environments, including important effects like multi-path propagation due to reflection and diffraction, and predict the resulting delay spread.

As future work, we would like to implement a full 3D beam tracer and a form of parameter optimization, that allows the user to tune the simulation by taking sparse measurements of the scene into account.

References

[CHCH06] CARR N. A., HOBEROCK J., CRANE K., HART J. C.: Fast gpu ray tracing of dynamic meshes using geometry images. In *GI '06: Proceedings of Graphics Interface 2006* (Toronto, Ont., Canada, Canada, 2006), Canadian Information Processing Society, pp. 203–209.

[CRR07] CATREIN D., REYER M., RICK T.: Accelerating radio wave propagation predictions by implementation on graphics hardware. In *Proc. IEEE Vehicular Technology Conference* (Dublin, Ireland, 2007), pp. 510–514.

[Dam99] DAMOSSO E. (Ed.): *COST Action 231: Digital mobile radio towards future generation systems, Final Report*. Office for Official Publications of the European Communities, Luxembourg, 1999.

[DSDD07] DACHSBACHER C., STAMMINGER M., DRETTAKIS G., DURAND F.: Implicit visibility and antiradiance for interactive global illumination. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 61.

[FMC99] FUNKHOUSER T., MIN P., CARLBOM I.: Real-time acoustic modeling for distributed virtual environments. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 365–374.

[HH84] HECKBERT P. S., HANRAHAN P.: Beam tracing polygonal objects. In *SIGGRAPH '84: Proceedings of*

Method	Accuracy	Time	Std. dev.	Delay spread	Reflections	Diffractions
Our method	5 m	1.8 s	6.18 dB	Yes	Yes	Yes
Woelfle [WWW*05]	10 m	36 s	6.41 dB	No	Yes	Yes
Rick [RM07]	5 m	0.05 s	4.5 dB	No	No	Yes
Schmitz [SK06]	5 m	9m 20s	5.82 dB	Yes	Yes	Yes

Table 1: *Comparison of our method to several other state of the art works. Our method supports all important propagation effects, and is apart from the work of Schmitz [SK06], the only method which simulates the delay spread.*

the 11th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1984), ACM, pp. 119–127.

[HSHH07] HORN D. R., SUGERMAN J., HOUSTON M., HANRAHAN P.: Interactive k-d tree gpu raytracing. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 167–174.

[ITY91] IKEGAMI F., TAKEUCHI T., YOSHIDA S.: Theoretical prediction of mean field strength for urban mobile radio. *IEEE Transactions on Antennas and Propagation 39* (1991), 299–302.

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 143–150.

[Kel62] KELLER J. B.: Geometrical theory of diffraction. *Journal of the Optical Society of America 52* (1962), 116–130.

[KGI*99] KIM S.-C., GUARINO B. J., III T. M. W., ERCEG V., FORTUNE S. J., VALENZUELA R. A., THOMAS L. W., LING J., MOORE J. D.: Radio propagation measurements and prediction using three-dimensional ray tracing in urban environments at 908 MHz and 1.9 GHz. *IEEE Trans. Veh. Technol. 48* (1999), 931–946.

[Leh93] LEHNERT A.: Systematic errors of the ray-tracing algorithm. *Applied Acoustics*, 38 (1993), 207–221.

[ORM07] OVERBACK R., RAMAMOORTHI R., MARK W. R.: A real-time beam tracer with application to exact soft shadows. In *EGSR* (2007).

[Rap95] RAPPAPORT T. S.: *Wireless Communications: Principles and Practice*. Prentice-Hall, Inc., 1995.

[RM07] RICK T., MATHAR R.: Fast edge-diffraction-based radio wave propagation model for graphics hardware. In *Proc. IEEE 2nd International ITG Conference on Antennas* (Munich, Germany, March 2007), pp. 15–19.

[RNFR96] RAJKUMAR A., NAYLOR B. F., FEISULLIN F., ROGERS L.: Predicting rf coverage in large environments using ray-beam tracing and partitioning tree represented geometry. *Wirel. Netw. 2*, 2 (1996), 143–154.

[SCS*08] SEILER L., CARMEAN D., SPRANGLE E., FORSYTH T., ABRASH M., DUBEY P., JUNKINS S., LAKE A., SUGERMAN J., CAVIN R., ESPASA R., GROCHOWSKI E., JUAN T., HANRAHAN P.: Larrabee: a many-core x86 architecture for visual computing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–15.

[SIR92] SCHAUBACH K. R., IV N. J. D., RAPPAPORT T. S.: A ray tracing method for predicting path loss and delay spread in microcellular environments. In *Proc. IEEE Vehicular Technology Conference* (May 1992), vol. 2, pp. 932–935.

[SK06] SCHMITZ A., KOBBELT L.: Wave propagation using the photon path map. In *PE-WASUN '06* (New York, NY, USA, 2006), ACM, pp. 158–161.

[SRK*09] SCHMITZ A., RICK T., KAROLSKI T., KOBBELT L., KUHLEN T.: Beam tracing for multipath propagation in urban environments. In *3rd European Conference on Antennas and Propagation* (2009), IEEE.

[Sta99] STAM J.: Diffraction shaders. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 101–110.

[TFNC01] TSINGOS N., FUNKHOUSER T., NGAN A., CARLBOM I.: Modeling acoustics in virtual environments using the uniform theory of diffraction. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 545–552.

[TGD04] TSINGOS N., GALLO E., DRETTAKIS G.: Perceptual audio rendering of complex virtual environments. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 249–258.

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Commun. ACM 23*, 6 (1980), 343–349.

[WWW*05] WAHL R., WÖLFLE G., WERTZ P., WILDBOLZ P., LANDSTORFER F.: Dominant path prediction model for urban scenarios. *14th IST Mobile and Wireless Communications Summit, Dresden (Germany)* (2005).