GPU Accelerated Ray Launching for High-Fidelity Virtual Test Drives of VANET Applications

Manuel Schiller, Alois Knoll
Lehrstuhl für Echtzeitsysteme und Robotik
Technische Universität München
Munich, Germany
{manuel.schiller,knoll}@in.tum.de

Marina Mocker, Thomas Eibert
Technische Universität München
Lehrstuhl für Hochfrequenztechnik
Munich, Germany
{marina.mocker,eibert}@tum.de

***Abstract*—Due to the complexity of Vehicular Ad Hoc Networks, future driving assistance systems need to be validated through virtual test drives in a simulated environment. An accurate modeling of the vehicle-to-vehicle communication channel is crucial to enable a precise evaluation of such network-aware applications. Since existing ray-based methods cause long computation times, a new parallel GPU-based ray-launching simulation method is presented. The algorithmic improvements allow a high utilization of the GPU computing power, which results in significantly faster simulations while achieving high accuracy. The validation of the simulation results against real-world measurements showed a high level of agreement.**

***Keywords*—VANET, Vehicular Communication, Radio Propagation Simulation, Ray launching, GPU computing**

I. Introduction

Vehicular Ad hoc Networks (VANETs) are envisioned to improve traffic safety and efficiency as well as driver comfort. A high variety of applications, such as cooperative driving and subsequently automated driving, are enabled through wireless ad hoc communication between the vehicles on the road. These applications, which often exhibit safety-critical features, need to be tested extensively before deployment in series production. Due to the costs and difficulties of conducting real-world experiments, simulation will be a key methodology for evaluating real-world implementations of VANET-based driving assistance systems in virtual test drives.

Accurate modeling of the vehicle-to-vehicle (V2V) communication channel, which is heavily influenced by traffic dynamics and road surroundings, is crucial to obtain valid simulation results when investigating upper layer protocols and applications [1]. Stochastic models are not suitable for the evaluation of safety applications with the impact of a single transmission being critical. Several radio-propagation models targeting large-scale VANET simulation have been developed [2], [3], which take into account location-specific link conditions. However, the simplifications made to reduce computational complexity render them unusable in virtual test drives. At the other end of the spectrum, ray tracing based methods provide an accurate model of the V2V communication channel at the cost of high computational effort. They are typically used for optimization of antenna systems [4]. The prohibitively long computation times for path calculations with high accuracy restrict their area of application to short simulation scenarios consisting of only very few vehicles.

In order to evaluate and validate real implementations of VANET applications in virtual test drives, the wireless communication channel needs to be modeled in high fidelity. This is particularly important for critical situations when Non-Line-Of-Sight (NLOS) conditions exist, which are caused by static, e.g. buildings and dynamic obstacles such as large vehicles [5]. However, simulation durations need to be kept reasonable to enable medium-scale scenarios, i.e. tens of vehicles.

Modern GPUs provide the facilities and computational power to perform fast ray-based methods [6]. In this paper we present radio propagation simulation based on ray launching which exploits the computational power of GPUs to achieve a high performance, deterministic model of the V2V communication channel for detailed and dynamic scenes.

The remainder of this paper is organized as follows: Section II gives an overview of the related work. In section III we describe our simulation model and the algorithmic improvements in detail. In section IV the ray-optical model is validated by means of measurements and compared against existing models. Section V concludes the paper and gives an outlook on future work.

II. Related Work

Before we give an overview of the related work, we distinguish between deterministic ray tracing and ray launching. Ray *tracing* is a generic term for propagating rays through a virtual scene. However, deterministic ray *tracing* is commonly referred to a deterministic calculation of propagation paths, whereas ray *launching* involves shooting many rays to sample the scene in a brute force manner [7].

Ray-based methods are well-suited for parallel computing architectures such as GPUs. Due to the advances in GPU technology, parallel ray tracing for image rendering has attracted a lot of research effort in the last years. Several optimizations, which enable fast rendering of highly detailed and dynamic scenes, have been developed such as efficient bounding volume hierarchies [8].

Maurer [9] was one of the first to apply ray tracing in order to model the V2V communication channel. Schmitz [10] used GPU based ray methods to speed up the simulation of radio wave propagation for static transmitter scenarios. Ray launching on GPUs was also employed in [11] to simulate radio propagation in indoor scenarios. It was shown that

978-1-4673-7813-0/15/$31.00 ©2015 IEEE

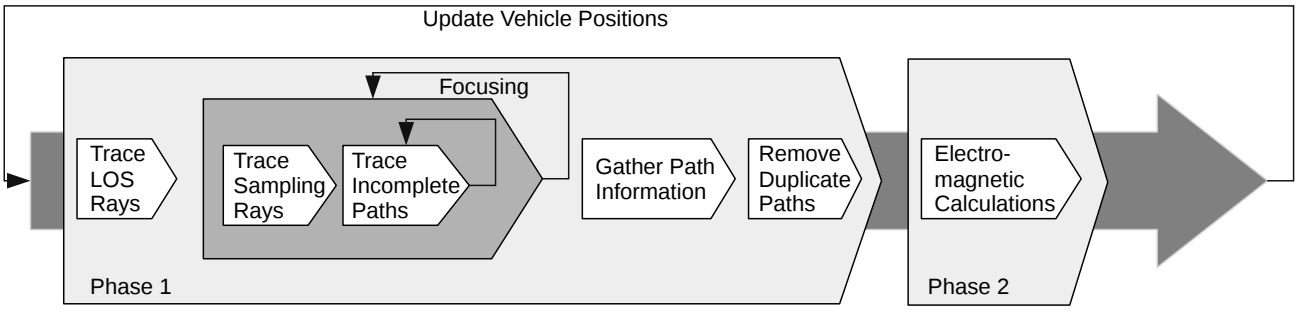

Fig. 1. Simulation pipeline comprising two main phases

exploiting the parallel computing power of GPUs results in substantial reduction in computing time. However, in [11] only static scenes without diffractions were considered which are challenging on GPUs due to their branching nature. In the following we show how wave propagation at diffractions can be handled efficiently on GPUs. Another domain in which ray-based methods are applied is the simulation of sound propagation [12]. GPU computation is there used to achieve interactive update rates.

III. GPU ACCELERATED ELECTROMAGNETIC WAVE PROPAGATION

As previously stated we want to perform highly detailed yet fast simulations of the V2V communication channel in order to conduct virtual test drives of VANET applications. In order to achieve this goal, we choose ray launching over deterministic ray tracing because it is better suited for fast computations of highly detailed and dynamic scenes with moving transmitters, receivers and obstacles. In the following we describe the concept and algorithmic improvements for the parallel scalability of our GPU based ray launching model.

A. Conceptual Overview

In order to reduce the overall simulation time, all steps of the propagation model are optimized for efficient parallel computations. In graphic rendering the shading of rays is usually done on the fly, i.e. colors are determined while the ray propagates through the scene. However, most of the rays that are shot in our simulation model never hit a receiver so computing their electromagnetic properties would waste computational resources. The model is therefore split into two separate phases as shown in figure 1. The first phase employs parallel ray launching to accelerate the visibility computations for the current scene configuration. The resulting propagation paths between transmitters and receivers are then passed on to the post processing phase, in which the material data of the environment as well as the antenna patterns are taken into account in order to compute the electromagnetic properties for each propagation path.

B. Ray Launching Phase

1) Coherent Ray Launching: As a first step, it is essential to identify the existence of Line-Of-Sight (LOS) paths because the LOS path dominates the received signal strength since it is only subjected to free space propagation loss. LOS rays are therefore shot through the scene to determine the existence of the LOS path between each sender-receiver pair. While the LOS paths can be evaluated quickly, multi-path propagation results in a considerably higher computational effort since a large number of rays need to be shot. The ray launching method is based on sending rays from each transmitter antenna to detect possible propagation paths between transmitter and receiver. In order to achieve accurate results, the sampling of the primary rays' directions must be uniform to avoid undersampling of specific parts of the scene. This can be achieved by using geodesic tessellated spheres [13], however this method limits the number of primary rays.

We therefore employ uniformly distributed random directions to allow an arbitrary amount of rays. Each primary query ray following a random direction can be processed in parallel, however, in order to gain high efficiency, the computation model of modern GPUs needs to be taken into account. The SIMT (single instruction multiple threads) execution model groups multiple threads in so called warps and exhibits the highest performance if all threads within a warp follow the same execution path [14]. When tracing rays in randomly distributed directions in parallel, this leads to a high thread divergence. Different intersection tests need to be performed when the rays encounter different scene objects and access incoherent memory locations. To avoid this efficiency problem we sort the randomly generated directions using a space-filling curve. We use the Z-order curve since it can be implemented and computed efficiently [15]. After the sorting process the initial directions can be traced coherently. Depending on the amount of initial directions, the sorting itself can be too time-consuming despite being performed in parallel as well. We therefore precompute the sorted random directions before the actual simulation takes place. In order to avoid aliasing artifacts, the uniformly sampled directions are rotated randomly for each transmitter in every time step.

2) Ray Interactions: In order to find the propagation paths we shoot rays from the center point of each transmitting antenna to sample the 3D space. As the rays propagate through the scene, they interact with the scene geometry and potentially intersect the receiver antennas, which are modeled as spheres. The size of these spheres determines how many rays are collected at each receiver. In analogy to the notation introduced by Heckbert [16] we generally define the interactions along the propagation paths between transmitter (T) and receiver (R) as either specular (S) or diffuse (D). We concentrate on reflections as specular and diffractions as diffuse interactions

in the following, however other ray interactions can be easily added into the model. Since we apply ray launching, a ray can miss (M) the scene geometry. Each valid path can therefore be characterized by the regular expression $T(S|D)^*R$. Figure 2 shows how the rays launched from the transmitter propagate through the scene to determine the visibility of surfaces and diffraction edges. So every ray launched from the transmitter queries the scene to evaluate if a given outgoing direction results in a hit of the receiver. This process creates a tree of possible propagation paths, which in the following is called the scene query tree.

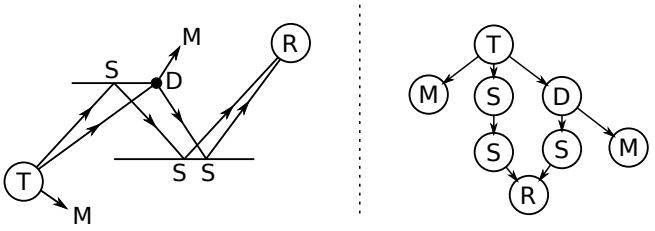

Fig. 2. Left: Rays propagate through the scene; Right: Resulting scene query tree

The surfaces of the 3D scene are modeled as triangular meshes with material properties attached to it, which determine for example if a specific triangle acts as a reflecting or refracting surface. When a ray hits a reflecting triangle, it is reflected on the surface and a new ray is traced in the direction of reflection. This process continues until the configured order of reflections is reached. Each specular reflection results in a single additional node entry in the scene query tree.

Since all meshes are assumed to only undergo rigid-body motion, edges which can cause a diffraction are detected in an off-line preprocessing step based on a pre-defined angle criterion. In the ray tracing phase, analogously to the reception sphere, the diffraction edges are modeled as cylinders centered around the edges with a configurable radius as shown on the right in figure 3. This allows to efficiently evaluate the diffractions at runtime by testing the intersection of a ray with the cylinder geometry. If a ray intersects a cylinder, it is considered to be close enough to the edge to cause a diffraction. The hit point on the cylinder surface is projected onto the diffracting edge. This point on the edge is then the new origin for the subsequent outgoing rays, which are generated according to the formulation of the Uniform Theory of Diffraction (UTD) [17]. The incident ray has the same angle relative to the edge as the outgoing rays, which leads to the new rays being sampled on a cone (left in figure 3).

Each diffraction leads to branching in several new paths in the scene query tree. This branching continues exponentially if multiple orders of diffractions are taken into account. In order to efficiently trace the outgoing ray paths of a diffraction in parallel, we terminate the tracing of this specific ray and save the path until the diffraction has occurred as an incomplete path. GPU memory must be used economically both to save memory bandwidth and to allow a high number of paths to be stored. Therefore we only save an implicit version of the path which just contains the information necessary to re-trace this specific path later on. Each path can be uniquely defined by its origin, the direction it is sent from the emitter and all following

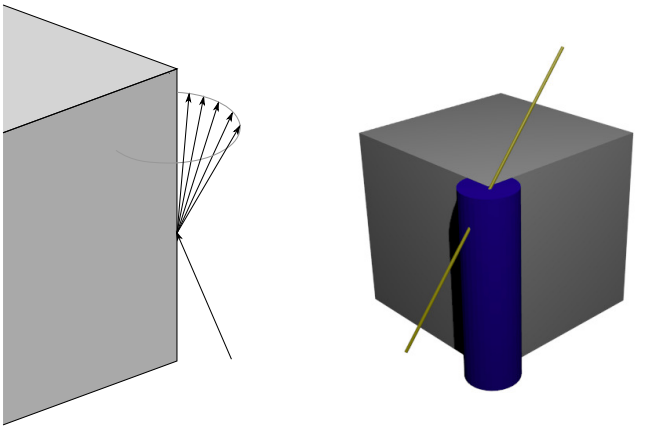

Fig. 3. Left: New rays are traced along the diffraction cone; Right: Diffraction edges are modeled as cylinders

directions after an interaction with a scene object has occurred. This method has the advantage that specular reflections, which are unambiguous, do not require any storage and therefore the amount of necessary memory is reduced drastically.

When all current ray tracing activity is finished, multiple new rays need to be traced from the previous stop location of each incomplete path. To improve coherency for the new rays, their origins are sorted by the Z-order curve before launching them. This method is also shown in [18] where the rays are additionally sorted by direction. However, as incomplete paths do not represent a single ray but a whole batch which in itself is already coherent, it is sufficient to only sort by the origin.

This process of tracing, terminating and restarting is repeated until there are no incomplete paths left. The scene query tree now contains the implicitly encoded propagation paths between emitters and receivers. In order to get the full information for each path, we perform a gathering step which consists of tracing one ray per path and following the propagation path through the scene while recording the information such as triangle IDs and material data. This detailed per-interaction data is then stored in the scene query tree.

The scene query tree might contain duplicate sequences of triangles and diffraction edges denoting the same propagation path. In order to efficiently eliminate duplicates, the ray sequences are first sorted in parallel by interaction type and primitive index. The duplicates are then removed from this reordered path list to ensure that only unique propagation paths are contained.

3) Focusing: The propagation paths between transmitter and receiver depend on the position of the transmitters, receivers as well as on the location of static and dynamic obstacles. A brute force solution to find these paths would sample the scene as fine as necessary to reach all receiver spheres on all relevant paths in order to avoid undersampling artifacts. As we target large scene dimensions (i.e. city scale), a very high number of rays is necessary for accurate results. Since most rays do never reach a receiver, a lot of computational resources would be wasted. We therefore apply a technique which we call “focusing”. Its main goal is to achieve a better utilization of computing resources while maintaining accuracy.

For reasons of simplicity we illustrate this technique based on an exemplary 2D street intersection in figure 4, however the same principle applies for 3D scenes.

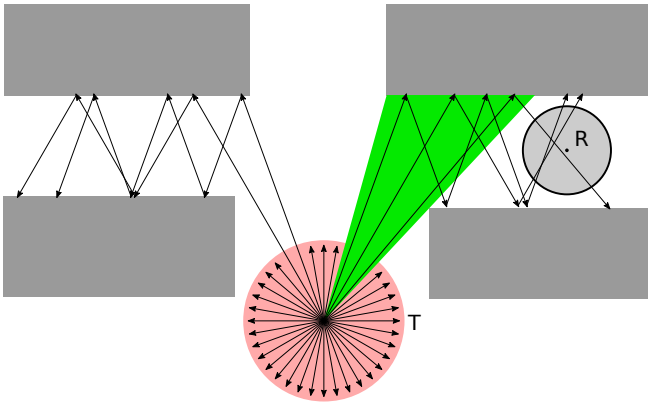

Fig. 4. Focusing applied for an exemplary street intersection

As a first step we quickly detect an approximate set of surfaces and diffraction edges which are directly or indirectly (through reflections and diffractions) visible to both receiver and transmitter. This is done by employing a large receiver sphere while coarsely sampling the scene with few rays. After this first coarse pass, the outgoing directions which lead to a hit are identified and are thus partitioned into sections of interest (green area in figure 4) and no-interest (red area in figure 4). Multiple rays are then traced into the sections of interest to sample this subset of the scene in more detail. The directions for the finer sampling are again taken from the precomputed uniformly random directions to achieve coherence between the rays which lie in each section of interest. Multiple focusing passes can be performed depending on the desired accuracy, the scene geometry as well as the computation time budget.

4) Exploiting Frame Coherence: If the movements of the vehicles and thus the changes of the scene configuration between succeeding time steps are relatively small, the results from the previous frame can also be used to avoid undersampling artifacts. After the determination of the propagation paths of a frame we store the outgoing directions of the successful paths which have hit a receiver sphere. When the position updates have been applied to the scene in the next time step, this direction information is then used in the coarse phase of the focusing step. By exploiting the similarity between small scene updates, paths which are still valid or have only slightly changed are then again retrieved in the next frame.

C. Electromagnetic Post Processing

All propagation paths that have been determined by the ray launching phase are passed on to the electromagnetic post processing phase. In this phase, the 3D electromagnetic wave propagation is calculated based on the principles of Geometrical Optics (GO) and UTD. The paths are first subjected to the full-polarimetric antenna gain pattern of the transmitter, which can be efficiently interpolated on GPUs [19]. Subsequently, the free space propagation and the effects of each recorded interaction are applied to the paths before the receiver antenna pattern is applied. These steps are performed in parallel, as all paths are independent from one another. Finally, a parallel reduction yields the superposition of the paths associated for every transmitter-receiver pair.

D. Implementation

All computations are performed on the GPU in order to avoid additional latency when copying intermediate results between CPU and GPU memory spaces. The ray launching phase is implemented using the NVIDIA OptiX framework, which provides the basic building blocks to implement ray tracing based algorithm on GPUs [20]. It features high performance acceleration structures with fast rebuilding between successive frames, which enables the simulation of highly dynamic scenes. We employ the sorting algorithms from the Thrust library [21], which offers state of the art parallel implementations for the CUDA programming model.

IV. Validation of the Proposed Model

We have shown the correctness of the electromagnetic calculations in [22] by comparing the simulation results of our GPU based ray launching model with the results of a full wave simulation. In order to validate our ray optical model against real world scenarios, we conducted an extensive set of measurements. In the following we describe an urban intersection scenario comprising both LOS and NLOS conditions and compare the results of our model with the recorded measurements.

A. Measurement Setup and Scenario

The measurements were recorded while driving through the city of Ingolstadt with two Audi A8L. Both cars were equipped with an Autotalks PANGAEA3[1], a development platform for vehicular communications which also features a GPS receiver, as well as a prototype antenna. The cars were configured to continuously send a 400 byte packet containing an increasing sequence number every $50\,\text{ms}$ with 6 Mbps. The transmit power was set to $10\,\text{dBm}$ while operating at $5.9\,\text{GHz}$. GPS positions are obtained with an update frequency of $5\,\text{Hz}$. The cars both logged received packets and signal strength as well as the GPS positions.

In the scenario for which we compare measurement and simulation, the two vehicles drove behind each other and took a right turn with a time-delay. This resulted in a LOS situation before the first car turned right, then in a NLOS situation before the second car turned right until finally LOS was reestablished. The intersection and the trajectory are shown in figure 5.

In order to reconstruct this scenario in our simulation, a 3D model of the environment is necessary. In [23] it has been shown theoretically that building data from OpenStreetMap[2] (OSM) is sufficiently precise for the use of channel modeling. Hence we converted the 2.5D data from OSM based on the guideline presented in [23] into a 3D model. This model was then enriched with the material data for the permittivity ϵ_r and the roughness δ as listed in table I. The diffraction edges were pre-computed off-line as previously described. Figure 6 shows the 3D model as well as the determined propagation paths for

[1] www.auto-talks.com/category/pangaea/

[2] www.openstreetmap.org

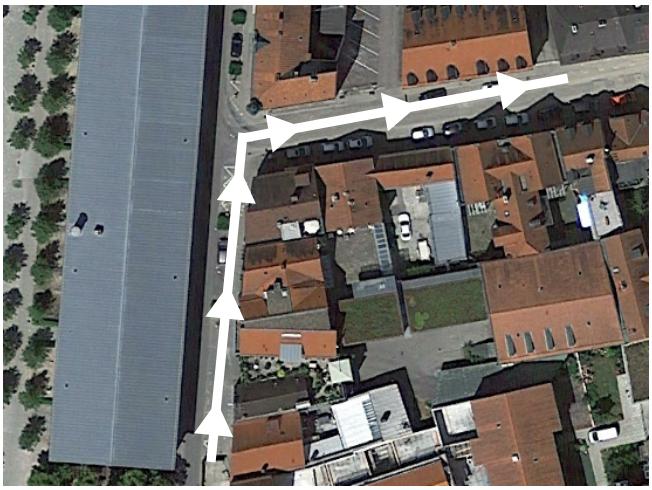

Fig. 5. Intersection at N48.766328°, E11.425775° with overlayed driving direction; Image courtesy of Google Earth

a specific time step. Diffracted rays are depicted in magenta, reflected rays in yellow and rays which hit the receiver turn green at the end.

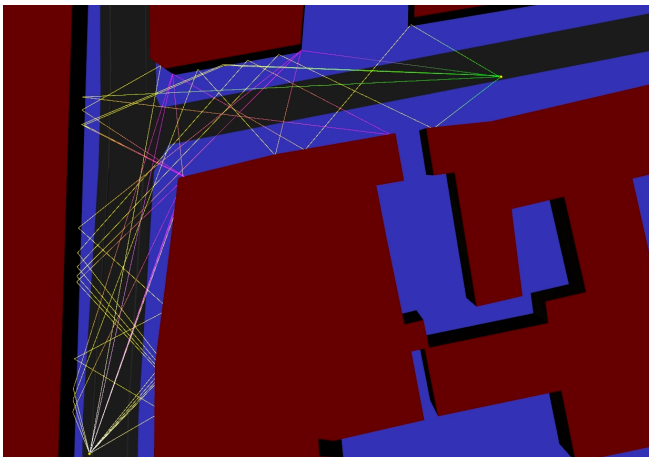

Fig. 6. Propagation paths in the 3D model

TABLE I. MATERIAL PARAMETERS USED IN VALIDATION SCENARIO

Material	ϵ_r	$\delta[m]$
Asphalt [24]	$4.54 - j0.36$	$0.4 \cdot 10^{-3}$
Concrete [9]	$5 - j0.1$	$0.4 \cdot 10^{-3}$

B. Evaluation

We simulated the above described scenario for a time step width of 100 ms. The duration of the whole scenario is 30 s which results in $N = 300$ discrete configurations. The maximum order of reflections was set to 4 and the order of diffractions was set to 1 for this scenario. The machine on which the simulation was performed is equipped with a NVIDIA Geforce GTX 680 GPU and a 3.6 GHz Intel Xeon CPU. CUDA version 6.5 and OptiX version 3.7 were used. For this configuration one time step can be computed on average in 103 ms.

The measurements recorded by both cars (averaged over 5 time steps) and the simulated received power are shown in figure 7. It can be seen that there is a good agreement between simulation and measurement for both the LOS and NLOS sections of the scenario as well as the transitions between these sections.

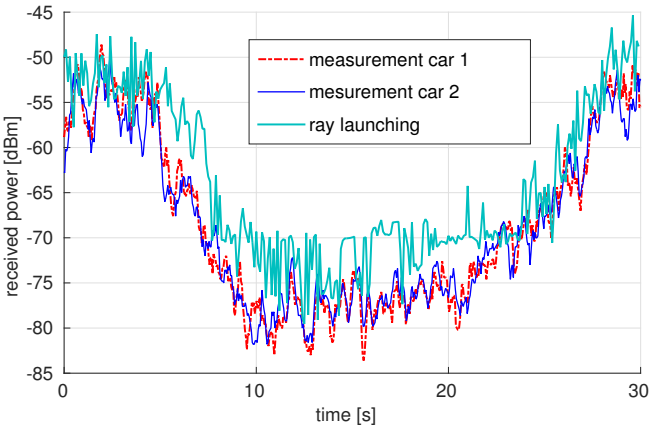

Fig. 7. Comparison of received power for measurements and ray launching simulation

The deviations between simulation and measurement can be quantified using the mean error

$$\mu = \frac{1}{N} \sum_{i=1}^{N} e_i \tag{1}$$

and the standard deviation

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} |\mu - e_i|^2} \tag{2}$$

based on the per time step error

$$e_i = p_{meas,i} - p_{sim,i} \tag{3}$$

where $p_{meas,i}$ is the measured and $p_{sim,i}$ the simulated received power for each time step

When comparing against the raw measurements recorded for car 1, this results in the mean error $\mu = -4.56$ dB and the standard deviation $\sigma = 3.73$ dB, which indicates that our model achieves accurate results with a slight offset.

We also simulated the same scenario using two other models which are less computationally intensive. Virtual-Source11p [2] is an empirical, low-complexity path-loss model specifically designed to predict NLOS reception at urban intersections. It consists of only a single formula taking into account the receiver's and transmitter's distance to the intersection center. It assumes that the intersection geometry is perpendicular, which is not fully met in our validation scenario. Since only one analytical formula has to be evaluated, one time step takes less than 0.5 ms in our MATLAB implementation. GEMV^2 [25] on the other hand uses a geography database to achieve a location-specific V2V channel model in LOS and NLOS conditions. It considers both static and dynamic obstacles and is based on a very simplified ray tracing method, which tries to capture only the most significant contributions.

Using its MATLAB implementation[3] one time step of the scenario can be computed on average in 12 ms.

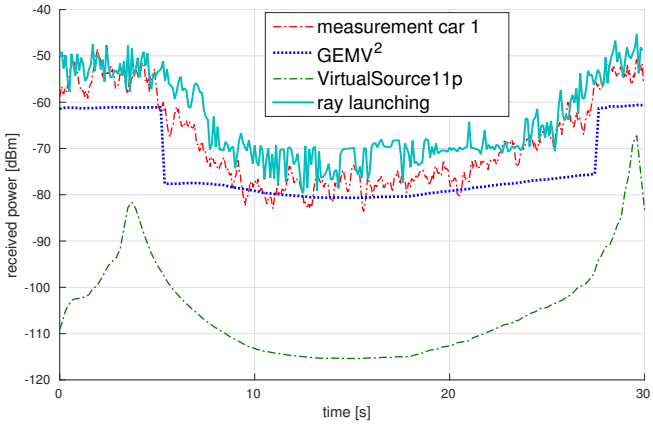

Fig. 8. Comparison of received power between measurement, GEMV^2 and VirtualSource11p

Figure 8 shows the comparison of the obtained measurement for one car as well as the results from the other two described models. Since the VirtualSource11p model is only valid for NLOS, the values for the LOS section of this scenario are obviously incorrect. The shape of the resulting curve follows the measurements with a great offset which suggests that the model is not calibrated well enough for this specific intersection. The street width in this scenario is approximately 10 m, which is significantly less than the narrowest street (18 m) used for calibrating the model [2]. A similar issue has also been reported in [26].

The results from the GEMV^2 model show a better match with the measurements. However, the transitions between LOS and NLOS are very sharp, which stems from the fact that the model explicitly distinguishes between these conditions and does not model the transitions specifically. While this model provides a very good accuracy to computational effort ratio, which makes it applicable in large-scale scenarios, it can be too imprecise when investigating such intersections in detail.

V. Conclusion and Future Work

The detailed modeling of the V2V communication channel based on ray-optical models requires a high computational effort, which usually leads to prohibitively long simulation durations. In this paper, we presented a ray launching based simulation model which exploits the parallel computational power of modern GPU hardware to achieve fast simulations. We showed algorithmic improvements to efficiently determine propagation paths in parallel so that a high utilization of the computing power of GPUs is achieved. This results in fast simulations, which enables the employment of this highly detailed propagation model in the envisioned use case of virtually test driving VANET applications. The validation of the proposed model exhibited a high level of agreement with real world measurements for both LOS and NLOS conditions. Finally, we compared the results against existing low-complexity models and pointed out that the precise calibration of empirical models is crucial as well as that these models fail to accurately predict transitions between LOS and NLOS conditions.

We are further optimizing the calculation of diffractions since they have the highest impact on computation times by employing bi-directional ray tracing and precomputing edge visibility as proposed in [27]. Additionally, we are investigating how the focusing approach can be coupled with the guidance algorithm presented in [12].

[3] available at vehicle2x.net

References

[1] F. Martinez, M. Fogue, M. Coll, J.-C. Cano, C. Calafate, and P. Manzoni, "Assessing the Impact of a Realistic Radio Propagation Model on VANET Scenarios Using Real Maps," in *9th IEEE International Symposium on Network Computing and Applications (NCA)*, July 2010, pp. 132–139.

[2] T. Mangel, O. Klemp, and H. Hartenstein, "A validated 5.9 GHz Non-Line-of-Sight path-loss and fading model for inter-vehicle communication," in *11th International Conference on ITS Telecommunications (ITST)*, August 2011, pp. 75–80.

[3] C. Sommer, D. Eckhoff, R. German, and F. Dressler, "A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments," in *8th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2011)*. Bardonecchia, Italy: IEEE, January 2011, pp. 84–90.

[4] L. Reichardt, J. Maurer, T. Fugen, and T. Zwick, "Virtual Drive: A Complete V2X Communication and Radar System Simulator for Optimization of Multiple Antenna Systems," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1295–1310, July 2011.

[5] M. Boban, T. Vinhoza, M. Ferreira, J. Barros, and O. Tonguz, "Impact of Vehicles as Obstacles in Vehicular Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, January 2011.

[6] T. Aila and S. Laine, "Understanding the Efficiency of Ray Traversal on GPUs," in *Proceedings of the Conference on High Performance Graphics 2009*. New York, NY, USA: ACM, 2009, pp. 145–149.

[7] H. Azodi, U. Siart, and T. F. Eibert, "A Fast 3-D Deterministic Ray Tracing Coverage Simulator Including Creeping Rays Based On Geometry Voxelization Technique," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 1, January 2015.

[8] T. Karras and T. Aila, "Fast Parallel Construction of High-quality Bounding Volume Hierarchies," in *Proceedings of the 5th High-Performance Graphics Conference*. New York, USA: ACM, 2013, pp. 89–99.

[9] J. Maurer, T. Fügen, T. Schafer, and W. Wiesbeck, "A new inter-vehicle communications (IVC) channel model," in *IEEE 60th Vehicular Technology Conference, (VTC2004-Fall)*, vol. 1, September 2004, pp. 9–13 Vol. 1.

[10] A. Schmitz and L. Kobbelt, "Efficient and accurate urban outdoor radio wave propagation," in *International Conference on Electromagnetics in Advanced Applications (ICEAA)*, September 2011, pp. 323–326.

[11] R. Felbecker, L. Raschkowski, W. Keusgen, and M. Peter, "Electromagnetic wave propagation in the millimeter wave band using the NVIDIA OptiX GPU ray tracing engine," in *6th European Conference on Antennas and Propagation (EUCAP)*, March 2012, pp. 488–492.

[12] M. Taylor, A. Chandak, Q. Mo, C. Lauterbach, C. Schissler, and D. Manocha, "Guided multiview ray tracing for fast auralization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 11, pp. 1797–1810, 2012.

[13] G. Durgin, N. Patwari, and T. Rappaport, "An advanced 3d ray launching method for wireless propagation prediction," in *IEEE 47th Vehicular Technology Conference*, vol. 2, May 1997, pp. 785–789 vol.2.

[14] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: A unified graphics and computing architecture," *IEEE Micro*, vol. 28, no. 2, pp. 39–55, 2008.

[15] B. Moon, Y. Byun, T.-J. Kim, P. Claudio, H.-S. Kim, Y.-J. Ban, S. W. Nam, and S.-E. Yoon, "Cache-oblivious ray reordering," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 3, pp. 1–10, 2010.

[16] P. S. Heckbert, "Adaptive Radiosity Textures for Bidirectional Ray Tracing," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '90. New York, USA: ACM, 1990, pp. 145–154.

[17] R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," *Proceedings of the IEEE*, vol. 62, no. 11, November 1974.

[18] J.-H. Nah, Y.-H. Jung, W.-C. Park, and T.-D. Han, "Efficient ray sorting for the tracing of incoherent rays," *IEICE Electronics Express*, vol. 9, no. 9, pp. 849–854, 2012.

[19] D. Ruijters, B. M. ter Haar Romeny, and P. Suetens, "Efficient GPU-based texture interpolation using uniform B-splines," *Journal of Graphics, GPU, and Game Tools*, vol. 13, no. 4, pp. 61–69, 2008.

[20] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison *et al.*, "OptiX: a general purpose ray tracing engine," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 66, 2010.

[21] N. Bell and J. Hoberock, "Thrust: a productivity-oriented library for CUDA," in *GPU Computing Gems*. Morgan Kaufmann, 2011.

[22] M. S. L. Mocker, M. Schiller, R. Brem, Z. Sun, H. Tazi, T. F. Eibert, and A. Knoll, "Combination of Full Wave Methods and Ray Tracing for Radiation Pattern Simulations of Antennas on Vehicle Roofs," in *9th European Conference on Antennas and Propagation (EUCAP)*, 2015.

[23] J. Nuckelt, D. Rose, T. Jansen, and T. Kurner, "On the use of OpenStreetMap data for V2X channel modeling in urban scenarios," in *7th European Conference on Antennas and Propagation (EuCAP)*, April 2013, pp. 3984–3988.

[24] E. S. Li and K. Sarabandi, "Low grazing incidence millimeter-wave scattering models and measurements for various road surfaces," *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 5, May 1999.

[25] M. Boban, J. Barros, and O. Tonguz, "Geometry-Based Vehicle-to-Vehicle Channel Modeling for Large-Scale Simulation," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4146–4164, Nov. 2014.

[26] T. Abbas, A. Thiel, T. Zemen, C. F. Mecklenbräuker, and F. Tufvesson, "Validation of a Non-Line-of-Sight Path-Loss Model for V2V Communications at Street Intersections," in *13th International Conference on ITS Telecommunications*, 2013, p. 5.

[27] C. Schissler, R. Mehra, and D. Manocha, "High-order diffraction and diffuse reflections for interactive sound propagation in large environments," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 39:1–39:12, July 2014.