Interactive Tracing of Radio Waves and Neuronal Fiber Pathways for Exploratory Visualization in Virtual Reality

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker Tobias Rick
aus Neuss

Berichter: apl. Prof. Dr.rer.nat. Torsten Kuhlen
Univ.-Prof. Dr.rer.nat. Leif Kobbelt
Univ.-Prof. Dr.med. Katrin Amunts

Tag der mündlichen Prüfung: 09.05.2012

Abstract

Exploratory visualization of physical simulations in virtual environments greatly benefits from interactively changing parameters with real-time feedback. We address two specific research questions by problem reformulation and implementation on the *Graphics Processing Unit* (GPU)'s many-core architecture towards a simulation and interaction in real-time: (1) simulating and manipulating wireless radio networks and (2) estimating and disambiguating neuronal fiber connectivity in the living human brain. We assist in the exploratory scientific analysis by developing two distinct applications for each domain area where we coupled simulation input and output directly with visual feedback and natural interaction via an interactive *Virtual Reality* (VR) interface.

Radio Wave Propagation Knowledge of the propagation behavior of radio waves is a fundamental prerequisite for planning and optimizing mobile radio networks. Propagation effects are usually simulated numerically, since real-world measurement campaigns are time-consuming and expensive. The computation should be fast, in order to provide a large number of simulations to select the best candidates from, and accurate, such that the simulation reflects the actual propagation behavior.

Diffraction along edges is a predominant effect for common mobile radio frequencies and is usually modeled by shooting a multitude of rays into the shadow cone of the diffracting edge which results in a large computational overhead. By exploiting the parallel compute capabilities of GPUs we utilize the concept of shadow volumes to directly render whole diffraction cones. Our approach transforms the problem of finding diffraction rays to repeated shadow computations, which can be done extremely fast on recent GPUs. Moreover, we achieved huge speedups by culling away propagation paths in the early stage of the computation that will not contribute to the signal reception. We developed distinct algorithms for specific propagation phenomena such as wall transmission,

diffraction over rooftops and into street canyons in order to optimize each ray path computation individually.

Propagation predictions provide the search space for automatic planning algorithms that explore a vast amount of network configurations to find good deployment schemes. However, complex urban scenarios demand for a great emphasis on site-specific details in the propagation environment which are often not covered in automatic approaches. Therefore, we combined the simulation of radio waves with an interactive exploration and modification of the propagation environment within a *Virtual Environment* (VE). The user is put in direct control over transmitter site locations and network size. In particular, we let the user manipulate the underlying building database within the VE. New buildings, that can be constructed by sketching their floor plan on the ground, are instantly integrated into the simulation. This effectively enables a dynamic planning process and demonstrates the effect of a specific entity in the environment on the overall propagation phenomena and network characteristics, such as coverage and interference.

Fiber Pathway Estimation Understanding the connectivity structure of the human brain is a fundamental prerequisite for the treatment of psychiatric or neurological diseases. Probabilistic tractography is a promising technique to account for the inherent uncertainties in *Magnetic Resonance Imaging* (MRI) data by estimating the likelihood that a fiber bundle takes its course through a particular voxel. We address two major aspects of probabilistic tractography – computation and visualization – by presenting (1) an effective algorithm that is able to compute probabilistic fiber tracts interactively and (2) an embedded real-time exploratory visualization of the results.

We achieve a real-time fiber estimation by a parallel implementation on the GPU. In addition to the basic probabilistic streamline integration, we also demonstrate how state-of-the-art extensions like multi-fiber orientation and loop checking can be adapted to benefit from the GPU's many-core architecture. Our visualization approach focuses on the assessment of fiber probabilities in relation to their structural context. In particular, we employ a semi-transparent direct volume rendering technique to display the course of the fiber in relation to its confidence. A VR interface lets the user take an active role in the exploration process: we use a magic lens technique to support the understanding of the structure-function relationship and to disambiguate between data modalities, such as MR, PET, fiber tracts or brain areas. Moreover, exploratory aspects are supported by a direct coupling between the computation and the visualization of fiber tracts. Seed regions for probabilistic fiber tracts can either be selected based on a subject-specific brain atlas, or chosen freely with a *Six Degrees of Freedom* (6DOF) device. The combination of these components provide the neuroscientist with an interface to explore brain connectivity that is very similar to the popular dye injection paradigm in a virtual wind tunnel.

List of Figures

List of Tables

Contents

CHAPTER 1

INTRODUCTION

1.1. Motivation

This chapter motivates the general thesis background which revolves around the real-time manipulation and visualization in VR for two specific application areas: simulating radio waves in urban environments and estimating fiber connections in the human brain. We approach this by effectively utilizing the computational power of the GPU's many-core architecture. We will therefore introduce two programming paradigms for using graphics hardware as a parallel computing platform: *General Purpose Computation on Graphics Processing Units* (GPGPU) for radio wave propagation as we will make heavy use of the GPU's rasterization engine, and *Compute Unified Device Architecture* (CUDA) for inferring brain connectivity using the latest hardware features for numerical stochastic integration.

Additionally, we will briefly introduce immersive scientific visualization in VR for exploring and analyzing complex data with an innovative and natural human-machine interface. After the general motivation, we will provide an overview of the thesis' contributions followed up with a list of the author's publications and close the chapter with an outline of this thesis. We refer to the individual chapters for a domain related motivation of the corresponding topics.

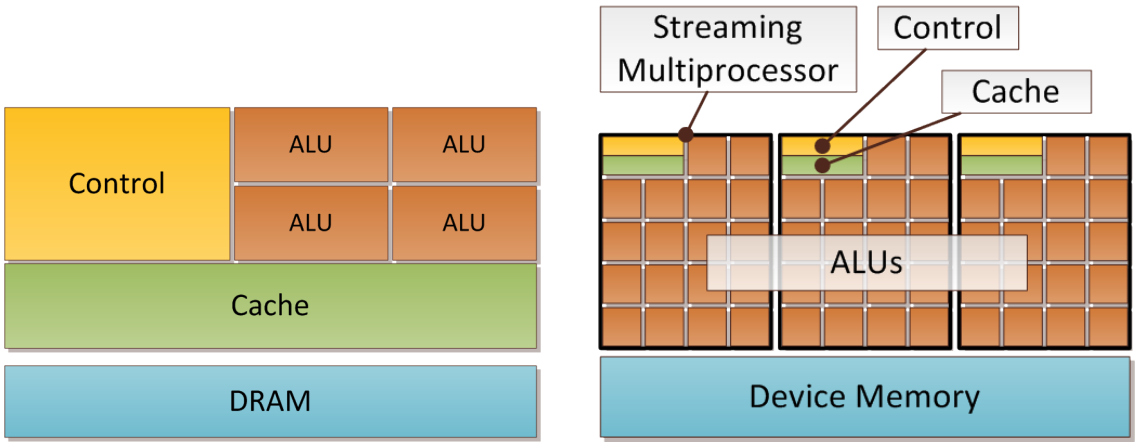


Figure 1.1.: A schematic illustration of modern CPU (left) and GPU (right) architectures [103, 105]. In contrast to the CPU which has dedicated support for data caching and flow control, the GPU is especially designed for computations with high arithmetic intensity.

1.1.1. Programmable Graphics Hardware

Until 1999 graphics cards had a non-programmable fixed-function architecture. Over the last decade they evolved to configurable pipelines and recently into fully programmable floating-point GPUs. Modern GPUs are extremely powerful computing devices. The number of *floating point operations per second* (FLOPS) of the GPU has increased dramatically in comparison to the CPU over the last few years, cf. [109]. At the time of writing (2011), a NVIDIA GPU (GeForce 580) with 512 CUDA cores achieves roughly 1,500 Giga FLOPS (GFLOPS). The performance of an Intel CPU (i7 980 XE) with six cores is about 107 GFLOPS [71]. For reference, in 2007 a NVIDIA GPU (GeForce G80) achieved approximately 300 GFLOPS whereas a Quad-Core Intel Xeon processor (3GHz) of this time had roughly about 80 GFLOPS.

The reason for this is that the GPU is specialized for computational intensive, highly parallel calculations. Rather than caring for data caching and flow control as the CPU, the GPU is especially designed to support data processing, cf. Figure 1.1. The GPU architecture follows the *Single Instruction Multiple Data* (SIMD) paradigm. Many processors simultaneously execute the same instructions on different parts of a data stream. We will now introduce two programming paradigms for programming graphics hardware: classical GPGPU and the recently introduced *Compute Unified Device Architecture* (CUDA).

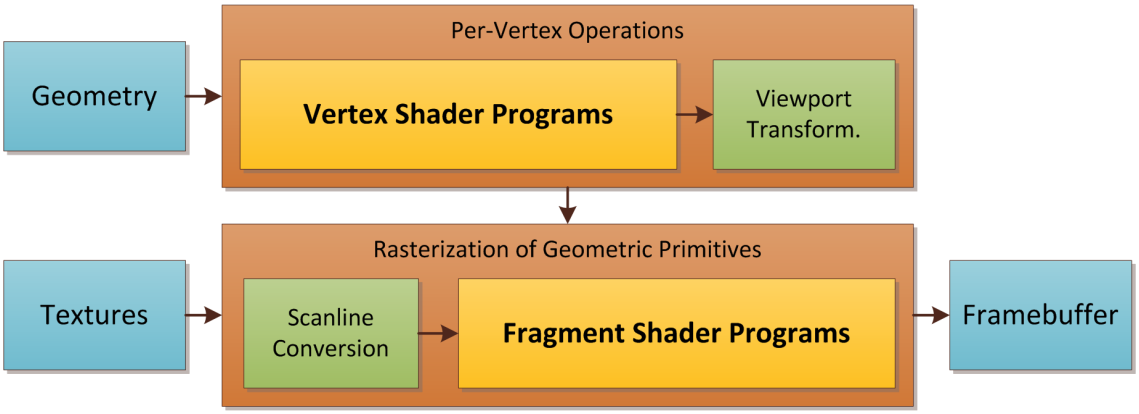


Figure 1.2.: The graphics rendering pipeline. Geometric primitives are loaded into the vertex buffer and transformed by multiple vertex processors in parallel. The rasterization samples the geometry into pixel positions such that multiple fragment processors can assign colors that are recorded in the frame buffer for the final image.

1.1.1.1. General Purpose Computations

Before the introduction of CUDA, the key challenge of programming graphics hardware was to correctly map problems to the graphical rendering context. Input data had to be transformed into images or geometry and algorithms had to be turned into image synthesis. A computation on the GPU consists of a pass through the various stages of the graphics rendering pipeline, see Figure 1.2. We refer to this pass as a rendering pass. Basic geometrical primitives like points or triangles are loaded into the vertex buffer. The primitives are described by their location in space, i.e., coordinates (vertices) and associated attributes like material or color. Additionally, data arrays (textures) of integer or floating point values can be allocated directly in graphics memory. First, multiple vertex processors execute in parallel the instructions from a user-written program (vertex program). Vertex programs operate on single vertices with access to their attributes and global read-only texture memory. Typically, geometric transformations like translation, rotation and projection are applied. In the subsequent step, the transformed geometry is sampled (rasterized) into discrete points (fragments). Each fragment corresponds to a single pixel (picture element) position on the screen, and typically has additional information like color and a depth value, i.e., the distance between viewer and the object which originally corresponded to the pixel. The fragment processors operate analogous to the vertex processors. A user-written fragment program is executed on each fragment in a parallel fashion. The fragment program is executed once per fragment. Most instructions are floating-point vector operations. Typically, lighting models are evaluated and corresponding fragment colors are assigned (shaded).

The frame buffer is a two-dimensional array of pixels. The task of the frame buffer is to assemble the final result of the GPU computation by collecting and recording all

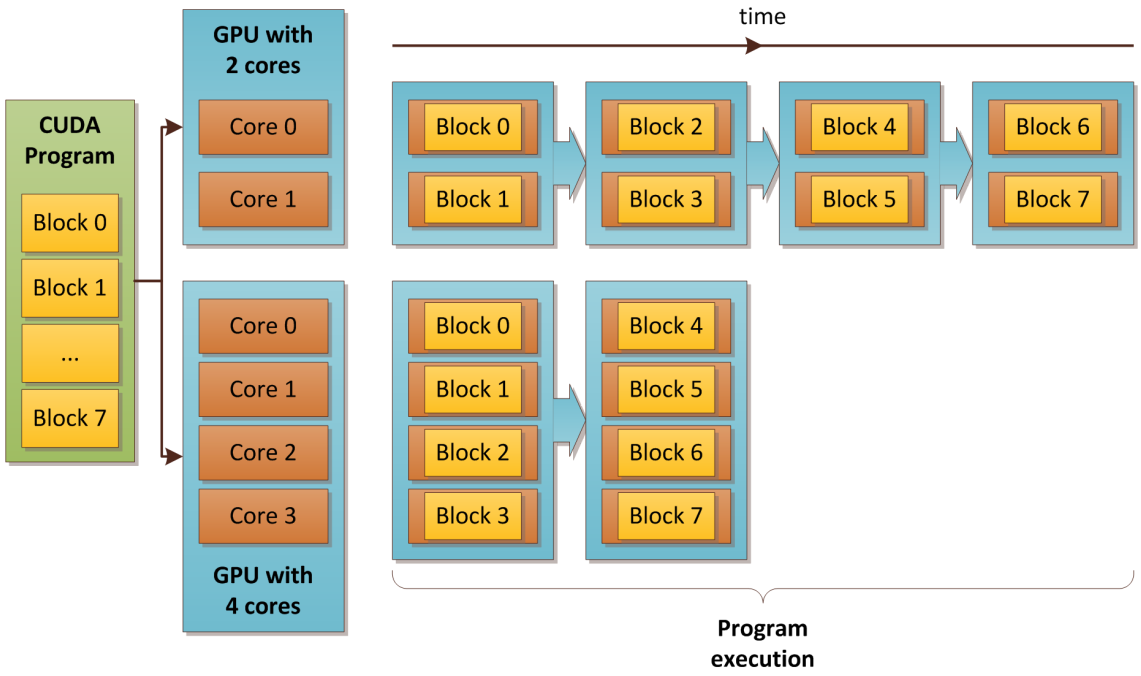


Figure 1.3.: If a problem can be partitioned into blocks of sub-problems that can be calculated independently from each other, the processing time scales with the number of available processor cores.

fragments. Frame buffer operations decide how the color from the incoming fragments is combined with the color already stored at the same pixel position. Thus, many fragments can contribute to the final color of a pixel. Commonly, the frame buffer contains color information and hence, it can be displayed on the screen. Information in the frame buffer can also be read back to host memory, which is especially useful for retrieving results of GPU computations. For further readings we point the interested reader to introductory texts and advanced techniques on graphics and shader programming in [1, 137] or [53]. A more general overview of general purpose computations on GPUs and on the evolution of GPU architectures is given for instance by Owens et al. in [109] and [108].

1.1.1.2. Compute Unified Device Architecture

In order to make the GPU computing platforms more accessible, graphics card manufacturers have developed special application programming interfaces (APIs), that can harness the GPU's resources without using the rendering pipeline. Mentionable is the OpenCL API by the Khronos Group [139] and NVIDIA's CUDA API [84, 105] which is also used in the present thesis. In contrast to classical GPGPU, OpenCL and CUDA offer a C like programming interface with random access to the device memory and full support for integer and bit wise operations.

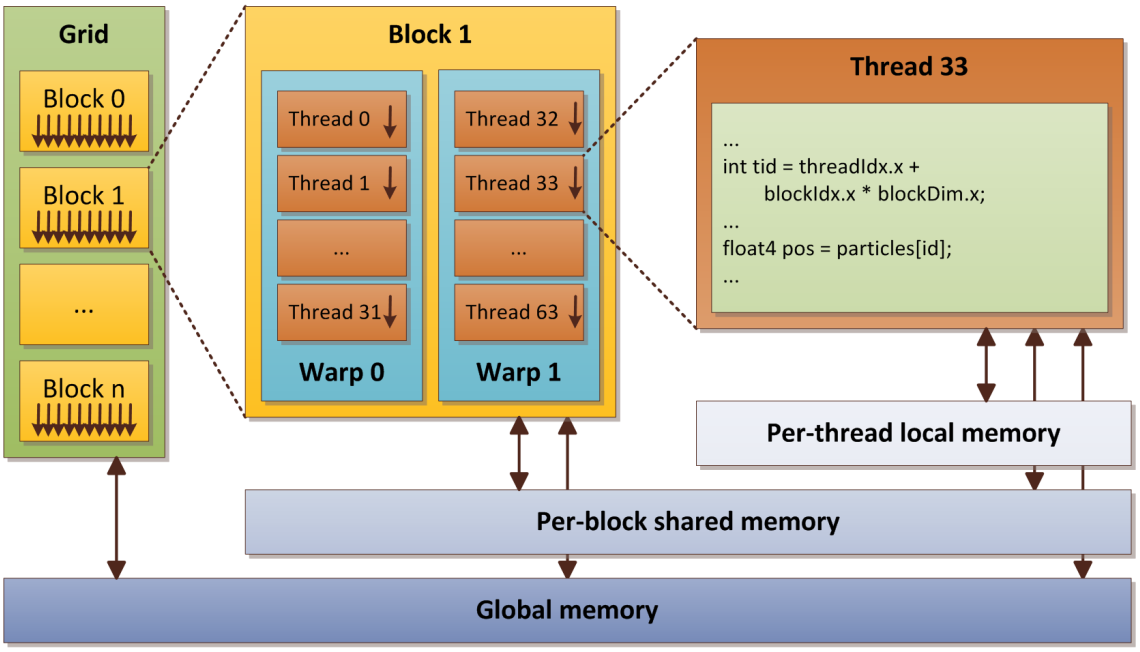


Figure 1.4.: The CUDA programming model organizes the threads in blocks and the blocks as grid. Each thread has a unique id that can be used to identify its part of the computation.

GPU Architecture The unified compute APIs also introduce a new terminology with respect to the GPU architecture. Instead of vertex and fragment shader units the modern GPU consists of several streaming multiprocessors (SMs), each of which can run hundreds of computational threads concurrently. In order to map threads to the SMs, the threads are organized in thread blocks. All threads of a block are processed by the same multiprocessor. Thus, computations must be partitioned into blocks that can be solved independently from one another, cf. Figure 1.3. Within each block, computations are again subdivided and assigned to individual threads of a block. Computations scale with an increasing number of available SMs as long as thread blocks can be processed independently from other blocks. In order to reach a good device occupation, CUDA based programs should therefore always assure that there are enough blocks to process.

CUDA Programming Model The CUDA programming model revolves around compute kernels which can be considered as the body of *for each* loops. Each kernel invocation requires a definition of the underlying block structure and eventually triggers a kernel execution on each thread. Figure 1.4 illustrates the organization of thread blocks that are themselves organized in a one- or two-dimensional grid structure, by which each block and thread gets its own index. This allows the execution of more threads at once than would fit into a single block. Within the kernel program, each thread can identify its part of the computation by accessing the indices of its block. In addition to its own

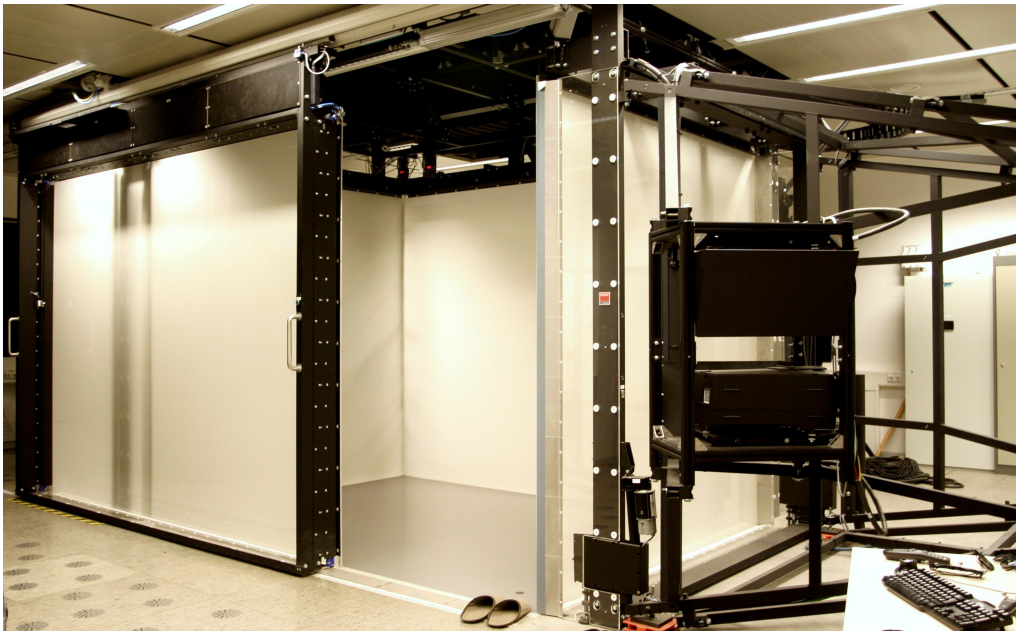


Figure 1.5.: The CAVE Virtual Environment with a sliding door at the Virtual Reality Group of RWTH Aachen University, Germany.

local memory, each thread can also access and share data with threads in the same block. The global memory is used to share data between all blocks of the grid and is persistent across launches of different kernels. The local- and shared memory banks are very limited in size, but they provide much lower access latency than the global memory which in contrast offers a much higher storage capacity at the cost of increased latency. The only cached memory is the device's texture memory unit which is read-only during a kernel execution. The global memory of the GPU is also accessible from the host system in order to copy data to and from the graphics device.

1.1.2. Immersive Visualization in Virtual Reality

We refer to *Virtual Reality* (VR) as "a computer generated environment that the user can interact with in real-time and experience with his natural senses". In VR the *immersion* of the user is typically realized by stereo displays with tracking hardware. The user tracking allows for an user centric projection instead of a classical desktop centered projection which increases spatial perception. State-of-the-art devices include head mounted displays and multi-wall display environments like a *Cave Automatic Virtual Environment* (CAVE), cf. Figure 1.5. *Interactivity* is enabled by spatial input devices with 6DOF that let the user experience and interact with the environment in a natural, direct way. Typical devices are for instance data gloves or tracked controllers like the A.R.T. Flystick (Figure 1.6). VR interfaces require a human-in-the-loop. This

Figure 1.6.: A controller with reflective (passive) markers for 6DOF interaction on the left and corresponding infrared optical tracking cameras on the right.

can be for instance a domain expert who uses immersive visualization techniques to search for patterns or anomalies in his data. This is also referred to as *imagination* and concludes the three I's of VR: immersion, interactivity and imagination.

We will now briefly introduce some of the general challenges of VR that we believe are of interest to the present work from a broad, big picture's point of view [145]. Real-time interaction in VR is much more challenging than in desktop applications for several reasons: (1) screen sizes tend to be very large and hence often have a very high resolution which in turn increases rendering load, (2) stereo rendering usually doubles the rendering requirements, and (3) the user is constantly in motion, frequent scene updates are crucial for head tracking. Low frame rates or lags tend to lead to motion sickness in a user centric projection which is an extremely undesired effect in VR. In this sense, maximizing frame rates can be considered in inherent challenge of every VR application.

Moreover, VR is still in the process of maturing and little consensus and accepted standards have emerged in terms of general methodology and hardware or software platforms. In particular, this thesis addresses certain issues of existing rendering and interaction techniques that arise when scaling and combining these techniques from their proof-of-concept prototypes to a real-world application.

Closely related are also the research challenges in scientific visualization as formulated by Johnson [72]. Four of these challenges are of special interest here and we summarize the take-home messages: work with application scientists to establish a mutual beneficial collaboration in "Think about the science", make visualizations interactive by leveraging available computing power in "Efficiently utilizing novel hardware architectures", interaction should evolve at least at the same pace as computing and visualization to make information easily accessible in " Human-computer interaction", and push visualization from a post processing step towards an integral part within the scientific computing pipeline in "Integrated problem-solving environments".

All of these challenges point towards exploratory scientific visualization, as a means to understand theoretical models and data, and VR as a potential tool to enhance the scientist's ability to access, manipulate and analyze his data to eventually develop, reject or accept hypotheses. Or, in other words, to help answer "what if?" questions. A lot of work has been published towards this end. Prominent examples of immersive visualization in VR are the pioneering work of Bryson et al. on the virtual wind tunnel in [31, 30], the visualization of flow structures through an artery by Forsberg et al. [56], or more recently the blood damage visualization in a ventricular assist device by Hentschel et al. [66].

1.2. Overview

1.2.1. Contributions

The main contributions of the present thesis can be summarized as follows (see Section 4.2 for a more detailed discussion):

- Although the concepts of shadow volumes are well studied in the field of computer graphics, to the best of our knowledge this is their first use in the context of radio wave simulation.

- Literature has discussed the principle of diffraction in great detail, however, its computation has remained computationally expensive. A major contribution is therefore an efficient formulation and implementation of the corresponding algorithms to compute diffraction ray paths in heterogeneous urban environments. Our approach has led to the first published real-time simulation of radio waves at a comparable prediction accuracy.

- We present mathematical models that describe how to derive the received signal strength from the computed ray paths and estimate unknown influences based on real-world measurements. In particular, we propose a closed form analytic solution for multi-path optimization with an arbitrary but fixed number of deflections and arriving paths. Former approaches to this had been limited either to single path models or needed an iterative solver that lacks a proof of convergence.

- We developed an VR interface that integrates state-of-the-art interaction components directly with a real-time simulation of radio waves. This itself may represent no original research per se, but presents a none the less significant contribution to the overall work showing how individual components work in combination to one another.

- The contribution of this thesis towards the estimation of neural fiber pathways is not a reinvention of the method itself but a massively parallel implementation on special purpose compute devices (GPUs) based on the original mathematical model of probabilistic tractography. It is therefore a predominantly technical contribution from a computer science's perspective but with a significant impact towards its application in the domain of neuroscience.

- A methodical contribution in probabilistic tractography algorithms has been made by extending the original algorithm by a probabilistic loop checking method that allows fiber tracts to evolve over longer distances.

- We combined and adapted state-of-the-art direct volume rendering and magic lens interaction to integrate and disambiguate between different neuroscientific data modalities. The contribution lies therefore not in a single aspect itself but in a combination that provides the basic framework for a subsequent interpretation and exploration of probabilistic tractography.

- We integrated each single aspect – computation, rendering and data interaction – into a VR interface. Again, this contribution is predominantly of technical nature, nevertheless non-trivial, showing how all individual contributions work together.

1.2.2. Publications

We divided the list of publications into three parts: Papers related to radio wave propagation, publications related to probabilistic tractography and additional publications which are indirectly related to this thesis.

1.2.2.1. Radio Wave Propagation

The author's publications on radio wave propagation discuss how real-time performance can be achieved for radio wave propagation predictions [37, 119, 120, 121, 124] and how the user can be provided with efficient interaction techniques for an exploratory analysis of the complex propagation phenomena [122, 123]. The computation of multi-path effects is not subject of this thesis, however, the work is published in [131, 132, 134].

Accelerating Radio Wave Propagation Predictions by Implementation on Graphics Hardware [37]. Fast radio wave propagation predictions are of tremendous interest, e.g., for planning and optimization of cellular radio networks. We propose the use of ordinary graphics cards and specialized algorithms to achieve extremely fast predictions. Our

implementation of the empirical COST-Walfisch-Ikegami model allows the computation of several hundred predictions in one second in a 7 km^2 urban area. Further, we present a ray-optical approach exploiting the programming model of graphics cards. This algorithm combines fast computation times with high accuracy.

This paper introduces key concepts that this thesis elaborates in the chapter on radio wave simulation.

Graphics Hardware Accelerated Field Strength Prediction for Rural and Urban Environments [119]. We present general acceleration techniques on graphics hardware for field strength prediction algorithms. We identify some basic building blocks of common propagation algorithms and propose methods which can be efficiently implemented on GPUs. We show that the use of such acceleration techniques leads to huge speedups in the evaluation time of propagation algorithms.

This work discusses the use of GPUs in radio wave propagation predictions in more general terms and contributes to the overview sections of the present thesis.

Hardware Acceleration Techniques for 3D Urban Field Strength Prediction [120]. Planning and optimization of radio networks are active research areas. Therefore, both fast and accurate radio wave propagation predictions are required. To fulfill those requirements we propose specialized algorithms on ordinary graphics cards. We present an efficient algorithm for determining the visibility between objects. Therefore, we exploit the discrete pixel structure on GPUs. This leads to an acceleration of up to 140 times compared to a CPU version.

This paper presents a direct mapping of ray launching methods from the CPU to the GPU. It is not discussed directly in the present thesis, however, has influenced the algorithm design for diffraction into street canyons.

Fast Edge-Diffraction-Based Radio Wave Propagation Model for Graphics Hardware [121]. Fast radio wave propagation predictions are of tremendous interest, e.g., for planning and optimization of cellular radio networks. We propose the use of ordinary graphics cards and specialized algorithms to achieve extremely fast predictions. We present a ray-optical approach for wave diffraction at building edges into street canyons, exploiting the programming model of graphics cards. More than twenty predictions per second are achieved in a 7 km^2 urban area with a mean squared error of less than 7 dB when compared with measurements.

This paper introduces key concepts that this thesis elaborates in the chapter on radio

wave simulation.

Accelerating Radio Wave Propagation Algorithms by Implementation on Graphics Hardware [124]. Radio wave propagation prediction is a fundamental prerequisite for planning, analysis and optimization of radio networks. In this chapter we focus on accelerating techniques for predicting the mean received signal in dense urban environments. We use graphics hardware to accelerate the overall computation. Here, the main challenge is to trick the graphics processors into general purpose computing by transforming input data into images and algorithms into image synthesis. Among the most time consuming tasks in ray tracing based wave propagation is identifying all possible interaction sources for deflected propagation rays. Our method is also based on ray tracing. However, interaction sources are found by tracing full line-of-sight beams instead of single rays. Each beam is constructed such that it covers all rays that fall inside its angular opening. Thus, fewer beams than rays have to be processed. Moreover, we trace beams by traversing discrete sections of rays within a beam. As prove of concept, we have implemented four basic propagation effects: (1) line-of-sight propagation, (2) wall penetration depth in non-line-of-sight, (3) diffraction into street canyons and (4) diffraction over building rooftops.

The distinct algorithmic approaches to wall transmission and vertical and horizontal diffraction are conceptually introduced and evaluated in this paper and are presented in an extended form in the present thesis.

A Virtual Reality System for the Simulation and Manipulation of Wireless Communication Networks [123]. The knowledge of the propagation behavior of radio waves is a fundamental prerequisite for planning and optimizing mobile radio networks. Propagation effects are usually simulated numerically, since real-world measurement campaigns are time-consuming and expensive. Automatic planning algorithms can explore a vast amount of network configurations to find good deployment schemes. However, complex urban scenarios demand for a great emphasis on site-specific details in the propagation environment which are often not covered by automatic approaches. Therefore, we have combined the simulation of radio waves with an interactive exploration and modification of the propagation environment in a virtual reality prototype application. By coupling real-time simulation and manipulation tasks we can provide an uninterrupted user-centered workflow.

This paper is presented in an extended version in the thesis.

Beam Tracing for Multipath Propagation in Urban Environments [131]. We present a novel method for efficient computation of complex channel characteristics due to mul-

tipath effects in urban microcell environments. Significant speedups are obtained compared to state-of-the-art ray-tracing algorithms by tracing continuous beams and by using parallelization techniques. We optimize simulation parameters using on-site measurements from real world networks.

This paper is an extension of the work presented in this thesis, in particular the concept of beam tracing. In consequence, this is not discussed any further here. However, the closed-form multi-path optimization as presented in this thesis is first published in this paper.

Simulation of Radio Wave Propagation by Beam Tracing [132]. 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.

This paper is an extension of the work presented in this thesis. In particular the concept of beam tracing for the efficient computation of reflection ray paths. In consequence, this is not discussed any further here.

Efficient Rasterization for Outdoor Radio Wave Propagation [134]. Conventional 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. We use a custom, parallel rasterization pipeline for creation and evaluation of the beams. We implement a subset of a standard 3D rasterization pipeline entirely on the GPU, supporting 2D and 3D frame buffers for output. 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 simu-

lation results with measurements from a real-world network. Furthermore, we account for characteristics of different propagation environments and estimate the influence of unknown components like traffic or vegetation by adapting model parameters to measurements.

Only the multi-path parameter optimization is introduced in this thesis whereas the algorithmic part of the paper is not discussed in this thesis any further.

1.2.2.2. Probabilistic Tractography

The author's publications in the field of probabilistic tractography address the visualization of probabilistic tracts in VR [148, 122, 125]. In particular, an application of the proposed techniques has also been published in a neuroscientific journal [36] (co-authorship). A publication on the parallel implementation of the tractography algorithm has been submitted at the time of writing (December 2011).

GPU Computation of Probabilistic Diffusion Tractography (submitted) Understanding the connectivity structure of the human brain is a fundamental prerequisite for the treatment of psychiatric or neurological diseases. Probabilistic diffusion tractography is a promising technique to account for the inherent uncertainties in DTI data by estimating the likelihood that a fiber bundle takes its course through a particular voxel. Inspired by the virtual wind tunnel paradigm we present a GPU implementation of probabilistic tractography that allows the scientist to interactively explore global connectivity of diffusion data. Our implementation is based on an established neuroscientific tractography model that relies on probabilistic streamline integration. We extent the original algorithm by a probabilistic loop checking that can compromise between accuracy and memory requirements to detect undesired loops in streamlines. We establish the anatomical plausibility of our method by comparing our results against a widely used reference implementation. Depending on the parameter setting, we achieve speedups of up to 100X enabling real-time tracing and interactive visualization.

At the time of writing this paper has just been submitted and undergoes a peer review process. The techniques and results presented in this paper are a summary of the corresponding thesis chapter on estimating global connectivity.

Visualization of Probabilistic Fiber Tracts in Virtual Reality [125]. Understanding the connectivity structure of the human brain is a fundamental prerequisite for the treatment of psychiatric or neurological diseases. Probabilistic tractography has become an established method to account for the inherent uncertainties of the actual course of fiber

bundles in MRI data. This paper presents a visualization system that addresses the assessment of fiber probabilities in relation to anatomical landmarks. We employ real-time direct volume rendering to display fiber tracts within their structural context in a VE. Thereby, we not only emphasize spatial patterns but furthermore allow an interactive control over the amount of visible anatomical information.

The visualization of probabilistic fiber tracts is an essential component of the visual coupling of the probabilistic simulation of global connectivity.

GPU Implementation of 3D Object Selection by Conic Volume Techniques in Virtual Environments [122]. In this paper we present a GPU implementation to accurately select 3D objects based on their silhouettes by a pointing device with 6DOF in a VE. We adapt a 2D picking metaphor to 3D selection in VEs by changing the projection and view matrices according to the position and orientation of a 6DOF pointing device and rendering a conic selection volume to an off-screen pixel buffer. This method works for triangulated as well as volume rendered objects, no explicit geometric representation is required.

The techniques presented in this paper are essential building blocks in the VR prototypes for the simulation of radio waves and the visualization of probabilistic fiber tracts.

Evaluating a Visualization of Uncertainty in Probabilistic Tractography [148]. In this paper we evaluate a visualization approach for representing uncertainty information in probabilistic fiber pathways in the human brain. We employ a semi-transparent volume rendering method where probabilities of fiber tracts are conveyed by colors and opacities. Anatomic orientation is provided by placing anatomic landmarks in form of cortical or functional defined brain areas. In order to quantify the effectiveness of our approach we have conducted a formal user study concerning preferred anatomic context information and coloring of fiber tracts.

This work is partly included in the chapter about probabilistic tractography.

Probabilistic fiber tract analysis of cytoarchitectonically defined human inferior parietal lobule areas reveals similarities to macaques [36]. The human inferior parietal lobule (IPL) is a multi-modal brain region, subdivided in several cytoarchitectonic areas which are involved in neural networks related to spatial attention, language, and higher motor processing. Tracer studies in macaques revealed differential connectivity patterns of IPL areas as the respective structural basis. Evidence for comparable differential fiber tracts of human IPL is lacking. Here, anatomical connectivity of five cytoarchitectonic human IPL areas to 64 cortical targets was investigated using probabilistic

tractography. Connection likelihood was assessed by evaluating the number of traces between seed and target against the distribution of traces from that seed to voxels in the same distance as the target. The main fiber tract pattern shifted gradually from rostral to caudal IPL: Rostral areas were predominantly connected to somatosensory and superior parietal areas while caudal areas more strongly connected with auditory, anterior temporal and higher visual cortices. All IPL areas were strongly connected with inferior frontal, insular and posterior temporal areas. These results showed striking similarities with connectivity patterns in macaques, providing further evidence for possible homologies between these two species. This shift in fiber tract pattern supports a differential functional involvement of rostral (higher motor functions) and caudal IPL (spatial attention), with probable overlapping language involvement. The differential functional involvement of IPL areas was further supported by hemispheric asymmetries of connection patterns which showed left-right differences especially with regard to connections to sensorimotor, inferior frontal and temporal areas.

This paper is a neuroscientific publication that showcases the proposed visualization technique of probabilistic fiber tracts.

1.2.2.3. Additional Publications

Although the publications presented in this section are only indirectly related to the content of the present thesis, they are listed for the sake of completeness.

Interactive Particle Tracing in Time-Varying Tetrahedral Grids [32]. Particle tracing methods are a fundamental class of techniques for vector field visualization. Specifically, interactive particle advection allows the user to rapidly gain an intuitive understanding of flow structures. Yet, it poses challenges in terms of computational cost and memory bandwidth. This is particularly true if the underlying data is time-dependent and represented by a series of unstructured meshes. In this paper, we propose a novel approach which maps the aforementioned computations to modern many-core compute devices in order to achieve parallel, interactive particle advection. The problem of cell location on unstructured tetrahedral meshes is addressed by a two-phase search scheme which is performed entirely on the compute device. In order to cope with limited device memory, the use of data reduction techniques is proposed. A CUDA implementation of the proposed algorithm is evaluated on the basis of one synthetic and two real-world data sets. This particularly includes an assessment of the effects of data reduction on the advection process accuracy and its performance.

Related to the present work in terms of GPU implementation techniques.

Comparing Steering-Based Travel Techniques for Search Tasks in a CAVE [149]. We present a novel bimanual body-directed travel technique, PenguFly (PF), and compare it with two standard travel-by-pointing techniques by conducting a between-subject experiment in a CAVE. In PF, the positions of the users head and hands are projected onto the ground, and travel direction and speed are computed based on direction and magnitude of the vector from the midpoint of the projected hand positions to the projected head position. The two base-line conditions both use a single hand to control the direction, with speed controlled discretely by button pushes with the same hand in one case, and continuously by the distance between the hands in the other case. Users were asked to travel through a simple virtual world and collect virtual coins within a set time. We found no significant differences between travel conditions for reported presence or usability, but a significant increase in nausea with PF. Total travel distance was significantly higher for the baseline condition with discrete speed selection, whereas travel accuracy in terms of coin-to-distance ratio was higher with PF.

Related to the present thesis in terms of basic navigation principles in VR.

Towards the Visualization of Spiking Neurons in Virtual Reality [150]. This paper presents a prototype that addresses the visualization of the microscopic activity structure in the mammalian brain. Our approach displays the spiking behavior of neurons in multiple layers based on large-scale simulations of the cortical microcircuit. We will apply this visualization to the activity of brain-scale simulations by coupling the microscopic structure with the macroscopic level. Thereby, we hope to convey an intuitive understanding of the concise interaction and the activity flow of pairs of distant brain areas.

This paper addresses brain visualization on a microscopic level wheres the corresponding chapter of this thesis focuses on the macroscopic visualization.

Interactive Particle Tracing with Cumulative Blood Damage Computation for Ventricular Assist Devices [78]. This paper presents the results of a student research project and deals with the interactive particle tracing of derived quantities. Additionally to common vector field advection a complex model for the computation of blood damage in a ventricular assist device is computed for every particle. Thereby, we not only depict hydraulic flow structures but furthermore convey information like deformation and stress of blood particles. The employed blood damage model models the morphological deformation of blood cells along pathlines. Due to the large amount of simulation data, previous work on interactive blood damage analysis for ventricular assist devices relied on the precomputation of pathlines along the flow field of the artificial pump with an incremental computation of the blood damage. Therefore, the main contribution of this work lies in the interactive computation of both, particle tracing and blood damage

computation based on an implementation on a parallel many-core compute platform. However, due to memory limitations we operate on a downsampled version of the original grid. Nevertheless, the interactivity of our proposed technique reveals important flow structures and especially enables the exploration in virtual reality systems and use of natural interaction techniques with the complex three-dimensional data according to the virtual windtunnel paradigm.

Related to the present thesis in terms on GPU implementation techniques.

1.2.3. Outline

This chapter gave some general background information on programmable graphics hardware as a parallel computing platform and immersive scientific visualization. We discussed two major GPU programming paradigms: general purpose computations which transform algorithms into graphics and a unified approach that uses a C-like programming languages. We also introduced immersive visualization in VR with respect to some broader research areas related to the present thesis.

Chapter 2 addresses our real-time implementation of radio wave propagation. We first cover the basic foundations of urban propagation models and then describe the core concepts and individual algorithms for predicting path loss on the GPU. After introducing models for received signal strength that can be calibrated to real-world measurements we present a VR interface towards an interactive manipulation of site-specific details. We conclude the chapter with an in-depth discussion of computation time and accuracy of the individual models.

We present our parallel GPU approach to probabilistic tractography in Chapter 3. After presenting the technical background and problem statement we introduce the underlying mathematical model of tractography and discuss aspects of our parallel implementation. We briefly describe our visualization and interaction interface and conclude the chapter with a detailed examination of our algorithm in terms of anatomical plausibility and run time.

Chapter 4 concludes the thesis with a summary, possible extensions of the presented work and discusses the thesis' contributions in more detail.

CHAPTER 2

RADIO WAVE PROPAGATION

Abstract Knowledge of propagation behavior of radio waves is a fundamental prerequisite for planning and optimizing mobile radio networks. Propagation effects are usually simulated numerically, since real-world measurement campaigns are time-consuming and expensive. Measurements are usually only provided in small quantities in order to inspect the simulation results for correctness and accuracy. To choose optimal antenna sites, many different candidates have to be simulated. Hence, the computation should be both fast, in order to cope with the vast amount of simulations, and accurate, such that the simulation reflects the actual propagation behavior. Automatic planning algorithms can explore a vast amount of network configurations to find good deployment schemes. However, complex urban scenarios demand for a great emphasis on site-specific details in the propagation environment which are often not covered in automatic approaches. Therefore, we combined the simulation of radio waves with an interactive exploration and modification of the propagation environment in a virtual reality. By coupling real-time simulation and manipulation tasks we are able to provide an uninterrupted user-centered workflow.

The work presented here can be classified into two categories: (1) how real-time performance can be achieved for radio wave propagation predictions [37, 119, 120, 121, 124] and (2) how the user can be provided with efficient interaction techniques for an exploratory analysis of the complex propagation phenomena [122, 123]. The computation of multi-path effects will not be discussed any further here, however the work is published in [131, 132, 134].

The remainder of the chapter is organized as follows. After a brief motivation and problem statement in Section 2.1 and Section 2.2, respectively, we introduce general background information and review previous work on radio wave propagation in Sec-

tion 2.3. We present our real-time simulation algorithm in Section 2.4, where we present the technical details for the computation of urban propagation phenomena and describe path loss models. Model parameters are retrieved by solving a constrained optimization problem which is formulated in Section 2.4.3. Then, we present a VR interface for the interactive manipulation of site-specific details in Section 2.5. Here, we first give an overview of the general application layout and features and discuss the technical realization of a VR prototype application. We give a performance analysis in terms of computation time, accuracy and usability in Section 2.6 and conclude the chapter in Section 2.7.

2.1. Motivation

Radio wave propagation prediction is a fundamental prerequisite for planning, analyzing and optimizing radio networks. For instance coverage analysis, interference estimation or channel and power allocation all rely on propagation predictions. In wireless communication networks optimal antenna sites are determined by either conducting a series of expensive propagation measurements or by estimating field strengths numerically. In order to cope with the vast amount of different configurations to select the best candidate from, numerical predictions have to be both, accurate and fast.

One important aspect in radio wave propagation is the prediction of the mean received signal strength which can be simulated by taking complex interactions between radio waves and the propagation environment into account (see Figure 2.1). Thus, the simulation of radio waves for propagation predictions becomes a computationally intensive task.

Much effort and interest has been put into the acceleration of ray optical approaches, since most ray tracing algorithms tend to be computational intensive and exhibit run times up to several hours. Therefore, we focus on the efficient implementation of wave guiding effects on graphics hardware for the frequency range of common mobile communication systems, i.e., several hundred MHz up to few GHz.

2.2. Problem Statement

Among the most time consuming tasks in computing ray interactions with the propagation environment is the problem of visibility between objects, i.e., the identification of all possible interaction sources for diffracted or reflected propagation rays. The algorithms we will present here, are specifically designed to reduce the computational cost of the

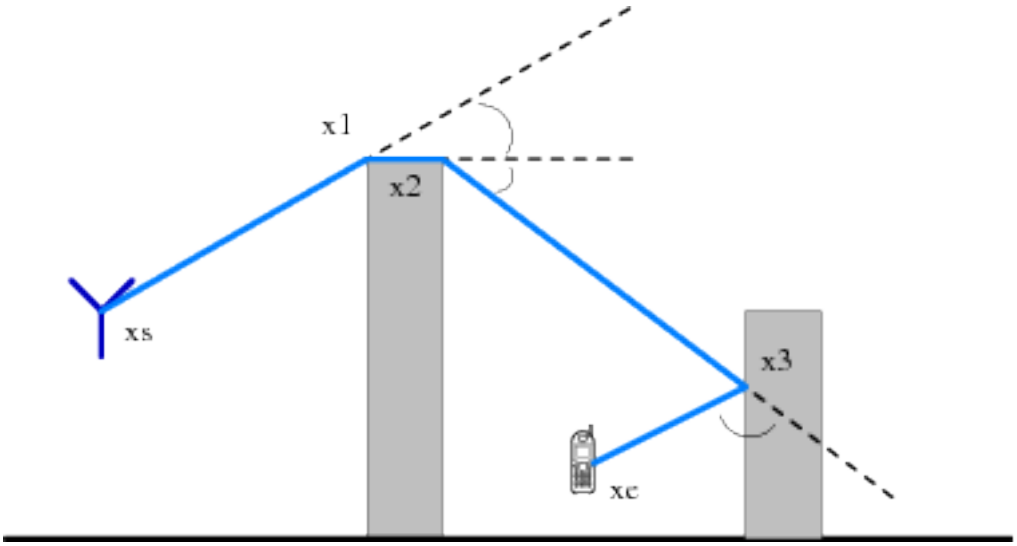


Figure 2.1.: Example propagation path (side profile). A ray is emanating from the radiation source, diffracted at the rooftop of the first building, and reflected at the second building into the street: $x_s \rightsquigarrow x_1 \rightsquigarrow x_2 \rightsquigarrow x_3 \rightsquigarrow x_e$.

visibility computations by exploiting special features of graphics cards.

Requirements The basic propagation phenomena are reflection, diffraction and scattering. All effects contribute to the radio signal distortions and give rise to signal fluctuations (fading) and additional signal propagation losses. We distinguish propagation effects according to the characteristics of the propagation environment by approximating which propagation paths (see Figure 2.1) are most likely to occur.

If receiver and transmitter antenna have an unobstructed direct path, they are in *Line-of-Sight* (LOS), otherwise in *Non-Line-of-Sight* (NLOS). An urban high-rise scenario consists predominantly of streets lined with tall buildings of several floors each. According to Andersen et al. [3] high building heights make significant contributions from diffraction over multiple rooftops rather unlikely. Therefore, if transmitter antennas are mounted below rooftops, dominant propagation effects are expected from reflection and diffraction into street canyons. See Figure 2.7(b) for an illustration. Furthermore, an urban low-rise scenario is characterized by wide streets and buildings with less than three floors. Antennas are usually mounted above average rooftop level and diffraction over rooftops (Figure 2.9(a)) becomes the dominant propagation effect. Propagation paths due to reflection are not considered in this chapter. However, we have presented an approach for the efficient computation of reflection paths in collaboration with Schmitz et al. in [134].

We formulate the requirements of our propagation algorithm as follows: (1) It is nec-

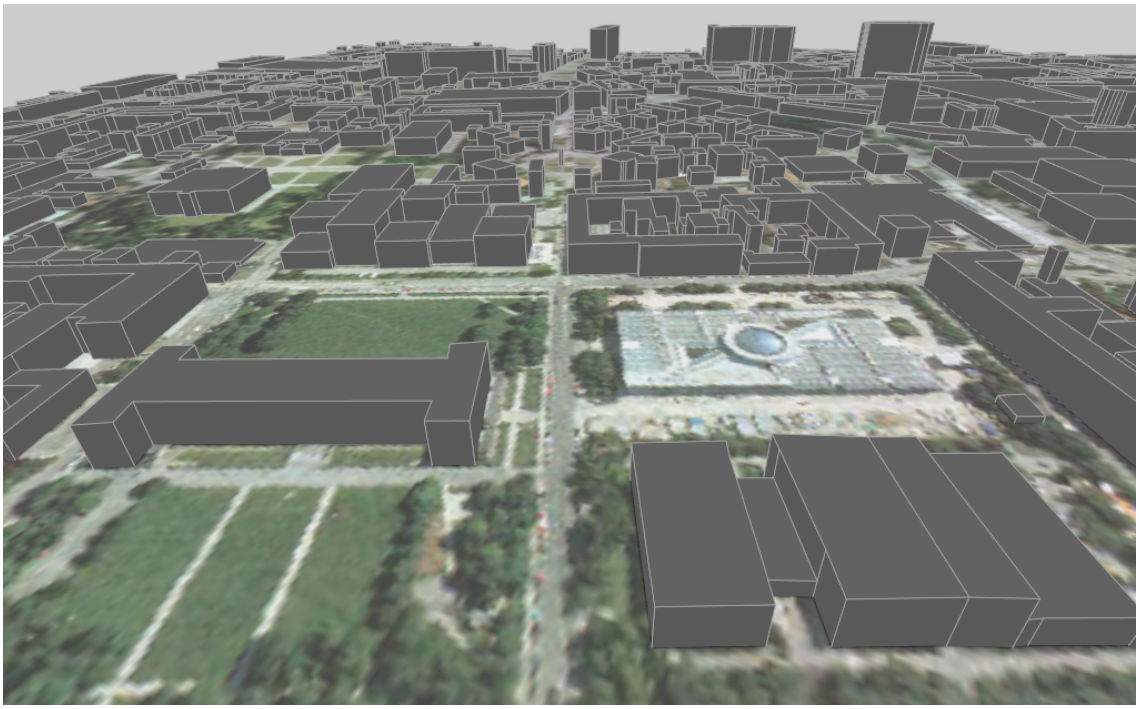


Figure 2.2.: Typical urban propagation environment with a simplistic city model and a satellite image for geographical reference.

essary to efficiently distinguish between regions in LOS and in NLOS. (2) In case of NLOS, we require our algorithm to compute different propagation paths, namely wall penetration depth, diffraction into street canyons and diffraction over rooftops.

Approach A promising approach is the use of ordinary graphics cards, nowadays available in every personal computer. With over 1000 Gigaflops, modern graphics hardware offers the computational power of a small-sized supercomputer. This is achieved by a strict parallel many-core architecture which can be accessed by a high level of programmability. The main challenge of utilizing graphics hardware for scientific computations is to trick the graphics processors into general purpose computing by casting problems as graphics: Input data is transformed into images and algorithms are turned into image synthesis.

Our method is based on ray tracing, however, interaction sources are found by tracing full line-of-sight beams instead of single rays. Each beam is constructed such that it covers all rays that fall inside its angular opening. Thus, fewer beams than rays have to be processed. Moreover, we do not actually compute any intersections between the propagation environment and radiation sources. We use a discrete sampling approach, i.e., we trace beams by traversing discrete points of all rays that lie within a beam. Our approach results in an acceleration of wave propagation algorithms by developing

a specialized implementation that exploits the parallel architecture of modern graphics hardware.

As prove-of-concept, we have implemented four basic propagation effects: (1) line-of-sight propagation, (2) wall penetration depth in non-line-of-sight, (3) diffraction into street canyons and (4) diffraction over building rooftops.

2.3. Background

2.3.1. Empirical Models

A widely adapted model (cf. [117]) for describing the path loss between transmitter and receiver at a distance d is the log-distance model

$$P^{\mathrm{dB}}_{\mathrm{logd}}(d) = c_f + \gamma \cdot 10 \cdot \log_{10}(d) \tag{2.1}$$

at a frequency f where

$$c_f = 20 \cdot \log_{10}\left(\frac{4\pi f}{c}\right)$$

is the frequency dependent loss and c denotes the speed of light. The path loss coefficient γ depends on the land cover and usually ranges between 2 (free space) and 3.5 (urban environment).

We will now introduce two well-known empirical models [60, 45], the COST-Hata model and the COST-Walfisch-Ikegami, in more detail. Both models are based on the log-distance model and describe the prediction of mean received signal strength in urban propagation environments and will later be compared against our approach. The COST-Hata model treats the propagation environment as either urban, suburban or rural. Additionally, the COST-Walfisch-Ikegami model distinguishes between propagation in LOS and NLOS. Since no actual propagation paths are considered both models are limited to the approximation of indirect wave fronts.

2.3.1.1. COST-Hata Model

For a receiver r at height Δh_r in distance d_r to a transmitter site with frequency f at height Δh_T the path loss for an urban propagation environment is given as

$$\begin{aligned} P^{\mathrm{dB}}_{\mathrm{urban}}(r) = {} & 33.9 \cdot \lg(f) + \left(44.9 - 6.55 \cdot \lg(\Delta h_{\mathcal{T}})\right) \cdot \lg(d_r) \\ & + g_{\text{COST-H}}(f, \Delta h_T, \Delta h_r) \end{aligned} \tag{2.2}$$

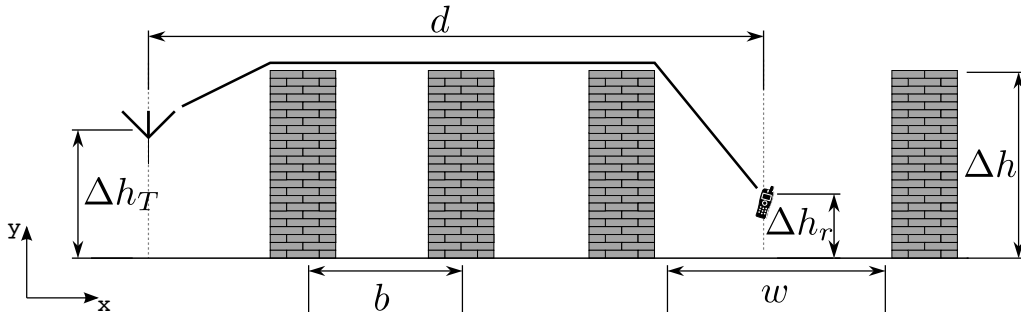


Figure 2.3.: This figure sketches the influencing terms for the COST-WI model (2.8). It takes into account statistics about the propagation environment like mean street separation b or mean building heights Δh.

similar to the log-distance model (2.1) with an additional empirical correction term

$$g_{\text{COST-H}}\left(f, \Delta h_T, \Delta h_r\right) = 46.3 - 13.82 \cdot \lg\left(\Delta h_T\right) - \Delta P^{\text{dB}}\left(f, \Delta h_r\right) + \begin{cases} 3 & \text{, urban center} \\ 0 & \text{, otherwise} \end{cases} \tag{2.3}$$

and a correction for receiver height gains

$$\Delta P^{\text{dB}}\left(f, \Delta h_r\right) = \big(1.1 \cdot \lg\left(f\right) - 0.7\big) \cdot \Delta h_r - \big(1.56 \cdot \lg\left(f\right) - 0.8\big). \tag{2.4}$$

The attenuation for suburban and rural environments is described as

$$P^{\text{dB}}_{\text{suburban}}\left(r\right) = P^{\text{dB}}_{\text{urban}}\left(r\right) - 2 \cdot \lg\Big(\lg\left(\frac{f}{28}\right)\Big)^2 - 5.4 \tag{2.5}$$

and

$$P^{\text{dB}}_{\text{rural}}\left(r\right) = P^{\text{dB}}_{\text{urban}}\left(r\right) - 4.78 \cdot \lg\big(\lg\left(f\right)\big)^2 + 18.33 \cdot \lg\left(f\right) - 40.94. \tag{2.6}$$

Hence, the attenuation for suburban and rural environments is based on the urban model with additional frequency dependent and constant correction terms.

2.3.1.2. COST-Walfisch-Ikegami Model

The COST-WI model is also based on the log-distance model (2.1) and similar to COST-Hata distinguishes between urban and suburban building density. Additionally, building information is considered to account for LOS and NLOS situations between sender and receiver. Further building information is only used in the form of mean building heights Δh, mean street widths w and mean building distances b, cf. Figure 2.3.

The LOS attenuation at a receiver location r is defined as

$$P_{\mathrm{LOS}}^{\mathrm{dB}}(r) = c_f + 26 \cdot \lg(d_r) + 10.2 \tag{2.7}$$

which is again a composition of frequency and distance dependency but with a constant empirical correction. The numerical values are derived from empirical measurements in various European cities. The rather low attenuation coefficient of the distance is explained by wave front propagation effects along street canyons.

The NLOS loss is described with varying empirical correction terms

$$P_{\mathrm{NLOS}}^{\mathrm{dB}}(r) = P_{\mathrm{F0}}^{\mathrm{dB}} + \max\left\{\Delta P_{\mathrm{F,rts}}^{\mathrm{dB}} + \Delta P_{\mathrm{F,msd}}^{\mathrm{dB}}, 0\right\} \tag{2.8}$$

where

$$P_{\mathrm{F0}}^{\mathrm{dB}} = 20 \cdot \lg\left(d_r \cdot \frac{4\pi f}{c}\right) = c_f + 20 \cdot \lg(d_r) \tag{2.9}$$

denotes the base loss for free space (isotropic) propagation at speed of light c in m/s. The additional correction terms approximate the influence of diffraction and scatter losses, cf. Figure 2.3 and will be discussed briefly in the following.

$\Delta P_{\mathrm{F,rts}}^{\mathrm{dB}}$ estimates the attenuation due to the last diffraction edge between a receiver in the street canyon and is referred to as roof-top-to-street-diffraction. It is influenced by the street width w and the orientation φ between the direct path and the respective street. The COST-WI model approximates this as

$$\begin{aligned}\Delta P_{\mathrm{F,rts}}^{\mathrm{dB}} = &-16.9 - 10 \cdot \lg(w) + 10 \cdot \lg(f) + 20 \cdot \lg(\Delta h - \Delta h_r) \\ &+ \begin{cases} -10 + 0.354 \cdot \varphi & , \text{falls } 0° \leq \varphi < 35° \\ 2.5 + 0.075 \cdot (\varphi - 35°) & , \text{falls } 35° \leq \varphi < 55° \\ 4.0 + 0.114 \cdot (\varphi - 55°) & , \text{falls } 55° \leq \varphi < 90° \end{cases}\end{aligned} \tag{2.10}$$

whereas multiple screen diffraction, i.e., the diffraction over multiple rooftops is described as

$$\Delta P_{\mathrm{F,msd}}^{\mathrm{dB}} = \Delta P_{\mathrm{F,msd,1}}^{\mathrm{dB}} + \Delta P_{\mathrm{F,msd,2}}^{\mathrm{dB}} + k_d \cdot \lg(d_r) + k_f \cdot \lg(f) - 9 \cdot \lg(b) \tag{2.11}$$

with

$$\Delta P_{\mathrm{F,msd,1}}^{\mathrm{dB}} = \begin{cases} -18 \cdot \lg(1 + \Delta h_T - \Delta h) & , \Delta h_T > \Delta h \\ 0 & , \text{otherwise} \end{cases}. \tag{2.12}$$

If the transmitting antenna is below the mean rooftop level an additional loss is added to account for the first diffraction to reach the roof level

$$\Delta P_{\mathrm{F,msd,2}}^{\mathrm{dB}} = \begin{cases} 54 & , \Delta h_T > \Delta h \\ 54 - 0.8\,(\Delta h_T - \Delta h) & , d_r \geq 0.5km \text{ and } \Delta h_T \leq \Delta h \\ 54 - 0.8\,(\Delta h_T - \Delta h)\,\frac{d}{0.5} & , d_r < 0.5km \text{ and } \Delta h_T \leq \Delta h \end{cases} \tag{2.13}$$

with

$$k_d = \begin{cases} 18 & , \Delta h_T > \Delta h \\ 18 - 15\frac{\Delta h_T - \Delta h}{\Delta h} & , \text{otherwise} \end{cases} \tag{2.14}$$

and

$$k_f = \begin{cases} -4 + 0.7\left(\frac{f}{925} - 1\right) & , \text{suburban} \\ -4 + 1.5\left(\frac{f}{925} - 1\right) & , \text{urban center} \end{cases}. \tag{2.15}$$

Since the COST-WI model takes only statistical information about the propagation environment into account, it performs best in uniform scenarios. However, heterogeneous environments like historically grown cities may require more sophisticated approaches that consider the actual propagation geometry. Nevertheless, the COST-Hata as well as the COST-WI model both require only few computations for a path loss prediction and are therefore often chosen for a fast preview of the propagation behavior in large radio networks. For reference, we implemented both models also on the GPU and will discuss their performance in terms of computation time and accuracy in more detail in Section 2.6.

2.3.2. Related Work

A theoretical foundation of radio wave propagation is given by Rappaport in [117]. Since a variety of approaches exists for solving the problem of predicting mean received signal strength we point the reader to [42] for an overview. In general, we distinguish between empirical (stochastic) channel models and deterministic propagation algorithms. Propagation models that approximate the path loss by a parametrized function like (2.1) can be categorized as empirical models, whereas deterministic approaches are often based on ray tracing. That is, they identify ray paths through the propagation environment based on wave guiding effects like reflection or diffraction.

Well-known empirical models are the work of Hata [64] and Ikegami et al. [70]. They propose to model the radio propagation phenomena by approximating the actual propagation loss (path loss) by a set of parametrized functions. Hata determined parameter values by conducting extensive measurement campaigns. Ikegami et al. extended Hata's work by analyzing the dependence of approximate equations for mean field strength in urban propagation environments with respect to height gain, dependence on street width, propagation distance and radio frequency. Due to their widespread use, we discussed both models in more detail in Section 2.3.1. Erceg et al. [52] pursued a similar approach and presented a stochastic channel model which can be applied to frequency ranges above two GHz (WiMAX). Additionally to height gains and propagation distance, they

included a parametrization of the environment into their model according to a flat or hilly terrain with either high or low vegetation density. Such empirical models are typically characterized by short evaluation time but are prone to prediction errors if their original model assumptions contradict the physical reality of the supplying area. Especially in Europe with its heterogeneous propagation environments of historically grown cities these models provide only limited value [45].

Therefore, most deterministic algorithms rely on the computation of actual propagation paths due to wave guiding effects like reflection, diffraction and scattering (see Figure 2.1). The principle of diffraction has been introduced in the field of electromagnetic by the Geometric Theory of Diffraction [77] and the Universal Theory of Diffraction [87] and is discussed in subsequent work in great detail, however, it has remained computationally expensive.

Typical deterministic approaches are often based on ray tracing, which was originally introduced by Whitted [158] to compute global illumination effects based on geometric optics for image synthesis. Although global illumination as formulated by Kajiya [76] and radio wave propagation are similar problem statements, different propagation effects like diffraction or interference become dominant when shifting from visible light to radio waves due to the different size of wavelengths. Since the computation of global lighting effects requires a huge amount of calculations, various approaches that focus on acceleration techniques by mapping global illumination algorithms onto the GPU [152, 19, 44, 153] have been developed. Furthermore, global illumination techniques have been used for different problems before, for instance for sound rendering. Notable here are the works of Tsingos et al. [141, 142] and Funkhouser et al. [59]. Most approaches focus on the efficient computation of reflection, however some work on the diffraction effect is described for instance by Stam [138] for application in computer graphics, whereas Tsingos et al. apply diffraction theory for modeling acoustics in virtual environments.

Ikegami et al. [69] showed that classical ray tracing can also be applied to the estimation of radio propagation losses. Due to complex interactions of radio waves and geometric structures, this is a time consuming task. However, high prediction accuracy can be achieved. For instance, [128, 51, 79] and [133] state that their predicted path loss values were generally within 4 to 8 dB of the measured path loss, which is considered as a very good result. Reasonable computation times are for instance achieved in [94] where a ray launching algorithm is used, which represents an urban environment as a grid of discrete blocks. Ray-object intersections are found by traversing the blocks by a line sampling method, thereby greatly reducing computation time. In order to further reduce the computational complexity, we presented a GPU-based approach to radio wave propagation in [37, 121] and [124], where we trace propagation paths in a discrete fashion by repeated rasterization of line-of-sight regions. Part of this work is presented in this chapter.

The idea of ray tracing can be further extended to the concept of beams, which are a continuum of rays. Beam tracing was first introduced by Heckbert and Hanrahan [65]. The benefits of beam tracing are mainly reduced intersection tests and less sampling problems. After a few iterations ray samples tend to become either too sparse or too dense, this can be alleviated by tracing beams. Work in this area includes application of real time rendering by Overbeck et al. [107] or audio rendering by Funkhouser et al. [59]. An application of beam tracing to the problem of radio wave propagation can be found in the work by Rajkumar et al. [116] and more recently by Schmitz et al. [132]. They especially address the issue of delay spread due to multi-path propagation.

Recent applications of VR to simulation tasks include but are not limited to the reconstruction of traffic flows in [146] or the interactive simulation of nanoparticle manipulation in [25]. To the best of our knowledge, this is the first application of radio wave propagation in VR.

Symbol	Description
$s = (x, y, z)$	Radiation source location with height $s.z$
$\mathcal{R}$	Discrete set of receiver points with constant ground level $\mathcal{R}.z$
$\mathcal{W}$	Set of all building walls w
$w = (p_0, p_1, p_2, p_3, \vec{n})$	Building wall with p_0, p_1 at ground level and p_2, p_3 at rooftop level. The normal vector $\vec{n}$ is perpendicular to the wall and points away from the associated building.
$\mathrm{NLOS}\,(s \to w)$	Shadow polygon cast by wall w when viewed from point s
$\mathrm{LOS}\,(s) \subseteq \mathcal{R}$	Subset of $\mathcal{R}$ that is in line-of-sight to s
$\mathrm{VDS}\,(s)$	Set of vertical (street) diffraction sources for a radiation source s
$\mathrm{VDB}\,(x) \subseteq \mathcal{R}$	Vertical diffraction beam with origin at the diffraction source x, i.e. a subset of $\mathcal{R}$ that forms a secondary wave front.
$\mathrm{HDB}\,(s \to w)$	Horizontal (roof) diffraction beam with origin s and diffraction edge w.

Table 2.1.: Algorithm Notation

2.4. Wave Propagation

Similar to [151], the input data requirements of our algorithms for radio wave propagation are building structures with corresponding building heights. The shape of rooftops is usually omitted and hence, a building is described by its polygonal outline and one height value. We refer to this representation as 2.5 dimensional. Building structures are usually given in vector format with a location accuracy in the order of 1 to 10 meter. In order to produce reliable results, height accuracy should be around 1 to 2 meter. Information about vegetation is not considered and terrain is assumed to be flat. Figure 2.2 shows a typical propagation environment.

In general, the output is a discrete two-dimensional raster image which serves as a role model for the discrete representation of results from GPU computations. An image consists of pixels that are organized in a regular array. Pixels are the data elements of this structure. However, pixels are not restricted to contain only color information. Pixel data is interpreted according to the current context of the GPU computation.

In order to provide a formal description of the presented algorithms we introduce the following notation (see Table 2.1): Let $s = (x, y, z)$ be a radiation source location, e.g., the transmitter antenna, where the height is referred to as $s.z$. All height values are relative to ground. A wall w of a building structure consists of two points p_0 and p_1 at ground level and two points p_2 and p_3 at rooftop level. Due to the 2.5 dimensional

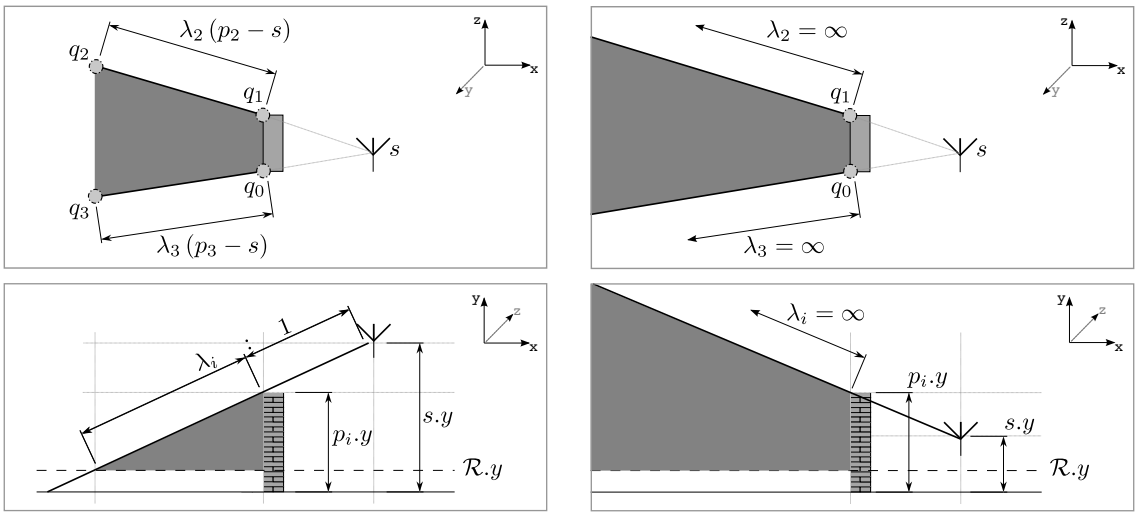


Figure 2.4.: Top and side profile of the shadow polygon (dark gray) according to the intercept theorem. Left: source height $s.z$ is greater than the wall height $p_i.z$. Right: source height $s.z$ is less or equal to the wall height $p_i.z$.

database, the wall point coordinates differ only in their height components, i.e. they rise perpendicular to the ground plane. The set containing all walls w is denoted by $\mathcal{W}$. Additionally, we associate a normal vector $\vec{n}$ with each wall. Normal vectors are perpendicular to the respective wall and point away from the corresponding building. Since the terrain is assumed to be rather flat and the receiver points are typically located 1.5 meter above ground we define a *receiver plane* $\mathcal{R}$ to be a discrete set of receiver points at constant height $\mathcal{R}.z$. We restrict ray path calculations to only those paths that intersect the receiver plane.

2.4.1. Algorithms

We will now describe algorithms, which are explicitly designed to run directly on graphics hardware. First, we present a method for determining LOS regions. Then we will show how this algorithm can be extended to provide additional NLOS information. Furthermore, we show how graphics cards can be exploited for ray path calculation due to diffraction into street canyons and diffraction over rooftops. GPU implementations are obtained by separating the calculation of these effects into distinct algorithms.

2.4.1.1. Line-of-Sight

The LOS algorithm computes a sampling of the receiver plane $\mathcal{R}$, where each location $p \in \mathcal{R}$ is marked whether it is in clear line-of-sight to a source s or not. This algorithm

will be one of the main building blocks in the computation of ray paths on graphics hardware.

The GPU computation is based on the concept of shadow volumes, cf. [54]. This technique constructs a polygonal representation of the shadow cone (shadow volume) for 3-dimensional triangular geometry. Since the description of urban propagation environments involves usually just a polygonal outline and one corresponding height value, we propose a specialized algorithm for this particular form of geometry (see [37]). The main idea is to extract the shadow of each building wall directly in the receiver plane. We refer to the intersection of the shadow cone and the receiver plane as shadow polygon. Regions are in LOS, if and only if they are in no shadow polygon. The construction of the shadow polygon of a wall w proceeds as follows: each shadow polygon is a quadrangle with corners (q_0, q_1, q_2, q_3). Points q_0 and q_1 are given by the corners of the wall p_0 and p_1 on the ground. The remaining two corners are determined by the intersection of the receiver plane $\mathcal{R}$ and each of the straight lines through the source point s and the wall point at roof level p_2 and p_3.

According to the intercept theorem (see Figure 2.4), $q_i, i \in \{2, 3\}$ is then given by

$$q_i = p_i + \lambda \cdot (p_i - s) \tag{2.16}$$

where

$$\lambda = \begin{cases} \frac{p_i.z - \mathcal{R}.z}{s.z - p_i.z} & \text{, if } s.z - p_i.z > 0 \\ \infty & \text{, otherwise} \end{cases} . \tag{2.17}$$

The corners of each shadow polygon are computed by a vertex program in parallel for each wall and sampled into discrete points by the subsequent rasterization phase. The result is a two-dimensional pixel array, every (discrete) receiver location lies either in LOS or inside a shadow polygon, hence in NLOS.

With q_2 and q_3 according to equation (2.16), we refer to the shadow polygon of a single wall w with source point s as

$$\text{NLOS}\,(s \to w) = \{p \in \mathcal{R} \mid p \in \text{polygon}\,(w.p_0, w.p_1, q_2, q_3)\}\,. \tag{2.18}$$

A *LOS beam* $\text{LOS}\,(s)$ is the set of discrete points in the receiver plane that do not lie in any shadow polygon of the source s

$$\text{LOS}\,(s) = \bigcap_{w \in \mathcal{W}} \{p \in \mathcal{R} \mid p \notin \text{NLOS}\,(s \to w)\}\,. \tag{2.19}$$

Figure 2.5 shows an example of a discretized receiver plane with pixels in LOS (dark gray) and NLOS (light gray) pixels. Although the result is two-dimensional, all shadow computations are performed using all 2.5-dimensional information.

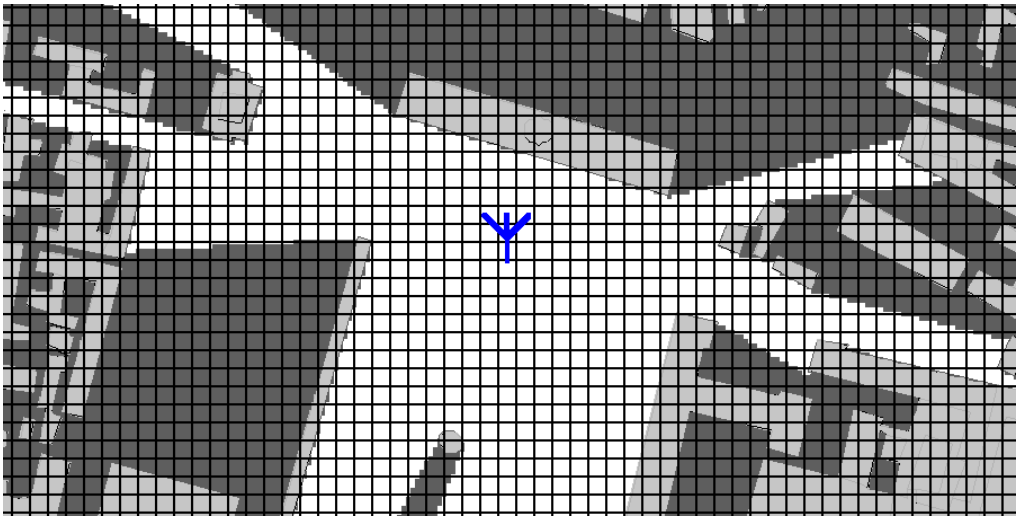


Figure 2.5.: Top view of the discrete non-line-of-sight beam (dark gray) with the source point in the center (blue symbol). Building interior is shown in light gray. The discretization is overemphasized by a black grid for visualization purposes.

Algorithm Details Algorithm 1 describes the realization of (2.16) and (2.17) as a vertex shader, i.e., it transforms geometry on a per vertex basis to construct the edges of the shadow polygon. Algorithm 2 implements (2.18), it calls the vertex processor with Algorithm 1 and assembles the fragments of each shadow polygon by rasterization. In the fragment processor all corresponding fragments are then shaded with a color that encodes NLOS. The implementation of a discrete LOS beam is detailed in Algorithm 4. Fragments are initialized with the set of discrete points in the receiver plane. Shadow fragments are subtracted from this set by repeated calls to Algorithm 2. For convenience, we also define the complementary NLOS beam in Algorithm 3.

A practical application of these algorithms is given in Algorithm 5. We detail the implementation of a two-slope model (similar to the COST-WI model) that is entirely performed on the GPU. To avoid multiple evaluations of the path loss model on fragments that coincide due to overlapping shadow regions, we employ a two-phased approach. We disable model evaluation by turning off color assembling, fill the depth buffer with zeros and accept all incoming fragments, regardless of their depth. Then, we render the shadow polygons of each wall with a fixed depth value of one. This will not change the shape of the polygon in the receiver plane. So far, no shading (and hence no model evaluation) has been performed but a one has been recorded in every fragment of the depth buffer that corresponds to the NLOS beam, whereas all other parts of the depth buffer are still zero (thus marking a LOS fragment). Now, we exploit the depth buffer test by setting the depth comparison function to accepts only fragments of the same depths as the one already in the depth buffer. Color assembling is activated again and we fill two buffers, one with the evaluation of the LOS model and the other with the evaluation of the NLOS model. These buffers are than renders as a flat texture with a depth value of

zero and one, respectively. The depth test makes sure that only fragments of the same depth value are shaded with the correct slope of the path loss model.

Algorithm 1 SHADOW_SHADER(Vertex p, s)

Note: p is vertex of a wall, $p.y$ corresponds to wall height. s is radiation source.

```
d ← p − s
if s.y > p.y then
    λ ← p.y / (s.y − p.y)
else
    λ ← ∞
end if
p ← p + λ · d
return p
```

2.4.1.2. Wall Transmission

The algorithm for LOS beams can be extended to provide additional *non*-line-of-sight information. This can include for instance the number of penetrated walls or material which can be achieved by taking all walls into account that intersect the direct ray from the source to a receiver as sketched in Figure 2.6.

We first provide a more thorough look at GPU frame buffer operations, which are an integral part of the algorithm for transmission depth. When fragments are collected and recorded in the frame buffer at the final stage of the rendering pipeline (cf. Section 1.1.1), frame buffer operations decide how fragments that fall on the same pixel position, contribute to the final color of that pixel. Commonly, the fragment with the lowest depth value, i.e., which is nearest to the viewer, determines (replaces) the pixel color. Alternatively, the final color can be a combination (interpolation) of the values of both fragments, the one already in the frame buffer and the new one, which wants to

Algorithm 2 NON_LINE_OF_SIGHT(Vertex $source, p, q$)

Note: Assemble fragments of shadow polygon.

```
p' ← SHADOW_SHADER(p, source)  {vertex processor}
q' ← SHADOW_SHADER(q, source)  {vertex processor}
poly ← polygon(p', p, q, q')
fragments ← RASTERIZE_POLYGON(poly)  {rasterizer}
for all frag ∈ fragments {fragment processor} do
    frag.color ← RGBA(NLOS) {mark fragment as NLOS }
end for
return fragments
```

Algorithm 3 NLOS_BEAM(Vertex *source*, WallSet $\mathcal{W}$)

Note: Collects all shadow fragments for a given source and wall set.

```
frags ← ∅
for all w ∈ W do
  frags ← frags ∪ NON_LINE_OF_SIGHT(source, w.p2, w.p3) {add shadow fragments}
end for
return frags
```

Algorithm 4 LOS_BEAM(Vertex *source*, WallSet $\mathcal{W}$)

Note: Collects all line-of-sight fragments for a given source and wall set.

```
frags ← R− NLOS_BEAM(source,W) {remove NLOS fragments from discrete receiver
points }
return frags {return remaining LOS fragments}
```

Algorithm 5 TWO_SLOPE(Vertex *source*, WallSet $\mathcal{W}$)

Note: A two slope model evaluates path loss differently in LOS and NLOS.

```
DISABLE_BUFFER(color_buffer) {disable color assembling }
CLEAR_BUFFER(depth_buffer,0) {fill depth buffer with zeros}
DEPTH_FUNC(always) {depth buffer test accepts all incoming fragments}
for all w ∈ W do
  p ← w.p2
  q ← w.p3
  p.z ← q.z ← 1 {set depth value z = 1}
  NON_LINE_OF_SIGHT(source, p, q) {render NLOS region}
end for
DEPTH_MASK(false) {set read-only access to depth buffer }
ENABLE_BUFFER(color_buffer) {enable color assembling}
DEPTH_FUNC(equal) {only accept fragments of same depth value}
tex_LOS ← EVAL_MODEL(LOS) {evaluate LOS model for all receiver points}
tex_NLOS ← EVAL_MODEL(NLOS) {evaluate NLOS model for all receiver points}
DRAW_TEXTURE(tex_LOS, z = 0) {render with depth value z = 0}
DRAW_TEXTURE(tex_NLOS, z = 1) {render with depth value z = 1}
return color_buffer
```

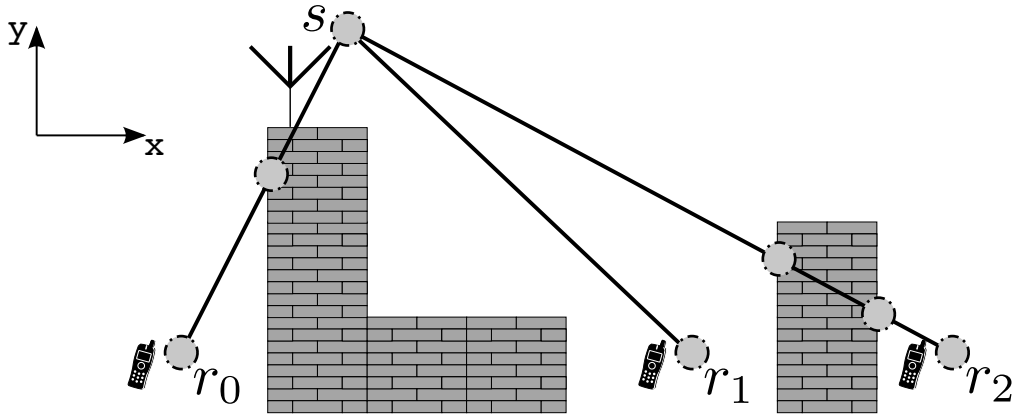


Figure 2.6.: The Wall transmission depth is taken into account by counting the number of walls in the direct path between the source and the receiver points. This is achieved by an additive blending of shadow polygons on the graphics card.

occupy the same pixel position. This technique is called blending. In image synthesis, blending is commonly used to draw translucent objects.

Here, we use the blending capability to increase the value in the frame buffer with every rendered shadow polygon. Thus, instead of a solid rendering (replacement of fragments) of the shadow polygons, we apply an additive blending. This effectively counts the number of shadow fragments at each receiver location. However, frame buffer operations like blending are currently implemented in hardware with a precision of 8 bits, only. This means that we could only count up to 255 wall transmissions, which may not be enough for large and complex scenarios. A solution is presented by graphics cards that offer buffers (textures) of higher precision like 32 bit. Blending has to be implemented by a user-written fragment program since high precision blending is not yet directly supported in hardware.

To overcome this drawback and to support propagation environments with an arbitrary number of walls, we employ a hybrid approach that combines 8 bit hardware and 32 bit software blending. Two 32 bit buffers are created and filled with zeros. These buffers will provide a so-called ping-pong scheme because current GPUs do not support a simultaneous read and write access to the same buffer. One buffer is referred to as the *next* buffer, it is the buffer which will be rendered to. The other buffer is referred to as the *current* buffer, it is bound as a texture and read from in the updating step. After each pass, the buffers are swapped, so that the *next* buffer becomes the *current* buffer and vice versa.

Hence, one buffer will keep track of the total transmission depth for every fragment and the other will store intermediate results. The actual rendering of the shadow polygons

is done into a third 8 bit frame buffer with hardware supported additive blending. The vertex buffer containing the wall geometry is rendered in separate chunks of 255 walls each. After each chunk rendering the transmission depth of every fragment is added to the *next* buffer. This is done by a fragment shader which reads the *current* buffer and the 8 bit buffer containing the transmission depths of the latest wall chunk. A simple add operation is performed and the result is stored in the *next* buffer. This procedure is repeated until all wall chunks have been processed. This approach is a multi-pass algorithm, for n walls it requires $\lceil \frac{n}{255} \rceil$ rendering passes. This number can be reduced if chunks of non overlapping shadow polygons are rendered in parallel.

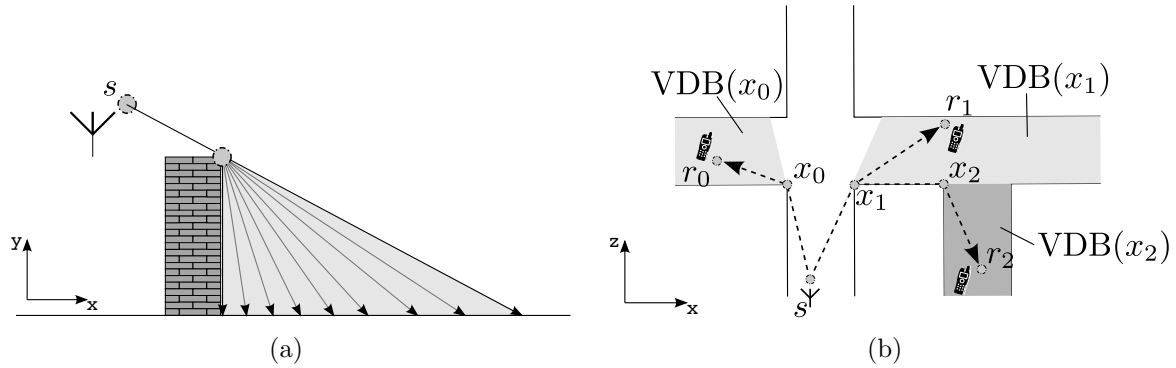
Algorithm Details The hardware blending part is sketched in Algorithm 6. The ping-pong scheme can be seen as an outer loop to this. We enable GPU alpha blending and set an additive blending function such that the incoming source color is multiplied by the source alpha value and added to the destination color. This way, different wall material coefficients can be incorporated in the corresponding color components. The accumulated NLOS information is recorded in the color buffer and gives the number of shadow polygons per fragment. This information can then in turn be used for a path loss model evaluation, cf. the Multi-Wall model [91] or Section 2.4.2.2.

Algorithm 6 WALL_TRANSMISSION(Vertex *source*, WallSet $\mathcal{W}$)

Note: Provide additional *non*-line-of-sight information.
ENABLE(blending) {enable GPU alpha blending}
BLEND_FUNC(additive) {set GPU blending function to $dest.col += src.col \cdot src.a$ }
CLEAR_BUFFER(color_buffer,0)
for all $w \in \mathcal{W}$ **do**
RGB *wall_type* $\leftarrow$ COLOR_FROM_WALL_TYPE(w)
SET_COLOR(*wall_type*,1/255) {color encodes wall material}
NON_LINE_OF_SIGHT($source, w.p_2, w.p_3$) {record NLOS fragments with current material properties in color buffer}
end for
return color_buffer

2.4.1.3. Diffraction into Street Canyons

Classical ray tracing algorithms for radio wave propagation commonly model diffraction effects by tracing a multitude of rays into the respective diffraction cone as illustrated in Figure 2.7(a). This is computationally intensive, as one ray (the one hitting the diffraction edge) is split into many secondary rays. Shooting fewer rays usually results in an under sampling of the propagation environment, leading to regions where no rays arrive, and consequently no path loss calculations can be performed.



(a) (b)

Figure 2.7.: (a) Diffraction is modeled by shooting a multitude of rays into the diffraction cone. (b) Diffraction beams propagate in nearby street canyons. Diffraction sources are $\mathrm{VDS}(s) = \{x_0, x_1\}$ and $\mathrm{VDS}(x_1) = \{x_2\}$. Geometric ray paths can be constructed for instance as $s \rightsquigarrow x_0 \rightsquigarrow r_0$ or $s \rightsquigarrow x_1 \rightsquigarrow x_2 \rightsquigarrow r_2$.

A solution to this problem is presented by the following approach. In order to achieve a high throughput of diffraction computations, we trace full beams instead of single rays. This has the advantage that beams cover a lot more area than single rays and thus, a lot fewer beams than rays have to be processed for a sufficient sampling of the propagation environment. The algorithm consists of three main steps. (1) Identify all potential diffraction sources for a propagation into street canyons. (2) Determine the propagation paths of the secondary wave fronts. (3) Compute geometrical paths in reverse order from the target location towards the diffraction source. This procedure can be applied recursively if propagation paths along multiple street canyons are desired. The idea is sketched in Figure 2.7(b).

Let s be a radiation source. According to Section 2.4.1.1 the set of discrete points that are in line-of-sight to s is defined by $\mathrm{LOS}(s)$. We define a *diffraction source* as an end point of a wall w that satisfies the following criteria: (1) A wall w is in LOS to the radiation source s if one of the end points $w.p_0, w.p_1$ is in LOS

$$w \in \mathrm{LOS}(s) \iff w.p_0 \in \mathrm{LOS}(s) \vee w.p_1 \in \mathrm{LOS}(s) \tag{2.20}$$

and (2) if the normal vector $\vec{n}$ of the wall w is facing away from the radiation source s

$$\left\langle w.\vec{n}, \frac{w.p_0 - s}{\parallel w.p_0 - s \parallel_2} \right\rangle < 0. \tag{2.21}$$

For a radiation source s, we collect all possible *vertical diffraction sources* in the set $\mathrm{VDS}(s)$

$$\mathrm{VDS}(s) = \bigcup_{w \in \mathrm{LOS}(s)} \begin{cases} \{w.p_j\} & \text{, if } \left\langle w.\vec{n}, \frac{w.p_0 - s}{\|w.p_0 - s\|_2} \right\rangle < 0 \\ \emptyset & \text{, otherwise} \end{cases} \tag{2.22}$$

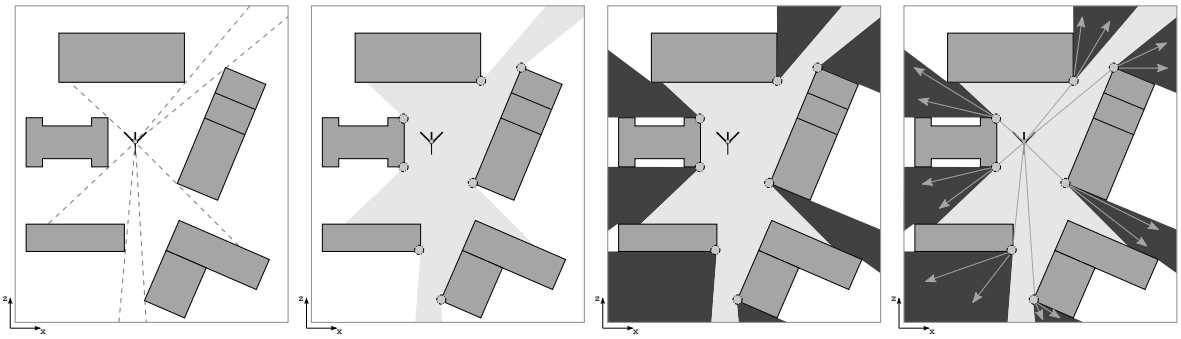


Figure 2.8.: Step-by-step illustration of the algorithm for vertical diffraction into street canyons. First, we identify all diffraction sources within the discrete LOS beam of the radiation source. Each diffraction source triggers in turn a computation of secondary LOS beams from where geometric ray paths can be reconstructed.

where each $w.p_j$ is chosen such that

$$j \in \{0,1\} \wedge \parallel w.p_j - s \parallel_2 \leq \parallel w.p_{1-j} - s \parallel_2 . \tag{2.23}$$

By construction, a diffraction source is always a point on the wall that is closest to the radiation source.

Now, we set up the *vertical diffraction beam* $\mathrm{VDB}(x)$ as the set of points that forms a secondary wave front originating at the diffraction source $x \in \mathrm{VDS}(s)$. With that notation we can write a diffraction beam as

$$\mathrm{VDB}(x) = \{r \in \mathcal{R} \mid r \in \mathrm{LOS}(x) \wedge r \notin \mathrm{LOS}(s)\} . \tag{2.24}$$

All points which have been in line-of-sight to the original radiation source s will not be part of the diffraction beam. Thereby, we ignore regions that would result in very large diffraction angles which would in turn not contribute to the overall signal level, significantly.

The final step consists of the reconstruction of geometric ray paths based on the beam information. For each receiver location in $\mathrm{VDB}(x)$, we create a ray path

$$\Upsilon^{\rightsquigarrow}(x) = \{v_0^{\rightsquigarrow}, \ldots, v_N^{\rightsquigarrow}\} = \{s \rightsquigarrow x \rightsquigarrow r \mid r \in \mathrm{VDB}(x)\} . \tag{2.25}$$

Thus, every point within the beam travels along its diffraction source towards the original radiation source. Diffraction paths of arbitrary length can be constructed by assigning the start points of the rays as new diffraction sources in a recursive fashion.

Algorithm Details Algorithm 7 describes the implementation of finding vertical diffraction sources as formally defined in (2.22). This involves checking every wall for LOS status and orientation, cf. (2.20) and (2.21), which is realized in a vertex shader in parallel

Algorithm 7 VERTICAL_DIFFRACTION_SOURCES(Vertex *source*, WallSet $\mathcal{W}$)

Note: Collects all vertical diffraction points for a given source and wall set.

```
frags_los ← LOS_BEAM(source, W) {get LOS fragments from source }
VDS ← ∅ {set of vertical diffraction sources}
for all w ∈ W do
  if w.p0 ∈ frags_los or w.p1 ∈ frags_los then
    if ⟨w.n, (w.p0 − source)/‖w.p0 − source‖2⟩ < 0 then
      if ‖ w.p0 − s ‖2 ≤ ‖ w.p1 − s ‖2 then
        VDS ← VDS ∪ {w.p0}
      else
        VDS ← VDS ∪ {w.p1}
      end if
    end if
  end if
end for
return VDS
```

since walls can be processed independently from one another. Figure 2.8 illustrates the process. Vertical diffraction can only occur at vertical building edges. Hence, for every wall this is a diffraction candidate, the diffracting edge is the one nearer to the radiation source. Since we are only interest in ray path with significant power contributions we choose the corresponding wall vertex at roof top level.

For better performance we integrated the computation of vertical diffraction beams (2.24) and ray paths (2.25) into one algorithm, Algorithm 8. If we would consider all possible ray interactions, a huge number of secondary rays would be constructed with no significant power contributions due to large diffraction angles. To counteract, we employ an early-out strategy and only keep the secondary rays with the lowest diffraction angles. This can efficiently be accomplished by exploiting depth buffer capabilities of the GPU. Initially, we fill the depth buffer with the angle π in radians. The depth buffer test is set to only accepts fragments of lesser values than those already at the same pixel position and we encode each respective diffraction angle in the fragment's depth value in the fragment shader. The actual deflection point is encoded in each fragment's color value to allow a subsequent reconstruction of the diffraction ray paths.

2.4.1.4. Propagation over Rooftops

The algorithm for diffraction into street canyons (see Section 2.4.1.3) always maps one wall edge to one source point for LOS since only ray paths that hit the receiver plane are computed. The calculation of propagation paths over rooftops is different because the diffraction source now spans over the whole edge of the roof, cf. Figure 2.9(a). Hence,

Algorithm 8 VERTICAL_DIFFRACTION_RAYPATH(Vertex *source*, WallSet $\mathcal{W}$)

Note: Constructs vertical diffraction ray paths for a given source and wall set.

$\Upsilon^{\rightsquigarrow} \leftarrow \emptyset$ {set of ray paths}
frags_src_los $\leftarrow$ LOS_BEAM($source, \mathcal{W}$) {get LOS fragments from radiation source }
VDS $\leftarrow$ VERTICAL_DIFFRACTION_SOURCES($source, \mathcal{W}$)
CLEAR_BUFFER(color_buffer,0)
CLEAR_BUFFER(depth_buffer,π)
for all v $\in$ VDS **do**
  frags_diff_los $\leftarrow$ LOS_BEAM($v, \mathcal{W}$) {get LOS from diffraction source v }
  {for all fragments that are in LOS to v but not to *source*}
  for all frag $\in$ frags_diff_los $-$ frags_src_los **do**
    frag.color $\leftarrow$ v {encode diffraction source as color}
    $\alpha \leftarrow \text{acos}\left(\langle \| v - source \|_2, \| frag.pos - v \|_2 \rangle\right)$ {compute diffraction angle}
    frag.depth $\leftarrow \alpha$ {encode diffraction angle as depth value}
    {depth test only keeps fragments with lower diffraction angle}
    if frag.depth $<$ depth_buffer[frag.pos] **then**
      depth_buffer[frag.pos] $\leftarrow$ frag.depth
      color_buffer[frag.pos] $\leftarrow$ frag.color
    end if
  end for
end for
for all frag $\in$ color_buffer **do**
  {create new ray path from source over diffraction point to current receiver location }
  $\Upsilon^{\rightsquigarrow} \leftarrow \Upsilon^{\rightsquigarrow} \cup \{source \rightsquigarrow frag.color \rightsquigarrow frag.pos\}$
end for
return $\Upsilon^{\rightsquigarrow}$

this section describes how propagation paths due to diffraction over rooftops can be calculated efficiently on the graphics processing unit. The algorithm basically consists of two steps: (1) A discretized version of the diffraction cones is constructed for every rooftop. (2) Ray paths are found by going backwards from each receiver location towards the transmitter.

An integral part of our method is the computation of diffraction beams for every rooftop simultaneously. Therefore, no identification of diffraction sources is required, we define the set of roof diffraction sources directly as the set of all building walls $\mathcal{W}$. By construction the recursion depth is automatically set to the total number of walls. By formulating the computation as follows, high computational performance is achieved by directly parallelizing the computation over the recursion steps, i.e. the diffraction cones for each ray path length are computed at the same time, independently from one another.

Let s be a radiation source. As illustrated in Figure 2.9(a), a *horizontal (roof) diffraction beam* $\mathrm{HDB}(s \to w)$ of a wall w directly corresponds to the shadow polygon

$$\mathrm{HDB}(s \to w) = \mathrm{NLOS}(s \to w). \tag{2.26}$$

Due to the construction of the beam, all geometric ray paths from $\mathrm{HDB}(s \to w)$ to the radiation source s have a deflection point somewhere on the wall w at rooftop level, e.g. $s \rightsquigarrow w \rightsquigarrow r$. For a point $r \in \mathrm{HDB}(s \to w)$ the exact deflection point x_r on the wall can be found as the intersection of the two lines $l_0 = (w.p_2, w.p_3)$ and $l_1 = (r, s)$ as sketched in Figure 2.9(b), hence

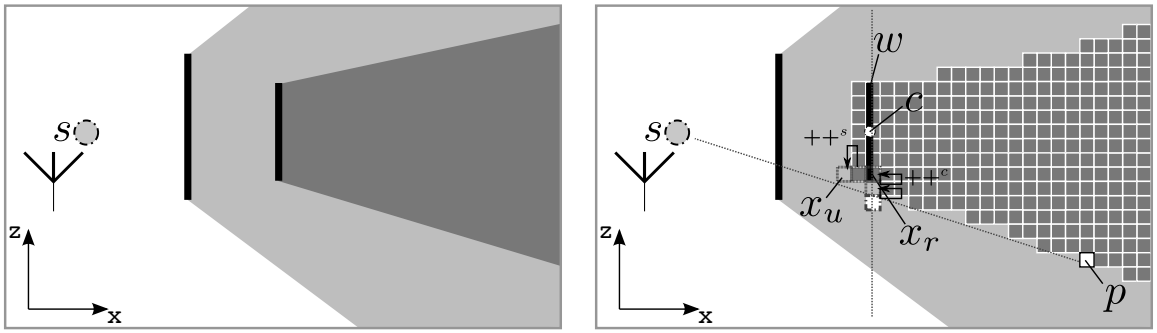
$$x_r = \mathrm{intersect}(l_0, l_1). \tag{2.27}$$

The intersection point can directly be computed for instance by using determinants [20, 5] whereas the height is given by the corresponding wall. The geometric ray paths for each beam $\mathrm{HDB}(s \to w)$ are then

$$\Upsilon^{\rightsquigarrow}(s, w) = \{v_0^{\rightsquigarrow}, \ldots, v_N^{\rightsquigarrow}\} = \{s \rightsquigarrow x_r \rightsquigarrow r \mid r \in \mathrm{HDB}(s \to w)\}. \tag{2.28}$$

Some care has to be taken when ray paths are concatenated due to diffraction over multiple rooftops. If two consecutive points of a ray path are in LOS to each other, all points in between must be removed from the final path, cf. Figure 2.13. This can be achieved by computing the upper convex hull of all points on a ray path. Further implementation details are discussed in the following paragraph.

Algorithm Detail An effective parallelization of the above method requires some non trivial steps which we will describe now in more detail. We parallelize our implementation in two ways, over the receiver points where ray paths may emerge, and in particular, over ray path recursion levels. First we explain the principles of our algorithm, briefly.



(a) (b)

Figure 2.9.: Illustration of roof diffraction beams. (a) The figure depicts two nested roof diffraction beams from a top view. (b) Due to discretization artifacts, we have to correct the geometric deflection point x_r to x_u by first advancing towards c, the center of the diffracting wall and then towards s, the original source, until we reach a new diffraction cone.

For each point in the receiver plane we determine which is the last diffracting wall on the ray path to the transmitter, i.e. which (if any) wall is the first to block the LOS condition. This information is written into the *Horizontal Diffraction Wall Map* (HDWM) and used to calculate the last diffraction point for each receiver point, i.e., the intersection of the ray path and the last diffracting wall. Iterating this procedure until LOS is reached will generate a sequence of points containing all possible diffraction points.

The HDWM is essentially the discrete representation of all horizontal diffraction beams (2.26) with an inverse lookup function: a query at a receiver point will give its diffracting wall (cf. Figure 2.10 bottom right). A pseudo code implementation is given in Algorithm 9. We let the transmitter "drop" from an infinite height down to its original height. For each height of the transmitter we record the NLOS fragments of the virtual diffraction sources in the color buffer. The fragments are encoded with each wall's unique ID. We use the depth buffer to ensure, that only the very first shadow casting wall is recorded. Thus, IDs are only written to the HDWM once: when a receiver points enters NLOS for the first time. Figure 2.10 illustrates the steps of the dropping algorithm and Figure 2.11 depicts a visualization of an HDWM for a small example scenario where each color encodes a discrete horizontal diffraction beam.

Horizontal diffraction ray paths can now be constructed in reverse order, starting at the receiver location, cf. Algorithm 10 and Figure 2.12. The preceding deflection x_r point is given by the intersection of the diffracting wall segment (from the HDWM) with the straight connection between the receiver and transmitter. The height of the deflection point must be corrected to the corresponding wall height. In this step, some effort is necessary to cope with discretization artifacts, cf. Algorithm 11: x_r may intersect the wall line but must not necessarily lie within the wall endpoints (i.e. on the line segment).

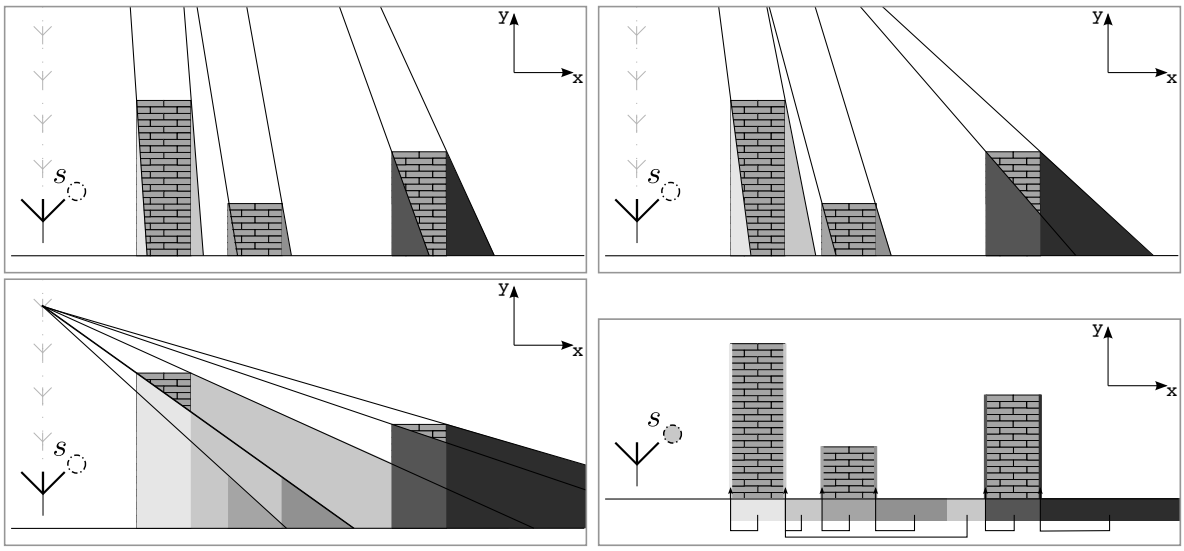


Figure 2.10.: The HDWM is constructed by "dropping" the transmitter from an infinite height to its original height. Unique wall IDs are recorded in the color buffer only when a receiver points enters NLOS for the very fist time. The last picture in the series shows the resulting HDWM with pointers for each region to its diffraction source.

In this case we proceed by traversing receiver pixels towards the radiation source (with the Bresenham [29] line drawing algorithm). The point for further diffraction lookups x_u is found similarly in order to make sure that the ray path progresses towards the radiation source, see Figure 2.9(b). As mentioned earlier, if two consecutive points on a concatenated ray path are in LOS to one another, all points in between must be removed, see Figure 2.13. We use Andrew's Monotone Chain algorithm [4] to construct the final convex ray path. As the points on the path are already sorted, this step can be skipped here.

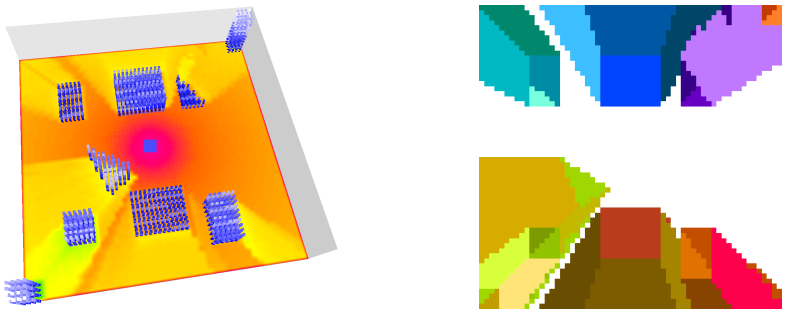


Figure 2.11.: Visualization of an HDWM (right) for a small example scenario (left). Each color encodes a discrete horizontal diffraction beam.

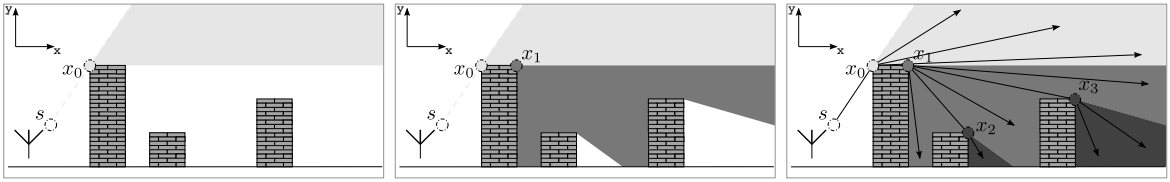


Figure 2.12.: Concatenation of multiple horizontal diffraction beams.

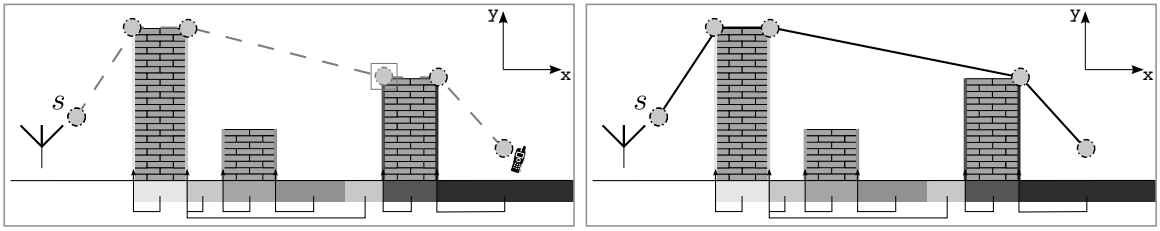


Figure 2.13.: The convex ray path is constructed by successively looking up points in the HDWM and removing intermediate points that are in LOS to one another.

Algorithm 9 HORIZONTAL_DIFFRACTION_WALL_MAP(Vertex *source*, WallSet $\mathcal{W}$)

Note: Constructs the HDWM by letting the transmitter "drop" from an infinite height.

```
CLEAR_BUFFER(color_buffer,0)
h ← h_0 {assign initial height}
while h ≥ source.y do
  for all w ∈ W do
    SET_COLOR(w.id) {color identifies diffracting edge }
    s' ← Vertex(source.x,source.y,h) {create virtual diffraction source at height h}
    NON_LINE_OF_SIGHT(s', w.p_2, w.p_3) {record NLOS fragments in color buffer}
  end for
  h ← (h − Δh) {decrease virtual source height}
end while
return color_buffer {color buffer contains HDWM}
```

Algorithm 10 HORIZONTAL_DIFFRACTION_RAYPATH(Vertex $source, receiver$)

Note: Ray paths are constructed in reverse order.
$\upsilon^{\rightsquigarrow} \leftarrow receiver$ {ray path ends at receiver location}
$p \leftarrow receiver$
buf_NLOS $\leftarrow$ NLOS_BEAM($source, \mathcal{W}$) {get NLOS from $source$ }
while buf_NLOS[p] $\neq$ LOS {as long as ray path has not yet reached line-of-sight} **do**
$(x_r, x_u) \leftarrow$ HORIZONTAL_DIFFRACTION_PREDECESSOR(p) {retrieve deflection and lookup location for p}
$\upsilon^{\rightsquigarrow} \leftarrow x_r \rightsquigarrow \upsilon^{\rightsquigarrow}$ {prepend to ray path }
$p \leftarrow x_u$
end while
$\upsilon^{\rightsquigarrow} \leftarrow source \rightsquigarrow \upsilon^{\rightsquigarrow}$ {ray path always starts at source }
$\upsilon^{\rightsquigarrow} \leftarrow$ CONVEX_HULL($\upsilon^{\rightsquigarrow}$) {remove deflection points that are not part of the convex hull}
return $\upsilon^{\rightsquigarrow}$

Algorithm 11 HORIZONTAL_DIFFRACTION_PREDECESSOR(Vertex p)

Note: Predecessor computes two points which may but must not coincide: the deflection point on the ray path x_r and the point for further diffraction lookups x_u
$w \leftarrow$ HDWM[p] {look up diffracting edge}
$l_0 \leftarrow (w.p_2, w.p_3)$ {line of diffracting edge}
$l_1 \leftarrow (source, p)$ {line of direct path }
$x_r \leftarrow$ INTERSECT(l_0,l_1)
while x_r not on w {account for discretization errors} **do**
p++ {shift by one voxel towards source}
$w \leftarrow$ HDWM[p] {additional lookup}
$l_1 \leftarrow (source, p)$ {line of direct path }
$x_r \leftarrow$ INTERSECT(l_0,l_1)
end while
$x_u \leftarrow x_r$
while HDWM[x_u] == w {do not stay on the same diffracting edge} **do**
x_u++ {shift by one voxel towards source}
end while
return (x_r, x_u)

2.4.2. Path Loss Calculation

This section describes how ray paths are mapped to the received signal strength. Every received signal experiences a basic propagation attenuation due to frequency dependent losses

$$c_f = 10 \log_{10} \left(\left(\frac{4\pi f}{c} \right)^2 \right) \tag{2.29}$$

with signal frequency f and speed of light c.

We could also incorporate an (unknown) dampening coefficient into the frequency dependent loss to account for additional losses due to vegetation or weather. Let $f' = f \cdot \rho$ for a dampening coefficient ρ, with the corresponding loss

$$c_{f'} = 10 \log_{10} \left(\left(\frac{4\pi f \cdot \rho}{c} \right)^2 \right) = 10 \log_{10} \left(\left(\frac{4\pi f}{c} \right)^2 \right) + 20 \log 10 \left(\rho \right). \tag{2.30}$$

We introduce an additional term c_0 as base loss with $c_0 = 20 \log 10 \left(\rho \right)$ and can write

$$c_{f'} = c_f + c_0. \tag{2.31}$$

The signal attenuation due to distance d between sender and receiver for a path loss coefficient γ can be written as

$$PL^{\mathrm{dB}}_{\mathrm{dist}} \left(d \right) = 10 \log_{10} \left(d^{\gamma} \right). \tag{2.32}$$

We have a clear unobstructed LOS path if a ray path is of the form

$$\upsilon^{\rightsquigarrow} = s \rightsquigarrow x \tag{2.33}$$

for a radiation source s and a receiver location x. We can write the corresponding path loss in dB as

$$PL^{\mathrm{dB}}_{\mathrm{LOS}} \left(\upsilon^{\rightsquigarrow} \right) = c_f + PL^{\mathrm{dB}}_{\mathrm{dist}} \left(\parallel x - s \parallel_2 \right). \tag{2.34}$$

The path loss prediction in regions with no direct LOS is known to be more complex. Consider a ray path $\upsilon^{\rightsquigarrow}$ given as

$$\upsilon^{\rightsquigarrow} = x_0 \rightsquigarrow \ldots \rightsquigarrow x_{i-1} \rightsquigarrow x_i \rightsquigarrow x_{i+1} \rightsquigarrow \ldots \rightsquigarrow x_N \tag{2.35}$$

for a radiation source x_0 and a receiver x_N.

The change of direction $\alpha_{\upsilon^{\rightsquigarrow}}^{(i)}$ according to the deflection of the wave front at x_i is given by

$$\alpha_{\upsilon^{\rightsquigarrow}}^{(i)} = \operatorname{acos}\left(\langle \parallel x_i - x_{i-1} \parallel_2, \parallel x_{i+1} - x_i \parallel_2 \rangle\right). \tag{2.36}$$

Let α be the change of direction of the corresponding deflection, based on the Taylor series [6] we approximate the (unknown) angle dependent attenuation by a polynomial of degree 2

$$g(\alpha) = a_0 + a_1\alpha + a_2\alpha^2. \tag{2.37}$$

We retrieve the unknown model coefficients (a_0, a_1, a_2) by a calibration to real-world measurements, thus modeling the stochastic influence of traffic and vegetation. The calibration can be realized by a formulation as a constraint least-square problem. The optimal parameter vector can then be calculated by common solver algorithms like Gauss-Newton or Levenberg-Marquardt (see [90]). We will come back to the parameter optimization in Section 2.4.3 when discussing multi-path effects.

Different types of wave guiding effects are taken into account by introducing distinct attenuation functions. The attenuation functions due to a single diffraction over rooftops and into street canyons are denoted by $g(\alpha)$ and $h(\beta)$ with deflection angles α and β, respectively.

The logarithmic attenuation for a rooftop diffraction with N_r deflection points and corresponding deflection angles $\alpha_{\upsilon^{\rightsquigarrow}}^{(i)}$ is described by the sum of each single attenuation

$$PL_{\text{roof}}^{\text{dB}}(\upsilon^{\rightsquigarrow}) = \sum_{i=1}^{N_r} g\left(\alpha_{\upsilon^{\rightsquigarrow}}^{(i)}\right). \tag{2.38}$$

The same holds for a series of diffractions along street canyons

$$PL_{\text{street}}^{\text{dB}}(\upsilon^{\rightsquigarrow}) = \sum_{j=1}^{N_s} h\left(\beta_{\upsilon^{\rightsquigarrow}}^{(j)}\right) \tag{2.39}$$

with N_s deflection points and corresponding angles $\beta_{\upsilon^{\rightsquigarrow}}^{(j)}$.

For the attenuation due to wall transmissions $PL_{\text{wall}}^{\text{dB}}(N_w)$ we use a model similar to the Multi-Wall model by [91] which depends only on the number of penetrated walls N_w of a ray path $\upsilon^{\rightsquigarrow}$

$$PL_{\text{wall}}^{\text{dB}}(\upsilon^{\rightsquigarrow}) = L_w * N_w \tag{2.40}$$

where L_w is a material constant describing the propagation loss per wall penetration. This can be seen as an approximation to the rooftop diffraction where the deflection angles are considered constant.

The overall attenuation of a receiver location r is hence the sum of the arriving ray paths $\Upsilon^{\rightsquigarrow} = \upsilon_0^{\rightsquigarrow}, \ldots, \upsilon_N^{\rightsquigarrow}$, we write informally

$$PL_{\text{all}}^{\text{dB}}\left(\Upsilon^{\rightsquigarrow}\right) = c_f + \sum_{\upsilon^{\rightsquigarrow} \in \Upsilon^{\rightsquigarrow}} \begin{cases} PL_{\text{dist}}^{\text{dB}}\left(\parallel x - s \parallel_2\right), & \upsilon^{\rightsquigarrow} \text{ of the form } s \rightsquigarrow x \\ PL_{\text{street}}^{\text{dB}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ due to diff. into street} \\ PL_{\text{roof}}^{\text{dB}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ due to diff. over roof} \\ PL_{\text{wall}}^{\text{dB}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ due to wall transmissions} \end{cases} . \quad (2.41)$$

From this general model we derive two specific models that rely on a subset of ray path calculations. We will show in Section 2.6.5 that these derivations will in practice provide a good compromise between computation time, numerical stability and prediction accuracy.

2.4.2.1. Roof Diffraction Model

In this section we introduce the *Roof Diffraction Model* (RDM) for urban environments. We assume that rays propagate in a straight line from the transmitter and may be diffracted downwards at the roof of buildings. The path loss for a ray path $\upsilon^{\rightsquigarrow}$ is modeled by

$$PL_{\text{RDM}}^{\text{dB}}\left(\upsilon^{\rightsquigarrow}\right) = c_f + c_0 + PL_{\text{dist}}^{\text{dB}}\left(d_{\upsilon^{\rightsquigarrow}}\right) + PL_{\text{roof}}^{\text{dB}}\left(\upsilon^{\rightsquigarrow}\right) \quad (2.42)$$

which yields

$$PL_{\text{RDM}}^{\text{dB}}\left(\upsilon^{\rightsquigarrow}\right) = c_f + c_0 + \gamma \cdot 10 \cdot \lg\left(d_{\upsilon^{\rightsquigarrow}}\right) + \sum_{i=1}^{N_r} g\left(\alpha_{\upsilon^{\rightsquigarrow}}^{(i)}\right). \quad (2.43)$$

The second term depends on the path loss exponent γ and the length of the diffracted path $d_{\upsilon^{\rightsquigarrow}} = \parallel x_N - x_0 \parallel_2$. N_r denotes the number of roof diffractions and $\alpha_{\upsilon^{\rightsquigarrow}}^{(i)}$ i-th diffraction angle. The function

$$g\left(\alpha\right) = a_0 + a_1\alpha + a_2\alpha^2, \quad \alpha \in \left[0, \frac{\pi}{2}\right] \quad (2.44)$$

with parameters $a_0, a_1, a_2 \in \mathbb{R}$ models the attenuation due to a diffraction angle α.

Adequate values for the parameter γ and the coefficients a_0, a_1, a_2 are retrieved from a calibration with measurement data, cf. Section 2.4.3 and Section 2.6.5.1. Route METRO202 of the COST-Munich scenario yields

$$\gamma = 2.76; \quad a_0 = 3.37; \quad a_1 = 0; \quad a_2 = 4.9.$$

The above values are used in this work when calculating the path loss for the RDM.

2.4.2.2. Edge Diffraction Model

We now present a different specialization of the general model (2.41), which we will refer to as the *Edge Diffraction Model* (EDM). Again, we assume that rays propagate in a straight line from the transmitter. Incoming wave fronts may be diffracted at vertical building edges which practically model diffraction into street canyons. We model the diffraction of multiple rooftops by counting the number of diffractions but, unlike the RDM, we do not consider the angle of each diffraction. This assumption is based on an informal observation that the total sum of deflection angles is more or less the same for most propagation paths and therefore can be treated as an empirical constant of the propagation environment that can be retrieved by parameter calibration. This claim is supported by the higher stability and thereby accuracy of the EDM over the RDM as presented in Section 2.6.

The path loss for a ray path $\upsilon^{\rightsquigarrow}$ is modeled by

$$PL^{\mathrm{dB}}_{\mathrm{EDM}}\left(\upsilon^{\rightsquigarrow}\right) = c_f + c_0 + \begin{cases} PL^{\mathrm{dB}}_{\mathrm{LOS}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ in LOS} \\ PL^{\mathrm{dB}}_{\mathrm{LOS}}\left(\upsilon^{\rightsquigarrow}\right) + PL^{\mathrm{dB}}_{\mathrm{VD}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ due to diff. at vertical edge} \\ PL^{\mathrm{dB}}_{\mathrm{HD}}\left(\upsilon^{\rightsquigarrow}\right), & \text{otherwise} \end{cases} \tag{2.45}$$

with

$$PL^{\mathrm{dB}}_{\mathrm{LOS}}\left(\upsilon^{\rightsquigarrow}\right) = \gamma_{\mathrm{LOS}} \cdot 10 \cdot \log_{10}(d_{\upsilon^{\rightsquigarrow}}) \tag{2.46}$$

$$PL^{\mathrm{dB}}_{\mathrm{VD}}\left(\upsilon^{\rightsquigarrow}\right) = \sum_{i=1}^{N_s} h\left(\alpha_r^{(i)}\right) \tag{2.47}$$

$$PL^{\mathrm{dB}}_{\mathrm{HD}}\left(\upsilon^{\rightsquigarrow}\right) = \gamma_{\mathrm{NLOS}} \cdot 10 \cdot \log_{10}(d_r) + W_r \cdot L_w \tag{2.48}$$

and the frequency dependent term c_f as above. Again, $d_{\upsilon^{\rightsquigarrow}} = \| x_N - x_0 \|_2$ denotes the path length between receiver and transmitter, and $\gamma_{\mathrm{LOS}}, \gamma_{\mathrm{NLOS}}$ the corresponding path loss exponents. N_s is the number of edge diffractions and $\alpha_r^{(i)}$ the i-th diffraction angle. The function

$$h\left(\alpha\right) = b_0 + b_1\alpha + b_2\alpha^2, \quad \alpha \in [0, \pi],$$

with parameters $b_0, b_1, b_2 \in \mathbb{R}$ models the attenuation due to a diffraction angle α. As above, N_w is the number of walls in the direct path from the transmitter to the receiver and L_w the loss per obstructing wall.

Adequate values for the parameters are usually obtained from a calibration with mea-

surement data. Route METRO202 of the COST-Munich scenario yields

$$\begin{aligned} \gamma_{\text{LOS}} &= 2.6007; \quad \gamma_{\text{NLOS}} = 3.4330; \quad L_w = 0.2522; \\ b_0 &= 1.4562; \quad b_1 = 0.4022; \quad b_2 = -0.0022. \end{aligned}$$

These values are used in this work when calculating the path loss for the EDM.

2.4.3. Model Parameter Calibration

The modeling of urban propagation environments often consists of polygonal building outlines with one height value per building, the so-called 2.5 dimensional description. Other influencing factors like building material, roof style, texture or vegetation are typically not included in the description of the urban models since the acquisition of this information is either very expensive or sometimes simply not available. However, a city with modern skyscrapers that consist predominantly of glass fronts and flat roof tops will certainly exhibit a different attenuation behavior than a small town with pitched roofs and fronts made of concrete [94, 131, 134].

In literature, it is therefore quite common to adapt propagation models to different types of environments. A qualitative description of a propagation environment would for instance consist of a classification into rural or urban, or by building density and street widths. However, such coarse grain classification usually reflects typical propagation characteristics very poorly. Therefore, we use an implicit description by adapting model parameters to different environments by calibration from real-world measurements. Thus, we model unknown components of the propagation environment like traffic or vegetation by introducing variable coefficients (model parameters) into our path loss calculation.

Since the logarithm does not change the basic behavior of a function and therefore preserves the minimum, we choose to optimize the logarithm of the path loss formula (2.41). Hence, the product in the numerator is transformed to a sum which can be expressed easily in matrix notation. The linear system is set up as:

$$C\frac{1}{d^{\gamma}}\prod_{i=1}^{N} f_i \rightarrow 1\log C - \gamma \log d + \sum_{i=1}^{N} \log f_i. \tag{2.49}$$

We can then formulate the adaption of model parameters as a constrained least-squares problem in order to minimize the mean squared error between predicted and measured

data:

$$\min_x ||F(x)||_2^2 = \min_x \sum_i F_i^2(x) \tag{2.50}$$

such that

$$\begin{aligned} F(x) &= M \cdot x - d && (2.51)\\ A \cdot x &\leq b && (2.52)\\ B \cdot x &= c. && (2.53) \end{aligned}$$

Each row of the matrix M corresponds to one measurement location, whereas the columns are formed by the arriving ray paths that reach the respective location, like travel distance of each arriving path and number of deflections. The vector d contains the measured path loss at each location. Hence, the optimal parameter vector $\hat{x}$ minimizes the mean squared error between the predicted and measured path loss with respect to the constraints (2.52) and additionally satisfying the equality constraints (2.53). Such constraints can incorporate expert knowledge on the propagation phenomena into the optimization problem, e.g., the path loss coefficient γ is known to be in the range between two for free space propagation and five inside densely populated cities. The optimal $\hat{x}$ can then be calculated by common solver algorithms like Gauss-Newton or Levenberg-Marquardt [90]. Section 2.6.5.1 shows a complete example of this procedure for the RDM model that works on all models with a single (strongest) path.

To the best of our knowledge, previous approaches to the adaption of model parameters from ray paths like the one of Mathar et. al [94] have been restricted to the optimization of the strongest ray paths, only. First, they adapt their model parameters to best match their least attenuated ray path at each receiver location based on an initial parameter vector. Then they cyclically iterate between the computation of strongest ray paths and corresponding parameter estimation because changes in the model parameters can result in different strongest paths in the next iteration. They fail to provide a strict proof for convergence of their alternating approach, however they claim to achieve good results in practice after two or three iterations. Obviously, this procedure heavily depends on the initial parameter vector and is not guaranteed to find the global optimum because the optimization algorithm has no access to the information of all incoming ray paths.

Multi-Path Models We propose an alternative approach that does not depend on a sequence of path computations and parameter estimations. In particular, we directly incorporate all incoming paths into the parameter estimation by binary encoding all

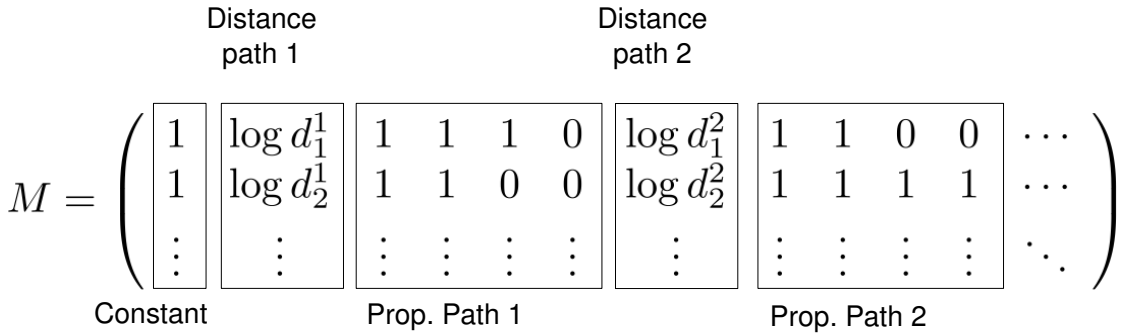


Figure 2.14.: Each row of the optimization matrix corresponds to one measurement location. Each column is formed by the travel distance and number of deflections of arriving ray paths at the respective location.

existing paths. Hence, we can achieve a closed form for the optimization algorithm and thereby inheriting all properties of the optimization procedure at hand.

The main idea is to incorporate the binary encoding in the optimization matrix M (cf. Fig. 2.14). For ease of understanding we describe the procedure only for the horizontal diffraction. However, the concept is of course not limited to propagation paths based on horizontal diffraction only, but can easily be extended to support paths that are based on other propagation phenomena.

Let R be the maximum recursion level of the diffraction and N the maximum number of arriving paths. Each row of M is then of the form (without the leading constant)

$$\left(\log d_1 \quad \delta_{1,1} \dots \delta_{1,R} \quad \dots \quad \log d_N \quad \delta_{N,1} \dots \delta_{N,R} \right) \tag{2.54}$$

where $\delta_{i,j}$ is a binary encoding such that

$$\delta_{i,j} = \begin{cases} 1 & \text{, if path } i \text{ has } j \text{ or more diffractions} \\ 0 & \text{, otherwise} \end{cases} . \tag{2.55}$$

Hence, we use the $\delta_{i,j}$'s to switch certain parts of the matrix M on or off, so to speak. Thereby providing the required uniform input matrix for the optimization algorithms while still supporting a varying number of arriving paths for each receiver location. An application of our approach can be found in [131, 132] and [134].

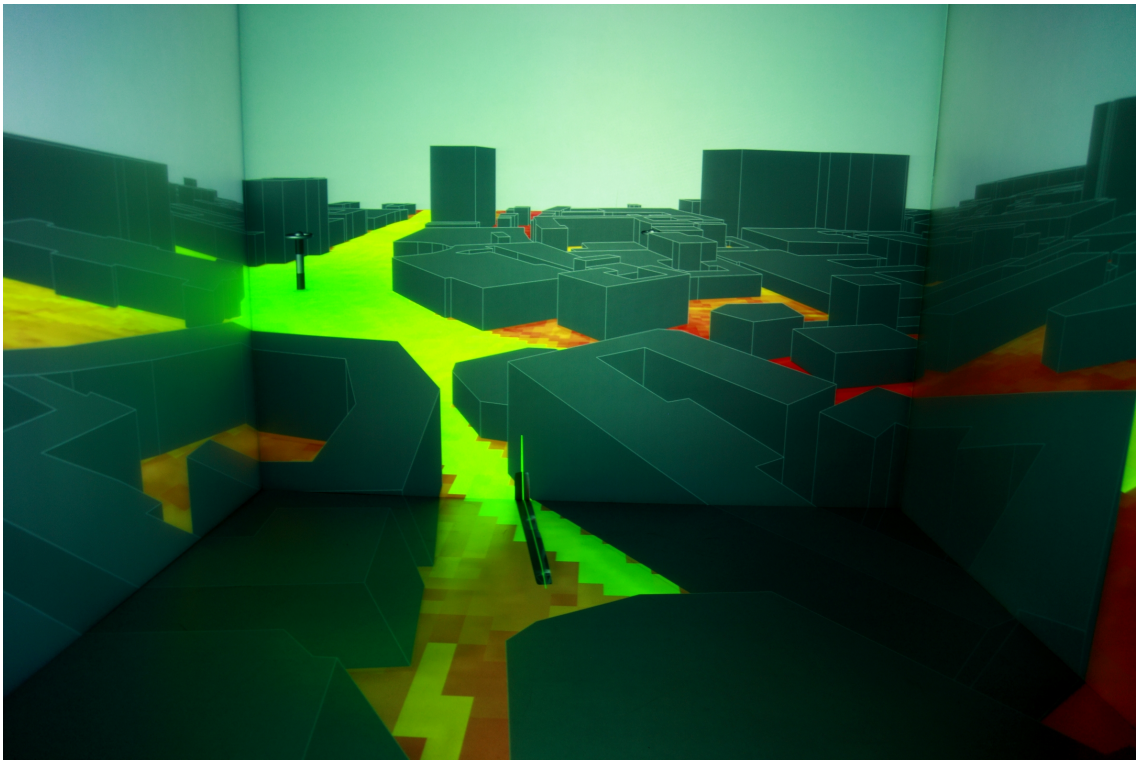


Figure 2.15.: Propagation prediction in the virtual CAVE environment.

2.5. Virtual Reality Interface

Dense urban areas demand for a great emphasis on site-specific details in the propagation environment which are often not covered in automatic approaches. City models may not be as up-to-date as recent satellite images of the same area and common issues are often incomplete or missing information in the geographical databases that serve as input to the propagation simulations. There may be critical sites or clinical areas (e.g. hospitals) where exposure to certain power levels may be hazardous to medical equipment and a minimum safe distance for radio transmitter sites has to be maintained. Areas along country borders are also often subject to restrictions in order to reduce interference with foreign signals.

With our real-time propagation simulation (cf. Section 2.4) we can now design an application that offers an interactive manipulation and modeling of the propagation environment that is directly coupled to the simulation. We present a prototype interface that integrates real-time simulations of propagation predictions with the interactive exploration and analysis in a VE. All user input and manipulation from within the VE is directly communicated to the simulation algorithm which immediately updates all propagation predictions such that the effect is instantly visible to the user. Parts of this work have been published in [123].

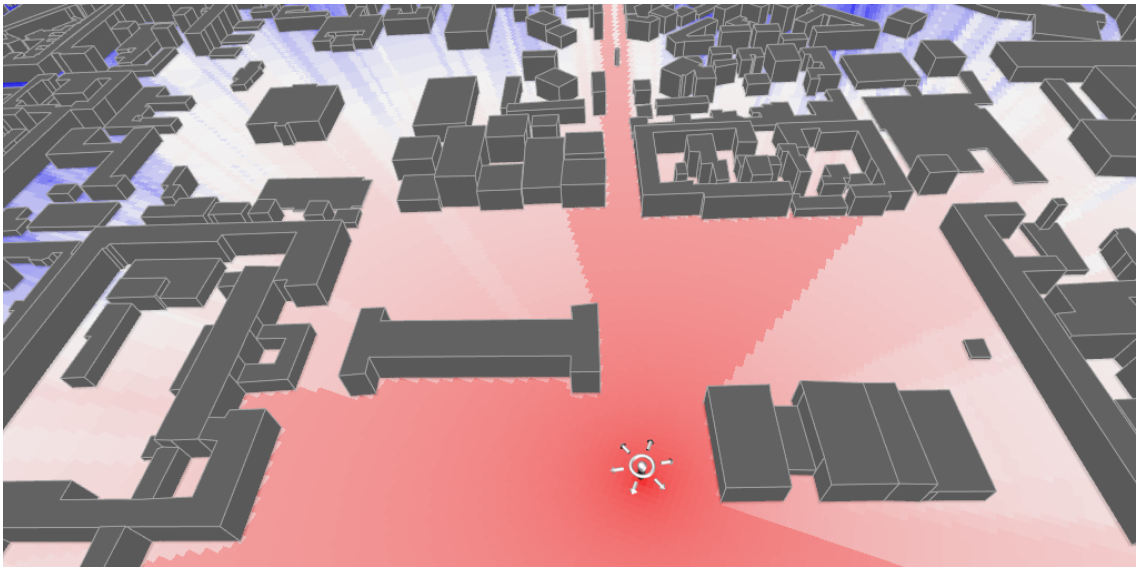


Figure 2.16.: Simulation of radio wave propagation with visualization of mean received field strength, colors ranging from red (strong) to white (medium) and blue (weak).

2.5.1. Overview

We formulate the conceptual requirements of our VR application based on informal discussions with domain experts as follows: (1) Visualization of the radio network within a geographical reference frame, (2) Modification and setup of transmitter sites, (3) In situ simulation of propagation effects and (4) Manipulation of site-specific details.

Furthermore, a fundamental prerequisite is the real-time requirement of the overall system in order to provide an interactive VR experience. We derive the main tasks from the above formulated requirements: (1) Adjustment of visualization parameters, (2) Configuration of transmitter sites and (3) Manipulation of the city model. We coupled our VR application directly to the simulation such that every user input triggers a real-time update of the simulation input parameters and thereby a subsequent computation of the propagation predictions.

2.5.1.1. Geographical Reference & Visualization

Geographical reference is usually provided by a specification of the supplying area in a global coordinate system. Coordinates are often given in latitudes and longitudes according to the World Geodetic System (latest revision: WGS84) or as grid based horizontal and vertical positions according to the Universal Transverse Mercator (UTM) system.

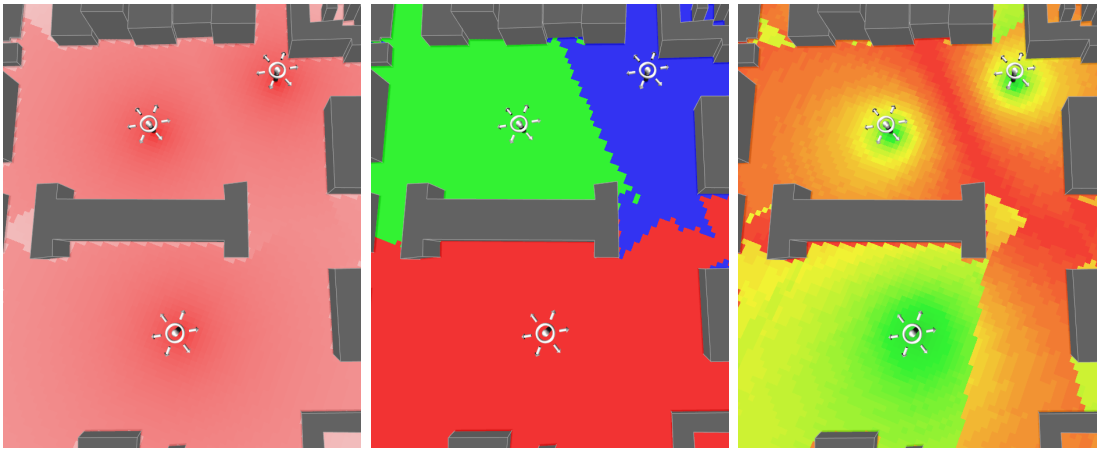


Figure 2.17.: The relationship of multiple antennas are displayed as network properties that reveal different statistics. The left image shows the coverage of field strength whereas the middle picture indicates from which antenna the strongest signal arrives. The right image depicts the ratio of interference from other signals, green stands for low interference, red for high interference. For good reception high coverage must be combined with low interference.

Context is provided in form of satellite images and a city model (cf. Figure 2.2) which for instance can be acquired from LIDAR data as demonstrated for the creation of virtual cities in [113]. Most of the time, satellite images are up-to-date and very accurate, whereas city models are usually very expensive and of rather coarse resolution (two to five meters). Often, we observe a gap between the information of recent satellite images and the corresponding building data of the city model. Building information may be incomplete or do not reflect the latest building development. Sometimes recent buildings are missing completely. We address this issue in Section 2.5.1.3.

As illustrated in Figure 2.16 transmitter sites are visualized on top of the geographical data and may be subject to further manipulation. The simulation results in terms of predicted signal strength or interference are displayed as a colored pixel image parallel to the ground plane. Initially, it depicts the propagation effects at 1.5 meters above ground which is common for analyzing cell phone reception.

2.5.1.2. Simulation of Propagation Effects

Ray tracing approaches are an established technique for radio wave propagation simulations, however, such approaches need to be extended to include diffraction, which is a predominant effect for common mobile radio frequencies. Diffraction along edges is usually modeled by shooting a multitude of rays into the shadow cone of the diffracting edge which usually results in a large computational overhead.

The key idea for implementing diffraction on the GPU is to utilize the concept of shadow volumes to mark regions which are in shadow. For the propagation of radio waves, only those propagation paths are of interest which pass through a certain height level above ground where cell phone reception is required. By applying a modified shadow volume technique, all pixels that are inside a diffraction cone are identified on an image plane, which is setup such as to correspond to the receiver plane. Hence, a GPU algorithm can implement the problem of finding diffraction rays as repeated shadow computations, which can be done extremely fast on recent graphics cards. Specifics about the algorithm and details of our GPU implementation can be found in Section 2.4.

Here, we want to discuss what our VR prototype has to know about the propagation algorithm, hence what is the input and the output of the radio wave simulation that the system and thereby the user will be aware of. We start with the data requirements of the simulation which basically is just a simplified city model. Shape of rooftops are usually omitted in common propagation algorithms. Thus, a building is described by its polygonal outline and one height value. The propagation loss is calculated as a function of city model, radiation parameters and location of the transmitting antenna. If multiple antennas are present in a network, the simulation has to be done separately for each given transmitter location. The simulation result is a simple pixel map where each pixel corresponds to a grid location within the supplying area that describes the mean signal attenuation.

Changes to a particular antenna lead to a complete recalculation of network properties. Currently, we provide the following statistics for network analysis: (1) Maximum intensity (MI), (2) Best server (BS) and (3) Carrier-to-Interference (CI). The MI view displays the attenuation of the strongest received signal strength among all antennas at each location. For the BS view each antenna is assigned a unique color and each receiving location is colored with the color of the strongest antenna, it provides a so-called sector view. The CI view shows the signal to interference ratio, hence the ratio between the strongest and all other combined signals. The CI value is a major indicator of network capacity in interference limited networks which basically are all current (3G and 4G) standards for mobile telecommunications. Figure 2.17 gives an impression of the three statistic views.

2.5.1.3. Manipulation of Site Specific Details

We will first describe the manipulation options for transmitter sites and then those for the city model.

Transmitter Sites An initial network setup with the description of transmitter site locations can for instance be computed by automatic cell planning algorithms or can

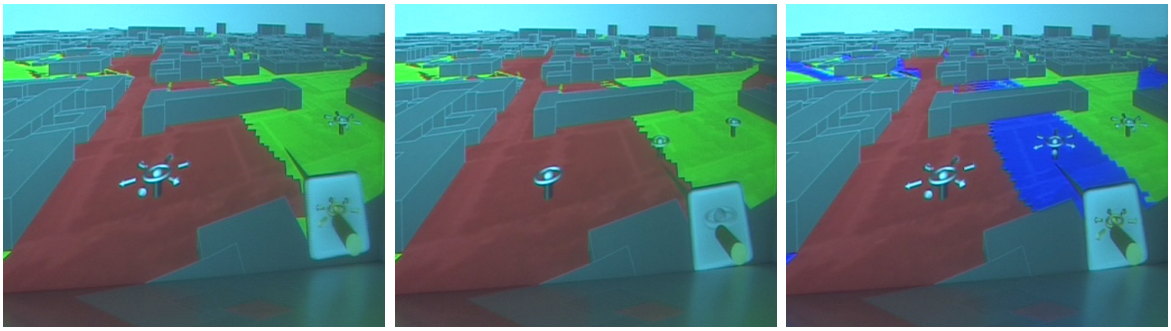


Figure 2.18.: The creation of a new transmitter site is depicted in the sector view where the receiver plane is colored according to the base station with maximum strength. When the mode for adding a transmitter site is entered, a wireframe model of a transmitter is rendered at the target location (middle image). The antenna is added to network by a button press and the propagation simulation immediately updates the sector view (blue for the new antenna in the right image).

simply be a hexagonal layout where antennas are evenly distributed over the supplying area. In our prototype application the user now can change at run time the location of antenna sites. If required, additional sites can be added to the network by pointing with a hand-held 6DOF device at the desired position and pressing the corresponding button. Figure 2.18 depicts the creation of a new transmitter site within the VR prototype. During all antenna operations, visual feedback and simulation updates are performed instantly in real-time. This is achieved by using a simulation algorithm which is implemented entirely on the GPU as introduced in Section 2.4.

City Model To account for missing or incorrect information in the city model database, we let the user manipulate the city model directly in the simulation environment. Sketching has been an established method for generating content on-the-fly, a recent example is content authoring in AR games as described in [63]. Since satellite images are already provided as geographical context information we let the user correct or create missing buildings by drawing their footprint on top of the satellite image, see Figure 2.19. A 3D model is created automatically be extruding the floor plan along the axis pointing upwards. For fine-tuning building heights or also to get in intuitive understanding of the influence on the radio wave propagation, the rooftop level of each building can be adjusted separately. All changes to the building database are directly communicated to the simulation algorithm which immediately updates all propagation predictions such that the effect is instantly visible to the user.

2.5.2. Realization

So far we have described the conceptual features of our prototype application from the view of the user. In this section we want to discuss issues that arose in the actual implementation of certain features and what challenges had to be faced to maintain interactive frame rates for simulation and manipulation tasks. We will focus on the differences and difficulties we faced when providing a virtual reality interfaces as opposed to common 2D desktop metaphors.

In order to realize the above described interaction we need the following basic building blocks: (1) navigation through the virtual environment, (2) selection of objects that can be subject to manipulation, (3) real-time simulation of propagation effects, (4) object manipulation based on 6DOF tracking data and (5) system control.

Setup The virtual reality toolkit that we use [7] is basically a master-slave system. Since our CAVE system consists of five walls in total each with passive stereo, it is operated by 10 rendering slaves, two for each wall and one for each eye per wall. The slaves are synchronized by a master node which handles all user inputs and tracking data, and distributes it to all rendering slaves. Therefore, in our usual setup every slave runs its own instance of the virtual reality application performing all computations on its own to minimize data transfer between nodes.

Navigation Since the user is equipped with a hand-held 6DOF device, navigation is achieved by a simple travel-by-pointing metaphor. Furthermore, the user can change the overall scale between being in the city and a world-in-miniature (WIM) view. This way, we provide an overview of the whole scene in the WIM and details can be provided on demand by standing inside the city model.

Selection We introduce selection as an extra paragraph here because the number of objects that the user can interact with proved to be a performance issue. Our test scenario of approximately 7 km^2 consists of 2086 distinct buildings with approximately 82,000 triangles in total. First tests showed heavy drops in performance when computing the intersections between selection ray and objects naively on every render node (which is default for the used VR toolkit). We then switched the computation to a designated node (the master) which would not do any renderings. We used an asynchronous connection that would broadcast the results from the selection process on a multicast address where every rendering node could listen and react accordingly. A refresh rate of 30 Hz to update object selection did not seem to introduce a noticeable lag to the system. By decoupling the selection process from the main body of the application and using a asynchronous

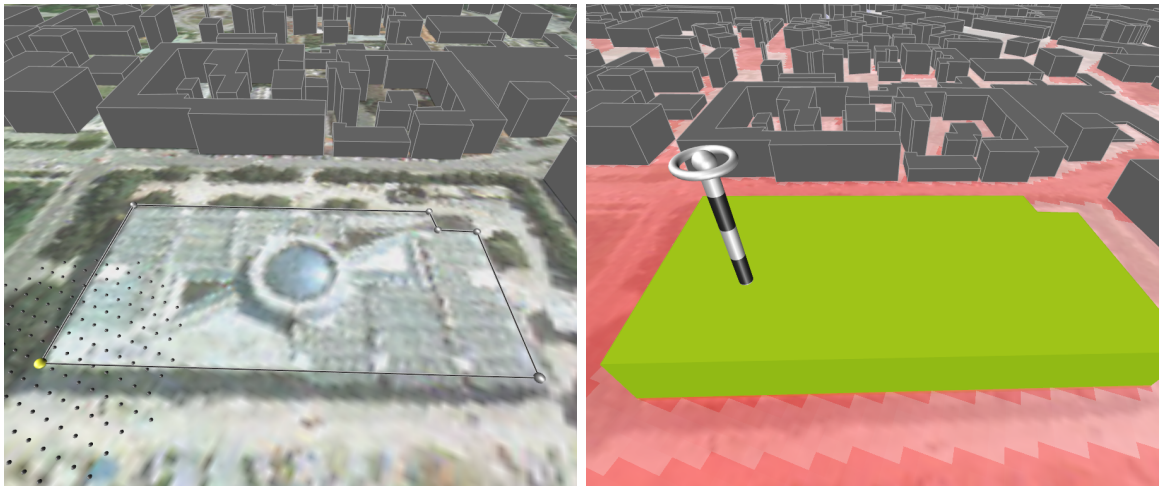


Figure 2.19.: A missing building is sketched directly on the satellite image. A grid raster is displayed around the cursor to assist the user in case of jitter. The effect of the new building becomes instantly visible in the simulation results.

multicast broadcast for scattering the information we maintained the same frame rate as without selection with no significant increase in network traffic.

Simulation The technical details for achieving a real-time simulation of a single transmitter site are described in Section 2.4. Here, we focus on how to couple the simulation with a changing propagation environment. In order to offer a real-time manipulation we had to update the city model database directly on the GPU. For computation efficiency the city model was initially transformed into a vertex buffer object in GPU memory. Additional information was attached to the vertex geometry (e.g. material properties and computational flags). For the manipulation of existing buildings (change of roof top level) we introduced an additional dependent texture lookup in the vertex shader that would transform the building geometry. The texture lookup maps unique building identifiers to a height value. To minimize data transfers between GPU (device) and CPU (host) we kept identical copies of the lookup table in host and device memory. Upon changes to building heights it is sufficient to upload only the modified part of the lookup table to texture memory. A subsequent simulation of the propagation prediction will then automatically use the updated texture for the dependent lookup. The remainder of the simulation code was left unchanged. Newly created buildings were registered to the simulation by introducing new vertex buffer objects which were created dynamically at run time.

Manipulation The main challenge was how to benefit from direct 3D interaction, avoid jitter or tracking inaccuracies without imposing restrictions on user interaction. For all manipulation tasks, extremely small hand movements were compensated similar to

Figure 2.20.: User stands in the VE with a tracked marker attached to his non-dominant hand and a tracked pointing device in his dominant hand. His left hand is slightly raised to look at a panel for system control. The selection of the panel icon is done with the pointing device in his right hand. Upon completion the panel is hidden by lowering the non-dominant hand.

the PRISM [57] technique. Furthermore, we split the manipulation of transmitter sites into two sub tasks, changing site location and changing antenna height. This effectively reduced the 3D manipulation task into a two-dimensional (changing location) and a one-dimensional task (change height). Though this might seem like a restriction at first, it turned out to be much more precise and would naturally stick to the planning process: first find out where your antenna site should be located and then adjust the height of the antenna tower for maximum coverage and minimum interference.

Sketching a new building on the satellite image also suffered heavily from jitter in hand movements. We assisted in the creation process by depicting discrete grid points over the image and building edges would snap onto the grid. A new building is then created by first entering the edit mode, one button adds the current grid point to the building outline whereas another button allows to undo the last operation. A building is finished by placing the last building corner in the vicinity of the first corner. Building walls are then extruded to generate a 3D model and visualization and simulation databases are updated accordingly.

System control We needed an interface that makes the control of visualization and network parameters accessible directly from within the VE. Common approaches are the use of 3D menus or small hand-held computer devices (e.g. Tablet-PC). However, we felt not comfortable with either of them. The 3D menus would clutter the visual field of the VE and are sometimes hard to use due to jitter. An additional hand-held device would encumber the user and would require him to switch context between immersive 3D and a 2D screen. We tried to combine both approaches by tracking the non-dominant hand and attaching a virtual screen to the hand position as soon as the user raises the hand above his waist. This idea is inspired from the use of a mobile smart phone that the user takes out of his pocket. The virtual screen depicts icons for controlling the application which can be selected by pointing and clicking with the device in the dominant hand. Figure 2.20 shows a picture of a user in the VE with his left hand raised and selecting a network statistic with his right hand.

2.6. Results

We will first introduce our benchmark scenario and present an informal evaluation of the VR interface (cf. Section 2.5). We will then define our prediction quality criteria and discuss the performance and quality of the presented propagation models. First, we will examine the two empirical models (COST-Hata and COST-WI) and then the two proposed ray optical models (RDM and EDM). In particular, we will compare our approach with results from literature and commercial tools, at least as far as data is publicly available.

2.6.1. Benchmark Scenario

We use building and urban micro cell measurements of downtown Munich, Germany for evaluating our methods. This data has been created during the COST 231 action as described by [45] and is now publicly available at [93]. The scenario comprises an area of 2,400 × 3,400 m (7 km^2) of approximately 18,000 building walls. The database offers a resolution of 5 m, the standard deviation of building heights was 8.56 m and that of the underlying terrain about 3.87 m.

A path loss measurement at 947 MHz has been performed by the Mannesmann Mobilfunk GmbH. Three different measurement routes are available and referred to as METRO200 (970 points) , METRO201 (355 points) and METRO202 (1,031 points). The scenario along with the transmitter location is depicted in Figure 2.21. The transmitter height is 13 m and measurements were performed at 1.5 m above ground. The path loss has been stored as an approximately 10 m sector average.

All predictions were performed on the whole area of 7 km^2 with a resolution of 5 meter. This requires the evaluation of more than $3 \cdot 10^5$ receiver points. The path loss calculations for the ray optical models RDM, EDM and *Cube-Oriented-Ray-Launching-Algorithm* (CORLA) [94] are all based on a calibration to the measurement points along route METRO202.

2.6.2. Virtual Reality Interface

We tested the COST 231 Munich scenario in a CAVE virtual environment of five walls each with passive stereo (1600 × 1200 pixels) which is operated by 10 rendering slaves (one for each eye per wall) and a master node which handles user inputs, tracking, data distribution and synchronization. Each node is equipped with an NVIDIA Quadro FX

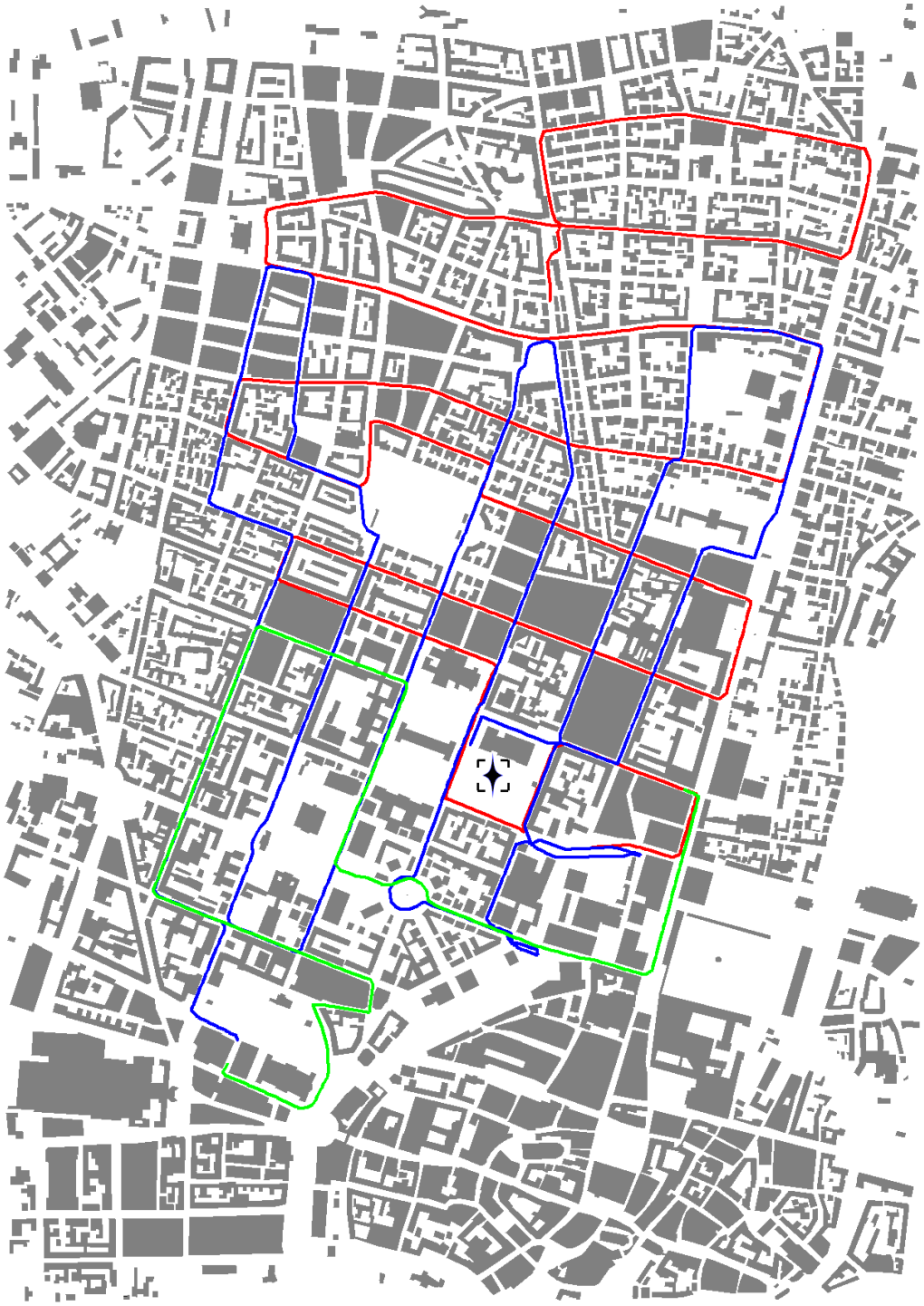


Figure 2.21.: This image shows the COST 231 scenario overview with buildings and measurement routes. The measurement routes METRO200, METRO201 and METRO202 are colored in red, green and blue, respectively. The transmitter location during measurements (roughly in the middle of the scenario) is indicated as a black star.

5600 graphics card. Latency of our infrared optical tracking system was between 80–120ms at an update rate of 60 Hz. The average frame rate did not drop below 40 frames per second for a network of eight transmitters and 2,086 buildings.

We performed a primary questioning of two domain experts (communication theory) regarding usability and features. They were first introduced by example in the basic features of the application. We asked them to create new antenna sites and adjust the position and height of the transmitters. We let them switch to different networks statistics and activate the various visualization settings (e.g. building wireframe vs. solid rendering, satellite image overlay on/off for geographic context) by using the control panel which was attached to a tracking device at their non-dominant hand. We also asked them to create an additional building that was outlined in the satellite image but was not available in the building database. After that, they were free to explore the radio wave simulation on their own.

Both found the potential of our application interesting. When asked for the worst feature they stated that they would like to have more control over the propagation simulation settings (e.g. antenna radiation pattern or power regulation). They liked the overview in the VE, especially the possibility to create missing buildings and stated that it would be great for debugging propagation algorithms. They assessed the overall application as good for finding weak spots in an initial planning phase.

2.6.3. Prediction Quality Criteria

We quantify the accuracy of our propagation predictions by the *Mean Error* (ME), the *Mean Squared Error* (MSE) and the *Standard Deviation* (STD) between prediction and measurement data. Let the number of measurement points be N, and r_i the ith measurement point. $M^{\mathrm{dB}}(r_i)$ denotes the measured and $PL^{\mathrm{dB}}(r_i)$ the predicted path loss at r_i.

We define the mean error as

$$\mathrm{ME} = \frac{1}{N}\sum_{i=1}^{N}\left[M^{\mathrm{dB}}(r_i) - PL^{\mathrm{dB}}(r_i)\right], \tag{2.56}$$

the mean squared error as

$$\mathrm{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[M^{\mathrm{dB}}(r_i) - PL^{\mathrm{dB}}(r_i)\right]^2} \tag{2.57}$$

and the standard deviation as

$$\mathrm{STD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left[M^{\mathrm{dB}}(r_i) - PL^{\mathrm{dB}}(r_i) - \mathrm{ME}\right]^2}. \tag{2.58}$$

2.6.4. (Semi-)Empirical Models

2.6.4.1. COST-Hata Model

As described in Section 2.3.1 the COST-Hata model treats the propagation environment as either urban, suburban or rural and is based on the log-distance model (2.1) with empirical correction terms. We compared the urban model

$$\begin{aligned} P^{\mathrm{dB}}_{\mathrm{urban}}(r) = {} & 33.9 \cdot \lg(f) + \left(44.9 - 6.55 \cdot \lg(\Delta h_{\mathcal{T}})\right) \cdot \lg(d_r) \\ & + g_{\text{COST-H}}(f, \Delta h_T, \Delta h_r) \end{aligned}$$

against the measurements in the COST 231 scenario, cf. Table 2.2. At first glance the COST-Hata model performs surprisingly well (mean STD of 8.4 dB) considering that building information are completely ignored and the computation time can be neglected. However, an exact point-to-point comparison (Figure 2.22) of prediction and measurement reveals the weakness of the model. The predicted values miss the actual attenuation sometimes by more than 10 dB. Only the basic shape of the measurement curve is reproduced.

Measurement Route	ME	MSE	STD
METRO200	−7.6 dB	12.2 dB	9.5 dB
METRO201	−2.1 dB	7 dB	6.7 dB
METRO202	−0.9 dB	8.9 dB	8.9 dB
All	-3.5 dB	9.4 dB	8.4 dB

Table 2.2.: Prediction accuracy in COST 231 Munich for the COST-Hata model.

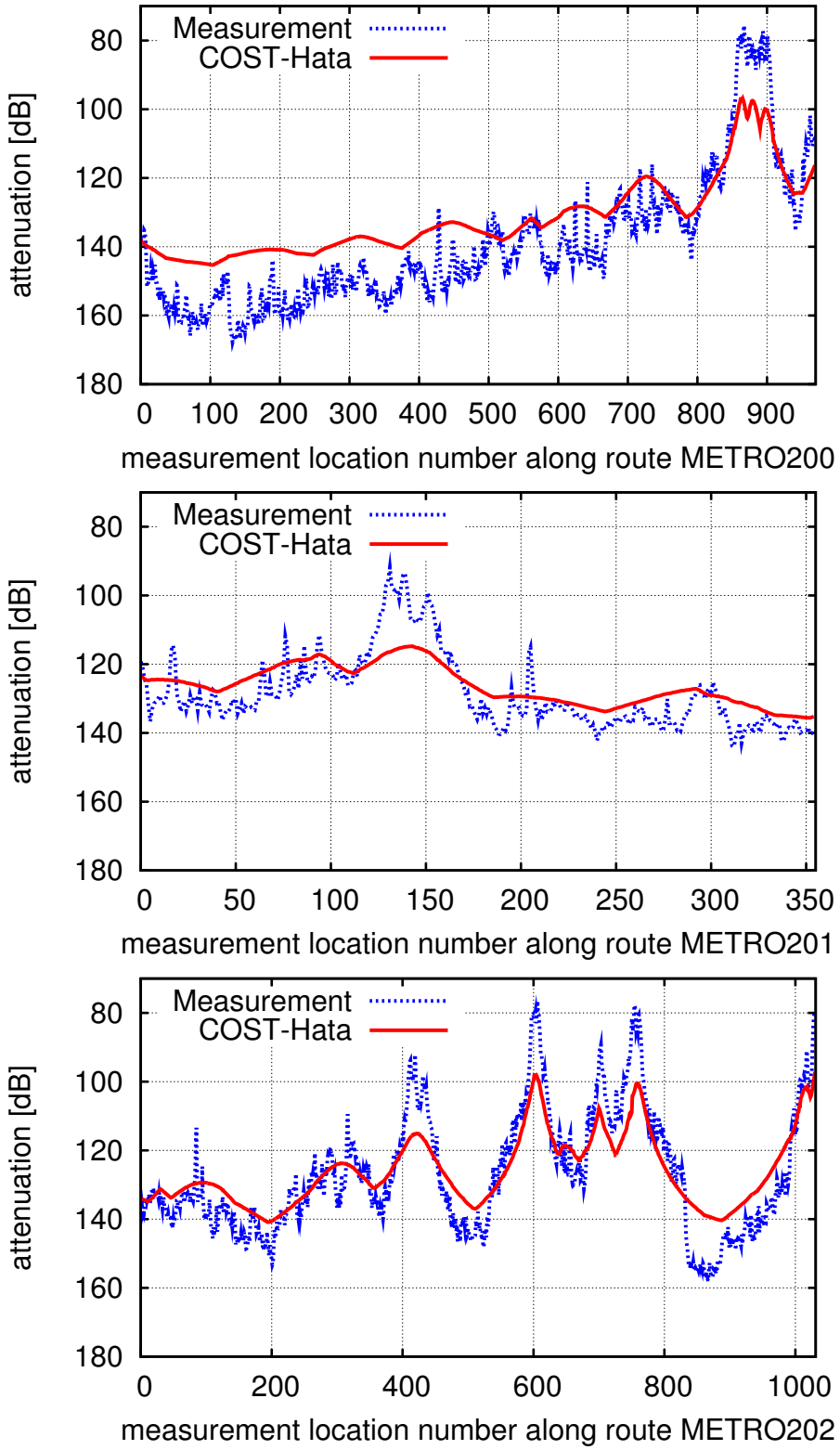


Figure 2.22.: Comparison between measured and predicted path loss for the COST-Hata model along the METRO routes in COST 231 Munich.

2.6.4.2. COST-Walfisch-Ikegami Model

The attenuation of the COST-WI model (cf. Section 2.3.1) depends whether the receiving location has an unobstructed LOS path or not. We set up our evaluation of this model as follows: For the fixed transmitter location in COST 231 Munich we first divide the receiver plane into points in LOS and the ones in NLOS according to the algorithm in Section 2.4.1.1. For the first set of points (LOS) we evaluate the path loss as

$$P_{\mathrm{LOS}}^{\mathrm{dB}}(r) = c_f + 26 \cdot \lg(d_r) + 10.2$$

and for the second set (NLOS) as

$$P_{\mathrm{NLOS}}^{\mathrm{dB}}(r) = P_{\mathrm{F0}}^{\mathrm{dB}} + \max\left\{\Delta P_{\mathrm{F,rts}}^{\mathrm{dB}} + \Delta P_{\mathrm{F,msd}}^{\mathrm{dB}}, 0\right\}.$$

Interestingly, our implementation of the COST-WI model achieves very high accuracy in terms of STD (5.9 dB) which is in the same order of magnitude as the deterministic models, cf. Table 2.3 and Table 2.13. Again, a comparison of the actual shape of the predicted and measured path loss reveals that the COST-WI model predominantly averages the overall attenuation but neglects most of the spikes and signal fluctuations, cf. Figure 2.24. The main difference to the COST-Hata model is the distinction between LOS and NLOS which is particularly visible in METRO201 (center) and METRO202 (bottom). This leads to a much better estimation of points in LOS by the COST-WI due to the additional degree of freedom. In general, the COST-WI model prediction overestimates the actual attenuation. The measured received signal strength is stronger than the COST-WI prediction which is probably due to additional wave guiding effects in street canyons that are neglected in the model.

Measurement Route	ME	MSE	STD
METRO200	−6.2 dB	9.5 dB	6.1 dB
METRO201	−10.7 dB	12.6 dB	5 dB
METRO202	−11.5 dB	14 dB	6.7 dB
All	-9.5 dB	12 dB	5.9 dB

Table 2.3.: Prediction accuracy in COST 231 Munich for the COST-WI model.

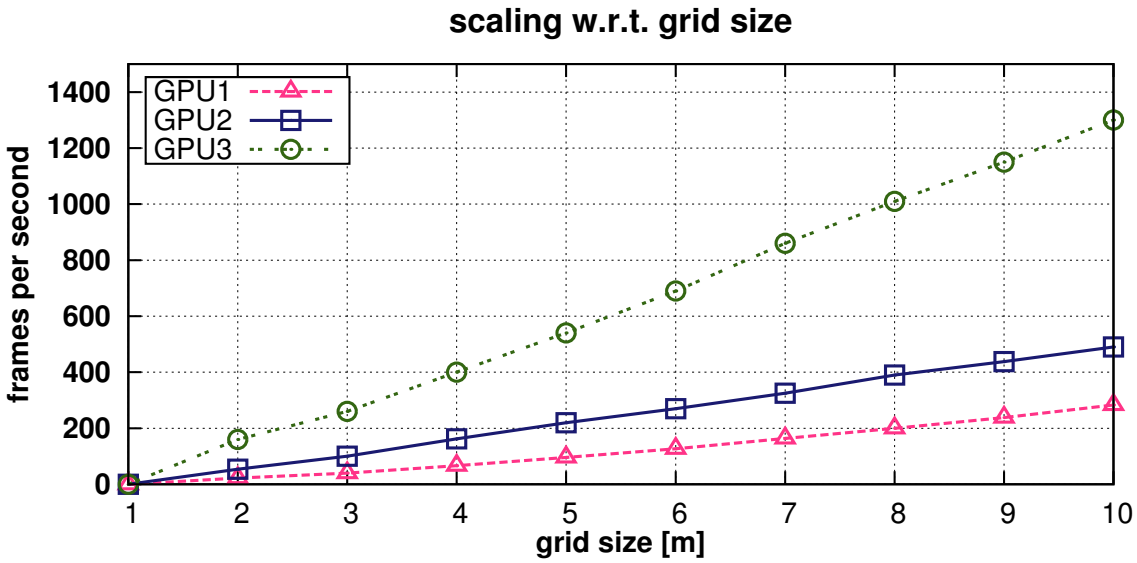


Figure 2.23.: Line-of-sight calculation time on different generations of graphics cards in the COST 231 Munich scenario.

Concerning the computational performance of our implementation, we identified the limiting factor of the model evaluation to be the computation of the LOS/NLOS region. The computation time of the evaluation of the actual path loss formula can again be neglected. Therefore, we analyzed the performance of the LOS computation with respect to scalability of the underlying resolution and number of shader units (cores). Table 2.4 gives an overivew of the hardware specifications. Figure 2.23 depicts the computations per second over the size of the underlying discrete grid. At the common resolution of 5 meters these cards achieve 100, 200 and over 500 LOS computations per second, respectively. Thus, we observe a speedup by a factor of 2 for every new generation of graphics cards. In terms of throughput, the 500 LOS computations per second can also be expressed as roughly $1.6 \cdot 10^8$ receiver points per second. The total number of receiver points is roughly $3 \cdot 10^5$ at a resolution of 5 meter. As point of reference, the commercial tool Winprop [8] states that their COST-WI implementation requires approximately 5 seconds at a resolution of 10 meters.

	type	year	core clock	core config	bandwidth (DDR3)
GPU1	6600GT	2004	500 MHz	3:8:8:4 *	14.4 GB/sec
GPU2	7800GT	2005	400 MHz	7:20:20:16 *	32.0 GB/sec
GPU3	8800GTX	2007	575 MHz	128:32:24 †	86.4 GB/sec

Table 2.4.: GPU Hardware Specifications. All graphics cards are NVIDIA Geforce cards. * core config is vertex shader : pixel shader : texture mapping unit : render output unit. † core config is unified shaders : texture mapping unit : render output unit.

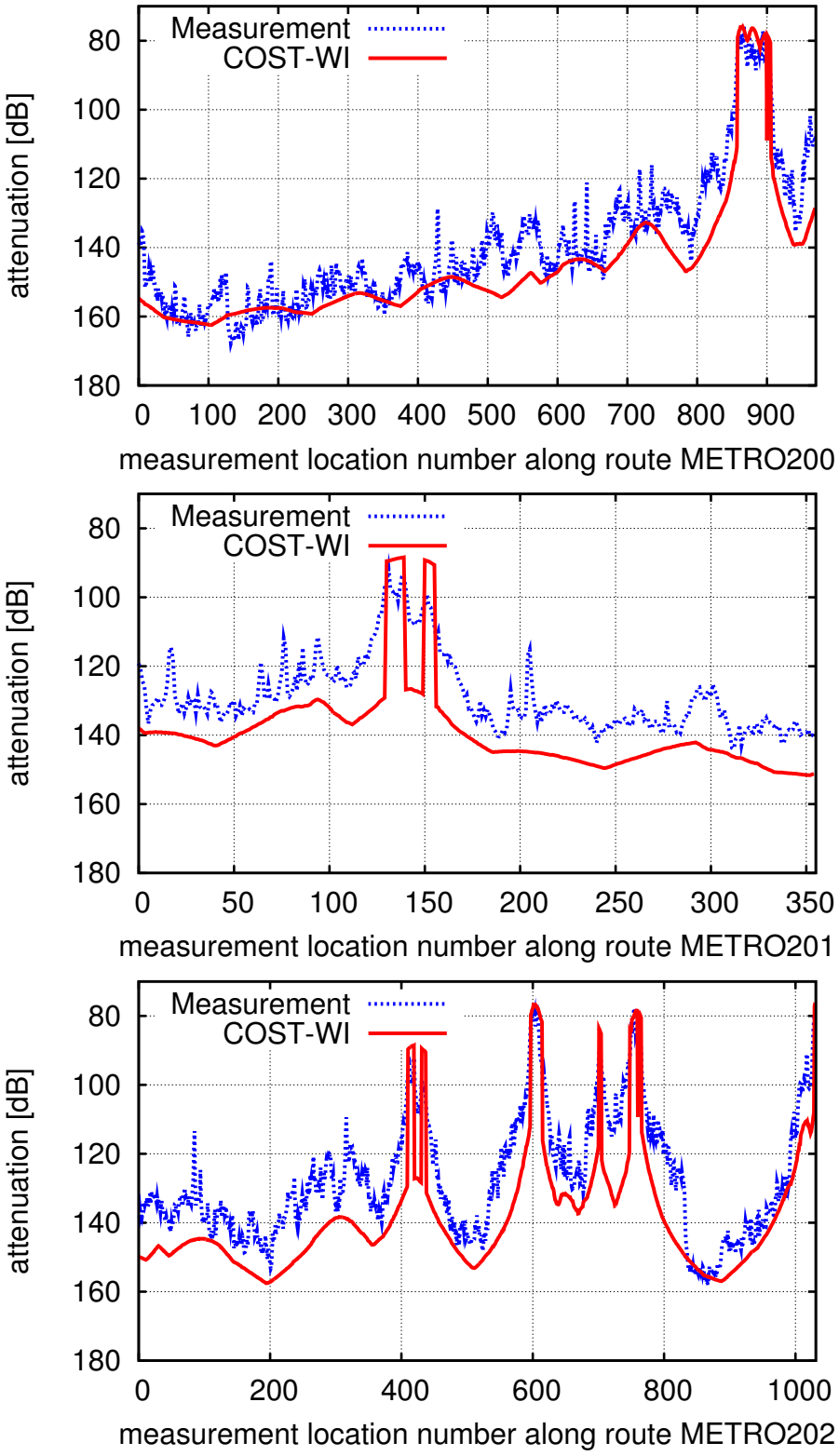


Figure 2.24.: Comparison between measured and predicted path loss for the COST-Walfisch-Ikegami model along the METRO routes in COST 231 Munich. The main difference to the COST-Hata model is the distinction between LOS and NLOS which is particularly visible in METRO201 (center) and METRO202 (bottom).

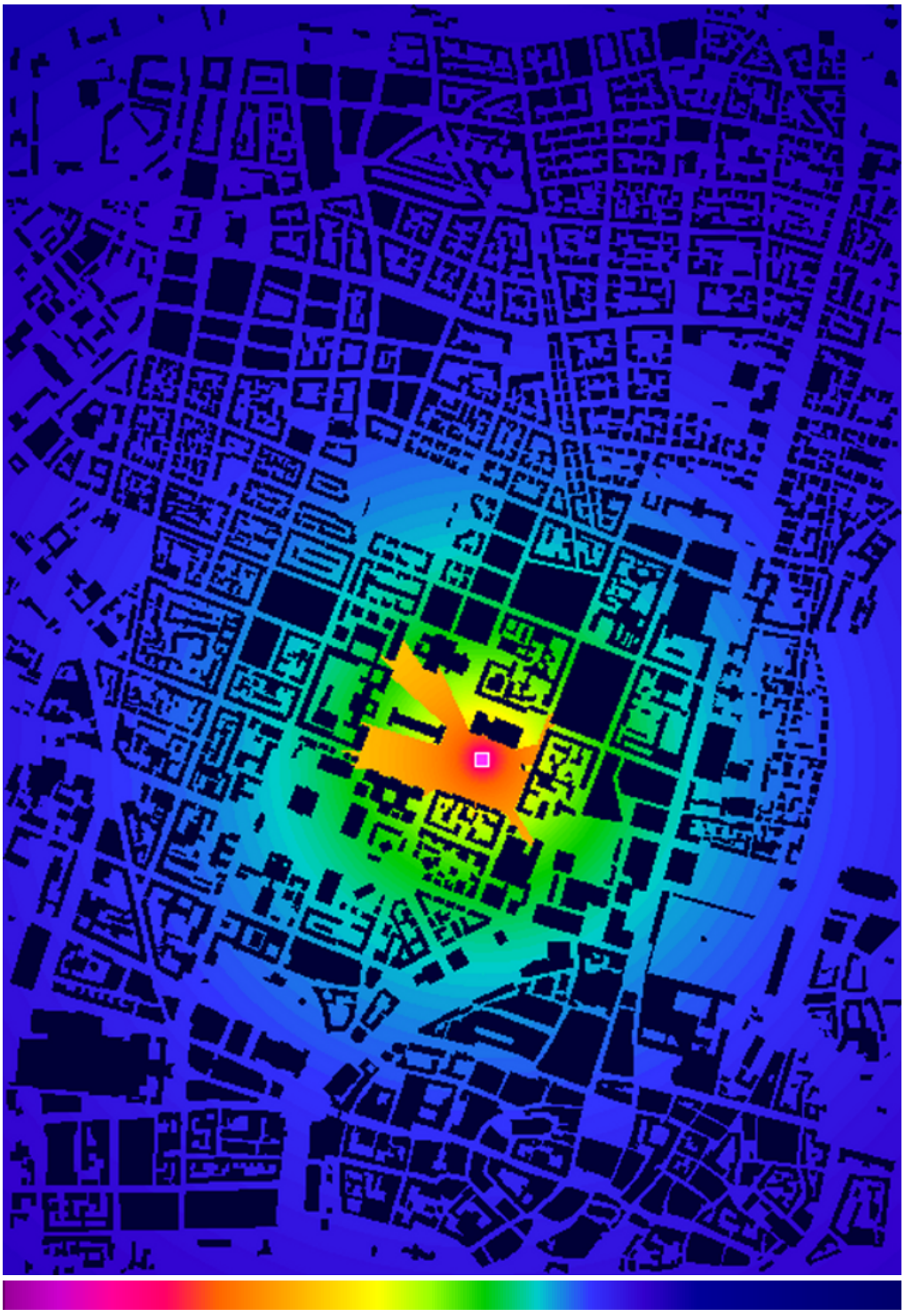


Figure 2.25.: This image shows the field strength prediction of the COST-WI model in the COST 231 scenario in downtown Munich, Germany.

2.6.5. Ray Optical Models

2.6.5.1. Roof Diffraction Model

We recall the path loss computation of the RDM from Section 2.4.2.1 as

$$PL_{\mathrm{RDM}}^{\mathrm{dB}}\left(\upsilon^{\rightsquigarrow}\right) = c_f + c_0 + \gamma \cdot 10 \cdot \lg\left(d_{\upsilon^{\rightsquigarrow}}\right) + \sum_{i=1}^{N_r} g\left(\alpha_{\upsilon^{\rightsquigarrow}}^{(i)}\right)$$

with the angle dependent attenuation $g\left(\alpha\right) = a_0 + a_1\alpha + a_2\alpha^2$.

Model Parameter Optimization We formulate physical constraints to the model parameters of the angle dependent loss as

$$\left.\begin{array}{lll} (a) & g\left(\alpha\right) & = a_0 + a_1\alpha + a_2\alpha^2 \geq 0 \\ (b) & g'\left(\alpha\right) & = a_1\alpha + 2a_2\alpha \geq 0 \end{array}\right\} \quad \forall\alpha \in \left[0, \frac{\pi}{2}\right] \tag{2.59}$$

where $(2.59a)$ ensures that diffraction does not increase received power and $(2.59b)$ that diffraction loss increases with increasing deflection angle. To fit our optimization framework (2.50), these constraints must be reformulated as equivalent linear constraints:

$$\begin{array}{lcl} g\left(0\right) & = & a_0 \geq 0 \\ g'\left(0\right) & = & a_1 \geq 0 \\ g'\left(\frac{\pi}{2}\right) & = & a_1 + a_2\pi \geq 0. \end{array} \tag{2.60}$$

With parameter vector $\boldsymbol{v} = \left(c_0, \gamma, a_0, a_1, a_2\right)^T$ we can formulate the linear optimization problem as

$$\hat{\boldsymbol{v}} = \arg\min_{\boldsymbol{v}} \sum_{i=1}^{N} \left(M^{\mathrm{dB}}\left(r_i\right) - PL^{\mathrm{dB}}\left(r_i\right)\right)^2 \tag{2.61}$$

with

$$\begin{array}{l} a_0 \geq 0 \\ a_1 \geq 0 \\ a_1 + a_2\pi \geq 0 \end{array} \tag{2.62}$$

where $\hat{\boldsymbol{v}}$ minimizes the mean squared error between predicted and measured data.

Solving (2.61) with constraints (2.62) for each individual measurement route yield the parameter vectors detailed in Table 2.5. We observe major differences between the

Optimized / Parameter Value	c_0	γ	b_0	b_1	b_2
METRO200	−53.36	5.02	0	0	4.22
METRO201	−29.04	3.95	1.5	4.27	−1.09
METRO202	−48.06	4.72	0.15	1.99	2.24

Table 2.5.: Parameter vectors according to constraints (2.62)

Optimized / MSE	METRO200	METRO201	METRO202	average
METRO200	6.1 dB	5.5 dB	6.6 dB	6.1 dB
METRO201	8 dB	4.4 dB	6.4 dB	6.3 dB
METRO202	6.9 dB	4.8 dB	6.1 dB	5.9 dB

Table 2.6.: Prediction accuracy of RDM calibrated with constraints (2.62).

parameter vectors which may diminish their ability to generalize well to other scenarios. The behavior of the angle dependent loss function is depicted in Figure 2.26 (left) for each parameter set per measurement route. In particular, the loss function of ROUTE201 deviates significantly in behavior from the ones of ROUTE200 and ROUTE202. We marked the range between 0 and $\frac{\pi}{4}$ (45 degree) since most diffraction angles are expected to fall into that range. Table 2.6 details the prediction quality with respect to the MSE for each pair of parameter vector and measurement route when using the physical constraints (2.62).

To compensate for the fluctuations in parameter characteristic and prediction quality we introduce an additional constraint by forcing the base loss c_0 to zero. Hence, the path loss will solely be determined by distance to sender and deflection angles. The modified constraints are

$$\begin{aligned} c_0 &= 0 \\ a_0 &\geq 0 \\ a_1 &\geq 0 \\ a_1 + a_2\pi &\geq 0. \end{aligned} \tag{2.63}$$

Again, we solve (2.61) for each measurement route, now with the more strict constraints (2.62). The resulting parameter sets are given in Table 2.7. We observe a significant improvement in stability over the previous parameters. The characteristics of the angle dependent loss functions are now in good agreement in the most important range between 0 and 45 degree, cf. Figure 2.26 (right). We notice a decrease in overall prediction quality (Table 2.8). However, we believe that the improved generality of parameters is worth the slight drop in accuracy.

Optimized / Parameter Value	c_0	γ	b_0	b_1	b_2
METRO200	0	3.06	2.09	0.059	5.37
METRO201	0	2.86	2.59	1.74	2.24
METRO202	0	2.76	3.37	0	4.9

Table 2.7.: Parameter vectors according to the more strict constraints (2.63).

Optimized / MSE	METRO200	METRO201	METRO202	average
METRO200	8.1 dB	6.9 dB	9 dB	8 dB
METRO201	9.8 dB	4.7 dB	7.5 dB	7.3 dB
METRO202	9.8 dB	4.8 dB	7.4 dB	7.3 dB

Table 2.8.: Prediction accuracy of RDM calibrated with the more strict constraints (2.63).

In the remainder of this work all path loss calculations were performed with parameters

$$c_0 = 0.0; \quad \gamma = 2.76; \quad a_0 = 3.37; \quad a_1 = 0; \quad a_2 = 4.9$$

which was retrieved from a calibration to route METRO202 in the COST 231 Munich scenario with additional physical constraints according to (2.63). Basically, the parameter vector represents unknown components like material properties or vegetation in the propagation environment. The main difference to the empirical models is the consideration of actual diffraction effects at building rooftops. As a result of the parameter optimization to the data points of route METRO202, the ME is almost zero there. For a fair comparison to the empirical models, the STD may be the major accuracy indicator since it removes the ME from the performance criteria.

Accuracy and Performance The RDM can achieve a much better reconstruction of the measured signal as illustrated in Figure 2.27. However, strong fluctuations are visible in METRO200 that happen at large distances between sender and receiving measurement points are due to the discrete nature of the GPU algorithm. A small change at the receiver location leads to a large change of the deflection angle of the roof diffractions which ultimately leads to discontinuities in the path loss computation. Informally speaking, the discrete resolution can lead to situations where rays are shot through buildings instead of being deflected. The numerical instabilities have a direct influence on the performance in terms of STD which is with 7.2 dB less accurate than the COST-WI prediction, cf. Table 2.9 and Table 2.3. However, the predicted signal strength of routes METRO201 and METRO202 as depicted in Figure 2.27 are in good agreement with the actual measurements.

A complete evaluation of the RDM in the COST 231 Munich scenario requires approximately 3 seconds at a resolution of 5 meters. We compared this to the model CORLA

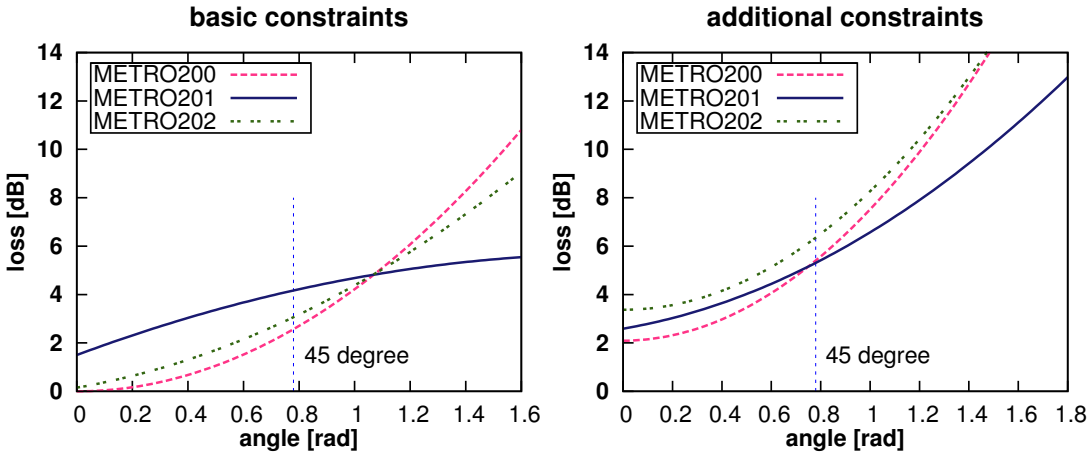


Figure 2.26.: Loss functions $g(\alpha)$ for angle dependent part of the RDM, with non-zero base loss (left) and additional physical constraints (right). Most diffraction angles are expected to fall in the range between 0 and $\frac{\pi}{4}$ (45 degree).

Measurement Route	ME	MSE	STD
METRO200	−4.9 dB	9.8 dB	9.5 dB
METRO201	−0.3 dB	4.8 dB	4.7 dB
METRO202	0.3 dB	7.4 dB	7.3 dB
All	-1.6 dB	7.3 dB	7.2 dB

Table 2.9.: Prediction accuracy in COST 231 Munich of RDM.

which we configured to consider only horizontal roof diffraction at a maximum recursion depth of 6. For a fair comparison we turned off the computation of reflection or vertical diffraction. At this setting CORLA requires roughly 9 seconds for a prediction in the same scenario. When judging the speedup of 3X we have to take into consideration that our RDM implementation inherently computes ray paths of all possible number of multiple roof diffractions which would correspond to an unbounded level of recursion.

For a deeper analysis of the computation time of our RDM implementation, we subdivided the evaluation into sub tasks: (1) the computation of LOS and the horizontal diffraction beams (HDBs), (2) the reconstruction of ray paths due to multiple roof diffractions and (3) the transformation into convex ray paths with a subsequent evaluation of the path loss formula (2.43).

As depicted in Table 2.10 we see that although the computation of the HDBs requires a lot of LOS evaluations it takes the lowest share of computation time with 0.05 seconds due to our highly efficient GPU implementation (cf. Figure 2.23). The reconstruction of diffraction paths of arbitrarily length is also performed in parallel on the GPU. However, the large number of memory accesses in proportion to floating point operations, every

Task/ Run time	NVIDIA 7800 GT AMD 3500
LOS / NLOS + diffraction beams	0.05s (GPU)
Ray path reconstruction	0.8s (GPU)
Convex hull of ray path + model evaluation	2.2s (CPU)
Total	**3.05s (GPU+CPU)**
CORLA (only roof diffraction with recursion depth of 6)	9s (CPU)

Table 2.10.: Run time analysis of the RDM at a resolution of 5 meter in the Munich scenario.

deflection point has to look up its predecessor until the original radiation source is reached, takes its toll with about 0.8 seconds. The algorithm for the upper convex ray path hulls has been implemented on the CPU only, due to a missing stack implementation on the GPU. Accordingly, all intermediate results have to be downloaded from the GPU and processed sequentially, hence this part makes up the largest portion of the overall run time with 2.2 seconds. Of course, it would be desirable to reduce the GPU-CPU bottleneck as much as possible. We will address this issue in the EDM implementation which can be performed entirely on the GPU.

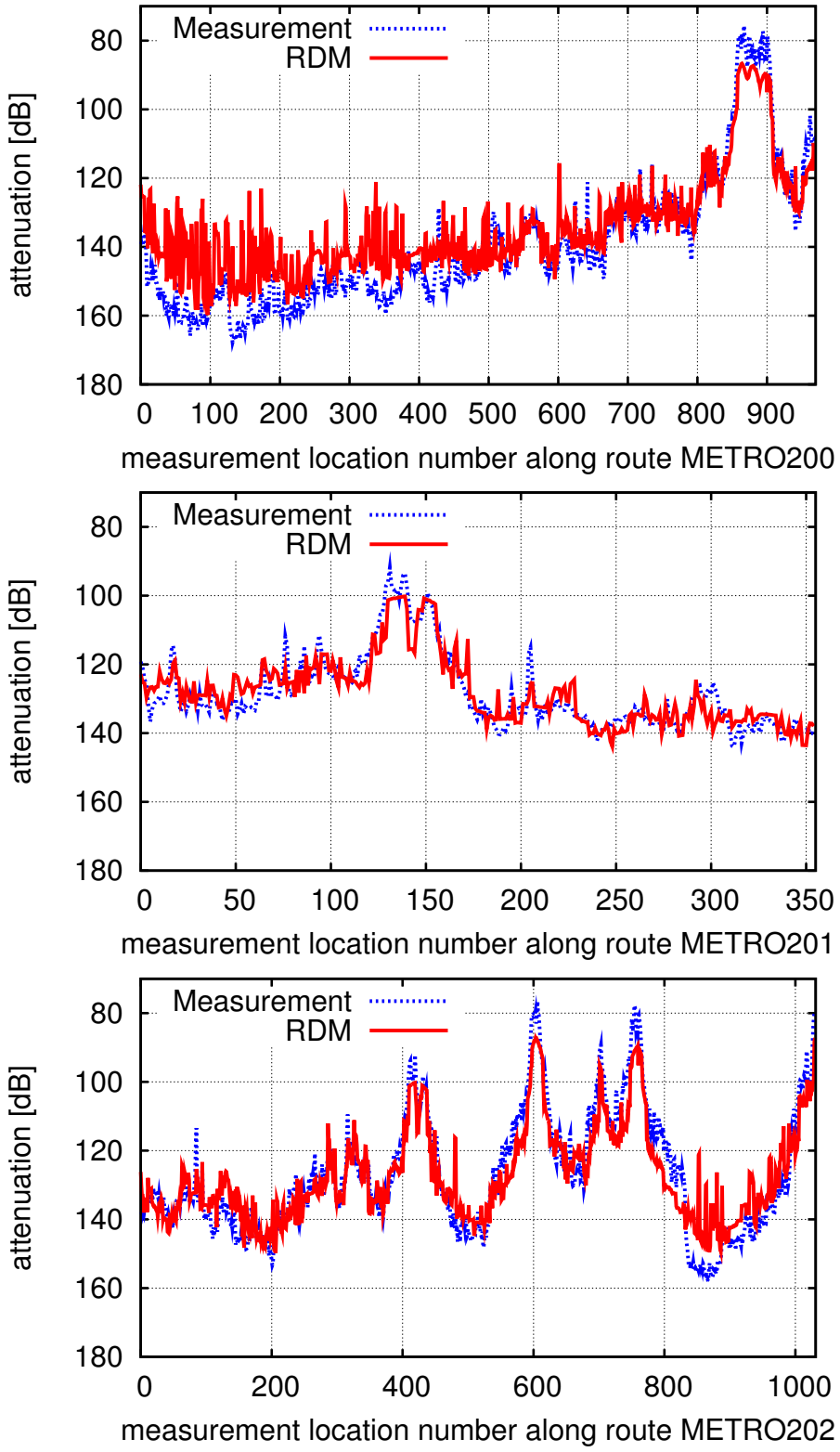


Figure 2.27.: Comparison between measured and predicted path loss for the Roof Diffraction Model (RDM) along the METRO routes in COST 231 Munich. The main difference to the empirical models is the consideration of actual diffraction effects on building rooftops.

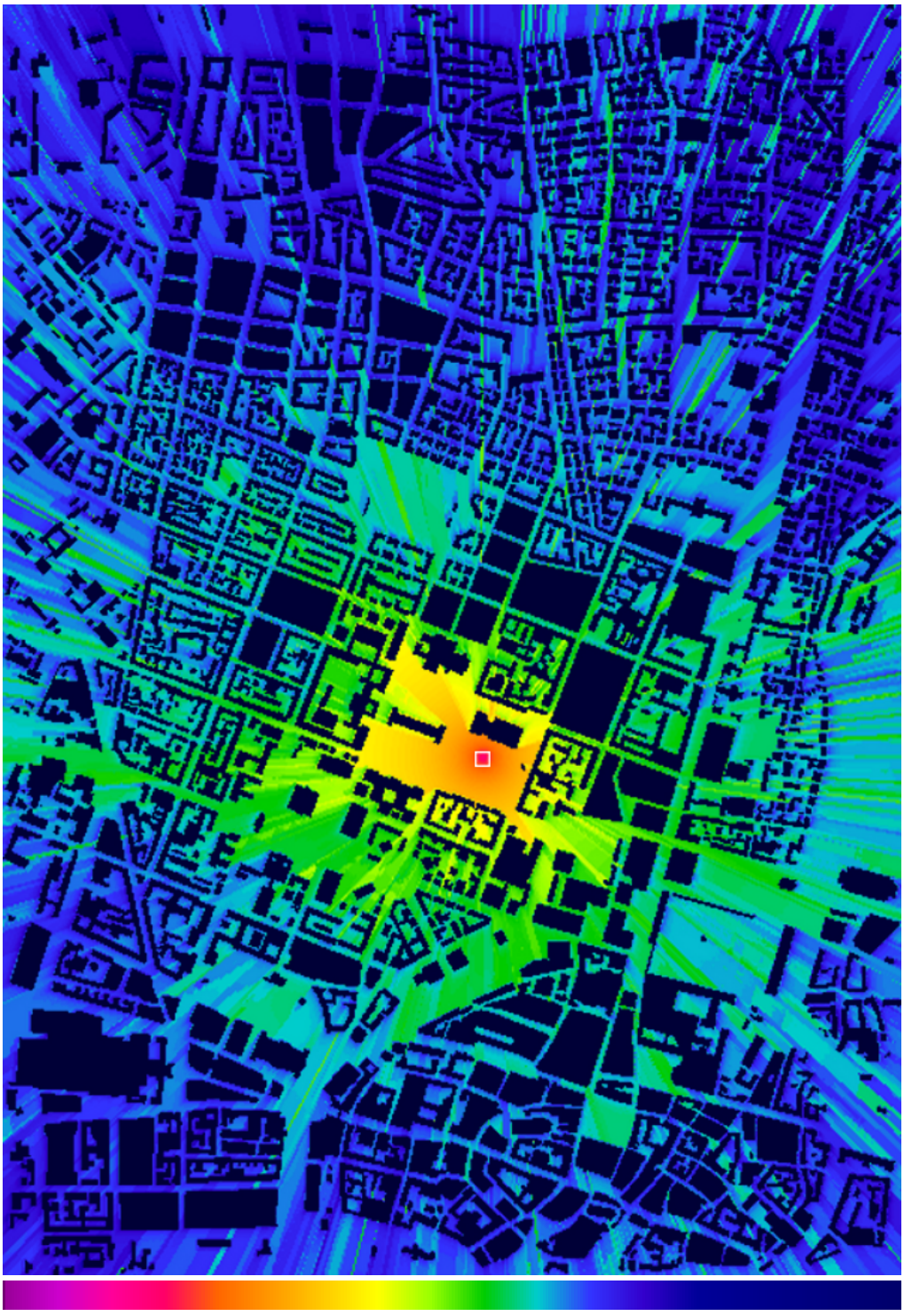


Figure 2.28.: This image shows the field strength prediction of the RDM in the COST 231 scenario in downtown Munich, Germany.

2.6.5.2. Edge Diffraction Model

We use the path loss formulation of the EDM as defined in Section 2.4.2.2

$$PL^{\mathrm{dB}}_{\mathrm{EDM}}\left(\upsilon^{\rightsquigarrow}\right) = c_f + \begin{cases} PL^{\mathrm{dB}}_{\mathrm{LOS}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ in LOS} \\ PL^{\mathrm{dB}}_{\mathrm{LOS}}\left(\upsilon^{\rightsquigarrow}\right) + PL^{\mathrm{dB}}_{\mathrm{VD}}\left(\upsilon^{\rightsquigarrow}\right), & \upsilon^{\rightsquigarrow} \text{ due to diff. at vertical edge} \\ PL^{\mathrm{dB}}_{\mathrm{HD}}\left(\upsilon^{\rightsquigarrow}\right), & \text{otherwise} \end{cases}$$

with the model parameters

$$\begin{aligned} \gamma_{\mathrm{LOS}} &= 2.6007; \quad \gamma_{\mathrm{NLOS}} = 3.4330; \quad L_w = 0.2522; \\ b_0 &= 1.4562; \quad b_1 = 0.4022; \quad b_2 = -0.0022. \end{aligned}$$

As always, all predictions were performed on the whole supplied area of approximately 7 km^2 with a resolution of 5 meters in the COST 231 Munich scenario. We have observed that in the particular scenario the first level of vertical diffraction into street canyons already provides sufficient prediction accuracy. The fixed angle approximation to the horizontal diffraction which is realized by determining the wall penetration depth, as described in Section 2.4.1.2, shows satisfying agreement with regions in deep NLOS, cf. Figure 2.29.

The achieved criteria for prediction quality are detailed in Table 2.11. On average a STD of 5.4 dB is achieved. It's worth noting that the STD and MSE are more or less the same for all measurement routes due to an almost zero ME. We conclude that the EDM model assumptions are well suited for the heterogeneous building development in the historically grown structures of the city of Munich and that the parameter vector which has been obtained by an optimization to the measurement points of route METRO202 generalizes nicely.

The strong fluctuations in the prediction by the RDM are greatly reduced in the EDM which in general achieves a much more stable reconstruction of the measured signal. It can be seen that although the path loss is still sometimes over- or underestimated, most of the important features in the measurement data are captured quite well by the propagation prediction. Among the considered models so far, we believe that the EDM performs best.

Table 2.12 gives an overview of different propagation predictions with respect to accuracy and run time for route METRO201. We use the CPU implementation of the ray launching algorithm CORLA by [94] as reference for our GPU implementation. In our test scenario, CORLA exhibits run times of about 9 seconds. When switching from the CPU implementation of CORLA to our GPU method we observe a speedup of roughly

Measurement Route	ME	MSE	STD
METRO200	2.1 dB	6.2 dB	5.9 dB
METRO201	0.1 dB	4.5 dB	4.5 dB
METRO202	−0.1 dB	5.7 dB	5.7 dB
All	0.7 dB	5.5 dB	5.4 dB

Table 2.11.: Prediction accuracy in COST 231 Munich of the EDM.

3X for the roof diffraction method where the most time (2 seconds) is currently consumed by the computation of upper convex hull, since this has to be executed on the CPU, due to a missing stack implementation on the GPU. The computation of propagation effects (diffraction into street canyon with transmission depth) required for the EDM can be computed at approximately 0.05 seconds and achieves roughly the same prediction accuracy. Hence, the diffraction into street canyon with the fixed angle approximation for horizontal diffraction results in a total speedup of a factor of 160X with respect to the CPU implementation of CORLA.

Prediction model	ME	STD	run time
CORLA [94] (CPU)	0.1 dB	4.2 dB	9 s
RDM (GPU)	−0.2 dB	4.7 dB	3.05 s
EDM (GPU)	0.1 dB	4.5 dB	0.05 s

Table 2.12.: Accuracy and run times of propagation models in COST 231 Munich along route METRO201.

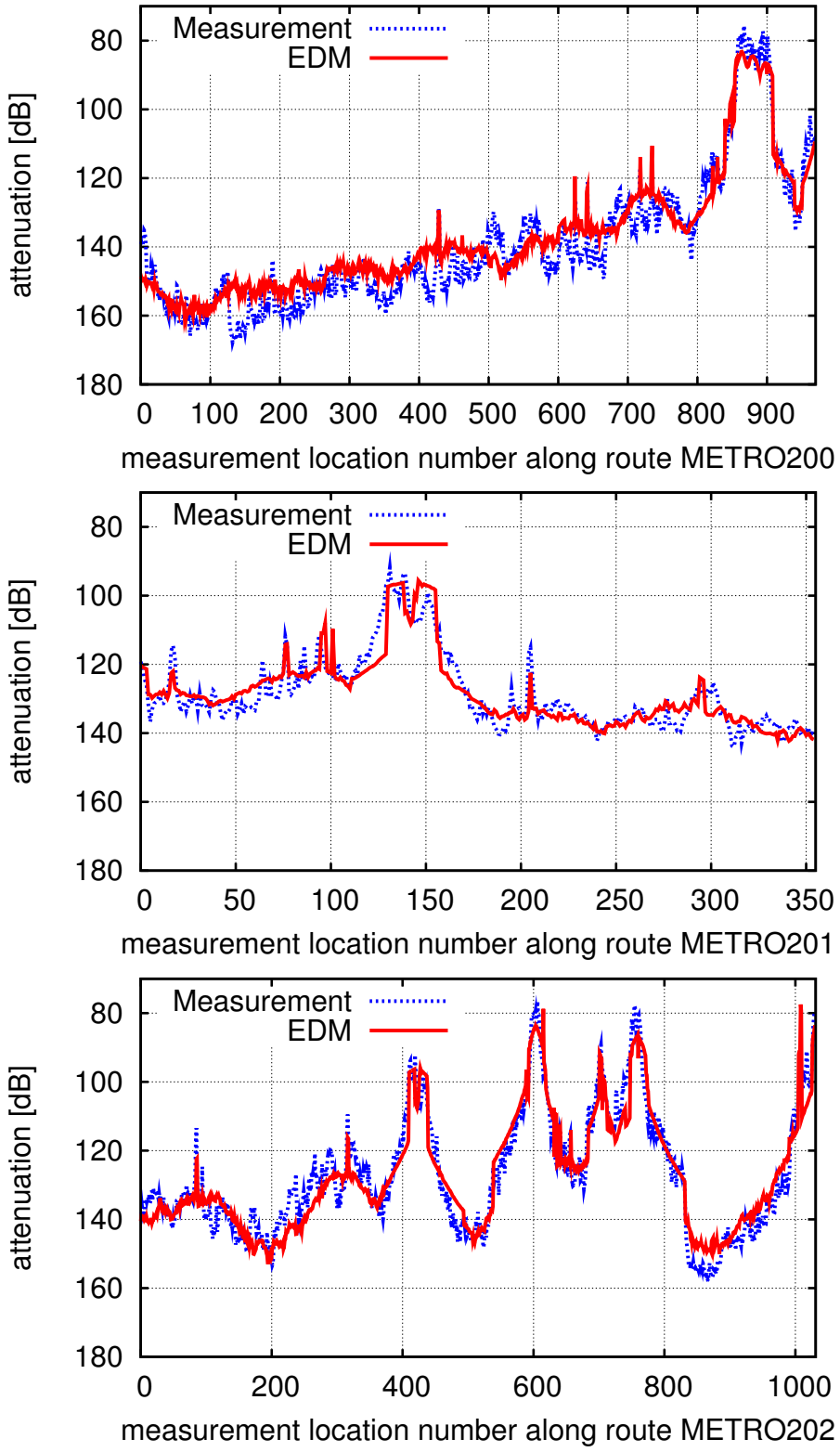


Figure 2.29.: Comparison between measured and predicted path loss for the Edge Diffraction Model (EDM) along the METRO routes in COST 231 Munich. The main difference to the empirical models is the consideration of actual diffraction effects into street canyons.

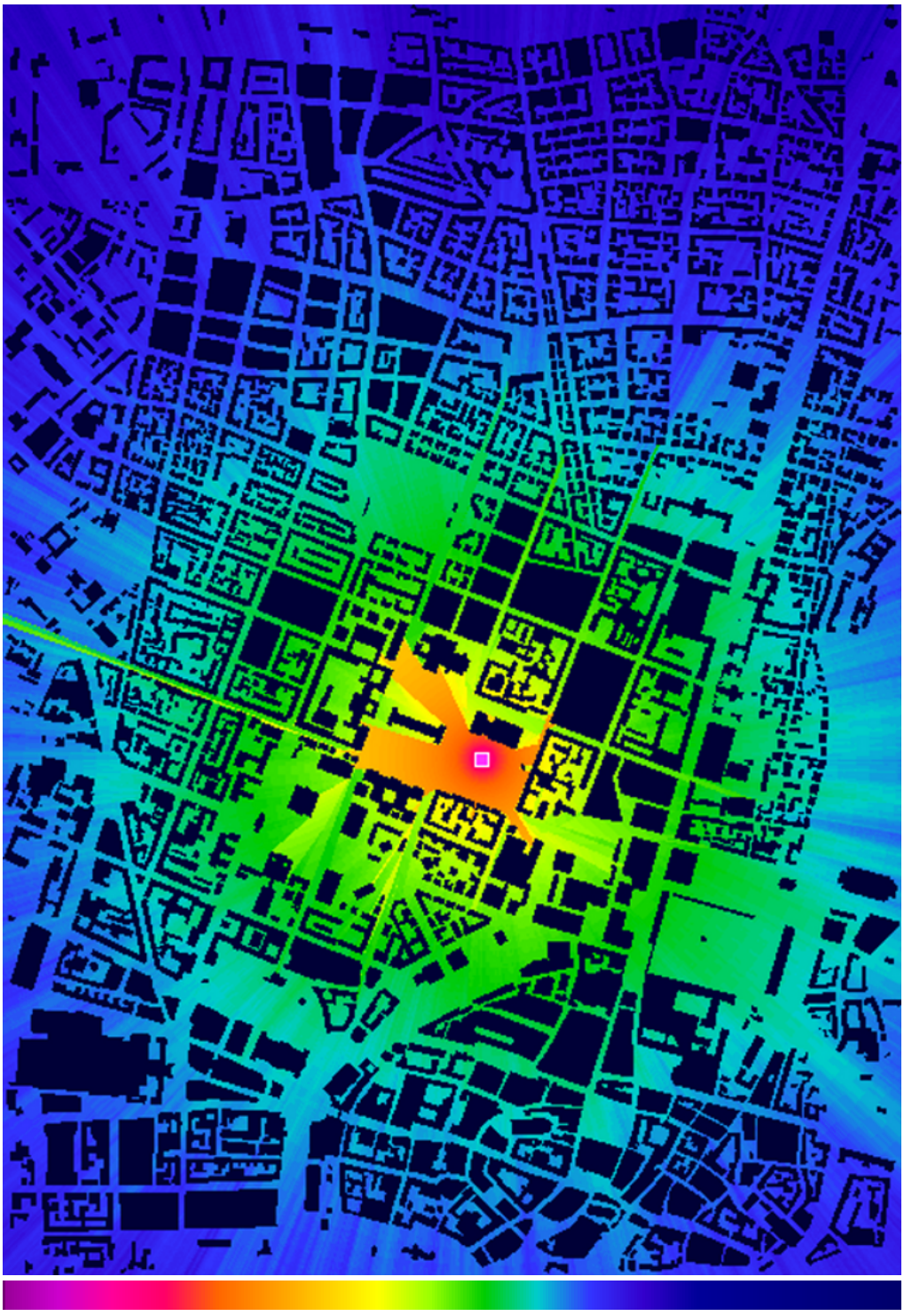


Figure 2.30.: This image shows the field strength prediction of the EDM in the COST 231 scenario in downtown Munich, Germany.

Model/Measurement Route	ME	MSE	STD
COST-Hata/METRO200	−7.6 dB	12.2 dB	9.5 dB
COST-Hata/METRO201	−2.1 dB	7 dB	6.7 dB
COST-Hata/METRO202	−0.9 dB	8.9 dB	8.9 dB
COST-WI/METRO200	−6.2 dB	9.5 dB	6.1 dB
COST-WI/METRO201	−10.7 dB	12.6 dB	5 dB
COST-WI/METRO202	−11.5 dB	14 dB	6.7 dB
RDM/METRO200	−4.9 dB	9.8 dB	9.5 dB
RDM/METRO201	−0.3 dB	4.8 dB	4.7 dB
RDM/METRO202	0.3 dB	7.4 dB	7.3 dB
EDM/METRO200	2.1 dB	6.2 dB	5.9 dB
EDM/METRO201	0.1 dB	4.5 dB	4.5 dB
EDM/METRO202	−0.1 dB	5.7 dB	5.7 dB

Table 2.13.: Overview of prediction accuracy in COST 231 Munich for our GPU implementation of the COST-Hata, COST-WI, RDM and EDM.

2.6.6. Comparison

First, we compare our GPU implementations with each other. Then, we briefly introduce well-known models from literature that have published results on their performance in the COST 231 scenario and give an overview of the prediction accuracies and compare them against the background of our results.

2.6.6.1. Comparison of GPU Implementations

Above, we have discussed the presented models in detail. Here, we give a broader comparison of the models to each other. For sake of completeness, an overview of the quality criteria of all implemented models is given in Table 2.13. In the Munich test site we can achieve higher prediction quality by an increasing computational complexity of the used models. The COST-Hata and the COST-WI model can offer a fast preview of the mean path loss, however rather large absolute errors are involved as can be seen from the large difference between their MSE and the corresponding STD. In general, the RDM can achieve a much better reproduction of the actual path loss measurements, but it suffers from instabilities of the discrete computation with a negative effect on the overall prediction quality. The instabilities have been greatly reduced in the EDM which performs best among the presented models. The MSE and STD are almost the same which is also reflected by a good reconstruction of the original signal as detailed in Figure 2.29.

2.6.6.2. Comparison with Literature

Most of the propagation models in Table 2.14 are based on empirical approaches or ray optical methods with either simplified analytical solutions or pure ray tracing techniques. The various approaches differ greatly on computation time efficiency, analytical path loss model and input data. Therefore, we will briefly summarize the essential idea and theoretical background of the individual models.

Ericsson The path loss in this model [21] is determined along paths following different streets with arbitrary angles of crossing streets. Internally, this is realized by a ray tracing approach. Furthermore, the model uses well known (empirical) path loss expressions for the loss between two isotropic antennas which is basically a recursive function of the number of deflection points between sender and receiver. Additionally, regions that are not reached with ray tracing are approximated by a COST-WI-like model for NLOS propagation to approximate multiple rooftop diffractions.

CNET The CNET model is an analytical, semi-deterministic approach [159]. It considers reflected and diffracted wave fronts below roof-top level. Propagation into street canyon can deal with street crossings of four corners (e.g. the Manhattan Grid). The maximum level of reflection recursion depths is given as 9, both for LOS and NLOS. Conductivity coefficients are approximated by heuristic parameters.

PPT This model by the Swiss Telecom [127] can handle arbitrary two-dimensional building geometry with individual permittivity and conductivity of each building wall, if available. However, the majority of their computations are done by using the same characteristic for every building. Ground reflection, scattering and over-roof-top propagation are neglected but specular reflections are computed by a ray tracing technique. Diffraction is computed by placing virtual sources on all building corners and traversing secondary propagation rays. The path loss is derived by a superposition of all incoming rays at a receiver location.

PPT TLM The propagation modeling is achieved by a transmission line matrix (TLM) technique [9] which is based on a direct discretization of the building structures in a two-dimensional grid. Solution approaches are similar to the Lattice Boltzmann Models (LBMs) which are commonly used in solving systems in computational fluid dynamics. In terms of wave propagation, space and time are represented in terms of finite elements and the propagation equations are solved with corresponding numerical schemes. The TLM assumes infinite building heights which implies a two-dimensional propagation

which is referred to as cylindrical source problem. The results are converted back to 3D by renormalization of the predicted results according to distance between transmitter and receiver.

COST-WI* see Section 2.3.1.

Uni-Valencia This approach separates the computation of propagation effects into rooftop diffraction and local scattering in receiver vicinity [34]. They follow an empirical computation approach with the argument that the building maps do not provide information of rooftop shape. Incident rays at the receiver are scattered at nearby walls and traced via ray tracing. If the transmitter is below the mean rooftop level, the model describes an additional diffraction loss, similar to the COST-WI approach.

CSELT This model predominantly estimates the path loss due to propagation over multiple rooftops [111]. It extends the COST-WI approximation by taking buildings of different heights within the vertical propagation plane into account. The buildings are treated as infinitely thin absorbing wedges and thus the Deygout's diffraction approach [48] is applied in this model.

PPT MCOR This is another model developed by the Swiss Telecom. The software is called MCOR and implements a multi-knife edge propagation and is a modified Deygout's methods proposed by [38]. The modification introduces a linear increase of the path loss due to multiple diffractions, thus the influence of the diffraction effect is decreased with increasing level of recursion. Additionally, a dual slope model is used to represent the distance dependent loss in LOS and NLOS.

Uni.-Karlsruhe This model proposes a three-dimensional ray tracing approach where base stations may be above as well as below building heights [41]. The ray tracing technique takes building information as either pixel grid data or vector oriented data. The result depends on a reasonable horizontal resolution of the supplying area. If not provided in the database, building heights are approximated by the number of floors and corresponding floor heights. The ray tracing implementation includes dominant propagation paths like rooftop diffraction in the vertical plane and diffraction into street canyons in the transverse plane. Within these planes, propagation paths are computed two-dimensionally.

Prediction model	METRO200		METRO201		METRO202		All
	ME	STD	ME	STD	ME	STD	STD
Ericsson [21] (R+E)	0.3	6.7	2.3	7.1	1.4	7.5	**7.1**
CNET [159] (R+O)	-2.1	6.9	-3.6	9.5	-0.2	5.6	**7.3**
PPT [127] (R)	-6.1	14.6	-6.7	15.5	-1.1	12.3	14.1
PPT [9] TLM (O)	0.8	13.8	6.7	21.7	6.5	12.9	16.1
COST-WI* (E)	10.8	7.7	15.4	5.9	16.3	7.3	**7.0**
Uni.-Valencia [34] (E+O)	0.2	8.7	-6.6	7.0	-7.4	10.3	8.7
CSELT [111] (E+O)	21.8	10.4	16.1	12.3	20.6	13.3	12.0
PPT MCOR [38] (O)	-3.3	7.0	-0.1	6.2	-1.1	7.6	**6.9**
Uni.-Karlsruhe [41] (R)	-4.3	8.5	2.4	9.1	-1.0	8.6	8.7
CORLA [94] (R)	1.0	6.1	0.1	4.2	0.1	5.6	**5.3**
Beamtrace [134] (R)	1.0	6.1	0.1	4.2	0.1	5.6	**5.3**
GPU EDM (R)	2.1	6.2	0.1	4.5	-0.1	5.7	**5.4**

Table 2.14.: Prediction accuracy in COST 231 Munich for various propagation models. Methods categorization: empirical (E), ray optical(R), other (O).

CORLA This model [94] is a ray launching algorithm. It represents the 3D propagation environment as a list of surfaces. Rays are launched from the transmitter on a discrete grid into predetermined directions and may be reflected or diffracted at surfaces. Diffractions are modeled by emitting new ray bundles into the respective diffraction cone whereas reflections only change the direction of existing rays. By concatenating deflection points, ray paths are constructed and based thereof the path loss is determined. Unknown model coefficients are determined by an iterative approach between calibration to measurement and computation of strongest propagation paths.

Beamtrace Schmitz et al.[134] presented an algorithm that uses beam tracing instead of classical ray tracing to efficiently compute propagation predictions. Their work is inspired by algorithms from global illumination in computer graphics. They focus on an efficient GPU implementation that relies on a custom rasterization pipeline. Their algorithm is explicitly designed to compute multi-path effects due to diffraction and refraction. Propagation Environment characteristics are also captured by calibration to measurements. Part of their work represents an extension to the algorithms presented in this thesis.

The quality criteria of the introduced models are summarized in Table 2.14 in terms of ME and STD. We first observe that the quality is very heterogeneous among the different models. The worst quality is about 16.1 dB by the PPT TLM. The highest accuracy is achieved by CORLA and our implementation of the EDM with 5.3 dB and 5.4 dB, respectively. It should be noted, that the MSE and STD can not be reduced arbitrarily. First, the description of the propagation environment is usually incomplete

like style of roof, type of material and vegetation. Furthermore, random effects like weather and traffic can only be modeled mathematically. Second, the measurements itself are subject to noise and uncertainty. It can be shown that two consecutive test runs within urban environments under comparable conditions exhibit already a STD of over 3 dB, cf. [126]. We see that whether the GPU or CPU is involved in the prediction computation has no significant influence on the propagation accuracy. The performance of the EDM exhibits similar accuracy as other state-of-the-art models.

2.7. Conclusions

This chapter showed how to exploit graphics hardware for accelerating the computation of radio wave propagation predictions. Our method traces discrete line-of-sight beams and reconstructs ray paths based on radiation source, beam origin and discrete deflections points within beams. The presented algorithms are designed to benefit from the computational power and parallel architecture of modern graphics cards. Core component of the *Graphics Processing Unit* (GPU) algorithms is a high throughput *Line-of-Sight* (LOS) computation which leads to computation times of 3 to 0.05 seconds in a scenario of 7 km^2 with roughly 18000 walls. This results in a speedup of 3X to 160X times compared to the CPU algorithm *Cube-Oriented-Ray-Launching-Algorithm* (CORLA) which performed best among state-of-the-art models in literature. Our performance increase is achieved by designing separate algorithms for distinct propagation effects such as diffraction over rooftops and diffraction into street canyons. An appropriate combination of propagation effects in the *Edge Diffraction Model* (EDM) can deliver propagation predictions at interactive rates. The accuracy of our propagation predictions is quantified by the *Mean Squared Error* (MSE) and the *Standard Deviation* (STD) between predictions and measurement data of 4 to 7 dB which is considered as a very good result. We conclude that the use of graphics hardware for field strength predictions does not diminish propagation accuracy but can significantly reduce computation time. We summarize the main contribution as the development of a run time efficient algorithm that accurately predicts mean received signal strengths in dense urban propagation environments.

Furthermore, we presented a VR prototype application for wireless communication networks that combines the real-time simulation with an interactive manipulation of the propagation environment. We identified three major interaction tasks: Adjustment of visualization and simulation parameters, setup and modification of transmitter sites and the manipulation of the city model which provides the computational basis for the simulation algorithm. All computational intensive tasks were performed on the GPU to achieve a real-time response directly in the *Virtual Environment* (VE). All user input is immediately communicated to the simulation algorithm which updates all propagation predictions such that the effect is instantly visible to the user. An expert review of two domain experts of communication theory revealed a promising potential of the VR interface. In general, they asked for more interactive control over propagation simulation settings such as radiation pattern or power regulation. They especially liked the overview in the VE and in particular the possibility to create missing buildings on the fly. They stated that the overall application of the presented algorithms and VR interface offers promising potential for finding weak spots in an initial or reviewing planning phase of communication networks. We hope that by coupling simulation and manipulation with a VR interface we contributed in the understanding of the underlying mathematical models and algorithms required for planning and optimizing radio networks.

CHAPTER 3

FIBER PATHWAY ESTIMATION

Abstract Understanding the connectivity structure of the human brain is a fundamental prerequisite for the treatment of psychiatric and neurological diseases. With *Diffusion Tensor Imaging* (DTI) we can measure water diffusion and in turn can infer white matter fiber tracts in the living human brain. However, due to the relative low resolution of DTI (about 2 – 3 mm^3) as compared to the diameter of an axon (about 1 μm) only dominant fiber directions are accounted for. By applying magnetic field gradients from different spatial directions, the inherent uncertainty within the diffusion data can be assessed and taken into account by probabilistic tractography when inferring the course of fiber tracts. In principle, the probability of how likely a fiber bundle takes its course through a particular voxel is calculated for any two voxels in the brain.

In this chapter, we address two major aspects of probabilistic tractography, computation and visualization, by presenting (1) an effective algorithm that is able to compute probabilistic fiber tracts in real-time and (2) a subsequent real-time exploratory visualization of its results. A real-time probabilistic tracing is achieved by a parallel implementation on the GPU. Besides the basic probabilistic streamline integration, we describe how state-of-the-art extensions like multi-fiber orientation and loop checking can be adapted to benefit from the GPU's many-core architecture. Our visualization approach focuses on the assessment of fiber probabilities in relation to their structural context. In particular, the comprehension of the course of the fiber in relation to its confidence is one of the most crucial steps. We employ a semi-transparent direct volume rendering technique to display fiber tracts in combination with anatomical landmarks. A VR interface lets the user take an active role in the exploration process by using magic lens techniques to disambiguate between data modalities and to support the understanding of the structure-function relationship. Moreover, exploratory aspects are supported by a direct coupling between the computation and visualization of fiber tracts.

The remainder of this chapter is structured as follows. After a brief motivation and problem statement in Section 3.1, we present the technical background and review related work in Section 3.3. Section 3.4 introduces a model of global connectivity and describes the algorithmic aspects of probabilistic tractography on the GPU. We describe the visualization and interaction of our VR interface in Section 3.5. Section 3.6 briefly discusses the anatomical validity of our fiber tract computation and then focuses on the computational performance in terms of run time and speedup. We conclude the chapter in Section 3.7.

3.1. Motivation

Neuroscientific research aims at understanding the structure-function relationship in the brain. Networks of communicating brain areas are required to fulfill motor, sensory as well as all mental and cognitive activities. The structural basis of such networks are nerve fibers connecting the participating brain areas. A profound knowledge about this connectivity structure is therefore necessary for understanding the computational activity of the brain. Consequently, the concept of the human "Connectome" has recently evolved a research strategy comparable to the Human Genome Project. One major goal of the Connectome Project is a complete mapping of the fiber bundles in the brain.

Diffusion tensor magnetic resonance imaging (DT-MRI) allows the assessment of white matter fiber tracts by inferring the course of fibers by measuring water diffusion in the living human brain. Based on their Brownian motion, water molecules prefer to move along directions with lowest resistance, which in the brain is provided along the Myelin sheaths. By applying magnetic field gradients from different spatial directions, the uncertainty within the diffusion data can be estimated and used for consecutive analysis. An effective *Diffusion Tensor* (DT) can be estimated within each voxel and leads to derived quantities such as mean diffusivity, principal diffusion direction and anisotropy of the diffusion ellipsoid [11].

To reconstruct fiber pathways based on the diffusion data, two main methods are currently used: deterministic tractography, and probabilistic tractography. Deterministic tractography tries to find the path from a seed to a target voxel based on the main diffusion direction within each voxel on the way. Uncertainty within the course of the fiber pathway is usually neglected. In contrast, probabilistic tractography explicitly accounts for the uncertainty of the actual fiber tracts. A local probability distribution of the diffusion direction is calculated for each voxel. A probabilistic algorithm then estimates global connectivity by finding the most probable course of a fiber between a seed and a target voxel. At each voxel the next fiber direction is determined based on the local probability distribution and its prior course [15]. As a result, probabilistic tractography does not provide a single fiber tract, but a probability distribution of possible fiber path-

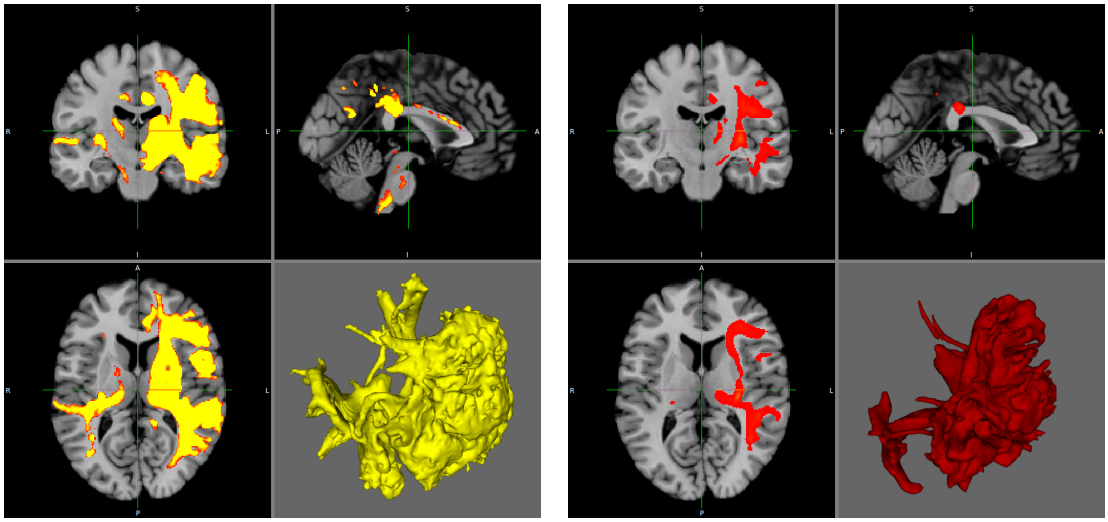


Figure 3.1.: Thresholding probabilities in 2D and corresponding isosurfaces in 3D. The result is heavily influenced by the choice of probability thresholds. The desired result is shown in Figure 3.2.

ways between seed and target voxels, ranging from voxels with a large number of passed traces to voxels with only a low number of passes.

3.2. Problem Statement

A major issue of current 3D visualization techniques in common DTI analysis tools is that no indication of uncertainty in the fiber tracts is contained in the final renderings. For instance in Figure 3.1 and Figure 3.2 (left), the rendering of tracts is achieved by extracting an isosurface from the fiber tract but with no further clues to anatomical details or probability distribution within the fiber tract. However, anatomical context information is crucial for the registration of the most likely course of a fiber pathway in relation to structural landmarks.

Moreover, a static visualization alone is often not sufficient. For a deeper understanding of the complex structures, interactivity is the key to provide context and to disambiguate structural relationships. A main challenge lies in the computational complexity of the underlying algorithm.

Requirements We address the above problem statement by formulating five conceptual requirements of an interactive exploration system based on an interdisciplinary discussion with DTI domain experts as follows: (1) The visualization should empha-

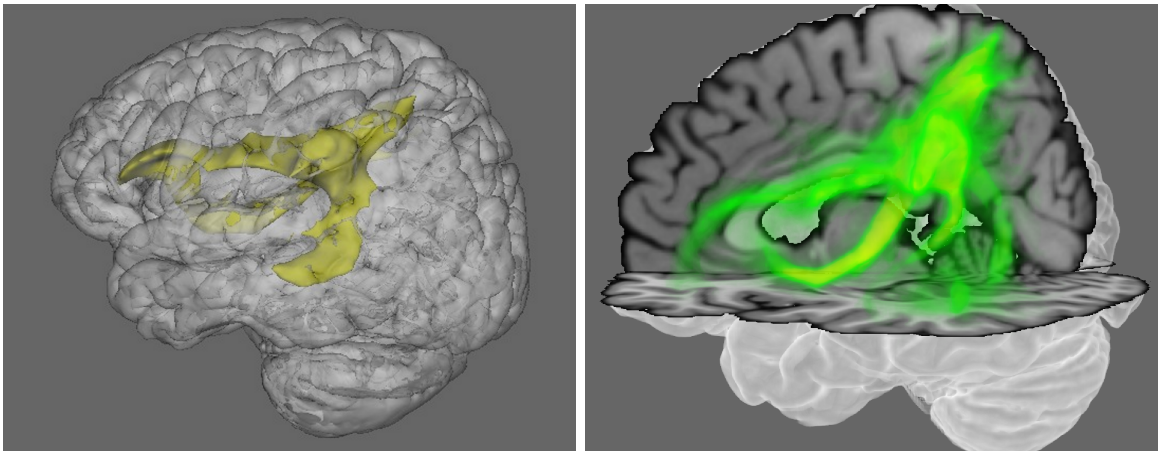


Figure 3.2.: The same fiber tract as in Figure 3.1 with desired thresholds depicted as an isosurface (left) and as probabilistic rendering (right). The probabilistic rendering also conveys the probability distribution within the fiber tract.

size spatial patterns and present the three-dimensional physical structure in an intuitive fashion. (2) The final rendering should convey the uncertainty within each fiber tract. (3) The location of fiber tracts within the human brain should easily be deduced by the anatomical context. (4) The understanding of the structure-function relationship should be supported by an on-demand computation of probabilistic fiber pathways. (5) None of the above requirements may interfere with the interactivity of the exploration system.

Approach We embed the interactive exploration system into a VE, which not only improves spatial perception due to stereoscopic projections but also enables the use of direct interaction techniques such that the user becomes an integral part of the visualization pipeline. We address the visualization of probabilistic fiber tracts by a direct volume rendering approach in order to provide semi-transparent renderings of the uncertainty in combination with structural information in real-time. A magic lens interaction metaphor, which we will refer to as *virtual flashlight*, lets the user gain fine-grain control over the amount of visible anatomical context. In particular, we implement the probabilistic fiber tracking algorithm in parallel on the GPU, thereby reducing the response time upon a seed query such that the exploration of global probabilistic connectivity becomes an interactive procedure.

We realize the above described system by using four basic techniques that have been adapted or developed further where required: (1) An interactive rendering of multimodal volumetric data sets in a VE, (2) a direct 3D magic lens interaction to disambiguate between data modalities, (3) a real-time selection within the VE with respect to special requirements due to data set characteristics, and (4) a parallel implementation of a probabilistic fiber tracking algorithm that enables interactive connectivity exploration.

The first three methods (rendering, interaction, selection) are essentially state-of-the-art algorithms that have been customized to explicitly work in VEs. We will therefore only briefly discuss them in Section 3.5 together with the VR prototype application. The main contribution of this chapter is the parallel implementation of the probabilistic fiber tracking algorithm which will be presented in more detail in Section 3.4.

3.3. Background

This section introduces basic DTI concepts, related work and the mathematical notation that will be used throughout this chapter.

3.3.1. DTI and Tractography

The following background information on DTI and Tractography is summarized from Tuch et al. [143] and Behrens et al. [18]. The neural architecture of the human brain consists of brain cells (neurons) that in turn consist of a cell body which processes signals received at its dendrites and transmits signals down its axons. Axons are responsible for passing signals to connected cells. Each axon is surrounded by a fatty substance known as Myelin to provide electrical insulation. Diffusion based MRI is a non-invasive technique that provides insight into this tissue architecture at the microscopic level. The diffusion of water molecules [55] (Brownian motion) is used as a macroscopic probe of orientation of axonal fibers in brain's white matter. Hereby, the technique relies upon the structure of the myelinated axonal sheaths. If a water molecule undergoing random motion encounters a barrier such as a Myelin wall, it is more likely to diffuse along the Myelin sheaths than across them. Thus, in white matter water diffusion is usually anisotropic.

Basser et al. characterized this diffusion anisotropy in [11]. They first introduced *Diffusion Tensor Imaging* (DTI) by allowing the diffusion process to be different in different directions. The diffusion is hereby characterized by a positive definite and symmetric 3×3 matrix which is commonly referred to as the *Diffusion Tensor* (DT) $\mathbf{D}$. Its three eigenvectors $\epsilon_1, \epsilon_2, \epsilon_3$ are orthogonal and form the principle diffusion directions. Thus, the DT represents a local orthogonal coordinate system. Each orthogonal axis is associated with an independent diffusion coefficient that are known as principle diffusivities and correspond to the three eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. The DT can be written as

$$\mathbf{D} = (\epsilon_1|\epsilon_2|\epsilon_3) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} (\epsilon_1|\epsilon_2|\epsilon_3)^T . \tag{3.1}$$

The DT allows the extraction of several scalar quantities, in particular the overall diffusivity and the diffusion anisotropy within a voxel. We will briefly introduce the most common ones and refer to [14] for a more general review of anisotropic water diffusion.

The *overall diffusion* in a voxel is described by the trace of the DT which corresponds to the sum of the eigenvalues

$$\mathrm{tr}\left(\mathbf{D}\right) = \sum_{i=1}^{3} \mathbf{D}_{ii} = \sum_{i=1}^{3} \lambda_i. \tag{3.2}$$

The *diffusion anisotropy* describes the degree to which diffusion in a individual voxel is preferred in one direction over others. Various measures of anisotropy have been proposed like the eigenvalue ratio

$$A_{\mathrm{ER}} = \frac{\lambda_1}{\lambda_3}, \tag{3.3}$$

the normalized eigenvalue ratio [155]

$$A_{\mathrm{NER}} = \frac{\lambda_1 - \lambda_3}{\lambda_1} \tag{3.4}$$

or the volume ratio [13]

$$A_{\mathrm{VR}} = \frac{\lambda_1 \lambda_2 \lambda_3}{\bar{\lambda}^3} \tag{3.5}$$

with $\bar{\lambda} = \left(\lambda_1 + \lambda_2 + \lambda_3\right)/3$. Yet the most commonly used measure is the *Fractional Anisotropy* (FA) [13]

$$\mathrm{FA} = \sqrt{\frac{3}{2\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2\right)}} \sum_{i=1}^{3} \sqrt{\left(\lambda_i - \bar{\lambda}\right)^2}. \tag{3.6}$$

The eigenvalues of the DT can also be used to describe shape properties that reflect the amount of anisotropy as linear c_l, planar c_p and spherical c_s configuration (cf. [157, 2]) with

$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}, \qquad c_p = \frac{2\left(\lambda_2 - \lambda_3\right)}{\lambda_1 + \lambda_2 + \lambda_3}, \qquad c_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}.$$

All expressions are normalized by the trace of the DT and are by design positive and sum to unity. In turn, this means that (c_l, c_p, c_s) can be regarded as a barycentric coordinate that represents a a mixture of linear, planar and spherical shape. It is apparent, that the linear metric offers the highest amount of anisotropy, followed by the planar and the spherical configuration.

In summary, diffusion anisotropy provides a measure for mean orientation of axons in voxels. The *Principal Diffusion Direction* (PDD) as reconstructed from DTI can already reveal major fiber pathways and are most commonly used as a propagator in deterministic streamline tractography. Moreover, the anisotropy measure is commonly used as a stopping criteria to end fiber tracts that enter regions with no principal direction.

However, the diameter of a typical axon is in the range of few μm whereas *Diffusion Weighted Image*s (DWIs) are typically acquired at a resolution of several mm^3 [14]. Hence, there may be hundreds of thousands of axons passing through each measured voxel. In particular, *Magnetic Resonance* (MR) images represent the average of the measured signal from all tissues per slice. Thus, each voxel in the image may represent more than one tissue type (or orientation). This phenomenon is commonly referred to as the *partial volume effect*. Moreover, if axons are crossing or exhibiting different orientations within an individual voxel, the diffusion data will contain only information of limited value. However, in deep white matter, axons tend to organize themselves in large fiber bundles which then in turn can be identified with DTI. At the same time, it is a major limitation of general tractography algorithms if they are restricted to the reconstruction of pathways in deep white matter regions, only.

Visualization of such major fiber pathways have been reported as visually pleasing and impressive, but have been difficult to interpret scientifically. In particular, deterministic algorithms have often failed in regions of low diffusion anisotropy where there is no obvious dominant fiber orientation. For this reason, tractography has also been of limited use when tracing connections into their gray matter targets. A promising approach to address this particular issue is probabilistic tractography as a method for inferring white matter connectivity in the presence of imperfect and low resolution diffusion data. It results in connectivity distributions with a quantitative description of belief in the trajectories. Furthermore, is enables tracing from gray matter sources to gray matter targets which is almost impossible with common streamline algorithms.

3.3.2. Related Work

Fiber Tractography Fiber tractography from *Diffusion Tensor Magnetic Resonance Imaging* (DT-MRI) was first introduced by Basser et. al [10, 12] as a method to calculate continuous fiber trajectories from the diffusion tensors. They successfully reconstructed known anatomic structures as the corpus callosum and the pyramidal tract. However, they found a strong sensibility of their algorithm in regions of low anisotropy. Jones et al. proposed a similar method in [74] as well as Weinstein et al. [154] and Mori et al. in [101], who also gives a good overview of the general tractography methodology in [102].

The strong sensitivity to low anisotropy is partially addressed by probabilistic ap-

proaches. Koch et al. [85] investigated connectivity by a *Monte-Carlo* (MC) simulation to determine whether a particle diffuses between two points based on the components of the local diffusion tensor. Additionally, they assessed functional connectivity by correlating levels of blood oxygenation between gray and white matter voxels. Björnemo et al. followed a similar approach in [27], using an extension to classical MC in order to facilitate importance sampling to estimate probability distributions of possible paths. Another MC approach is proposed by Parker and Alexander [110] who estimated a *probability density function* (*pdf*) of fiber orientation with MC simulation per voxel. They modeled single fiber orientation by a Gaussian density function and multiple fibers by a mixture of Gaussian densities. Moreover, they explicitly took the effect of noise directly from the DWIs into account.

Xiangfen Zhang et al. [164] addressed the issue of reducing inherent noise in DTI. They proposed a wavelet transform with anisotropic nonlinear diffusion to remove noise while preserving texture and edges. Wu and Xie [160] also specifically focused on noise reduction in DTI. Instead of denoising the DWI (which are used to acquire the DTI) directly, they employ on anisotropic filtering technique that is smoothing the tensor considering structure and characteristic of its eigenvalues and eigenvectors.

Mangin et al. [92] followed an alternative approach and inferred anatomical connectivity based on an energy minimization procedure that defines a trade-off between local information on voxel level as provided by the diffusion data and a priori information based on anatomical plausibility. Behrens et al. [15] proposed a general probabilistic tractography algorithm that tries to find the most probable course of a fiber between a seed and a target voxel. In each voxel they tried to find the most likely prosecution of the fiber based on the local probability distribution and its prior course. Other approaches [89, 73] used statistical bootstrapping methods that derive the uncertainty from the data themselves (as opposed from acquisition thereof) to estimate dispersion and errors in classical tractography results. In particular, Bermann et al. [22] combined bootstrapping with Q-ball imaging [144] to further discriminate multiple fiber populations per voxel. Behrens et al. [16] also incorporated multiple fiber orientations within a single voxels in their probabilistic tractography scheme to increase tracking sensitivity. Similar, Qazi et al. [115] proposed a two-tensor model that incorporates two principal diffusion directions to trace through regions of low anisotropy with deterministic tensor streamline integration.

An alternative probabilistic approach based on particle filtering in a non-linear state space model is presented by Zhang et al. in [161]. Recently, Momayyez and Siddiqi [100] investigated a novel approach to fiber tracking by characterizing the underlying water diffusion process as a 3D random walk that is described by stochastic differential equations. They showed that fiber trajectories correspond to curves of least energy between source and sink regions.

Mittmann et al. [98] addressed computational issues of classical streamline fiber track-

ing by comparing implementations on CPU clusters and on GPU. They concluded, that GPU implementations are much faster in general and hence are better suited for interactive applications. Yet, they found CPU clusters to be suitable for batch processing large amounts of fiber tracts due to the maturity of cluster computing. McGraw and Nadar [96] presented a GPU implementation of a stochastic connectivity mapping which is closely related to the present paper. However, the used a different underlying model of global connectivity and do not considered multi-fiber models or loop checking individual streamlines.

Visualization A variety of visualization techniques for tensor fields and tractography have been proposed. Here, we provide only a small overview.

The DT can directly be visualized by glyphs, where small graphical icons represent local tensor properties by mapping tensor eigenvectors and eigenvalues to the orientation and shape of geometric primitives. Instead of ordinary ellipsoids that may mislead the orientation of tensors with rotational symmetry (as common for diffusion tensors), Kindlmann [80] proposed the use of superquadric glyphs. He showed that a superquadric tensor glyph exhibits better shape and orientation clues with better profile and shading. In [156] Westin et al. presented an approach for a similar problem where they used a composition of several geometric shapes to convey the local diffusion tensor. They proposed to use several glyphs: a sphere with the radius of the smallest eigenvalue, a disk with the radius of second largest eigenvalue and a rod with the length twice the largest eigenvalue. Additionally, the composited glyphs were colored according to their shape property (linear, planar or spherical). However, in sections of crossing fibers the glyphs take on a misleading spherical shape due to partial volume effects. In [82] Kindlmann et al. used glyph packing for multivariate visualization of DT-MRI scans of a patient with a brain tumor to enable data inspection at discrete points while additionally providing large-scale continues structures for reference. Their approach is based on carefully distributing glyphs throughout the field (using energy profiles) to reduce the visual emphasis of the regular sampling grid of the tensor data and emphasize continuous features.

Glyph based visualization techniques suffer from visual clutter when too many structures are represented and the local information they represent is diminished. Direct volume visualization techniques have also been generalized and applied for the visualization of diffusion tensor fields in order to present large-scale structures. Kindlmann and Weinstein [81] addressed the use of direct volume rendering for 2nd order tensor fields by directly assigning color, lighting and opacity with a two-dimensional color map on the unit sphere.

Texture-based visualization techniques perform filtering to reveal the local curvature of vector fields by texture orientation and frequency. The texture synthesis technique

known as *Line Integral Convolution* (LIC) was introduced by Cabral and Leedom [33]. A low-pass filter convolves an input noise texture along streamlines to exploit spatial correlations in the flow direction. The resulting image exhibits highly correlated intensities along streamlines and uncorrelated intensity values across streamlines. Hsu [68] generalized LIC to the visualization of diffusion tensor fields by incorporating the principal and second-order anisotropy of the water diffusion and applied it to the visualization of myocardial architectures. McGraw et al. [97] adapted LIC to cope with singularities in the diffusion tensor field of the dominant eigenvector. Zheng and Pang [165] used LIC for the visualization of anisotropy in symmetric tensor fields. Their proposed technique provides a global continuous representation of the underlying tensor field that reflects all eigenvectors. When the tensors are almost linear, their technique reduces to the original LIC.

Unlike stationary representations, interactive particle tracing [83] has been used for visualizing vector fields of principal diffusion direction. Kondratieva [86] used GPU particle tracing for the visualization of 3D diffusion tensor fields. A large amount of particles can be traced at interactive frame rates to convey the vector field of principal directions. Different visual representation of the particles like a diffusion dependent ellipsoidal shape were used for the visualization.

Moreover, spatial continuity of principal diffusion direction can be conveyed by using streamlines or streamtubes, i.e. integral curves that are tangent to the vector field of principal directions. Zhang et al.[162] combined streamtubes and streamsurfaces for the visualization to emphasize linear and planar anisotropy. Streamtubes are used to represent primarily linear diffusion whereas streamsurfaces represent predominantly planar structures. In their application to brain tumor surgery they found a good correlation of the extracted streamtubes to major neural pathways.

Moberts et al. [99] addressed the issue of visual clutter when visualizing a large amount of individual white matter tracts. They proposed a framework to validate clustering methods for fiber bundles. They found that a hierarchical clustering using a single-link and a fiber similarity measure based on the mean distance performed best among the investigated methods. Enders et al. [50] performed classical streamline tracking with a subsequent clustering of streamline bundles which are then rendered as surfaces that wrap bundles of fibers. The surface is generated around the center line of a fiber bundle. The boundary curves are determined based on the variance of points within equidistant planes with normals that are tangent to the center line. From a surgical point of view, this can be interpreted as a safety zone around fiber bundles. It should be noted that streamline visualization may give a misleading impression of certainty about the location of fiber tracts since a single line is displayed between two points. Moreover, DTI does not measure actual fiber connectivity but water diffusion which can be used to *infer* connectivity.

Other work on fiber tract visualization has specifically focused on how to efficiently

access and distinguish between different fiber bundles and connectivity structures. Zimmermann et al. [168] proposed a method for modeling and classifying white matter tracts using classification trees in conjunction with a spatial representation of the individual fiber to capture the characteristic behavior of specific anatomical structures. Song Zhang et al. [163] presented a technique for clustering integral curves from DTI into anatomically plausible bundles. They employ a sampling and culling strategy for the generation of integral curves with a subsequent clustering based on a proximity measure that is calculated between every pair of curves. Proximity thresholds can be interactively determined by an expert user. Schultz et al. [135] followed an approach that considers the asymptotic behavior of probabilistic fiber tracking and defined concepts of flow topology (critical points, basins, faces) in terms of brain anatomy. Their resulting fuzzy feature definitions reflect the inherent uncertainty in brain connectivity. Sherbondy et al. [136] used dynamic queries to assist the exploration and interpretation process of fiber tracts. They proposed a simple query language in combination with box or ellipsoid-shaped regions to give interactive access to a precomputed database of fiber pathways and their statistical properties. Chen et al. [40] combined glyph and streamline visualization techniques by introducing merging ellipsoids. Each ellipsoid represents a local tensor and either blends with neighboring ellipsoids into a continuous curve representation or breaks away as a local glyph. They also presented an exploration interface to fiber structures in [39] that augments classical 3D visualization with 2D embeddings that removes visual clutter while preserving relationship between fiber bundles. In particular, their framework allows a quick and accurate selection of fiber bundles.

More recent work addressed the issue of parameter sensitivity in fiber tracking methods as presented by Brecheisen et al. [28]. They developed a visualization tool that lets the user explore how small variations in parameter values affect the output of fiber tracking. Prckovska et al. [114] considered recent developments in imaging technologies and presented a multi-field visualization framework that combines classical DTI with *High Angular Resolution Diffusion Imaging* (HARDI). They applied a classification scheme based on HARDI anisotropy to select the fiber model depending whether regions exhibit single fiber bundle coherence, areas of crossing, or more complex fiber structure.

3.3.3. Stochastic Background

When working with uncertainty, randomness and probabilistic expressions, a clear and concise notation is required. Therefore, we will introduce the notation and basic expressions used in this work, which is greatly inspired by Mathar [95]. We will introduce the terms *random variable*, *random vector*, *probability density function*, *joint distribution*, and *conditional and marginal density*. We will refer to the observation space in general as Ω. In one dimension this can be for instance the set of real numbers $\mathbb{R}$.

Random Variable and Density Function Let $f : \mathbb{R} \to \mathbb{R}^+$ be a non-negative real-valued function with

$$\int_{-\infty}^{\infty} f(t)\, dt = 1.$$

A real-valued random variable X is called continuous if

$$F_X(x) = \mathcal{P}(X \leq x) = \int_{-\infty}^{x} f(t)\, dt \quad \forall x \in \mathbb{R}.$$

$F_X(x)$ refers to the *cumulative distribution function* of X. The function f is called the *probability density function* (*pdf*) or simply the *density* of X. We refer to the *pdf* of X also as f_X.

The calculation of probabilities $\mathcal{P}$ with the *pdf* of a random variable X is described as follows. Let $I = (a, b]$ be the interval between $a \in \mathbb{R}$ and $b \in \mathbb{R}$, including a and excluding b. It then holds that

$$\mathcal{P}(X \in I) = \mathcal{P}(a < X \leq b) = \int_{a}^{b} f_X(t)\, dt.$$

$\mathcal{P}(a < X \leq b)$ is so to speak the probability that a realization of X lies in the range between a and b.

Random Vector and Joint Density Often, we are interested in more than only a single parameter or in a subset of parameters. This will require *random vectors*, *joint distributions* and *marginal distributions*. In the following we will mark vectors and matrices by a boldface notation. $\mathbf{A}^T$ will mark the transpose of the vector or matrix $\mathbf{A}$.

A function

$$\mathbf{X} = (X_1, \ldots, X_N)^T : \Omega \to \mathbb{R}^N$$

is called a *random vector* if $X_1, \ldots, X_N$ are random variables.

The *joint distribution* of $\mathbf{X}$ is defined as

$$F_{\mathbf{X}}(x_1, \ldots, x_N) = \mathcal{P}(X_1 \leq x_1, \ldots, X_N \leq x_N)$$

and we write $\mathbf{X} \sim F_{\mathbf{X}}$.

The *joint density* $f_{\mathbf{X}} : \mathbb{R}^N \to \mathbb{R}^+$ of $\mathbf{X}$ is then analogously defined as

$$F_{\mathbf{X}}(x_1, \ldots, x_N) = \int_{-\infty}^{x_N} \cdots \int_{-\infty}^{x_1} f_{\mathbf{X}}(t_1, \ldots, t_N)\, dt_1 \cdots dt_N \quad \forall x_1, \ldots, x_N \in \mathbb{R}.$$

The joint density can be simplified in the special case of *Independent Identically Distributed* (*iid*) random variables $X_1, \ldots, X_N$. Let $\mathbf{X} = (X_1, \ldots, X_N)^T$ be a random vector of *iid* random variables X_i, then the following holds

$$f_{\mathbf{X}}(x_1, \ldots, x_N) = f_{\mathbf{X_1}}(x_1) \cdots f_{\mathbf{X_N}}(x_N) \tag{3.7}$$

and $\prod_{i=1}^{N} f_{\mathbf{X_i}}(x_i)$ is the joint density of $\mathbf{X}$. In fact, $\mathbf{X}$ is called *iid* if and only if equation (3.7) holds.

Conditional Density If we are interested in only a subset of parameters we require the *conditional density*. Let (X, Y) be a continuous random vector with joint density $f_{X,Y}(x, y)$. The *conditional density* of X given Y is defined as

$$f_{X|Y}(x|y) = \begin{cases} \frac{f_{X,Y}(x,y)}{f_Y(y)} & , \text{if } f_Y(y) > 0 \\ f_X(x) & , \text{if } f_Y(y) = 0 \end{cases}.$$

The first case ($f_Y(y) > 0$) can be rewritten as

$$f_{X,Y}(x, y) = f_{X|Y}(x|y) f_Y(y) = f_{Y|X}(y|x) f_X(x).$$

Informally speaking, the chance of seeing both events X and Y is equal to the chance of seeing event X given that we have seen event Y multiplied by the chance of seeing event Y and vice versa. The statement leads directly to Bayes' theorem

$$f_{X|Y}(x|y) = \frac{f_{Y|X}(y|x) f_X(x)}{f_Y(y)}. \tag{3.8}$$

Marginal Density For a random vector $\mathbf{X} = (X_1, \ldots, X_N)^T$ every X_i has the *marginal density*

$$f_{X_i}(x) = \underbrace{\int \ldots \int}_{N-1} f_{\mathbf{X}}(u_1, \ldots, u_{i-1}, x, u_{i+1}, \ldots, u_N)\, du_1 \ldots du_{i-1} du_{i+1} \ldots du_N. \tag{3.9}$$

In general, for a subset of indices $I = \{i_1, \ldots, i_K\} \subset \{1, \ldots, N\}$ the marginal density of the random vector $\mathbf{X}_I = (X_{i_1}, \cdots, X_{i_K})^T$ is

$$f_{\mathbf{X}_I}(x_{i_1}, \ldots, x_{i_K}) = \underbrace{\int \ldots \int}_{N-K} f_{\mathbf{X}}(\ldots) \prod_{j \notin I} du_j.$$

Thus, the marginal density of a subset of random variables is obtained by integrating over the joint density over all the values of the other variables.

Application Context Random variables can be used to account for uncertainty within a system that we want to model. By observing data from real-world measurements of the system, we can make assumptions how these random variables should be distributed in our model. Informally speaking, we want to learn from the observation whereas our true interest lies not in the observation itself but in specific features of the system that generated that data.

Let M be a model of such a system with parameter vector $\omega \in \Omega$ that represents unknown influences. In general, the parameter vector may model noise in the system, or in the context of radio waves, building materials and information about vegetation or land use. In the context of fiber tractography in the human brain, the parameters can be used to describe the distribution of local fiber directions. Each respective model defines how the parameters (and other input data) are combined with one another to generate a prediction of a possible outcome. Let a real-world measurement of the system be described by Y. We can write the probability of predicting the data Y with our model M and a corresponding parameter set ω as $\mathcal{P}(Y|\omega, M)$. Ideally, we will find a set of parameters $\hat{\omega}$ that maximizes that probability. We could then say that M with the optimal parameters $\hat{\omega}$ is a good approximation to the actual physical system that we wanted to model. In the *Maximum Likelihood* (ML) formulation this set of parameters is described as

$$\hat{\omega} = \arg\max_{\omega} \mathcal{P}(Y|\omega, M).$$

Thus, we would vary over all possible parameter vectors and choose the one that maximizes the conditional probability. It is apparent that this approach may not be feasible for real-world problems, in particular, if the parameter domain is continuous. Moreover, it is more common to update the belief in model parameters based on the observed data and the model: $\mathcal{P}(\omega|Y, M)$, which is also referred to as the posterior distribution of the model parameters. Again, this can be rewritten according to Bayes and yields

$$\mathcal{P}(\omega|Y, M) = \frac{\mathcal{P}(Y|\omega, M)\,\mathcal{P}(\omega|M)}{\mathcal{P}(Y|M)}$$

which is again not trivially computed since the denominator

$$\mathcal{P}(Y|M) = \int_{\Omega} \mathcal{P}(Y|\omega, M)\,\mathcal{P}(\omega|M)\,d\omega$$

is an integral which is usually not solvable analytically. However, we can approach the integral numerically with *Markov-Chain-Monte-Carlo* (MCMC) sampling techniques like Metropolis-Hasting or Gibbs sampling. In this case, we usually restrict the sampling to the distribution of parameters of interest only which can be obtained by marginalization

$$\mathcal{P}(\omega_I|Y, M) = \int_{\Omega - I} \mathcal{P}(\omega_I, \bar{\omega}_I|Y, M)\,d\bar{\omega}_I.$$

ω_I are all parameters of interest and $\bar{\omega}_I$ are all the other parameters.

The latter derivation is the key component to estimating the distribution of local fiber orientations (details can be found in [17, 16] and [110]). In the next section, we will now jump directly to the inference of global connectivity.

3.4. Probabilistic Tractography

The algorithm for probabilistic tractography consists of two major steps: (1) estimate the distribution of parameters of interest locally at every voxel and (2) infer global connectivity between any two voxels based on the local distributions.

The local parameters of interest are the local fiber orientations that can be expressed as spherical coordinates (ϕ, θ). We consider the local model of diffusion according to the diffusion tensor model [11, 13] for a single fiber orientation as well as for multi-fiber fields [16]. The estimation of local parameters has to be performed only once, and hence can be done in an off-line preprocessing step. Details of the local parameter estimation are sketched in Section 3.3.3 and discussed in detail by Behrens et al. in [17] and [16]. For the remainder of this chapter, we assume that the local parameter distributions have already been calculated.

The estimation of global connectivity [17] is derived from the local fiber orientations. It describes the probability that a connection between any two points passes through a particular voxel. Thus, the global connectivity is given as a spatial probability distribution with a discrete probability value at each voxel. The estimation of global connectivity has to be performed very often: for every seed voxel of interest and for a large number of samples to increase the signal-to-noise ratio in our inherent probabilistic framework. Due to the independence of computation between voxels, this tasks maps well to an implementation on the GPU.

3.4.1. A Model of Global Connectivity

This section introduces the general notation as used in the remainder of this chapter and provides the mathematical model of global connectivity as introduced by Behrens at al. in [17]. Certain formulations may differ from the original model in order to ease the correspondence in the algorithmic description that follows in Section 3.4.2.

Let $\mathcal{V}$ be the space of all voxels, $U, V \in \mathcal{V}$ individual voxels, $N_\mathcal{V}$ the number of voxels, and V_i the ith voxel. We refer to the observed DTI data for all voxels as $Y_\mathcal{V}$. Let $(\theta, \phi)_\mathcal{V}$ be a complete set of (principal) diffusion directions. For the remainder of this chapter, we assume that we already have the local parameter distribution of fiber orientation $f_{\theta,\phi|Y_\mathcal{V}}$ for the single-fiber case and $f_{\theta_1,\phi_1|Y_\mathcal{V}}, \ldots, f_{\theta_M,\phi_M|Y_\mathcal{V}}$ for multi-fiber fields with M modeled fibers per voxel.

In case of no uncertainty about fiber orientations we can write the probability that a connection exists between voxels U and V given the (deterministic) knowledge of local

fiber orientations as a delta function

$$\mathcal{P}\left(\exists\ U \leftrightsquigarrow V | \left(\theta,\phi\right)_{\mathcal{V}}\right) = \begin{cases} 1 & \text{, there exists a streamline from } U \text{ to } V \\ 0 & \text{, otherwise} \end{cases}.$$

It can be computed by common streamline algorithms that follow the path of principal diffusion direction. The probability will be one if there exists a streamline connecting U and V and zero otherwise.

When uncertainty about fiber orientation is introduced into the model the connectivity probability must be expressed as the existence of connection and all local fiber orientations given the observation. This can be expressed as $\mathcal{P}\left(\exists\ U \leftrightsquigarrow V, \left(\theta,\phi\right)_{\mathcal{V}} | Y_{\mathcal{V}}\right)$, transformed into its *pdf* notation, reformulated with Bayes and expanded to

$$\begin{aligned} f_{\exists\ U \leftrightsquigarrow V,\left(\theta,\phi\right)_{\mathcal{V}}|Y_{\mathcal{V}}} &= f_{\exists\ U \leftrightsquigarrow V|\left(\theta,\phi\right)_{\mathcal{V}}} f_{\left(\theta,\phi\right)_{\mathcal{V}}|Y_{\mathcal{V}}} \\ &= f_{\exists\ U \leftrightsquigarrow V|\left(\theta,\phi\right)_{\mathcal{V}}} \prod_{i=1}^{N_{\mathcal{V}}} f_{\left(\theta,\phi\right)_{V_i}|Y_{V_i}}. \end{aligned}$$

Hence, the connectivity probability is the product of all probabilities of fiber connections that would contribute in the streamline case for any fixed set of fiber orientations. By marginalizing over all possible fiber orientations at every voxel the connectivity probability of any two points U and V given the observation data $Y_{\mathcal{V}}$ is described by

$$\begin{aligned} f_{\exists\ U \leftrightsquigarrow V|Y_{\mathcal{V}}} = \int_0^{2\pi} \int_0^{\pi} \cdots \int_0^{2\pi} \int_0^{\pi} & f_{\exists\ U \leftrightsquigarrow V|\left(\theta,\phi\right)_{\mathcal{V}}} \\ & f_{\theta_{V_1},\phi_{V_1}|Y_{\mathcal{V}}} \ldots f_{\theta_{V_{N_{\mathcal{V}}}},\phi_{V_{N_{\mathcal{V}}}}|Y_{\mathcal{V}}} d\theta_{V_1} d\phi_{V_1} \ldots d\theta_{V_{N_{\mathcal{V}}}} d\phi_{V_{N_{\mathcal{V}}}}. \end{aligned} \tag{3.10}$$

Since all local *pdf*s are generated by a sampling technique (e.g. MCMC, cf. Section 3.3.3) no analytical solution exists for (3.10). In consequence, it has to be computed numerically as detailed in the following section.

3.4.2. Algorithms

We subdivide the computation into four basic parts and two extensions. Algorithm 12 describes the core of the tracing procedure: starting from a seed voxel, the current position is propagated along a random direction at each step until a stopping criteria is met. Algorithm 13 shows how a local (random) direction is determined by a query into the discrete representation of local fiber distributions. Algorithm 14 updates the current diffusion direction based on the previous direction and a new local fiber orientation. Finally, all probabilistic streamline samples are collected and recorded as sketched in Algorithm 15 to construct a spatial *pdf* that reflects the connectivity distribution of the original seed voxel to every other voxel.

Additionally, these algorithms can be extended to support multiple fiber orientations per voxel as well as to check for artificial loops. Algorithm 16 shows how multiple fiber orientations per voxel are incorporated into the tracing procedure. The direction that matches the current streamline direction best is selected among all possible local fiber directions. We introduce an additional random component in Algorithm 17 to compromise between the memory footprint and accuracy of the loop check. This reduces the chance of artificially increasing the probability within loops. We will now discuss the individual algorithms in greater detail.

Probabilistic Streamline Integration Algorithm 12 sketches the main compute kernel of the probabilistic streamline procedure. It generates a probabilistic streamline that starts at the seed voxel U. Since a voxel is usually represented by discrete cell indices we first transform it into a continuous form to support sub-voxel accuracy in the tracing procedure. Similar to common streamline algorithms, the front of the streamline is propagated by moving a certain step Δ into the direction of diffusion. Typical step sizes correspond to half or a quarter voxel. Since diffusion directions are represented as *pdf*s we draw a random sample from the corresponding distribution (cf. Algorithm 13). This procedure is repeated until a stopping criteria is met. Possible stopping criteria may be an anisotropy measure (cf. Section 3.3.1), an anatomical curvature threshold or simply leaving the voxels that actually belong to the brain.

Random Neighbor Interpolation In principle, the local *pdf*s exist only in continuous space. However, they are provided on discrete grid points due to the MR acquisition method. Any kind of interpolation between the fiber orientations may result in averaged values which no longer presents the original orientation very well. As proposed by Behrens et al. [17] and also adapted by McGraw and Nadar in [96] we use a probabilistic data interpolation scheme. The main idea is to exploit the inherent probabilistic nature of the method and simply pick one of the neighboring fiber orientations at random. The interpolation scheme is set up such that in the parameter dimension u, the probability

Algorithm 12 TRACE_PROBABILISTIC_STREAMLINE(Voxel U)

Note: Generate a probabilistic streamline that starts at voxel U
  Vertex $p \leftarrow U$ {start at U, trace with sub-voxel accuracy}
  repeat
    $s^{\rightsquigarrow} \leftarrow p$ {append current position to streamline}
    $(\theta, \phi) \leftarrow$ PROB_SAMPLE$(p, f_{\theta,\phi|Y_{\mathcal{V}}})$
    {draw a random sample at p from $f_{\theta,\phi|Y_{\mathcal{V}}}$, cf. Algorithm 13}
    $\vec{d} \leftarrow$ UPDATE_DIR$(\vec{d}, \theta, \phi)$ {update direction}
    $p \leftarrow p + \Delta\vec{d}$ {move p along diffusion direction}
  until stopping criteria
  return $s^{\rightsquigarrow}$

of choosing data from location $\lfloor u \rfloor$ is

$$\mathcal{P}\left(\lfloor u \rfloor | u\right) = \lceil u \rceil - u$$

and for choosing $\lceil u \rceil$

$$\mathcal{P}\left(\lceil u \rceil | u\right) = 1 - \mathcal{P}\left(\lfloor u \rfloor | u\right).$$

$\lfloor . \rfloor$ denotes the floor and $\lceil . \rceil$ the ceiling function. This is implemented in Algorithm 13 by computing the fractional part of the sub-voxel query position and drawing a uniformly distributed sample $\rho \sim U(0, 1)$. If the sample is less than the fractional part we round down, otherwise we round up to the next integer voxel. Hence, two probabilistic streamlines that pass through the same voxel may have different interpolated values and therefore can experience different diffusion directions. Another random sample is drawn to select the bin of the discrete *pdf* representation for the interpolated voxel position. Hence, four uniformly distributed random values have to be drawn in the process, three for each component of the probabilistic interpolation and one for selecting the bin.

Direction Update Algorithm 14 sketches how the next streamline direction is determines based on the previous direction and a new fiber orientation. Its main purpose is to have a consistent direction to avoid going forth and back. We are not applying any curvature thresholds here, which are commonly used in deterministic algorithms, since we rely on loop checking (cf. Algorithm 17) as a major stopping criteria.

Creating a Spatial Probability Distribution Function With probabilistic streamlines generated with Algorithm 12 we are already able to create a spatial *pdf* that describes the connectivity distribution of the seed to every other voxel. The construction is sketched in Algorithm 15. Each voxel is initialized with zero, i.e. no connectivity. Each probabilistic streamline is traversed and at every new (integer) voxel position the connectivity

Algorithm 13 PROB_SAMPLE(p, $f_{\theta,\phi|Y_\mathcal{V}}$)

Note: Sample from a discrete *pdf*
{probabilistic interpolation}
for all $i \in \{x, y, z\}$ **do**
$r \leftarrow \lceil p.i \rceil - p.i$
$\rho \leftarrow$ UNIFORM_RANDOM(0,1) {draw random sample $\rho \sim U(0,1)$ }
$U_i \leftarrow \begin{cases} \lfloor p.i \rfloor & , \rho < r \\ \lceil p.i \rceil & , \text{otherwise} \end{cases}$
end for
$\nu \leftarrow$ UNIFORM_RANDOM(0,1) {choose among B bins by drawing random sample $\nu \sim U(0,1)$}
$b \leftarrow \lfloor \nu \cdot (B-1) + 0.5 \rfloor$ {round to discrete index}
discrete_pdf $\leftarrow$ SAMPLE_PDF($f_{\theta,\phi|Y_\mathcal{V}}$) {transform the continuous *pdf* $f_{\theta,\phi|Y_\mathcal{V}}$ into a discrete representation (cf. [17], usually done in a preprocessing step) }
return discrete_pdf[b][U]{return sample from the discrete *pdf* at voxel U of bth bin}

Algorithm 14 UPDATE_DIR($\vec{d}_{\text{prev}}, \theta, \phi$)

Note: Update diffusion direction based on previous direction and local fiber orientation
$\vec{d} \leftarrow \text{Vector}(\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)$
if $\left\langle \vec{d}, \vec{d}_{\text{prev}} \right\rangle < 0$ **then**
$\vec{d} \leftarrow -\vec{d}$
end if
return $\vec{d}$

count is increased. This procedure is iterated until all streamlines are processed. A sufficient number of streamlines is required to convergence in a stable distribution. A normalization over the total number of streamlines guarantees a value range between zero and one. As discussed in Section 3.4.4, the construction of the spatial *pdf* is a major synchronization point in the computation. Data from all individual samples is collected and is read from and written into the same block of memory.

Algorithm 15 CREATE_SPATIAL_PDF(Streamlines $s_1^{\rightsquigarrow}, \ldots, s_K^{\rightsquigarrow}$)

Note: Creates a spatial *pdf* from a set of probabilistic streamlines

```
PDF ← MEMSET(N_V, 0) {initialize all voxels of the pdf with zeros}
for all i ∈ {1, ..., K} do
  cur ← VOXEL_POS(s_i^⇝[1]) {retrieve initial voxel position}
  PDF[cur]++ {increase voxel count at current position}
  for j = 2 → LENGTH(s_i^⇝) do
    next ← VOXEL_POS(s_i^⇝[j]) {retrieve next voxel position}
    if next ≠ cur then
      cur ← next
      PDF[cur]++ {increase voxel count for every new voxel position}
    end if
  end for
end for
for all i ∈ {1, ..., N_V} do
  PDF[i] ← PDF[i] /K {normalize PDF by total number of streamlines}
end for
return PDF
```

Algorithm 16 MULTI_FIBER(p, $\vec{d}_{\text{prev}}$, $f_{\theta_1,\phi_1|Y_\mathcal{V}}, \ldots, f_{\theta_M,\phi_M|Y_\mathcal{V}}$)

Note: Determines next fiber orientation in a multi-fiber field.

```
for all i ∈ {1, ..., M} do
    (θ_i, φ_i) ← PROB_SAMPLE(p, f_{θ_i,φ_i|Y_V})
    d_i ← Vector(cos φ_i sin θ_i, sin φ_i sin θ_i, cos θ_i)
    α_i ← ⟨d_prev, d_i⟩
end for
k ← INDEX_OF_MINIMUM(|α_1|, ..., |α_M|)
p ← p + Δd_k {move p along selected diffusion direction}
```

3.4.3. Extensions

In addition to the basic algorithm we implemented two major extensions: a multi-fiber model and loop checking, which both improve the overall signal-to-noise ratio and offer significant advantages in the sensitivity when tracking in non-dominant fiber populations or in regions of low anisotropy.

Multiple Fiber Orientations Behrens et al. [16] presented a direct extension to the probabilistic tractography method that incorporated multiple fiber orientations per voxel. The basic principle relies on automatic relevance determination when constructing the local distributions of fiber orientation. If supported by the data, their method locally extracts the distribution of more than one fiber orientation. The probabilistic tracing (cf. Algorithm 12) can be extended to incorporate that knowledge. The main idea is to draw a sample from all local distributions and compute each local fiber direction every time a streamline is updated. Then, the fiber direction that is closest to parallel to the current streamline direction is chosen to propagate the current front. The general approach is sketched in Algorithm 16. In their experiment with a multi-fiber fit to a 60 direction data set [16], Behrens et al. found that about a third of voxels with an FA > 0.1 were able to support more than one fiber orientation, but no single voxel in their data set supported more than two orientations. Therefore, we optimized our actual implementation for the two-fiber case which has certain benefits on the compute device as discussed in Section 3.4.4.

Stopping Criteria and Loop Checking Stopping criteria are usually used to discard invalid streamlines from the overall computation. Common stopping criteria are usually thresholds on the anisotropy measure or on the local curvature. Although heavily used in most deterministic algorithms, strict stopping criteria may reduce the ability of the algorithm to trace through regions of noise, or low anisotropy. Conversely, if stopping criteria are chosen too lax, streamlines may turn back to already visited locations, thus,

Algorithm 17 LOOP_CHECK(ID,cur,next,loop_check)

Note: Perform loop checking on a coarse voxel grid based on per thread memory
  if cur == next **then**
    return false {continue tracing within same voxel}
  end if
  belief ← loop_check[ID + next] {our belief that we have seen the supervoxel before}
  $\rho \leftarrow$ UNIFORM_RANDOM(0,1) {draw random sample $\rho \sim U(0,1)$ }
  if belief > ρ·threshold **then**
    return true {loop detected: reject new streamline front and stop tracing}
  else
    loop_check[ID + next] ← belief++ {increase our belief}
    return false {accept new streamline front and continue tracing}
  end if

artificially increasing the probability in these loops which is usually an undesired behavior. Checking for loops in the probabilistic streamlines can therefore greatly reduce the need for curvature thresholds without introducing major changes in the probability distribution. However, two main issues have to be addressed in the implementation of loop checking: computational performance and memory requirements. In particular, the memory requirement in terms of size and number of memory accesses is the limiting factor since the actual tracing is performed in lightweight compute kernels on the device.

We considered two approaches how to implement loop checking. We could either store the path history with each streamline front or for each voxel store whether it has seen a particular streamline. The first approach would require a dynamic memory block (or a fixed block of by memory with enough excess capacity) with a subsequent self-intersection test. In any case, this easily introduces a lot of divergent branches for a set of streamlines with different histories. Again, this is an undesired behavior when executing a large number of lightweight compute kernels in parallel. The second approach will not introduce a divergent branch, the loop check is simply a lookup of the streamline's current front position in memory. However, the additional memory footprint can be substantial. Each voxel has to offer read/write access for a bit vector the length of the current sample population.

Since the tracing algorithm is inherent probabilistic, a probabilistic approximation of the loop check may not lead to significant different results. Thus we propose a hybrid approach of a low-resolution representation of visited areas in combination with a random component that represents the *believed* visited state of a voxel.

We reduced the memory footprint of the second approach by combining blocks of voxels into supervoxels, thereby creating a coarse resolution level of the tracing environment for loop checking. Additionally, we introduced a random component to account for artifacts

that may result from the lower resolution. The inputs to the loop checking Algorithm 17 are the current thread ID, the current and next supervoxel position of the streamline and a pointer to the memory block reserved for loop checking a fixed number of streamlines concurrently. The size of this memory block must be the number of concurrent threads times the number of supervoxels. We reserve eight bits for each entry that is initialized with zero and increased each time a streamline visits the supervoxel. To prevent an overflow we stop increasing the counter at 255 visits. The value of each entry presents our belief that the particular supervoxel has been visited. Each time a streamline's *next* position enters a previously visited supervoxel we draw a uniformly distributed random sample. If our belief of having seen the supervoxel is greater than the random sample we reject the new front of the streamline and stop its tracing, otherwise we accept the new front and continue tracing while increasing our belief of having seen the corresponding supervoxel. Thereby, we increase the rejection probability each time a sample is accepted.

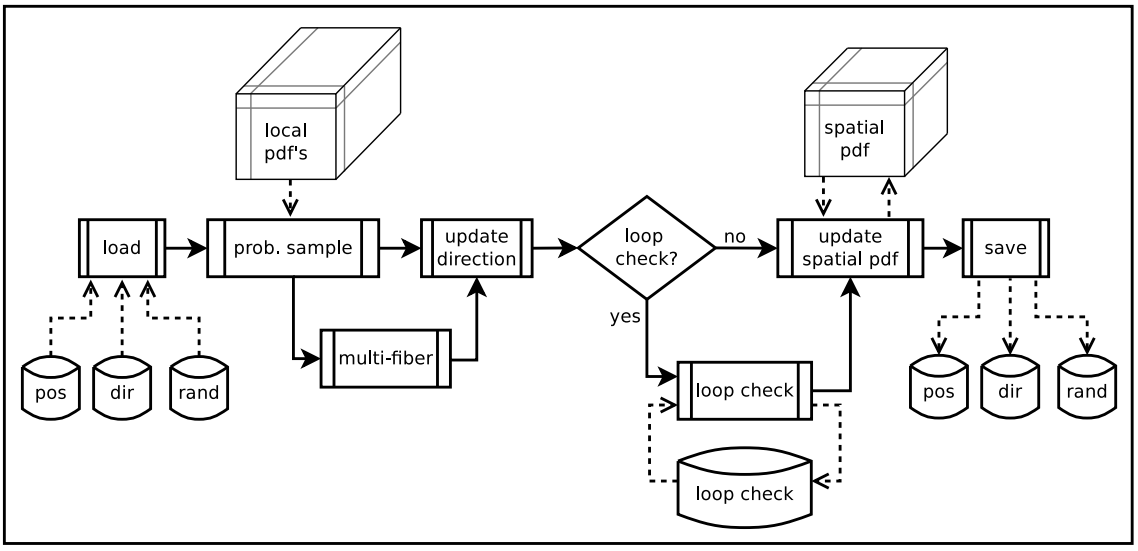


Figure 3.3.: Layout of the compute kernel. Solid arrows symbolize control flow whereas dashed arrows represent memory accesses. The arrow direction indicates a read (outgoing arrow) or write access (incoming arrow).

3.4.4. Implementation Details

Computational Flow For efficiency reasons, the computational flow of our implementation differs slightly from the algorithmic description in some details. First, the computation has been packed into one kernel execution, see Figure 3.3. Each kernel invocation computes one update step for all streamline samples concurrently. Second, the construction of actual streamlines is omitted in order to reduce the memory footprint of the computation. Intermediate results are directly recorded in the spatial *pdf*. Consequently, visual feedback can be provided by rendering the scalar field of the spatial *pdf* after each step.

At the beginning of an update step, each thread reads its sample position, direction and random generator state. The local fiber orientation is retrieved and the current fiber direction updated accordingly (Algorithm 13 and Algorithm 14). If multiple fiber orientations per voxel are available, the one that is best aligned to the current direction determines the next sample position (Algorithm 16). If a loop is detected (Algorithm 17), a sample is marked as invalid and is discarded in subsequent computations. Whenever a sample position advances to a new voxel position the corresponding entry in the spatial *pdf* is increased accordingly (Algorithm 15). Finally, the updated sample position, direction and random generator state is saved and the computation returns the control back to the host. This procedure is iterated until the spatial *pdf* has converged or a certain number of steps has been reached.

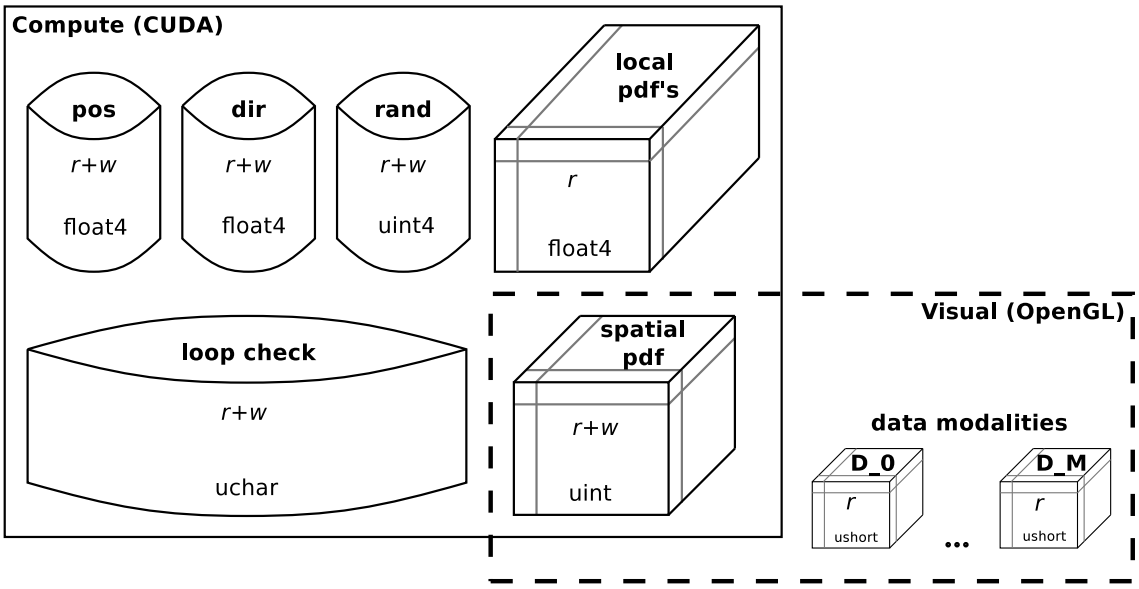


Figure 3.4.: Memory layout of the probabilistic streamline integration. Read and write access is required for each sample's position, direction, random generator state, and loop check data. The spatial *pdf* memory is updated at every step of the computation and can be used to visualize intermediate results.

Memory Layout Figure 3.4 illustrates the memory layout of our algorithm. We require read and write access to the position, direction, random generator state and loop check data for each sample. Thus, each attribute occupies a block of global memory. Let S be the total number of samples. Then, sample position and direction and generator state each require $4S \cdot 32\text{bit}$. In addition, the loop check state of each sample accounts to $N_{\mathcal{V}}/N_{\mathcal{SV}} \cdot 8\text{bit}$ if each supervoxel consists of $N_{\mathcal{SV}}$ voxels. In practice, the supervoxel memory tends to dominate the others in size.

The discrete representation of the *pdf*s describing the distribution of local fiber orientation is packed into a 3D texture to benefit from cached read accesses. We realigned the memory structure of the *pdf*s to fit them into a 3D texture block. Thus, memory offsets have to be computed before each texture fetch to access the realigned memory. Let the number of discrete bins per parameter of interest be B (depends on the local parameter estimation). The local estimation used in this work produced $B = 50$ bins. When represented as floating point values, the two parameters of interest for the probabilistic tracing (θ, ϕ) thus require $2 \cdot 32\text{bit}$. Since 3D textures can store up to 4 values (RGBA) per texel, two value slots were still unoccupied and we were able to pack the second local fiber orientation (θ', ϕ') of the two-fiber model into the same memory location. In general, for a multi-fiber model with M local orientations, the memory requirement of the local *pdf*s accounts to $N_{\mathcal{V}} \cdot \lceil M/2 \rceil \cdot 4B \cdot 32\text{bit}$.

The spatial *pdf* keeps track how often a voxel has been visited by a probabilistic stream-

line and thus also needs to be read from and written to by all threads concurrently. Alternatively, the construction of the spatial *pdf* would require a huge amount of dedicated memory per sample. Since memory is a very rare resource on the compute device (cf. loop checking in Section 3.4.3) we use the same (global) memory block for all threads for collecting the connectivity counts at the cost of thread level synchronization. Thus, total memory requirement is $N_{\mathcal{V}} \cdot 32\text{bit}$. Moreover, if visual feedback is desired, the block of memory representing the connectivity distribution has to be accessible from the rendering pipeline. To this end, the corresponding memory block is registered as a 3D texture after each compute run and mapped to compute memory prior to it. The memory for additional visual cues, for instance structural MRI and other data modalities, must then fit in the remaining device memory.

Random Number Generation An essential building block of the implementation is the generation of random numbers directly on the GPU. A variety of random number generators exist, some of which have been analyzed how well they perform on the GPU in terms of computation time and randomness [67, 88, 166]. Previous GPU implementations of probabilistic algorithms (e.g. McGraw et al. in [96]) have avoided that issue by generating random numbers on the CPU and packing them into a texture. If a huge number of random numbers are required this approach has high memory requirements.

We use a hybrid approach in our implementation. First, we generate a random seed vector on the CPU for every thread that will run in parallel on the device. These numbers are also packed into a texture for cached access in the compute kernel. Additionally, we have implemented a variant of the Hybrid Taus generator [67] which is a combination of a linear congruential generator and Tausworthe algorithm [140] directly in the compute kernel. A detailed analysis of the performance of this generator is discussed by Zhmurov et al. in [166]. We use the same constants as in [67] for Tausworthe algorithm and the congruential generator with periods p_i:

$$p_1 = 2^{31} - 1, \quad p_2 = 2^{30} - 1, \quad p_3 = 2^{28} - 1, \quad p_4 = 2^{32}$$

where the combined period is the least common multiple of p_1, p_2, p_3 and p_4 which is approximately 2^{121}.

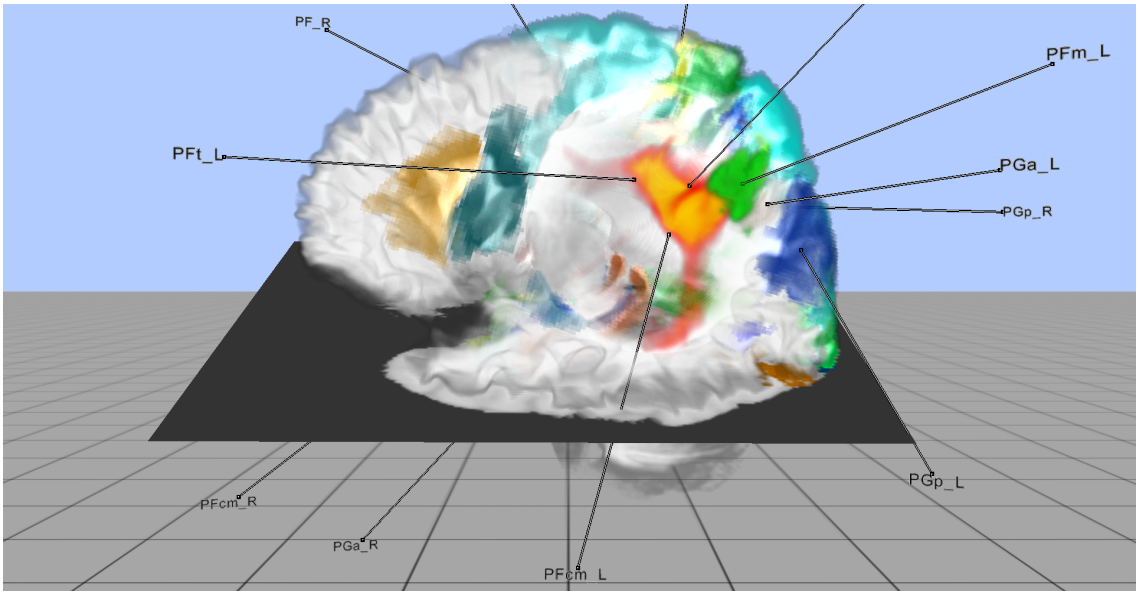


Figure 3.5.: Multi-modal brain visualization displaying brain's white matter, brain areas, a probabilistic fiber tract and the corresponding seed region. White matter and brain areas are clipped by a magic lens to reveal the fiber tract inside.

3.5. Virtual Reality Interface

This section presents a VR prototype application of the probabilistic tractography that combines the interactive computation of probabilistic fiber tracts from the previous section with a real-time visualization and interaction to assist in exploratory discoveries and the development of an intuitive understanding.

Our VR interface implementation is based on the virtual reality toolkit ViSTA [7]. In consequence, our prototype application runs on a broad range of systems, ranging from common desktop computers to immersive virtual environments (cf. Figure 1.5 left). The remainder of this section presents our choice of visualization and interaction techniques and discusses how we tailored existing state-of-the-art techniques to our specific requirements.

3.5.1. Visualization

In probabilistic tracking algorithms, individual streamlines no longer carry significant information, and hence standard visualization techniques like streamlines or streamtubes [130, 118] are no longer appropriate. Conveying the uncertainty in the rendering is an inherent requirement for neuroscientists to evaluate probabilistic tractography.

In most current visualizations uncertainty is only represented on two-dimensional slices whereas 3D representations of probabilistic fiber tracts are often generated by extracting opaque isosurfaces for certain probability ranges (cf. Figure 3.1 and Figure 3.2).

However, the visualization of the probability in three dimensions is an essential step for the registration of the most likely course of a fiber bundle and can be addressed by using direct volume rendering in order to provide semi-transparent renderings of volumetric data in real-time.

Moreover, anatomical and multi-modal information is required to reveal the fiber in its correct context. Data modalities may include, but are not limited to structural *Magnetic Resonance Tomography* (MRT), cortical or functionally defined brain areas, diffusion data and fiber tracts or information on the biochemical level (*Positron Emission Tomography* (PET)). Thus, an appropriate visualization should be able to display multiple , transparent voxel-based information, simultaneously. State-of-the-art techniques for direct volume rendering needs to be adapted for an interactive visualization in a VE. The main reason for this is that the process of rendering transparent objects, which usually relies on either depth sorting or ray casting, is a complex process in general and becomes even more demanding the more objects are involved. Recent work on that particular issue in medical visualization has also been investigated by Beyer et al. in [24, 23]. In a VE the real-time requirements of a visualization are even stricter because of the high pixel resolution of VR screens and in particular because of frequent changes in viewer orientation and position due to head tracking.

However, we can exploit the fact, that most medical data sets are already registered in a common reference space. Hence, we can fine-tune our implementation of direct volume rendering specifically to this kind of data. We adapt the classical direct volume rendering to efficiently handle multiple co-registered data sets as follows: We interleave the individual data sets into one vector-valued data field. The proxy geometry (texture slices or cubes [61, 62]) is setup such that it represents the shared reference space and is rendered only once. This avoids depth-sorting multiple proxy geometries. A special shader program handles each integration step of the individual data sets separately. The temporary integration values are combined (e.g. maximum intensity or weighted sum) into a single value and standard volume integration can be performed. This way, fiber probability and structural information can be classified according to separate transfer functions but form a consistent and correctly depth-sorted transparent image without introducing significant overhead. Additional anatomical cues are provided by opaque cross sections of the brain, which also can provide an unbiased view on the original data as sometimes preferred by domain scientists. In particular, the cross sections can be used as occluders to terminate the volume rendering early. In our implementation we have integrated this step directly in the setup of the proxy geometry.

Figure 3.5 shows a visualization of a brain with four different data channels: segmented white matter, brain areas, probabilistic fiber tract, and tract seed region. The brain

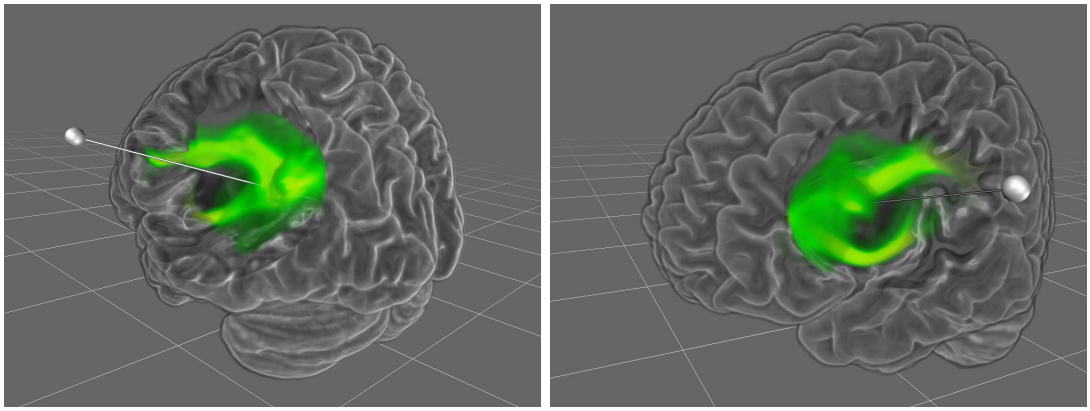


Figure 3.6.: The virtual flashlight: structural data is clipped by a magic lens to reveal the fiber tract inside. Images show slightly different locations of the flashlight position to show the boundaries of the fiber tract.

areas shown here are from the Jülich-Düsseldorf cytoarchitectonic atlas[167] and the corresponding annotations are positioned automatic within the VE by a component of the underlying VR toolkit as described by Pick et al. [112]. Typically, we encode high probabilities of the fiber tract with bright and opaque colors and low probabilities by dark and transparent colors.

3.5.2. Interaction

In addition to visualization, the intuitive exploration of data can also benefit greatly from interactive manipulation and direct interaction where the user takes an active role. Both are an integral part of every interactive VR system. This section describes a magic lens interaction for data disambiguation, a selection technique for brain areas, and seeding strategies on how to define starting regions for the probabilistic tractography.

3.5.2.1. Virtual Flashlight

We have incorporated a direct interaction metaphor into our VR application, the *virtual flashlight*. The basic principle of the virtual flashlight is similar to magic lenses which were first discussed by Bier et al. [26] as a 2D see-through user interface that changes the representation of content in a special window. A well-known example is the magnifying glass. In [147], Viega et al. extended the concept of magic lenses to VEs. They presented an implementation of volumetric lenses that uses hardware clipping of geometric primitives to reveal the inner structure of objects. Fuhrmann et al. [58] used a magic box to present a higher-resolution of a flow visualization in order to focus attention on these regions and investigate them in more detail.

In the presented VR prototype, the user can directly control the amount of visible anatomical structure by a 3D interaction device (cf. Figure 1.5 right), similar to the beam of a flashlight. Figure 3.6 illustrates the concept of how interesting parts of the probabilistic fiber tracts can be revealed and referenced with anatomical landmarks with reduced occlusion or visual clutter. This allows a more accurate inspection of the anatomic structure in the direct vicinity of fiber pathways.

Our implementation of the virtual flashlight is realized by mediating between two components of the ViSTA VR Toolkit: the interaction widgets and the volume rendering. The interaction widgets provide the user interface that can be modified either by a mouse on a desktop or 6DOF devices in a VE. For the virtual flashlight we use a spherical widget that can change it location and size to reflect the round organic shape of structures within the brain. The widget's information is communicated to a specialization of our volume renderer that uses an alternative transfer function for fragments within the magic lens' range.

3.5.2.2. Object Selection

Object selection is a basic interaction block in order to trigger further actions or manipulations. Thus, brain areas and fiber tracts, key components of the probabilistic tractography computation, have to be accessible directly from the VR interface. We noticed two issues that had to be addressed specifically since the objects that are subject to selection are (1) brain areas of usually non-convex shape and densely packed to one another, and (2) probabilistic fiber tracts which essentially are volumetric distributions in space and therefore have no explicit geometrical representation. In turn, this may introduce selection ambiguity when selection is done on the basis of coarse bounding volumes. Hence, our desired selection algorithm should distinguish between object based on their silhouette and be able to select polygonal as well as volumetric objects in a VE with a 6DOF pointing device.

For these reasons we adapted a 2D picking metaphor to 3D selection in VEs. Similar to the desktop picking we select the object that corresponds to the pixel that is seen through the eye of a 6DOF pointing device. The technique is easily implemented by a back buffer selection method. All selectable objects are rendered twice. The second render pass assigns unique object identifiers instead of colors. In a VE, this means changing the projection and view matrices for the second render pass according to the position and orientation of the pointing device. Hence, the perspective view frustum of the pointing device represents a conic selection volume. The opening angle of the selection volume can be adjusted by changing the field of view of the perspective projection. Visible pixels of the objects within the selection volume will be recorded by their identifiers in the off-screen buffer. State-of-the-art scoring schemes based on pixel based object statistics (e.g. time and distance ranking) can then be applied to choose the active object. In [122],

we have shown that this method works fast enough in a VE for triangulated as well as volume rendered objects with no explicit geometric representation. In the following we will present the basic steps.

Selection Rendering The relevant transformations are M_{Model} which transforms from object space into world space, M_{View} which transforms world space into eye space, and a final projection which maps eye space into screen space. Let M_{Proj} be a standard perspective projection matrix, M_{VCP} the projection matrix of a viewer centered projection that accounts for motion parallax in a head-tracked VE, and $M_{Tracking}$ the transformation of the tracked pointing device from eye space into world space.

Each object is rendered twice, first for its visual representation and then for writing its selection ID. Let the point p be the coordinate of an object defined in local object space. Usually the transformation from object space into the screen space of the VE is achieved by

$$p' = M_{VCP} \cdot M_{View} \cdot M_{Model} \cdot p.$$

In selection mode the projection of p into the screen space of the pointing device is

$$p_s = M_{Proj} \cdot M_{Tracking}^{-1} \cdot M_{Model} \cdot p.$$

Hence, if the first render pass for visual representation looks something like this

```
1. load_matrix(M_VCP · M_View · M_Model)
2. use_shader(fancy_lighting)
3. for_each(Object obj)
4.   obj.render()
```

the second render pass for selection simply would be

```
5. load_matrix(M_Proj · M_Tracking^-1 · M_Model)
6. use_shader(render_id)
7. for_each(Object obj)
8.   load_selection_id(obj.ID)
9.   obj.render().
```

Due to performance reasons, we are using an unused texture unit to pass object IDs as colors instead of using uniform shader variables.

Create Pixel Based Object Statistics The result of the selection rendering process is a two-dimensional pixel array P. Each pixel position contains a value that is either zero (no object) or an integer number (ID) that corresponds to an object. By construction, the

center pixel position c corresponds to the center of the selection cone. A popular scoring metric for instance used by [47] is the projected distance d between the cone center and an object at pixel position p which is $d = ||c - p||_2$. In [75] different rankings for object selection candidates have been introduced. Object statistics are created based on the time and stability of presence within the selection volume, distance to the volume's origin or center proximity. The pixel array P allows the computation of the above mentioned rankings, the choice of metric of course depends on user preference and application.

We extend the basic algorithm by describing how to get the actual contact point on the selected object and how to select regions of volumetric objects that have been rendered with a direct volume rendering technique.

Contact Point on Selected Object We can exploit the structure of our selection implementation to directly deliver the contact point on the selected object by saving the three-dimensional vertex positions to the off-screen buffer in addition to the object ID. Hence, each pixel points to an object via an identifier and to the 3D location that was mapped to the corresponding pixel position. The implementation on the GPU is straightforward, a vertex program would simply pass the transformed vertex position to the fragment shader.

Selection with Direct Volume Rendering When using a pixel based selection, everything that can be rendered can be selected with little additional effort. Explicit geometric representations are not required. Therefore, the same method can be used to select parts of volume rendered objects by using an alternative transfer function in the selection render pass that maps density values to object IDs. To ensure the uniqueness of object IDs, blending must be disabled and the texture lookup to the transfer function should use nearest neighbor interpolation. In consequence, the selection is restricted to parts that are closest to the selection cone. The rendering of the volume in the selection pass can be done at a very coarse sampling distance. In practice one-fifths of the original sampling distance turned out to be sufficient.

3.5.2.3. Seeding Strategies

The main user input to our probabilistic tractography algorithms is the definition of seed locations. We have integrated two typical seeding strategies in our VR interface: seeding from a predefined brain area and free seeding from a user defined location.

Seeding from a Brain Area We realized seeding from a specific brain area in our VR interface by letting the user point to a predefined set of areas on the brain and start the

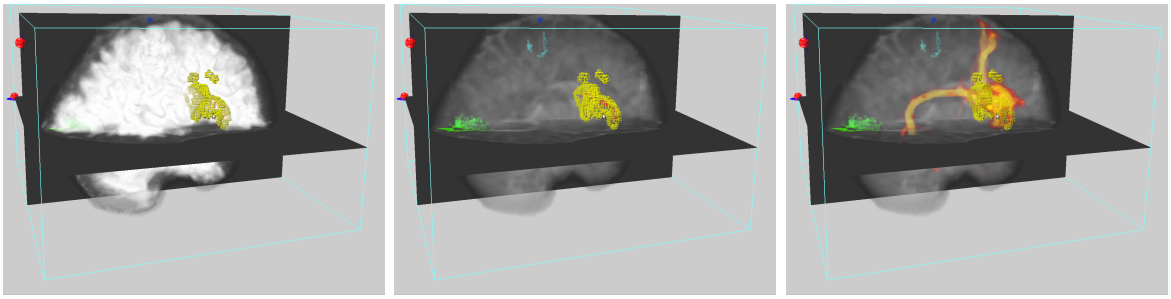


Figure 3.7.: Seeding from Broca's area. The left image shows the white matter segmentation and the seeding region for anatomic reference. The middle and right image show the evolution of the probabilistic fiber tract from the beginning until its convergence.

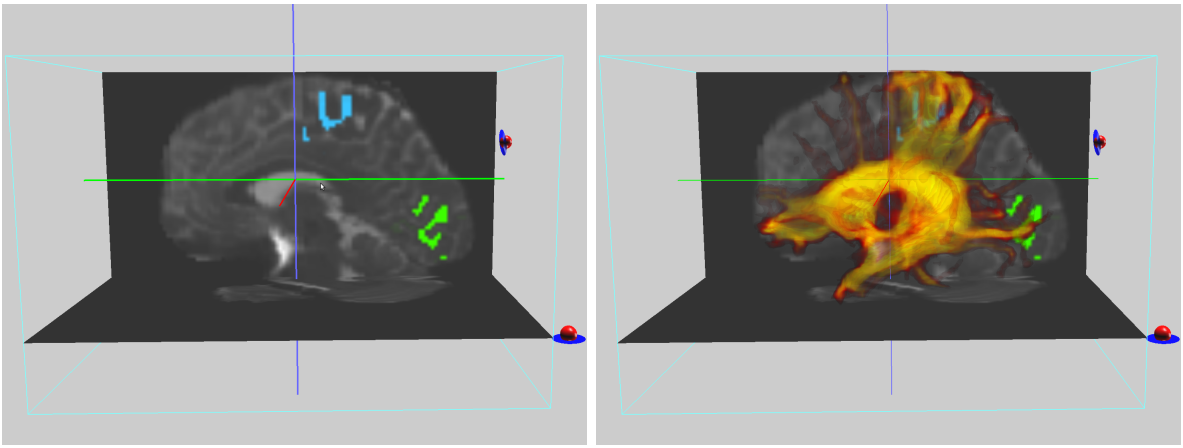


Figure 3.8.: Free seed placement in the corpus callosum. The left image shows the cross hairs cursor at the location of the seed origin and the right image depicts the fiber tract after a certain number of iterations.

tractography process by a button click. Interactive visual feedback is provided by rendering the current spatial probability distribution after each tracking step of all samples. Figure 3.7 shows the selection of Broca's area on the gray-white matter interface and the corresponding tracking result. As seeding and tracking happens in diffusion space, brain areas have to be mapped from their reference space to the individual diffusion space. In [36] brain areas are mapped onto the gray-white matter interface in diffusion space to allow fiber tracts to start in anisotropic regions and seeding is based on the center voxel of the mapped brain area. As this mapping may easily introduce mapping artifacts, our interface can also be used as a fast preview tool to see if an individual mapping has led to the desired results. The selection of brain areas is realized with the above described selection algorithm in order to be able to accurately select areas on the whole cortex.

Free Seeding Our free seeding strategy lets the user decide at which voxel the tractography algorithm should start without any restrictions on the location. This strategy

allows seeding from deep gray matter voxels as well as within white matter as illustrated in Figure 3.8 and is meant as an exploratory approach to figure out good seeding positions. In a VR setup, free seeding interaction is naturally achieved by direct 3D interaction with a tracked 6DOF device where the seeding origin can be defined by simply moving or pointing the tracked device to the desired location.

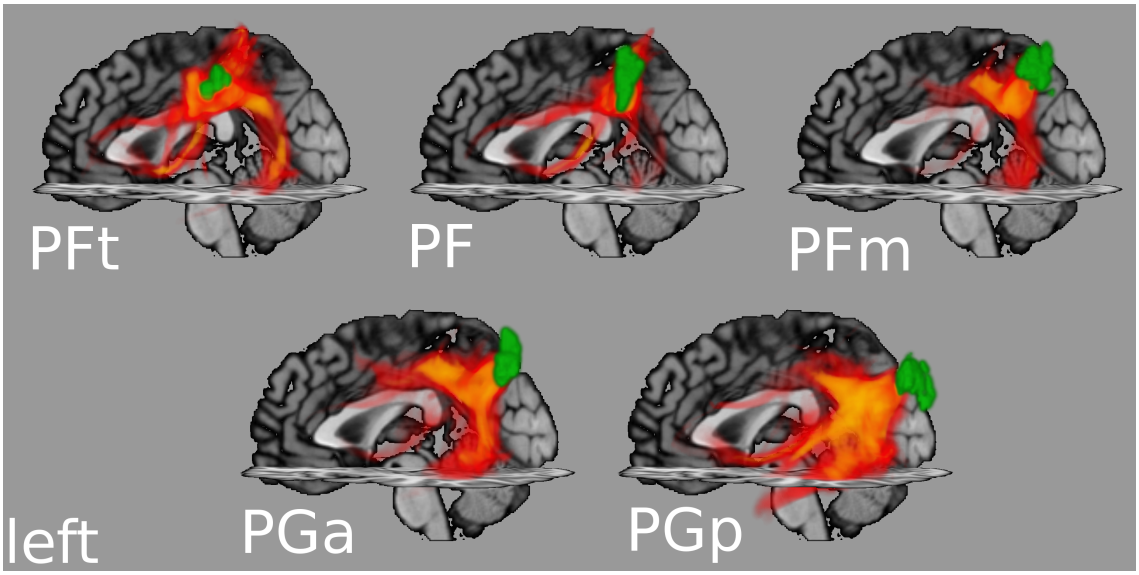


Figure 3.9.: Visualization of probabilistic fiber tracts of inferior parietal lobule areas (left hemisphere). The resulting tracks are volume rendered by coding probabilities as colors (yellow high, red low) and opacities. The corresponding seed regions are depicted by the green areas.

3.6. Results

This section presents the results of our parallel implementation of the probabilistic tractography algorithm. After a short remark on the visualization of probabilistic fiber tracts, we address the anatomical plausibility of our obtained results to establish the correctness of the computations. The main focus of the work was to accelerate the computation of global connectivity. Consequently, we cover the achieved performance in greater detail with respect to scalability, concurrent execution and speedup of different compute conditions.

3.6.1. Visualization

In a recent study [36], Caspers et al. debate how the structural and functional heterogeneity of the *Human Inferior Parietal Lobule* (IPL) relates to local connectivity patterns by analyzing the anatomical connectivity of five cytoarchitectonically defined IPL areas (PFt, PF, PFm, PGa, PGp) using DTI and probabilistic tractography. The IPL is a multi-modal brain region with marked functional heterogeneity that is reflected by a structural segregation into several cytoarchitectonic areas [35, 15]. Figure 3.9 illustrates how our proposed visualization technique (cf. Section 3.5.1) has been used for the presentation of the corresponding probabilistic fiber tracts.

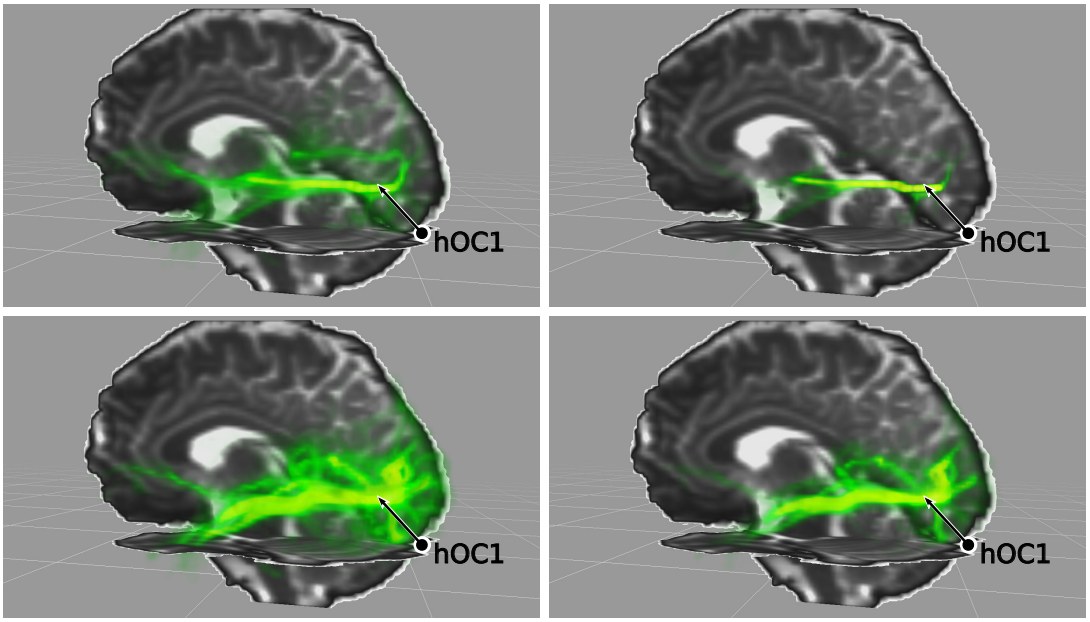


Figure 3.10.: Visualization of probabilistic fiber tracts that were seeded from the primary visual cortex in the left hemisphere. The first row shows the results from the FSL tool suite [43] and the second row from our GPU algorithm. The first column depicts the results without loop checking ad the second column with loop checking (supervoxel size 5). Fiber tracts computed with loop checking of a supervoxel size of 10 did not reveal apparent visual differences and are therefore omitted.

3.6.2. Anatomical Plausibility

To the best of our knowledge, there is now apparent or established method for comparing the course of probabilistic fiber tracts automatically. Dauguet et al. [46], for instance, used segmentation and registration in combination with visual comparison and a statistical analysis when comparing fiber tracts derived from in-vivo DTI with histological tracer reconstruction on a macaque brain. Whereas Okada et al. [106] compared fiber tracking results from 3.0-T and 1.5-T MR Imaging only by a visual (manual) inspection and classification of domain experts in the field of neuroradiology.

Apparently, manual inspection for all tractography computations in our benchmark setup (cf. Table 3.1) is unfeasible. In an informal discussion with our domain expert we agreed to conduct a spot-checked inspection of selected tractography computations where the results are commonly known anatomic structures. In particular, we decided on two specific seed locations: hOC1 (primary visual cortex) and PMC 4a (primary motor cortex), both in a male subject (M1). The expected course of the hOC1 tract should exhibit a more or less horizontal propagation towards the frontal lobe. Seeding from primary motor should reveal a connectivity towards the spinal cord.

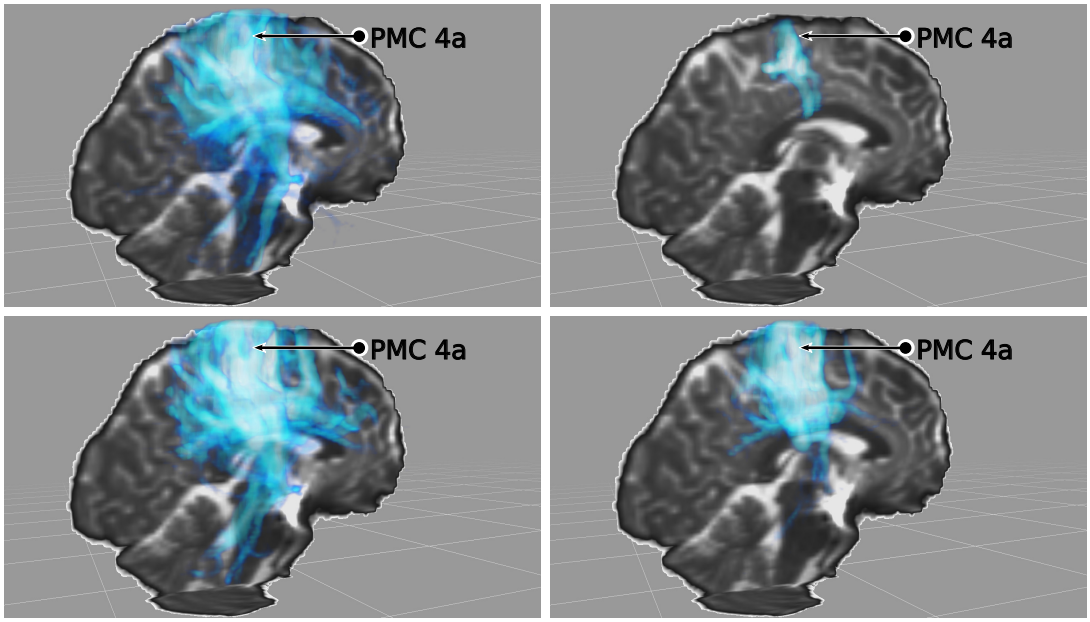


Figure 3.11.: Visualization of probabilistic fiber tracts that were seeded from the primary motor cortex in the right hemisphere. Again, the first row shows the results from the FSL tool suite [43] and the second row from our GPU algorithm. The first column depicts the results without loop checking ad the second column with loop checking (supervoxel size 5). Fiber tracts computed with loop checking of a supervoxel size of 10 did not reveal apparent visual differences and are therefore omitted.

Tractography parameters were set to a sample population of 100,000 with 5,000 steps at a step size of 0.25 (quarter voxel). We computed a probabilistic fiber tract for each of the five compute conditions (detailed in the next section). Two of five compute conditions estimated fiber tracts with the established FSL [43] tool suite and three with our GPU implementation. The visual inspection was not blind, meaning the domain expert knew the compute condition of each tract.

Figure 3.10 and Figure 3.11 show the visualization of the resulting fiber tracts. The first row depicts the results from the FSL condition, the second row from our GPU computation. The first column displays tracking results without loop checking and the second column with loop checking.

The domain expert stated that the fiber tracts were in fact comparable. All compute conditions produced fiber tracts with the correct connectivity trend towards the frontal lobe (hOC1) and towards the spinal cord (PMC 4a), respectively. Seeding in the primary visual cortex with loop checking did reveal a slightly clearer fiber tract with our probabilistic loop checking approach. This was not observed for the FSL computation. Moreover, loop checking with FSL in the seeding from the primary motor cortex slightly decreased the signal-to-noise ratio and in turn lessened the connectivity impression to-

wards the spinal cord. In contrast, the probabilistic loop checking of our implementation lead to an significant improvement of the fiber connectivity between the seed in the primary motor cortex and the spinal cord. In summary, the domain experts assessed the connectivity distributions of our implementation as plausible. She noticed a slight emphasis towards regions near the seed location with our approach compared to FSL. Nevertheless, voxel further away from the seed were still reached, although sometimes with lesser probabilities and less distinct forks. Having established anatomical plausibility of our results, we will now go on to discuss the computational performance.

3.6.3. Tractography Performance

After defining performance metrics and benchmark conditions we will discuss scalability, the performance impact of loop checking and the two-fiber model, and speedup of our GPU implementation.

Performance Metrics Let T_{trace} be the total computation time per trace in seconds, N_{trace} the number of trace steps and, S_{trace} the number of samples per trace. We define the computation time per step simply as $T_{\text{step}} = \frac{T_{\text{trace}}}{N_{\text{trace}}}$. To incorporate the number of traced samples into the performance measure we define million samples per second per trace as $\text{M}S_{\text{trace}}/\text{sec} = \frac{S_{\text{trace}} \cdot 10^{-6}}{T_{\text{trace}}}$

and million samples per second per trace step as $\text{M}S_{\text{step}}/\text{sec} = \frac{S_{\text{trace}} \cdot 10^{-6}}{T_{\text{step}}}$, respectively.

All timings exclude the transfer time from hard drive to host memory but include host to device memory transfer times. If not noted otherwise, all performance figures are given as averages over all (ten) individual patient data sets. Measures relating to individual patients are marked with subject number and gender.

The rate at which data is transferred between host and device, i.e. the bandwidth, is another important performance indicator that is affected by the choice of memory in which data is stored and accessed, how data is laid out and in which order it is accessed. Moreover, comparing theoretical bandwidth and effective (observed) bandwidth offers insight into performance and optimization efforts. The peak theoretical memory bandwidth in GB/sec of a compute device is defined as

$$\text{BW}_{\text{th}} = M_c \cdot 10^6 \cdot \frac{M_{iw}}{8} \cdot d_r \cdot \frac{1}{1024^3}.$$

Where M_c describes the memory clock in MHz, M_{iw} the memory interface width in bit and d_r the memory data rate. When using *Double Data Rate* (DDR) memory (as most current devices do), the data rate d_r is two. The effective bandwidth in GB/sec can be computed as

$$\text{BW}_{\text{eff}} = \frac{(B_r + B_w)\,/1024^3}{\text{time}}$$

where B_r is the number of bytes read per kernel, B_w the number of bytes written per kernel and the computation time is in seconds (cf. [104]).

Benchmark Variables and Compute Conditions Our benchmark variables are implementation type, size of super-voxel, step size, number of steps and samples, data set,

variable	values
implementation	FSL (CPU) our method (GPU)
size supervoxel	0* (no lc) 5 (lc5) 10† (lc10)
num. steps	5,000
step size	0.25
num. samples	10K 25K 50K 75K 100K 125K 250K 500K 750K 1M
data set subject	M1 F1 M2 F2 M3 F3 M4 F4 M5 F5
seed region	hOC1_L hOC1_R PMC_4a_L PMC_4a_R
seed type	center voxel area mask†

Table 3.1.: Algorithm parameters and their values for the performance benchmarks: 0* disables loop checking, † this setting was only measured on the GPU device.

seed region and seed type. Table 3.1 gives an overview of the possible values each variable can take. In particular, the implementation type is either the FSL suite [43] or our GPU implementation. The size of the supervoxels defines at which resolution we perform the loop checking, a value of zero disables loop checking. We fixed the number of steps to $N_{\text{trace}} = 5,000$ with a step size of 0.25 (quarter voxel). The number of samples S_{trace} ranged between 10K and 1M. Ten individual data sets were made available [36]: five male and five female subjects each with individual mapped brain areas for the primary visual (hOC1) and primary motor cortex (PMC 4a), each for the left and right hemisphere. These brain areas were used for seeding in the corresponding areas. We distinguished between two seeding strategies, seeding from the center voxel of an area and seeding from all area voxels.

We refer to a *Compute Condition* (CC) as a particular configuration of benchmark variables. For convenience we define five CCs that are used through the remainder of this section:

$$
\begin{aligned}
\text{CC}_{CPU} &= \text{(FSL, no lc, *, *, *, *, *, center voxel)} \\
\text{CC}_{CPU\_LC} &= \text{(FSL, lc5, *, *, *, *, *, center voxel)} \\
\text{CC}_{GPU} &= \text{(our method, no lc, *, *, *, *, *, *)} \\
\text{CC}_{GPU\_LC5} &= \text{(our method, lc5, *, *, *, *, *, *)} \\
\text{CC}_{GPU\_LC10} &= \text{(our method, lc10, *, *, *, *, *, *)}
\end{aligned}
$$

The asterisk (*) is a placeholder for all available values of a variable. If not otherwise noted, performance results were averaged over the number of samples, data sets, and seed regions. The supervoxel size of 10, was not available in FSL and consequently could only be computed with our method. Moreover, the seeding from all area voxels (area mask) was only computed with our method as a direct comparison in terms of computation time to FSL would have been unfair towards FSL due to different implementation specific issues. In consequence, comparisons between CPU and GPU are always performed with seeding from the center voxel only.

3.6.3.1. Scalability

We discuss the scalability of our approach with respect to two measures: total trace time and throughput in terms of samples per second. In addition, we investigate possible performance variation due to different seed origins and strategies and present the effective bandwidth of our implementation. The data shown Figure 3.12, Figure 3.13, Figure 3.15, and Figure 3.16 is averaged over all individual subjects and all seeding types for each particular sample population and compute condition whereas the data shown in Figure 3.14 is averaged only over individual subjects, thereby preserving seed specific influences.

Total Trace Time Figure 3.12 shows the total trace time T_{trace} over an increasing sample population (10K to 1M samples). x- and y-axis scales are logarithmic to depict the behavior of CPU and GPU conditions in one plot. In both CPU conditions (CC_{CPU} and CC_{CPU_LC}) samples are processed sequentially, in consequence the total trace time increases linearly. The computation time for the GPU conditions exhibits an improved scaling due to increasing device occupation with an increasing number of samples. In particular, 1M samples can still be computed in under 10 seconds on the GPU whereas 25K samples on the CPU already take more than 10 seconds to trace. We observe no significant differences between CC_{GPU} and CC_{GPU_LC5} , however, CC_{GPU_LC10} performs slightly better than the other GPU conditions.

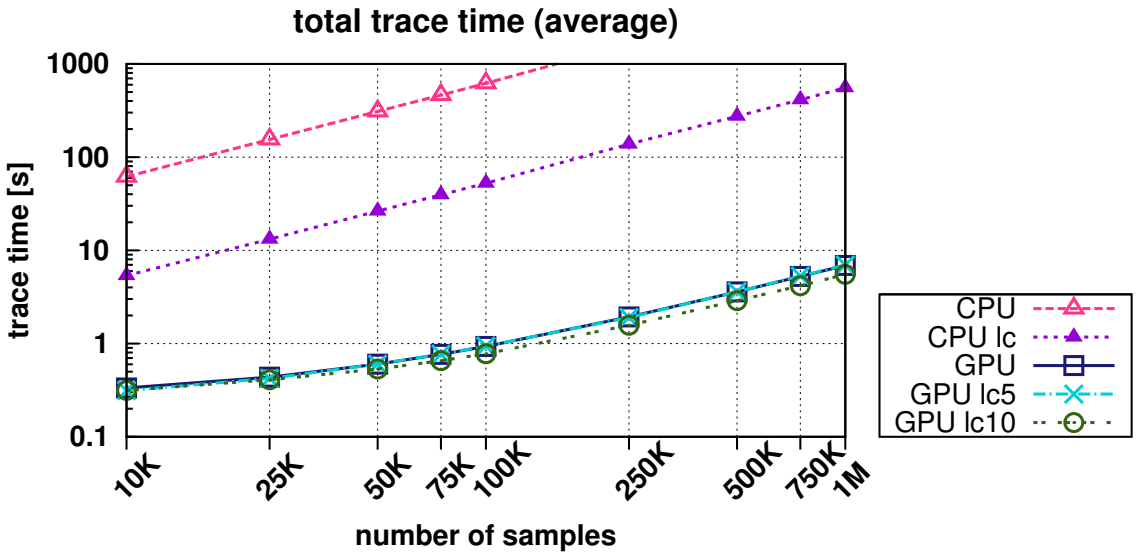


Figure 3.12.: Scaling of total trace time with respect to the number of samples.

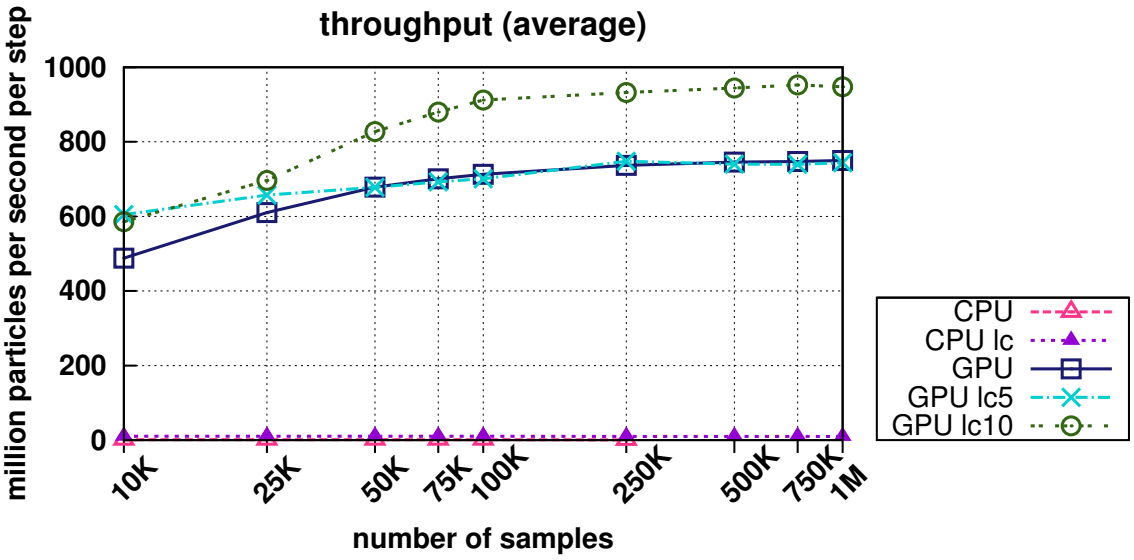


Figure 3.13.: Computational throughput in terms of million samples per second per step. More samples lead to an increased device occupation that reaches its maximum after 100K samples.

Throughput Figure 3.13 depicts the number of samples per second per trace step $\mathrm{MS}_{\mathrm{step}}$/sec. Since normalized to time and sample population, the $\mathrm{MS}_{\mathrm{step}}$/sec can be considered as a measure of computational throughput. Apparently, the CPU conditions do not show any change in throughput with an increasing sample population. However, on the GPU throughput is directly related to device occupation. Consequently, we observe an improved throughput for a higher number of samples. For the CC_{GPU} and CC_{GPU_LC5} conditions, the maximum throughput (700 million samples) is reached at a sample size of 100K samples. At the same sample population, the CC_{GPU_LC10} condition reaches its maximum of approximately 900 million samples per second per step.

Performance Variation The course of a fiber tract greatly varies based on its seeding region. Hence, we address the question whether the seeding region or strategy also has a strong influence of the total computing time. We refer to this as performance variation with respect to seeding. Figure 3.14 shows the total trace time T_{trace} for each seeding configuration in the GPU conditions averaged over all ten subject. In particular, seed regions are the primary visual cortex (hOC1) and the primary motor cortex (PMC 4a) for both, the left and the right hemisphere. Each computation was performed with every GPU condition for both seeding types (center voxel and area mask) with a fixed sample population of 100K. We observe that the average computation times within one condition are almost constant for both, different seed areas and different seeding strategies.

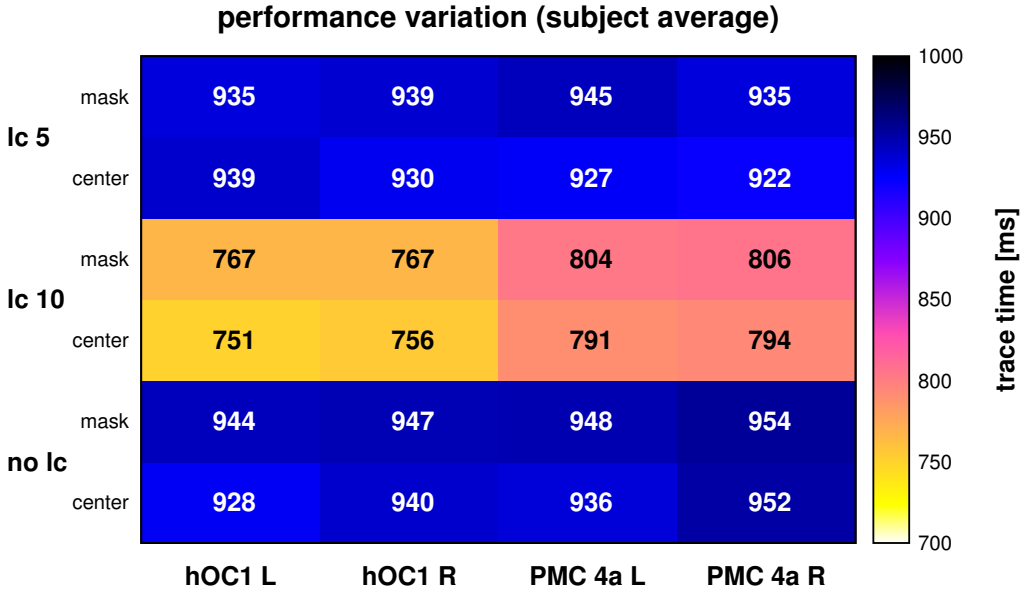


Figure 3.14.: This figure shows that the total trace time is almost independent of seeding area and strategy and only depends on the compute condition.

Concurrent Execution The total trace time greatly depends on the number of threads that can process the samples concurrently on the compute device. Apparently, the required amount of memory increases with larger sample populations and thereby limits the number of samples that can be traced in parallel. To this end, we analyzed the number of concurrent kernel executions on various compute conditions and memory restrictions. For convenience, we consider the unit of parallelism as the number of kernel invocations on the GPU. We considered two different compute devices, the first one was equipped with 1.5 GB of main memory and the other with 6 GB. About 1 GB was required for storing input data like the probability distributions of local fiber directions and some device memory was already allocated by other applications (possibly the operating system) such that a remaining amount of 230 MB was left for the fiber tracking on the first device and roughly 4.1 GB of device memory on the second device. Figure 3.15 shows the percentage of the sample population that can be processed concurrently on the compute devices for the three GPU conditions CC_{GPU}, CC_{GPU_LC5} and CC_{GPU_LC10}. As already discussed previously, loop checking requires considerable amount of device memory. Accordingly, we observe that roughly 12.5K samples fit into the 230 MB of device 1 and about 250K into the 4.1 GB of device 2 in the CC_{GPU_LC5} conditions (supervoxel size of 5). When doubling the supervoxel size in the CC_{GPU_LC10} condition, memory requirements are greatly reduced. Device 1 is now able to process 100K samples within one kernel invocation whereas the memory of device 2 is already sufficient to execute the tracing procedure for all 1M samples with one single call. No subdivision into several kernel invocations has to be done for the CC_{GPU} condition without loop checking for both devices. The memory is sufficient for storing the positions, directions and random seed state for all 1M samples.

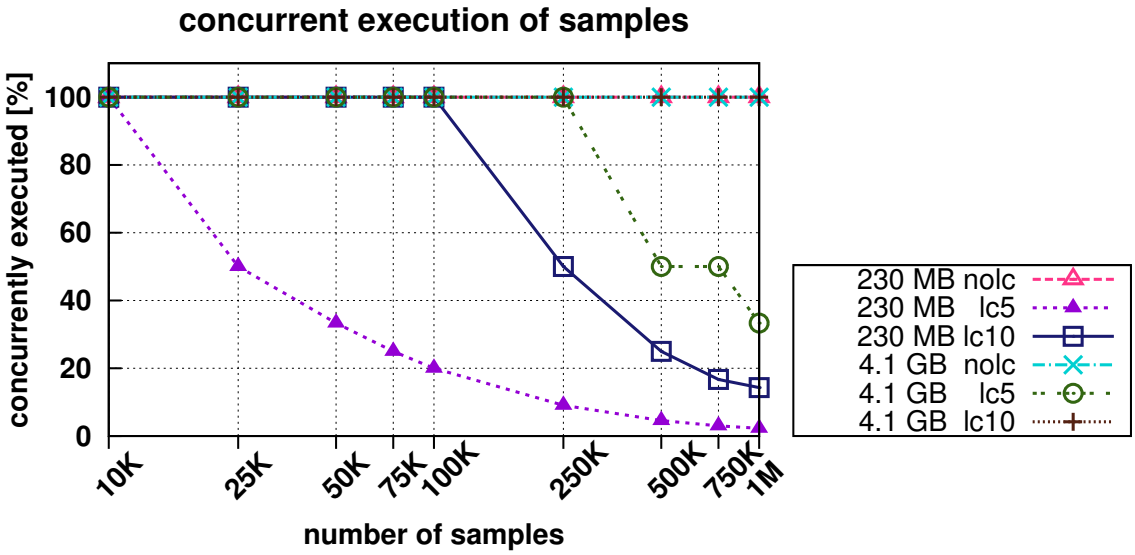


Figure 3.15.: Percentage of concurrent executions for an increasing number of samples and two devices with different available memory.

Effective Bandwidth Figure 3.16 shows the effective bandwidth for the GPU compute conditions for an increasing number of samples. For reference, our GPU had a peak theoretical memory bandwidth of 165 GB/sec. We used the effective bandwidth computation as described in the beginning of this section. Each kernel invocation accessed 16 bytes for each of the following: local fiber orientations, sample position, direction and random generator state. An additional 4 bytes were required for the count of the spatial distribution and 1 byte for checking a loop in a supervoxel. Only the access to the fiber orientations was read-only. In total, this sums up to 69 bytes that were read per kernel, and 53 bytes that were written per kernel. Disabling loop checking would save a byte on either side. We will discuss the effective bandwidth in more detail in the following section on the loop checking extension.

3.6.3.2. Performance Impact of Multiple Fiber Orientations and Loop Checking

Multiple Fiber Orientations We argued in Section 3.4.3 that it is sufficient to optimize an actual implementation of the multi-fiber model for the two-fiber case. Due to our memory layout for the storage of local fiber distributions (cf. Section 3.4.4), the introduction of a second fiber orientation revealed no significant degradation in performance. The main reason for this is that the memory layout on the device is optimized for a word size of four floats (RGBA), i.e. a read access to texture memory. Thus, the memory access for the single fiber model used only two float values (ϕ, θ) such that the second fiber orientation can be accessed with no further cost for the additional two components (ϕ', θ'). In consequence, we performed all our benchmark directly with the two-fiber

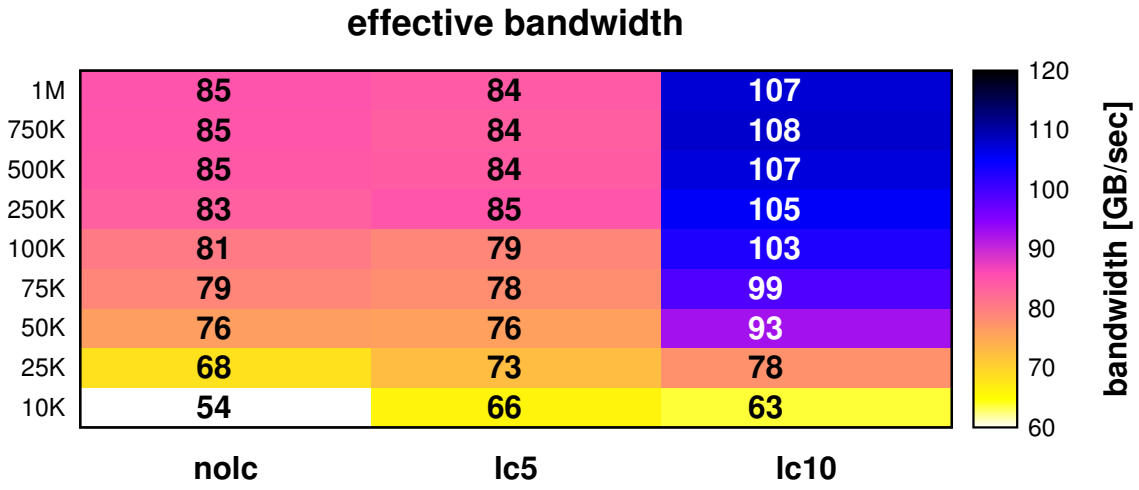


Figure 3.16.: Effective bandwidth for the GPU compute conditions.

orientation model. This was also in agreement with domain experts who preferred using the two-fiber orientation model over the single-fiber model for increased computational accuracy.

Loop Checking In contrast, loop checking exhibits a measurable difference in computation time and has therefore been included in all performance considerations as a separate compute condition. Moreover, loop checking can have a significant influence on the effective bandwidth as depicted in Figure 3.16. When increasing the size of the supervoxels used for loop checking, the effective bandwidth is increased from 85 GB/sec (in the CC_{GPU} condition) to 107 GB/sec (in the CC_{GPU_LC10} condition); samples that enter a loop can be discarded early, thereby allowing other samples to take their place within the thread pool. In summary, loop checking increases the per-thread memory requirements significantly, thus decreasing the number of concurrent kernel executions (cf. Figure 3.15). However, loop checking also provides an early out mechanism for samples that enter a loop thereby allowing traces to terminate prior to reaching the desired trace length and in turn increasing the overall throughput (cf. Figure 3.13).

3.6.3.3. Speedup

We define the speedup as the ratio of total trace times. Specifically, we refer to the individual factors as

$$S_{\text{no lc}} = \frac{T_{\text{trace}}\left(\mathrm{CC}_{CPU}\right)}{T_{\text{trace}}\left(\mathrm{CC}_{GPU}\right)}, \quad S_{\text{lc5}} = \frac{T_{\text{trace}}\left(\mathrm{CC}_{CPU\_LC}\right)}{T_{\text{trace}}\left(\mathrm{CC}_{GPU\_LC5}\right)}, \quad S_{\text{lc10}} = \frac{T_{\text{trace}}\left(\mathrm{CC}_{CPU\_LC}\right)}{T_{\text{trace}}\left(\mathrm{CC}_{GPU\_LC10}\right)}.$$

Figure 3.17 gives an overview of the speedup factors achieved by the different compute conditions over an increasing number of samples. The total trace time of the GPU

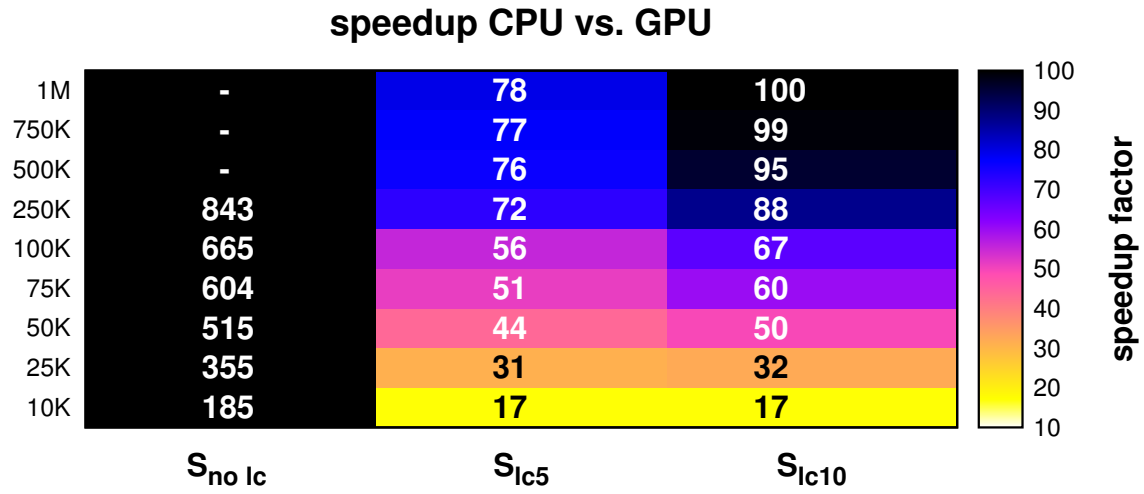


Figure 3.17.: Speedup factors in terms of total trace time ratio between CPU and GPU conditions.

methods include the host-device memory transfer times.

$S_{\text{no lc}}$ (first column) ranges from 185X for 10K samples to 843X for 250K samples. We did not compute the speedup for this condition for more than 250K samples due to the excessive computation time of the CPU condition. This condition utilizes the computational power of the device best: it requires only an initial data transfer between host and device and it has a low per-thread memory requirement such that all samples can be processed with one kernel invocation (cf. Figure 3.15). Loop checking greatly reduces the total trace time for both, serial and parallel, implementations: Samples can be discarded early as soon as a loop is detected (cf. Figure 3.12). However, the concurrency level of the GPU conditions now greatly depends on the amount of available device memory (cf. Figure 3.15). Still, we observe significant speedups: 17X for 10K samples to 78X for 1M samples for S_{lc5}. If the per-thread memory requirement is reduced by increasing the size of the supervoxels as in the CC_{GPU_LC10} condition, more kernels can be executed concurrently. This is reflected in slightly higher speedups for S_{lc10}: 17X for 10K samples to 100X for 1M samples. In summary, the speedup of our GPU implementation depends on various parameters such as number of samples and per-thread memory which can be adjusted as a trade-off between quality and speed (although we discussed quality only briefly in terms of plausibility).

3.7. Conclusions

We presented a parallel implementation of a probabilistic tractography algorithm that estimates the likelihood of connectivity between any two voxels. The method is based on a stochastic sampling of the tractography integral (3.10). Samples move along a multi-fiber field based on their current direction and the local distributions of fiber orientations. Samples that run into a loop are detected and discarded from the computation. The current front of every active sample is recorded in a spatial buffer. With a sufficient number of samples that buffer represents a spatial connectivity profile based on the relative frequency of visited samples.

Based on an expert review of fiber tracts from the primary motor and primary visual cortex we found a good overall agreement of our results in consistency with known anatomical structures. Moreover, we extended the original algorithm by a probabilistic loop checking approach which shows promising results in terms of an improved tract development compared to the deterministic loop check. However, a generalization of this observation would require further investigations from a neuroscientists' perspective.

By mapping the tractography algorithm to the many-core architecture of the GPU we were able to significantly reduce the overall run time. A sample population of 100K could be computed in under one second, a population of 1M samples took about ten seconds. Although fiber tracts may vary greatly depending on their seed location, we showed that the run time of our implementation is almost independent of seed location and seed strategy. We found an optimal device occupation with samples sizes of 100K or more which corresponds to a throughput of roughly 750 million samples per second per step (no loop checking and loop checking with supervoxel size 5). With a supervoxel size of ten, the throughput could be increased to approximately 950 million samples per second per step. The major limitation of our approach is the limited amount of device memory. In particular, our loop checking method requires a considerable amount of per thread memory which in turn limits the number of concurrent executions per kernel invocation. This limitation can be somewhat alleviated by increasing the size of the supervoxels. However, when compared to a CPU reference implementation, we obtained huge speedups: 185X to 800X without loop checking, 17X to 78X with loop checking (supervoxel size 5) and up to 100X with a supervoxel size of ten for 10K and 1M samples, respectively.

We also combined the tractography computation with an interactive visualization of the resulting fiber tracts by a direct volume rendering where probabilities are encoded as colors and opacities. Our proposed representation of probabilistic fiber tracts has already been used in several neuroscientific publications [49, 36]. Chapter A in the appendix also shows some additional visualizations which have been created with our VR prototype (cf. Section 3.5) by our collaborators at the Institute of Neuroscience and Medicine at the Research Center Jülich, Germany.

CHAPTER 4

CONCLUSIONS

4.1. Summary & Future Directions

We presented two specific application areas for exploratory visualization in VR: simulating radio waves and estimating neural fiber pathways. We put a special focus on the acceleration of the underlying computational methods to assist the human-in-the-loop pose "what if?" questions.

In the remainder of this chapter we will summarize the individual chapters, discuss possible extensions and future ideas, and identify the contributions of this thesis.

4.1.1. Radio Wave Propagation

Being able to predict the propagation behavior of radio waves is a crucial premise for every planning and optimizing algorithm for wireless networks. Moreover, propagation predictions must be both, accurate and fast, to be useful in real-world applications. Although most empirical models require little computational effort, they are known to perform poorly in heterogeneous urban environment, in particular in historically grown cities. Although ray tracing approaches can provide a much better prediction quality, high run times (in comparison to empirical approaches) have so far inhibited wide-spread use in large-scale scenarios.

The prediction of radio waves is quite similar to computing global illumination from a

computer graphics point of view. Due to the different wave lengths of common mobile radio frequencies, diffraction at building edges becomes a dominant propagation effect. Thus, in order to benefit from the acceleration of global illumination algorithms, those approaches have to be adapted to also incorporate diffraction efficiently. Towards this end, we presented distinct algorithms to compute dominant propagation paths due to diffraction at street corners and over building rooftops. Our main method is based on the repeated computation of discrete LOS regions which can efficiently be determined on the GPU. We adapted a basic shadow volume algorithm to work particularly fast on the special 2.5D geometry of common building databases, where a building is represented only by its polygonal outline and its height.

Moreover, we developed urban micro cell prediction models that derive the mean received signal strength directly from our diffraction ray paths. A key component of these models is a calibration technique to estimate unknown environmental influences like building material or vegetation directly from measurements. We show that this calibration process is not limited to dominant path models only but can also be extended to a closed-form solution for multi-path models. Our EDM model can be evaluated in under 0.05 seconds with the proposed GPU implementation and achieves a standard deviation between prediction and measurement of 4 to 7 dB. This is an extremely accurate result, considering that two consecutive measurement test runs already vary approximately around 3 dB.

In summary, we developed a run time efficient prediction algorithm that approximates dominant ray paths based on diffraction. We combined this algorithm with an accurate prediction model of the mean received signal strength in dense urban propagation environments.

In addition, we developed a VR interface to let the user interactively explore an urban propagation environment. While the user take active control over antenna placement he receives instant feedback on the propagation behavior of individual transmitter sites. Additional transmitter sites can also be constructed and network statistics show overall coverage, cell sizes and inter-cell interference estimates. We also made the building database dynamic and let the user change building heights and immediately show the influence on the propagation prediction. In particular, we developed a sketching based interface that lets the user outline a building on a geographical referenced satellite image directly in the VE. Thereby we further strengthened our exploratory approach to assist the domain scientist investigating or developing hypothesis about the data.

Future Directions The work of this thesis on the computation of diffraction paths could be extended to also include propagation paths due to reflection or scattering which may have a significant influence depending on the building material and vegetation in the propagation environment. Additionally, the computation of multi-path effects are of

great interest when designing channel coding and control algorithms. Since our computation is optimized for the estimation of the strongest paths, alternative approaches should be explored to compute a number of arriving rays per receiving location. Moreover, a limitation of the proposed algorithms is that they all require a discrete transformation of the propagation environment at a very early stage of the computation which sometimes introduces aliasing artifacts. Keeping a continuous data representation as long as possible is therefore a desirable goal in future algorithms. Some of the above points have already been addressed by Schmitz et al.[134] as an extensions to the present work. Moreover, the effect of vegetation, terrain elevation or dynamic traffic has been neglected so far and would be an interesting area to explore in future research.

4.1.2. Fiber Pathway Estimation

Chapter 3 presented a prototype implementation of the probabilistic tractography algorithm as introduced by Behrens et al. [15] which we adapted to specifically benefit from the GPU's many-core architecture. Based on the underlying mathematical model of probabilistic tractography (3.10), we derived the individual computational steps, examined possible computing device related restrictions, and formulated corresponding algorithms.

The computational core, as sketched in Algorithm 12, starts the tracing from a seed voxel and propagates the seed's position along local (random) fiber orientations until a stopping criteria is met. A new fiber orientation can depend on the previous direction to assure an anatomically consistent fiber course. All intermediate positions of the fiber streamlines are recorded in a global buffer that represents the global connectivity distribution of the original seed voxel to every other voxel after a sufficient number of iterations. We also implemented two state-of-the-art extensions: a multi-fiber model and loop checking. The multi-fiber model increases sensitivity in regions of low anisotropy or when tracking non-dominant fiber directions, thus, further supporting the general methodological issue of crossing fibers. Checking the generated traces for loops is of particular importance to avoid an artificial increase of probability within loops. We presented an extension to current approaches that also makes use of the inherent probabilistic framework and allows the trade-off between memory requirement and accuracy. We established the anatomical plausibility of our implementation with an expert review, where we also discussed possible advantages of our probabilistic loop checking approach with respect to the resulting fiber structure.

We show that our GPU implementation significantly reduced the overall run time compared to a reference CPU implementation. Populations of 100K samples can be computed in under one second and 1M samples can still be processed in less than ten seconds. A major limitation of our approach is the memory consumption of loop checking which reduces the number of concurrent executions per kernel invocation. However, we obtain

huge speedups over the reference implementation, ranging from 185X to 800X without loop checking for 10K and 1M samples, respectively. With loop checking we get speedups in the range of 17X to 78X with a supervoxel size of five and up to 100X with a supervoxel size of ten, again for 10K and 1M samples. The size of the supervoxel is a major criteria to trade a lesser memory requirement for a reduced loop checking accuracy. However, it remains an open issue if a reduced loop checking accuracy may significantly affect the resulting fiber tracts. This is an issue best addressed in close collaboration with domain scientists who are able to appropriately assess fiber tract difference since an overall ground truth is still subject to an ongoing discussion.

In addition to an efficient parallel implementation, we put a special focus on a direct coupling between the tracing procedure and a visualization of intermediate results such that the user can direct the computation towards a desired outcome if necessary. Our visualization of probabilistic fiber tracts relies on the direct volume rendering of their spatial distributions. By encoding the probabilities with color and opacity, the course of the fiber structure is displayed in relationship to its confidence and anatomical landmarks. We adapted a classical ray casting approach to support multiple data modalities and integrated it into the visualization library FlowLib [129] of the ViSTA VR Toolkit [7]. The value of our visualizations can also be seen by their usage in the neuroscientific publications of Eickhoff et al. [49] and Caspers et al. [36].

In summary, we integrated the computation and the visualization into a VR prototype application that incorporated the following aspects: (1) An interactive computation of probabilistic fiber tracts to assist in exploratory discoveries and the development of an intuitive understanding, (2) a real-time visualization of probabilistic fiber tracts that is directly coupled to the tract computation and conveys the course of uncertainty, and (3) a real-time interaction to disambiguate between multiple data modalities and to reveal fibers in their structural context.

Future Directions We currently use the voxel resolution given by the DTI measurements also for recording samples in the spatial *pdf* (cf. Algorithm 15 in Section 3.4.2). Since we already trace with subvoxel accuracy, we could easily extend Algorithm 15 to also record the connectivity profile at a higher resolution. In turn, the resulting probabilistic fiber tracts may exhibit finer structures and smoother transitions between probabilities.

Additional seeding strategies could be integrated in the VR interface to further support the exploratory analysis process. Of special interest would be a seeding with additional way points. Only fiber tracts which reach all of the way points would be included in the final connectivity profile. However, this would require a change in the algorithmic structure, instead of showing intermediate results after each step, the validity of a single streamline could only be determined after a certain number of steps, i.e., when all way

points have been visited. Such a change would either require extra device memory per sample or a memory transfer to the host after each integration step.

We use cytoarchitectonically brain areas [167] that are also described as probability maps. Currently, seeding masks are constructed by thresholding against a certain probability value (e.g. 90 percent) and uniformly distributing seeds within the mask. Since our approach allows the seeding of much more samples than was possible before, we could also integrate the probability profile of the brain areas into the seeding procedure. The main idea is to distribute the number of seeding points per voxel in the brain mask weighted with the local probability.

4.2. Contributions

We structure the main contributions of this thesis based on the individual chapters and sections.

Radio Wave Propagation

- The principle of diffraction has been introduced in the field of electromagnetics by the Geometric Theory of Diffraction [77] and the Universal Theory of Diffraction [87] and is discussed in subsequent work in great detail. However, the computation has remained computationally expensive. Work in computer graphics has focused on visually pleasing representations of the diffraction effect on small-scale objects such as a compact disc [138]. Formulating and implementing efficient algorithms for the computation of diffraction ray paths in heterogeneous urban environments is therefore the major contribution of Section 2.4. Moreover, this approach has led to the first published real-time simulation of radio waves at a comparable prediction accuracy.

- Although the concept of shadow volumes [54] is well studied in computer graphics, to the best of our knowledge this is their first use in the context of radio wave simulation. We adapt the principal idea and construct geometric representations of diffraction cones to let the GPU rasterize areas of diffraction. Moreover, we exploit the special structure of the underlying 2.5D building database to speed up the cone construction, cf. Section 2.4.1.1. We apply this method to effectively reconstruct ray paths due to diffraction into street canyons and over building rooftops.

- We present mathematical models that describe how to derive path loss and received signal strength from the computed ray paths. A key component of these models is the estimation of unknown influences from a small measurement sample. Section 2.4.2 shows how we can significantly increase prediction accuracy and discusses some stability issues of the calibration process with respect to different physical constraints. This estimation procedure has previously been proposed only to single path models as an iterative algorithm that lacks a proof of convergence [94]. Although the computation of multi-path effects is not discussed in detail in this thesis, we present a closed-form solution for multi-path optimization with an arbitrary but fixed number of deflections and arriving paths in Section 2.4.3.

- We integrated the individual components in a VR interface where the user can explore the propagation environment and interact with antenna placement, network statistics, and especially with a dynamic building database. In particular, we discuss the necessary steps to make established interaction techniques in VR work in combination to each other. The combination of these techniques itself may

represent no original research per se, but form a technical and not less significant contribution towards the implementation of real-world systems.

Tracing Neuronal Fiber Pathways

- Although quite novel, probabilistic tractography itself is a published and accepted method for the estimation of brain connectivity [15, 89, 73]. The contribution of this thesis is not a reinvention or a completely new approach of the probabilistic tractography method. In contrast, we investigated how the promising approach of Behrens et al. [15, 16] can be adapted to benefit from a massively parallel implementation on special purpose compute devices (GPUs). We decomposed the original mathematical model into parallel tasks that scale with a huge number of cores, each with very limited computing capabilities. It is therefore a predominantly technical contribution from a computer science's perspective but with a significant impact towards its application in the domain of neuroscience. In discussion with domain experts, we found that a limiting factor on the application of probabilistic tractography is in fact computation time, in particular when executing large-scale studies on hundreds of human brains. Since the algorithm is inherently probabilistic, a large number of samples are required for a reliable connectivity distribution, further adding to the overall computation time. Reducing the computation time by a factor of 100 for a practical computing condition (cf. Section 3.6.3.3) is therefore a main contribution of Chapter 3.

- Moreover, a methodical contribution in probabilistic tractography algorithms has been made by extending the original algorithm by a probabilistic loop checking method that allows fiber tracts to evolve over longer distances. Previously, loop checking has been performed on a coarser grid resolution and traces that revisit a coarse grid voxel were discarded. In consequence, we observed a tendency for fiber tracts to stop earlier (even when not running in loops) than their counterpart with no loop checking (cf. Figure 3.10 and Figure 3.11). Instead of a hard decision to discard traces we introduced a soft state that describes our belief of revisiting a previously seen supervoxel. We would then continue to trace samples based on thresholding our loop belief against a random component. The first results show a promising potential for our probabilistic loop checking in terms of more evolved and longer fiber tracts, consistent with the expected outcome.

- We also addressed the issue of interpreting the resulting probabilistic fiber tracts, in particular, the integration and disambiguation of the various neuroscientific data modalities (e.g. MRT, PET, brain areas). We approached this by adapting state-of-the-art direct volume rendering techniques to easily visualize multiple data sets and customizing it to the special requirements of VEs. Moreover, we combined the static visualization with interaction techniques (magic lens interaction) to dis-

ambiguate between different data modalities. Therefore, the contribution of this thesis to neuroscientific visualization lies not in a single aspect itself but in a combination and adaption of established methods that provide the basic framework for an interpretation and exploration of probabilistic tractography.

- We developed a VR application that combines each single aspect of the probabilistic tractography – computation, rendering and interaction – by a direct coupling between simulation and visualization of probabilistic fiber tracts. This contribution is predominantly of technical nature, nevertheless non-trivial, integrating all individual contributions into a real-world application.

In relation to the big picture as described in Chapter 1.1 the present thesis contributed to the two specific application areas by allowing the user to be an integral part of the workflow, observe computation results in real-time and to react appropriately. We enabled an interactive manipulation by decomposing and reformulating the domain specific algorithms into parallel sub tasks and demonstrated their scalability in the presence of a huge number of cores as available on current graphics hardware. Moreover, the significant increase in computing power has not only lead to an ever increasing amount of data sizes, but also to more and more complex algorithms. Controlling and making the best possible use of such algorithms is therefore crucial to exploit the full potential of a human-in-the-loop to see patterns and structures arise. Our major motivation to use VR interfaces is therefore to put the user in more control of the computing pipeline, or in a manner of speaking, to put the user directly at the interface between computation and visualization with the means to influence both.

APPENDIX A

BRAIN DATA & ADDITIONAL VISUALIZATIONS

Data Acquisition for Probabilistic Tractography The institute of Neuroscience and Medicine (INM-2) Research Center Jülich, Germany provided us with diffusion-weighted data from ten healthy, right-handed human subjects (five males, five females). Diffusion data was acquired on a 3.0 T Tim-Tri Siemens whole-body scanner (Siemens, Erlangen, Germany) with a maximum gradient strength of 40 mT m^{-1}, using a 12-channel phased-array head coil for signal reception. Subjects had no history of neurological or psychiatric disease, or head injury. All subjects gave informed, written consent to participate in the study which was conducted by the INM-2 and approved by the local Ethics Committee of the RWTH Aachen University.

Diffusion-weighted images were acquired using a twice-refocused spin-echo sequence (axial slices, slice thickness: 1.8 mmm, number of slices: 75, matrix = 128 × 128, field of view = 230 × 230 mm^2, bandwidth = 1502 Hz / pixel, reconstruction using an iPAT GRAPPA-scheme, final voxel resolution of 1.8 × 1.8 × 1.8 mm^3). The diffusion sensitive gradients were distributed along 60 directions in the icosahedral scheme [74]. For each set of diffusion-weighted data, 60 volumes with b-value = 800 s / mm^2 and 7 volumes with b-value = 0 s / mm^2 were obtained. For each subject, the entire diffusion measurement was repeated four times in successive sessions for subsequent averaging. The total scan time for the diffusion-weighted imaging protocol was about 50 min.

Additional Visualizations The visualizations depicted here were made by the neuroscientists from the Institute of Neuroscience and Medicine (INM-2) Research Center Jülich, Germany using our VR interfaces as described in Section 3.5.

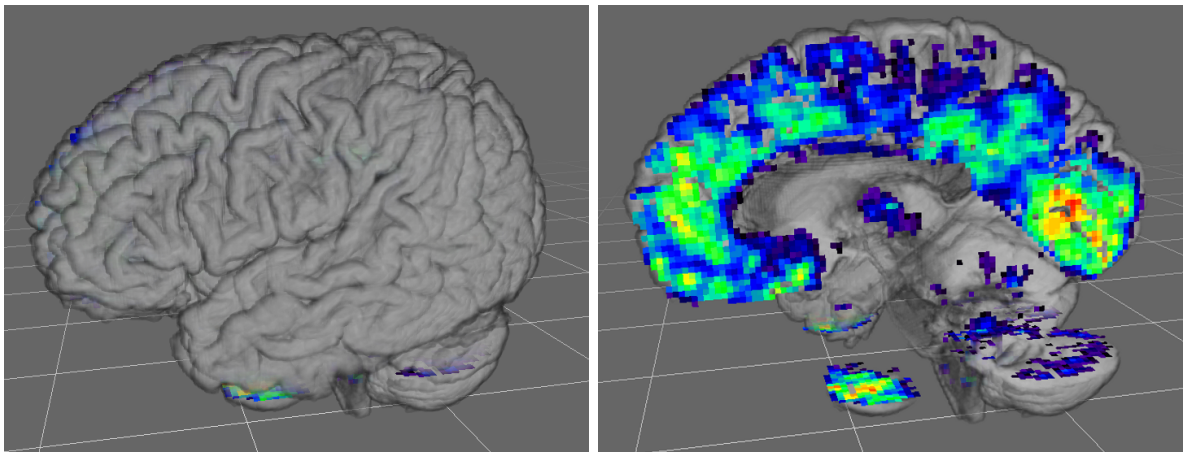


Figure A.1.: This figure shows a combination of MRT and PET that were recorded with different imaging devices and then co-registered. The MRT image provide structural context information for the PET image which is shown in distinct slices (PET resolution of approximately 2 mm^3).

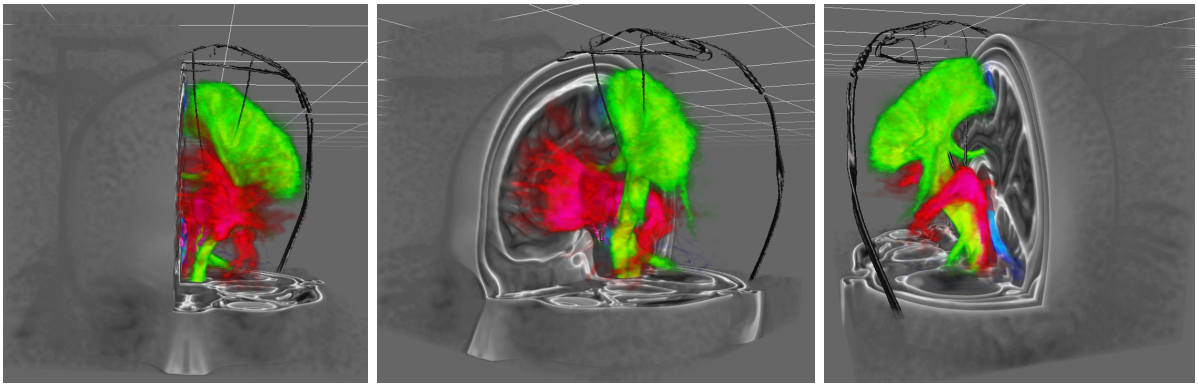


Figure A.2.: This data set examines the post-operative location of stimulation electrodes (black structure). Next to the electrodes are the core areas: subthalamic nucleus (nucleus subthalamicus), substantia nigra (substantia nigra), and red nucleus (nucleus ruber) and the fiber tracts: pyramidal tract (tractus pyramidalis), dentatorubrothalamic tract, (tractus dentatorubrothalamicus), anterior thalamic radiation, and medial forebrain bundle. The electrodes were reconstructed from CT data which had not been co-registered within the remaining MRT data. This leads to the increased distance between the electrodes and the rest of the brain and the skull as seen in the visualization.

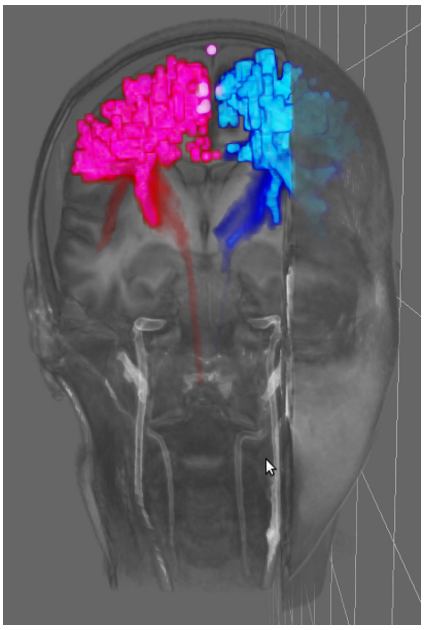


Figure A.3.: This MR image shows a lesion on the right hemisphere near the motor cortex the brain. The affected regions are sulcus centralis, gyrus precentralis passing to sulcus frontalis superior, and gyrus postcentralis.

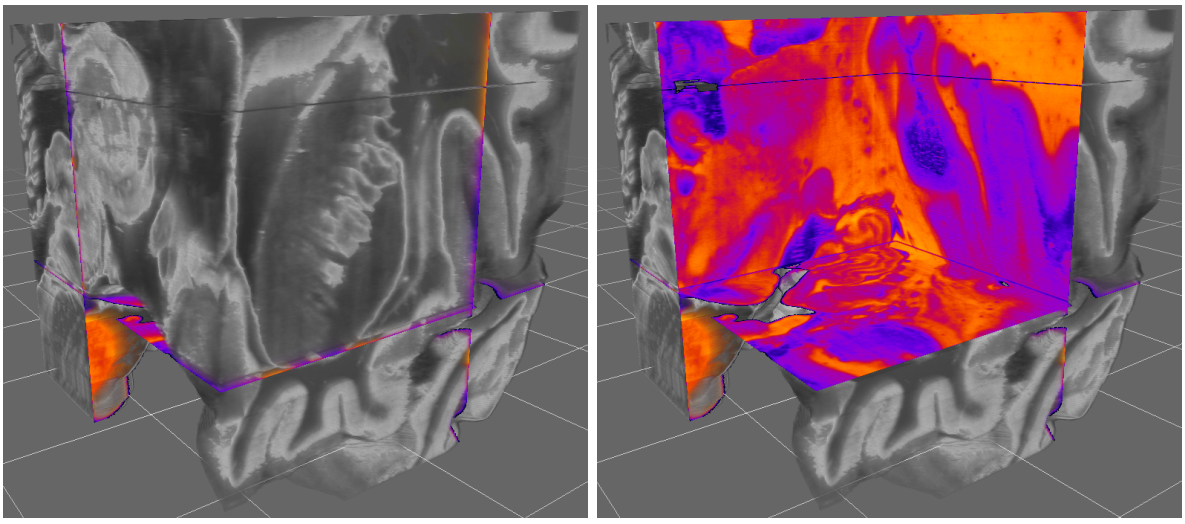


Figure A.4.: This data set consists of 512^3 voxels and shows a small subsection of a 3D reconstructed blockface data set from PLI near the cerebellum/brainstem.

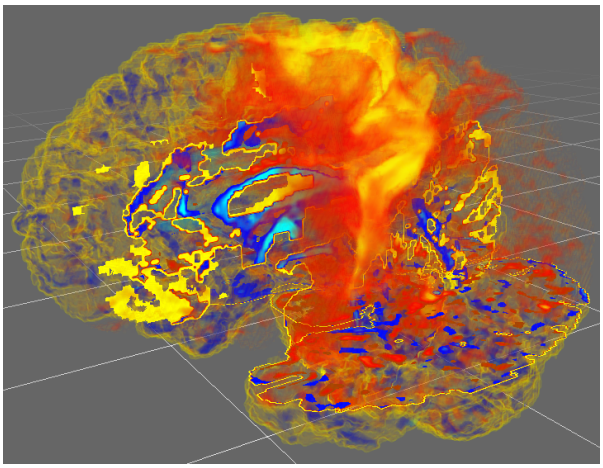


Figure A.5.: This figure shows the decrease in brain volume. The data set was reconstructed from images taken in intervals of several months. The visualizations shows a particular strong change in the motor region.

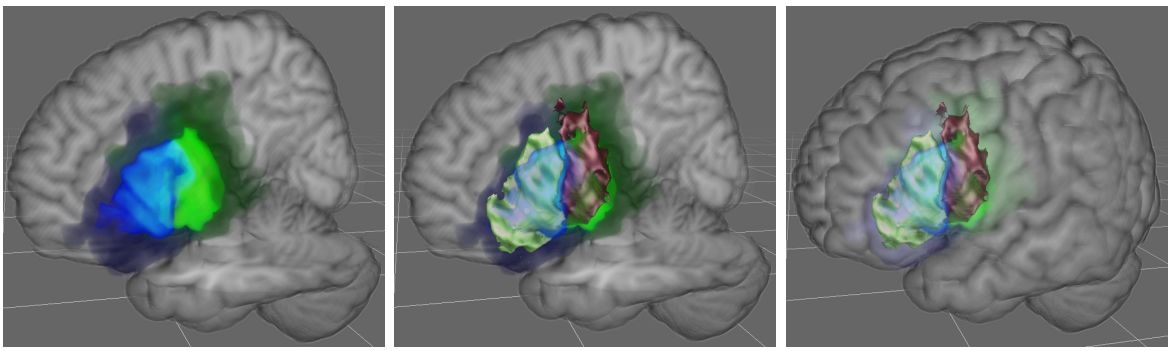


Figure A.6.: This image series shows different visualization of probabilistic brain areas. The left image shows volume rendering of the probability distribution, the middle images overlays the volume rendering with a thresholded geometric representation and the right image adds the structural information to the volume rendering.

APPENDIX B

TERMINOLOGY

^{T}	transpose of a vector or matrix
$\|x\|$	absolute value of a scalar x
$\| \vec{x} \|_2$	length of vector $\vec{x}$
$\langle \vec{x}, \vec{y} \rangle$	scalar product of two vectors $\vec{x}$ and $\vec{y}$
$\wedge$	logical operator *and*
$\vee$	logical operator *or*
$\bigcap$	set operator *intersection*
$\bigcup$	set operator *union*
$\emptyset$	empty set
c	speed of light
f	frequency
γ	distance dependent path loss coefficient
$PL^{\text{dB}}()$	path loss in dB
$\upsilon^{\rightsquigarrow}$	ray path, for instance $\upsilon^{\rightsquigarrow} = x_{i-1} \rightsquigarrow x_i \rightsquigarrow x_{i+1}$ describes a ray path that starts at x_{i-1}, changes direction at x_i and arrives at x_{i+1}
$\Upsilon^{\rightsquigarrow}$	set of ray paths $\upsilon_0^{\rightsquigarrow}, \ldots, \upsilon_N^{\rightsquigarrow}$

Table B.1.: General notation of radio wave propagation in Chapter 2.

$\mathbb{R}$	set of real numbers
Ω	observation space, in one dimension this can be for instance $\mathbb{R}$
$F_X(x)$	cumulative distribution function of a random variable X
$f_X(x)$	*probability density function* (*pdf*) of a random variable X
$\mathcal{P}(X \in I)$	probability that a realization of X lies in I
$f_{X,Y}(x,y)$	joint *pdf* of X and Y
$f_{X\|Y}(x\|y)$	conditional *pdf* of X given Y
(ϕ, θ)	fiber orientation in spherical coordinates
$\mathcal{V}$	space of all voxels
$N_\mathcal{V}$	number of voxels
U, V	individual voxels
V_i	ith voxel
$Y_\mathcal{V}$	observed DTI data for all voxels
$(\theta, \phi)_\mathcal{V}$	set of diffusion directions
$\mathcal{P}(\theta, \phi \| Y_\mathcal{V})$	local distribution of fiber orientation

Table B.2.: General notation of neural pathway estimation in Chapter 3.

Uniform Distribution A random variable X is *uniformly distributed*, $X \sim U(a,b), a < b \in \mathbb{R}$, if

$$\begin{aligned} f_X(x) &= \begin{cases} \frac{1}{b-a} & , a \leq x \leq b \\ 0 & , \text{otherwise} \end{cases} \\ &= \frac{1}{b-a} \cdot \mathrm{I}_{[a,b]}(x) \quad , \text{with } \mathrm{I}_A(x) = \begin{cases} 1 & , x \in A \\ 0 & , x \notin A \end{cases}, A \subseteq \mathbb{R}. \end{aligned}$$

APPENDIX C

ACRONYMS

6DOF	*Six Degrees of Freedom*
CAVE	*Cave Automatic Virtual Environment*
CC	*Compute Condition*
CORLA	*Cube-Oriented-Ray-Launching-Algorithm*
COST-WI	*COST-Walfisch-Ikegami*
CPU	*Central Processing Unit*
CT	*Computed Tomography*
CUDA	*Compute Unified Device Architecture*
DDR	*Double Data Rate*
DT	*Diffusion Tensor*
DTI	*Diffusion Tensor Imaging*
DT-MRI	*Diffusion Tensor Magnetic Resonance Imaging*
DVR	*Direct Volume Rendering*
DWI	*Diffusion Weighted Image*
EDM	*Edge Diffraction Model*
FA	*Fractional Anisotropy*
GPGPU	*General Purpose Computation on Graphics Processing Units*
GPU	*Graphics Processing Unit*
HARDI	*High Angular Resolution Diffusion Imaging*
HDWM	*Horizontal Diffraction Wall Map*
iid	*Independent Identically Distributed*
IPL	*Human Inferior Parietal Lobule*
LIC	*Line Integral Convolution*
LOS	*Line-of-Sight*
MC	*Monte-Carlo*
MCMC	*Markov-Chain-Monte-Carlo*

ME *Mean Error*
ML *Maximum Likelihood*
MR *Magnetic Resonance*
MRI *Magnetic Resonance Imaging*
MRT *Magnetic Resonance Tomography*
MSE *Mean Squared Error*
NLOS *Non-Line-of-Sight*
PDD *Principal Diffusion Direction*
pdf *probability density function*
PET *Positron Emission Tomography*
PLI *Polarized Light Imaging*
RDM *Roof Diffraction Model*
RV *Random Variable*
SIMD *Single Instruction Multiple Data*
STD *Standard Deviation*
TF *Transfer Function*
VE *Virtual Environment*
VR *Virtual Reality*

Curriculum Vitae

Personal Data

Name	Tobias Rick
Date of birth	30.01.1981
Place of birth	Neuss, Germany
Citizenship	Germany

Education

2000	General qualification for University entrance: Abitur, Rhein-Gynmnasium Sinzig
2001 – 2006	Academic studies at RWTH Aachen University, major subject Computer Science and minor subject Electrical Engineering
2006 – 2011	Research Assistant at RWTH Aachen University

BIBLIOGRAPHY

[1] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering*. Natick, MA: A K Peters, Ltd., 2002.

[2] Andrew L. Alexander, Khader Hasan, Gordon Kindlmann, Dennis L. Parker, and Jay S. Tsuruda. A geometric analysis of diffusion tensor measurements of the human brain. *Magnetic Resonance in Medicine*, 44(2):283–291, 2000.

[3] J.B. Andersen, T.S. Rappaport, and S. Yoshida. Propagation measurements and models for wireless communication channels. *IEEE Communications Magazine*, 33:42–49, 1995.

[4] A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[5] Franklin Antonio. *Faster line segment intersection*, pages 199–202. Academic Press Professional, Inc., 1992.

[6] G. Arfken. *Mathematical Methods for Physicists*. Academic Press, 3rd edition, 1985.

[7] Ingo Assenmacher and Torsten Kuhlen. The ViSTA virtual reality toolkit. In *The SEARIS Workshop on IEEE VR, Reno*, 2008.

[8] AWE Communications GmbH. WinProp. http://www.awe-communications.com.

[9] J.-F. Wagen B. Chopard, P. Luthi. Wave propagation in urban microcells: A massively parallel approach using the tlm method. In *COST 231 TD 33*, 1995.

[10] Peter J. Basser. Fiber-tractography via diffusion tensor MRI (DT-MRI). In *Proceedings of the International Society for Magnetic Resonance in Medicine*, page 1226, 1998.

[11] Peter J. Basser, James Mattiello, and Denis Lebihan. MR diffusion tensor spectroscopy and imaging. *Biophysical Journal*, 66:259–267, 1994.

[12] Peter J. Basser, Sinisa Pajevic, Carlo Pierpaoli, Jeffrey Duda, and Akram Aldroubi. Fiber tractography using DT-MRI data. *Magnetic Resonance in Medicine*, 44:625–632, 2000.

[13] Peter J. Basser and C. Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI. *Journal of Magnetic ResonanceB*, 111:209–219, 1996.

[14] C. Beaulieu. The basis of anisotropic water diffusion in the nervous system - a technical review. *NMR in Biomedicine*, 15:435–455, 2002.

[15] T.E. Behrens, H. Johansen-Berg, M.W. Woolrich, S.M. Smith, C.A.M. Wheeler-Kingshott, P.A. Boulby, G.J. Barker, E.L. Sillery, K. Sheehan, O. Ciccarelli, A.J. Thompson, J.M. Brady, and P.M. Matthews. Non-invasive mapping of connections between human thalamus and cortex using diffusion imaging. *Nature Neuroscience*, 6(7):750–757, 2003.

[16] T.E. Behrens, H.J. Johansen-Berg, S. Jbabdi, M.F. Rushworth, and M.W. Woolrich. Probabilistic diffusion tractography with multiple fibre orientations: What can we gain? *Neuroimage*, 34(1):144–155, 2007.

[17] T.E. Behrens, M.W. Woolrich, M. Jenkinson, H. Johansen-Berg, R.G. Nunes, S. Clare, P.M. Matthews, J.M. Brady, and S.M. Smith. Characterization and propagation of uncertainty in diffusion-weighted MR imaging. *Magnetic Resonance in Medicine*, 50(5):1077–1088, 2003.

[18] Tim Behrens. *MR Diffusion Tractography: Methods and Applications*. PhD thesis, Oxford Centre for Functional Magnetic Resonance Imaging of the Brain and Department of Engineering Science University of Oxford, 2004.

[19] Carsten Benthin, Ingo Wald, and Philipp Slusallek. A scalable approach to interactive global illumination. *Computer Graphics Forum*, 22(3):621–630, 2003.

[20] Jon Louis Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979.

[21] J.-E. Berg. A recursive method for street microcell path loss calculations. In *in Proc. Personal, Indoor and Mobile Radio Conference PIMRC*, pages 140–143, 1995.

[22] J. I. Berman, S. Chung, P. Mukherjee, C. P. Hess, E. T. Han, and R. G. Henry. Probabilistic streamline q-ball tractography using the residual bootstrap. *Neuroimage*, 39(1):215–222, 2008.

[23] Johanna Beyer. *GPU-based Multi-Volume Rendering of Complex Data in Neuroscience and Neurosurgery.* PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2009.

[24] Johanna Beyer, Markus Hadwiger, Stefan Wolfsberger, and Katja Buhler. High-quality multimodal volume rendering for preoperative planning of neurosurgical interventions. *IEEE Transactions on Visualization and Computer Graphics*, 13:1696–1703, 2007.

[25] K.C. Bhavaraju. Interactive virtual reality simulation for nanoparticle manipulation and nanoassembly using optical tweezers. In *Proc. IEEE Virtual Reality Conference*, pages 251 –252, 2009.

[26] Eric Bier, Maureen Stone, and Ken Pier. Enhanced illustration using magic lens filters. *IEEE Computer Graphics and Applications*, 17(6):62–70, 1997.

[27] M. Björnemo, A. Brun, R. Kikinis, and C.-F. Westin. Regularized stochastic white matter tractography using diffusion tensor MRI. In *Medical Image Computing & Computer Assisted Intervention*, pages 435–442, 2002.

[28] Ralph Brecheisen, Anna Vilanova, Bram Platel, and Bart ter Haar Romeny. Parameter sensitivity visualization for DTI fiber tracking. *IEEE Transactions on Visualization and Computer Graphics*, 15:1441–1448, 2009.

[29] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[30] Steve Bryson and Michael Gerald-Yamasaki. The distributed virtual windtunnel. In *Proceedings of Supercomputing '92*, pages 275–284, 1992.

[31] Steve Bryson and Creon Levit. The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *IEEE Visualization*, pages 17–24, 1991.

[32] Michael Bussler, Tobias Rick, Andreas Kelle-Emden, Bernd Hentschel, and Torsten Kuhlen. Interactive particle tracing in time-varying tetrahedral grids. In *Proc. Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pages 71–80, 2011.

[33] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 263–270. ACM, 1993.

[34] N. Cardona, P. Moeller, and F. Alonso. Applicability of the Walfisch-type urban propagation models. In *Electronical Letters*, volume 31, 1995.

[35] S. Caspers, S. Geyer, A. Schleicher, H. Mohlberg, K. Amunts, and K. Zilles. The human

inferior parietal cortex: cytoarchitectonic parcellation and interindividual variability. *Neuroimage*, 33(2):430–448, 2006.

[36] Svenja Caspers, Simon B. Eickhoff, Tobias Rick, Anette von Kapri, Torsten Kuhlen, Ruiwang Huang, Nadim J. Shah, and Karl Zilles. Probabilistic fibre tract analysis of cytoarchitectonically defined human inferior parietal lobule areas reveals similarities to macaques. *Neuroimage*, 58:362–380, 2011.

[37] D. Catrein, M. Reyer, and T. Rick. Accelerating radio wave propagation predictions by implementation on graphics hardware. In *Proc. IEEE Vehicular Technology Conference*, pages 510–514, 2007.

[38] J. H. Causebrook. Signal attenuation by buildings and trees. In *COST 231 TD 106*, 1990.

[39] Wei Chen, Zi'ang Ding, Song Zhang, Anna MacKay-Brandt, Stephen Correia, Huamin Qu, John Allen Crow, David F. Tate, Zhicheng Yan, and Qunsheng Peng. A novel interface for interactive exploration of DTI fibers. *IEEE Transactions on Visualization and Computer Graphics*, 15:1433–1440, 2009.

[40] Wei Chen, Song Zhang, Stephen Correia, and David F. Tate. Visualizing diffusion tensor imaging data with merging ellipsoids. *IEEE Pacific Visualization Symposium*, 0:145–151, 2009.

[41] D. J. Cichon and W. Wiesbeck. Comprehensive ray optical propagation models for indoor and outdoor environments. In *Proc. Commsphere*, pages 201–208, 1995.

[42] L. M. Correia, editor. *COST Action 273: Mobile Broadband Multimedia Networks, Final Report*. Academic Press, 2006.

[43] FSL 4.1, 2008. http://www.fmrib.ox.ac.uk/fsl/.

[44] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Frédo Durand. Implicit visibility and antiradiance for interactive global illumination. In *ACM SIGGRAPH*, page 61. ACM, 2007.

[45] E. Damosso, editor. *COST Action 231: Digital mobile radio towards future generation systems, Final Report*. Office for Official Publications of the European Communities, 1999.

[46] Julien Dauguet, Sharon Peled, Vladimir Berezovskii, Thierry Delzescaux, Simon K. Warfield, Richard Born, and Carl-Fredrik Westin. Comparison of fiber tracts derived from in-vivo DTI tractography with 3D histological neural tract tracer reconstruction on a macaque brain. *Neuroimage*, 37:530–538, 2007.

[47] Gerwin de Haan, Michal Koutek, and Frits H. Post. IntenSelect: Using dynamic object

rating for assisting 3d object selection. In Erik Kjems and Roland Blach, editors, *Proceedings of the 9th Internationial Immersive Projection Technology Workshop (IPT) and 11th Eurographics Workshop on Virtual Environment (EGVE)*, pages 201–209, 2005.

[48] J. Deygout. Multiple knife-edge diffraction of microwaves. *IEEE Transactions on Antennas and Propagation*, 14(4):480–489, 1966.

[49] Simon B. Eickhoff, Saad Jbabdi, Svenja Caspers, Angela R. Laird, Peter T. Fox, Karl Zilles, and Timothy E. J. Behrens. Anatomical and functional connectivity of cytoarchitectonic areas within the human parietal operculum. *The Journal of Neuroscience*, 30(18):6409–6421, 2010.

[50] Frank Enders, Natascha Sauber, Dorit Merhof, Peter Hastreiter, Christopher Nimsky, and Marc Stamminger. Visualization of white matter tracts with wrapped streamlines. In *IEEE Visualization*, page 7, 2005.

[51] V. Erceg, S. J. Fortune, J. Ling, J. Rustako, and R. A. Valenzuela. Comparison of a computer-based propagation prediction tool with experimental data collected in urban microcellular environments. *IEEE Journal on Selected Areas in Communications*, 15:677–684, 1997.

[52] Vinko Erceg, Larry J. Greenstein, Sony Y. Tjandra, Seth R. Parkoff, Ajay Gupta, Boris Kulic, Arthur A. Julius, and Renee Bianchi. An empirically based path loss model for wireless channels in suburban environments. *IEEE Journal on Selected Areas in Communications*, 17:1205–1211, 1999.

[53] R. Fernando and M. Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Reading, MA: Addison-Wesley, 2003.

[54] Randima Fernando. *GPU Gems*, chapter Effective Shadow Volume Rendering, pages 137–166. Addison-Wesley, 2004.

[55] A. Fick. On liquid diffusion. *Philosophical magazine*, 10(63):30–39, 1855.

[56] Andrew S. Forsberg, David H. Laidlaw, Andries van Dam, Robert M. Kirby, George E. Karniadakis, and Jonathan L. Elion. Immersive virtual reality for visualizing flow through an artery. In *IEEE Visualization*, pages 457–460, 2000.

[57] Scott Frees, G. Drew Kessler, and Edwin Kay. PRISM interaction for enhancing control in immersive virtual environments. *ACM Trans. Comput.-Hum. Interact.*, 14(1):2, 2007.

[58] Anton Fuhrmann and Eduard Gröller. Real-time techniques for 3D flow visualization. In *IEEE Visualization*, pages 305–312. IEEE Computer Society Press, 1998.

[59] Thomas Funkhouser, Patrick Min, and Ingrid Carlbom. Real-time acoustic model-

ing for distributed virtual environments. In *ACM SIGGRAPH*, pages 365–374. ACM Press/Addison-Wesley Publishing Co., 1999.

[60] N. Geng and W. Wiesbeck. *Planungsmethoden für die Mobilkommunikation*. Springer, 1998.

[61] M. Hadwiger, C. Sigg, H. Scharsach, K. Brhler, and M.Gross. Real-time ray-casting and advanced shading of discrete isosurfaces. *Proceedings of Eurographics 2005*, pages 303–312, 2005.

[62] Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, Daniel Weiskopf, and Klaus Engel. *Real-Time Volume Graphics*. A. K. Peters, 2006.

[63] N. Hagbi, R. Grasset, O. Bergig, M. Billinghurst, and J. El-Sana. In-place sketching for content authoring in augmented reality games. In *Proc. IEEE Virtual Reality Conference*, pages 91 –94, 2010.

[64] Masaharu Hata. Empirical formula for propagation loss in land mobile radio services. *IEEE Transactions on Vehicular Technology*, 29:317–325, 1980.

[65] Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. In *ACM SIGGRAPH*, pages 119–127. ACM, 1984.

[66] Bernd Hentschel, Irene Tedjo, Markus Probst, Marc Wolter, Marek Behr, Christian Bischof, and Torsten Kuhlen. Interactive blood damage analysis for ventricular assist devices. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1515–1522, 2008.

[67] Lee Howes and David Thomas. *Efficient Random Number Generation and Application Using CUDA*, chapter 37, pages 805–830. Addison-Wesley, 2007.

[68] E. Hsu. Generalized line integral convolution rendering of diffusion tensor fields. In *Proceedings of the International Society for Magnetic Resonance in Medicine*, page 790, 2001.

[69] Fumio Ikegami, Tsutomu Takeuchi, and Susumu Yoshida. Theoretical prediction of mean field strength for urban mobile radio. *IEEE Transactions on Antennas and Propagation*, 39:299–302, 1991.

[70] Fumio Ikegami, Susumu Yoshida, Tsutomu Takeuchi, and Masahiro Umehira. Propagation factors controlling mean field strength on urban streets. *IEEE Transactions on Antennas and Propagation*, 32:822–829, 1984.

[71] Intel Corporation, 2011. http://www.intel.com.

[72] Chris Johnson. Visualization viewpoints: Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[73] D. K. Jones. Tractography gone wild: probabilistic fibre tracking using the wild bootstrap with diffusion tensor MRI. *IEEE Transactions on Medical Imaging*, 27(9):1268–74, 2008.

[74] D.K. Jones, A. Simmons, S.C. Williams, and M.A Horsfield. Non-invasive assessment of axonal fiber connectivity in the human brain via diffusion tensor mri. *Magnetic Resonance in Medicine*, 42(1):37–41, 1999.

[75] Ed Kaiser, Alex Olwal, David McGee, Hrvoje Benko, Andrea Corradini, Xiaoguang Li, Phil Cohen, and Steven Feiner. Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality. In *ICMI '03: Proceedings of the 5th International Conference on Multimodal Interfaces*, pages 12–19, 2003.

[76] James T. Kajiya. The rendering equation. In *ACM SIGGRAPH*, pages 143–150. ACM Press, 1986.

[77] J. B. Keller. Geometrical theory of diffraction. *Journal of the Optical Society of America*, 52(2):116–130, 1962.

[78] Dennis Kempin, Peter Schumacher, Gernot Ziegler, and Tobias Rick. Interactive particle tracing with cumulative blood damage computation for ventricular assist devices. In *6. GI-Workshop Virtuelle und Erweiterte Realität*, 2009.

[79] S.-C. Kim, B. J. Guarino, T. M. Willis III, V. Erceg, S. J. Fortune, R. A. Valenzuela, L. W. Thomas, J. Ling, and J. D. Moore. Radio propagation measurements and prediction using three-dimensional ray tracing in urban environments at 908 mhz and 1.9 ghz. *IEEE Transactions on Vehicular Technology*, 48:931–946, 1999.

[80] G Kindlmann. Superquadric tensor glyphs. In *Proceedings of IEEE TVCG/EG Symposium on Visualization 2004*, pages 147–154, 2004.

[81] Gordon L. Kindlmann and David M. Weinstein. Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *IEEE Visualization*, pages 183–189, 1999.

[82] Gordon L. Kindlmann and Carl-Fredrik Westin. Diffusion tensor visualization with glyph packing. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1329–1336, 2006.

[83] Peter Kipfer, Mark Segal, and Rüdiger Westermann. UberFlow: A GPU-based particle engine. In *Proceedings of Graphics Hardware 2004*, pages 115–122. ACM Press, 2004.

[84] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach.* Morgan Kaufmann., 2010.

[85] M. Koch, D. Norris, and M. Hund-Georgiadis. An investigation of functional and anatomical connectivity using magnetic resonance imaging. *Neuroimage*, 16(1):241–250, 2002.

[86] Polina Kondratieva, Jens Krüger, and Rüdiger Westermann. The application of GPU particle tracing to diffusion tensor field visualization. In *IEEE Visualization*, 2005.

[87] 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*, 62(11):1448–1461, 1974.

[88] W. B. Langdon. A fast high quality pseudo random number generator for nvidia cuda. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2511–2514, New York, NY, USA, 2009. ACM.

[89] M. Lazar and A. Alexander. Bootstrap white matter tractography (boot-trac). *Neuroimage*, 24(2):524–532, 2005.

[90] K. Levenberg. A method for the solution of certain problems in least squares. *Appl. Math.*, 2:164–168, 1944.

[91] M. Lott and I. Forkel. A multi-wall-and-floor model for indoor radio propagation. In *Proc. IEEE Vehicular Technology Conference*, volume 1, pages 464–468, 2001.

[92] J.-F. Mangin, C. Poupon, Y. Cointepas, D. Rivire, D. Papadopoulos-Orfanos, C. A. Clark, J. Rgis, and D. Le Bihan. A framework based on spin glass models for the inference of anatomical connectivity from diffusion-weighted MR data. *NMR in Biomedicine*, 15:481–492, 2002.

[93] Mannesmann Mobilfunk GmbH, Germany. Cost 231 - urban micro cell measurements and building data, 1999.

[94] R. Mathar, M. Reyer, and M. Schmeink. A cube oriented ray launching algorithm for 3d urban field strength prediction. In *IEEE International Conference on Communications*, pages 5034 – 5039, 2007.

[95] Rudolf Mathar. *Informationstheorie, diskrete Modelle und Verfahren.* Teubner Verlag, 1996.

[96] T. McGraw and M. Nadar. Stochastic DT-MRI connectivity mapping on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1504–1511, 2007.

[97] Tim McGraw, Baba C. Vemuri, Zhizhou Wang, Yunmei Chen, Murali Rao, and Thomas Mareci. Line integral convolution for visualization of fiber tract maps from DTI. In *Medical Image Computing & Computer Assisted Intervention*, pages 615–622, 2002.

[98] A. Mittmann, M.A.R. Dantas, and A. von Wangenheim. Design and implementation of brain fiber tracking for gpus and pc clusters. In *21st International Symposium on Computer Architecture and High Performance Computing*, pages 101–108, 2009.

[99] B. Moberts, A. Vilanova, and J.J. van Wijk. Evaluation of fiber clustering methods for diffusion tensor imaging. In *IEEE Visualization*, pages 65–72, 2005.

[100] P. Momayyez and K. Siddiqi. 3d stochastic completion fields for fiber tractography. *Computer Vision and Pattern Recognition Workshop*, 0:178–185, 2009.

[101] Susumu Mori, Barbara J. Crain, V. P. Chacko, and Peter C. M. Van Zijl. Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging. *Annals of Neurology*, 45(2):265–269, 1999.

[102] Susumu Mori and Peter C. M. van Zijl. Fiber tracking: principles and strategies a technical review. *NMR in Biomedicine*, 15(7-8):468–480, 2002.

[103] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6:40–53, 2008.

[104] NVIDIA. CUDA C Best Practices Guide, 2011.

[105] NVIDIA. CUDA C Programming Guide, 2011.

[106] Tsutomu Okada, Yukio Miki, Yasutaka Fushimi, Takashi Hanakawa, Mitsunori Kanagaki, Akira Yamamoto, Shin-Ichi Urayama, Hidenao Fukuyama, Masahiro Hiraoka, and Kaori Togashi. Diffusion-tensor fiber tractography: intraindividual comparison of 3.0-T and 1.5-T MR imaging. *Radiology*, 238(2):668–78, 2006.

[107] Ryan Overback, Ravi Ramamoorthi, and William R. Mark. A real-time beam tracer with application to exact soft shadows. In *EGSR*, 2007.

[108] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[109] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports*, pages 21–51, 2005.

[110] G. Parker and D. Alexander. Probabilistic monte carlo based mapping of cerebral connections utilising whole-brain crossing fibre information. In *Proc. of Information Processing in Medical Imaging*, pages 684–695, 2003.

[111] M. Perucca. Small cells characterization using 3-d database and deygout diffraction approach. In *COST 231 TD 54*, 1994.

[112] Sebastian Pick, Bernd Hentschel, Marc Wolter, Irene Tedjo-Palczynski, and Torsten Kuhlen. Automated positioning of annotations in immersive virtual environments. In *Proceedings of the Joint Virtual Reality Conference of EuroVR - EGVE - VEC*, pages 1–8, 2010.

[113] C. Poullis and S. You. Automatic creation of massive virtual cities. In *Proc. IEEE Virtual Reality Conference*, pages 199 –202, 2009.

[114] Vesna Prckovska, Tim H. J. M. Peeters, Markus van Almsick, Bart ter Haar Romeny, and Anna Vilanova i Bartroli. Fused DTI/HARDI visualization. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints), 2010.

[115] A.A. Qazi, G. Kindlmann, L. O'Donnell, S. Peled, A. Radmanesh, S. Whalen, A.J. Golby, and C.-F. Westin. Two-tensor streamline tractography through white matter intra-voxel fiber crossings: Assessed by fMRI. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops.*, pages 1–8, 2008.

[116] A. Rajkumar, B. F. Naylor, F. Feisullin, and L. Rogers. Predicting rf coverage in large environments using ray-beam tracing and partitioning tree represented geometry. *Wirel. Netw.*, 2(2):143–154, 1996.

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

[118] Guido Reina, Katrin Bidmon, Frank Enders, Peter Hastreiter, and Thomas Ertl. GPU-based hyperstreamlines for diffusion tensor imaging. In *Proceedings of the Joint EG/IEEE VGTC Symposium on Visualization*, pages 35–42, 2006.

[119] M. Reyer, T. Rick, and R. Mathar. Graphics hardware accelerated field strength prediction for rural and urban environments. In *European Conference on Antennas and Propagation (EuCAP)*, pages 1–5, 2007.

[120] M. Reyer, T. Rick, and R. Mathar. Hardware acceleration techniques for 3D urban field strength prediction. In *Proc. ITG Wave Propagation in Communication, Microwave Systems and Navigation*, pages 16–20, 2007.

[121] T. Rick and R. Mathar. Fast edge-diffraction-based radio wave propagation model for graphics hardware. In *Proc. IEEE 2nd International ITG Conference on Antennas*, pages 15–19, 2007.

[122] T. Rick, A. von Kapri, and T. Kuhlen. GPU implementation of 3D object selection by conic volume techniques in virtual environments. In *Proc. IEEE Virtual Reality Conference*, pages 243–246, 2010.

[123] T. Rick, A. von Kapri, and T. Kuhlen. A virtual reality system for the simulation

and manipulation of wireless communication networks. In *Proc. IEEE Virtual Reality Conference*, pages 111–114, 2011.

[124] Tobias Rick and Torsten Kuhlen. Accelerating radio wave propagation algorithms by implementation on graphics hardware. In *Wave Propagation in Materials for Modern Applications*, pages 103–122. 2010. ISBN: 978-953-7619-65-7.

[125] Tobias Rick, Anette von Kapri, Svenja Caspers, Katrin Amunts, Karl Zilles, and Torsten Kuhlen. Visualization of probabilistic fiber tracts in virtual reality. In *Studies in Health Technology and Informatics*, volume 163, pages 486–492, 2011.

[126] K. Rizk. Propagation in microcellular and small cell urban environ- ment. In *Ph.D. dissertation, Ecole Polytechnique de Lausanne*, 1997.

[127] K. Rizk, J.-F. Wagen, and F. Gardiol. Ray tracing based path loss prediction in two microcellular environments. In *in Proc. Personal, Indoor and Mobile Radio Conference PIMRC*, pages 384–388, 1994.

[128] K. R. Schaubach, N. J. Davis IV, and T. S. Rappaport. A ray tracing method for predicting path loss and delay spread in microcellular environments. In *Proc. IEEE Vehicular Technology Conference*, volume 2, pages 932–935, 1992.

[129] Marc Schirski, Andreas Gerndt, Thomas van Reimersdahl, Torsten Kuhlen, Philipp Adomeit, Oliver Lang, Stefan Pischinger, and Christian Bischof. ViSTA flowlib - a framework for interactive visualization and exploration of unsteady flows in virtual environments. In *Proceedings of the 7th International Immersive Projection Technology Workshop (IPT) and 9th Eurographics Workshop on Virtual Environment (EGVE)*, pages 77–85, 2003.

[130] Marc Schirski, Torsten Kuhlen, Martin Hopp, Philipp Adomeit, Stefan Pischinger, and Christian Bischof. Virtual tubelets - efficiently visualizing large amounts of particle trajectories. *Computers & Graphics*, 29(1):17–27, 2005.

[131] A. Schmitz, T. Rick, T. Karolski, L. Kobbelt, and T. Kuhlen. Beam tracing for multipath propagation in urban environments. In *European Conference on Antennas and Propagation (EuCAP)*, pages 2631–2635, 2009.

[132] A. Schmitz, T. Rick, T. Karolski, T. Kuhlen, and L. Kobbelt. Simulation of Radio Wave Propagation by Beam Tracing. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 17–24, 2009.

[133] Arne Schmitz and Leif Kobbelt. Wave propagation using the photon path map. In *PE-WASUN '06*, pages 158–161. ACM, 2006.

[134] Arne Schmitz, Tobias Rick, Thomas Karolski, Torsten Kuhlen, and Leif Kobbelt. Effi-

cient rasterization for outdoor radio wave propagation. *IEEE Transactions on Visualization and Computer Graphics*, 17:159–170, 2011.

[135] T. Schultz, H. Theisel, and H.-P. Seidel. Topological visualization of brain diffusion MRI data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1496–1503, 2007.

[136] Anthony Sherbondy, David Akers, Rachel Mackenzie, Robert Dougherty, and Brian Wandell. Exploring connectivity of the brain's white matter with dynamic queries. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):419–430, 2005.

[137] D. Shreiner, J. Neider, M. Woo, and T. Davis. *OpenGL Programming Guide*. Reading MA: Addison-Wesley, fourth edition, 2003.

[138] Jos Stam. Diffraction shaders. In *ACM SIGGRAPH*, pages 101–110, 1999.

[139] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in Science and Engineering*, 12:66–73, 2010.

[140] Robert C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, (19):201–209, 1965.

[141] Nicolas Tsingos, Thomas Funkhouser, Addy Ngan, and Ingrid Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *ACM SIGGRAPH*, pages 545–552. ACM, 2001.

[142] Nicolas Tsingos, Emmanuel Gallo, and George Drettakis. Perceptual audio rendering of complex virtual environments. In *ACM SIGGRAPH*, pages 249–258. ACM, 2004.

[143] David Tuch. *Diffusion MRI of Complex Neural Architecture*. PhD thesis, Harvard-MIT Division of Health Sciences and Technology, 2002.

[144] David S. Tuch. Q-ball imaging. *Magnetic Resonance in Medicine*, 52(6):1358–1372, 2004.

[145] Andries van Dam, Andrew Forsberg, David H. Laidlaw, Joseph LaViola, and Rosemary Michelle Simpson. Immersive virtual reality for scientific visualization: A progress report. *IEEE Computer Graphics and Applications*, 20(6):26–52, 2000.

[146] J. van den Berg, J. Sewall, Ming Lin, and D. Manocha. Virtualized traffic: Reconstructing traffic flows from discrete spatio-temporal data. In *Proc. IEEE Virtual Reality Conference*, pages 183 –190, 2009.

[147] John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3d magic lenses.

In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 51–58, New York, NY, USA, 1996. ACM.

[148] Anette von Kapri, Tobias Rick, Svenja Caspers, Simon B. Eickhoff, Karl Zilles, and Torsten Kuhlen. Evaluating a visualization of uncertainty in probabilistic tractography. In *Medical Imaging 2010: Visualization, Image-Guided Procedures, and Modeling*, volume 7625, pages 762534–762546, 2010.

[149] Anette von Kapri, Tobias Rick, and Steven Feiner. Comparing steering-based travel techniques for search tasks in a CAVE. In *Proc. IEEE Virtual Reality Conference*, pages 91–94, 2011.

[150] Anette von Kapri, Tobias Rick, Tobias C. Potjans, Markus Diesmann, and Torsten Kuhlen. Towards the visualization of spiking neurons in virtual reality. In *Studies in Health Technology and Informatics*, volume 163, pages 685–687, 2011.

[151] R. Wahl, G. Wölfle, P. Wertz, P. Wildbolz, and F. Landstorfer. Dominant path prediction model for urban scenarios. In *14th IST Mobile and Wireless Communications Summit*, 2005.

[152] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *Proceedings of the 13th Eurographics workshop on Rendering*, EGRW '02, pages 15–24. Eurographics Association, 2002.

[153] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient GPU-based approach for interactive global illumination. In *ACM SIGGRAPH*, SIGGRAPH '09, pages 91:1–91:8. ACM, 2009.

[154] David Weinstein, Gordon Kindlmann, and Eric Lundberg. Tensorlines: advection-diffusion based propagation through diffusion tensor fields. In *IEEE Visualization*, pages 249–253, 1999.

[155] C.-F. Westin, S. E. Maier, B. Khidhir, P. Everett, F. A. Jolesz, and R. Kikinis. Image processing for diffusion tensor magnetic resonance imaging. In *Medical Image Computing & Computer Assisted Intervention*, Lecture Notes in Computer Science, pages 441–452, 1999.

[156] C.-F. Westin, S. E. Maier, H. Mamata, A. Nabavi, F. A. Jolesz, and R. Kikinis. Processing and visualization of diffusion tensor MRI. *Medical Image Analysis*, 6(2):93–108, 2002.

[157] C. F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F. A. Jolesz. Geometrical diffusion measures for MRI from tensor basis analysis. In *Proceedings of the International Society for Magnetic Resonance in Medicine*, Vancouver Canada, 1997.

[158] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980.

[159] J. Wiart. Microcell modelling when base station is below roof tops. In *Proc. 44th IEEE Vehicular Technology Conference VTC*, pages 200–204, 1994.

[160] Xi Wu and Mingyuan Xie. DTI noise reduction using anisotropic filtering. *Intelligent Information Technology Applications, 2007 Workshop on*, 2:94–96, 2009.

[161] F. Zhang, E. R. Hancock, C. Goodlett, and G. Gerig. Probabilistic white matter fiber tracking using particle filtering and von mises-fisher sampling. *Medical Image Analysis*, 13(1):5–18, 2009.

[162] Song Zhang, Çagatay Demiralp, and David H. Laidlaw. Visualizing diffusion tensor MR images using streamtubes and streamsurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):454–462, 2003.

[163] Song Zhang, Stephen Correia, and David H. Laidlaw. Identifying white-matter fiber bundles in DTI data using an automated proximity-based fiber-clustering method. *IEEE Transactions on Visualization and Computer Graphics*, 14:1044–1053, 2008.

[164] Xiangfen Zhang, Hong Ye, and Hongmei Zhang. Multi-channel wavelet-based diffusion method for denoising DTI images. *BioMedical Engineering and Informatics, International Conference on*, 2:178–182, 2008.

[165] Xiaoqiang Zheng and Alex Pang. HyperLIC. *IEEE Visualization*, 0:33, 2003.

[166] A. Zhmurov, K. Rybnikov, Y. Kholodov, and V. Barsegov. Generation of random numbers on graphics processors: Forced indentation in silico of the bacteriophage HK97. to appear Journal of Physical Chemistry B.

[167] K. Zilles and K. Amunts. Receptor mapping: architecture of the human cerebral cortex. *Current Opinion in Neurology*, 22(4):331–339, 2009.

[168] Gali Zimmerman-Moreno, Arnaldo Mayer, and Hayit Greenspan. Classification trees for fast segmentation of DTI brain fiber tracts. *Computer Vision and Pattern Recognition Workshop*, 0:1–7, 2008.