ABSTRACT

Title of dissertation: VISUAL COMPUTING TOOLS
FOR STUDYING
MICRO-SCALE DIFFUSION

Sujal Bista, Doctor of Philosophy, 2014

Dissertation directed by: Professor Amitabh Varshney
Department of Computer Science

In this dissertation, we present novel visual computing tools and techniques to facilitate the exploration, simulation, and visualization of micro-scale diffusion. Our research builds upon the latest advances in visualization, high-performance computing, medical imaging, and human perception. We validate our research using the driving applications of nano-assembly and diffusion kurtosis imaging (DKI). In both of these applications, diffusion plays a central role. In the former it mediates the process of transporting micron-sized particles through moving lasers, and in the latter it conveys brain micro-geometry.

Nanocomponent-based devices, such as bio-sensors, electronic components, photonic devices, solar cells, and batteries, are expected to revolutionize health care, energy, communications, and the computing industry. However, in order to build such useful devices, nanoscale components need to be properly assembled together. We have developed a hybrid CPU/GPU-based computing tool to understand complex interactions between lasers, optical beads, and the suspension medium. We demonstrate how a high-performance visual computing tool can be used to accelerate an optical tweezers simu-

lation to compute the force applied by a laser onto micro particles and study shadowing (refraction) behavior. This represents the first steps toward building a real-time nano-assembly planning system. A challenge in building such a system, however, is that optical tweezers systems typically lack stereo depth cues. We have developed a visual tool to provide an enhanced perception of a scene's 3D structure using the kinetic depth effect. The design of our tool has been informed by user studies of stereo perception using the kinetic-depth effect on monocular displays.

Diffusion kurtosis imaging is gaining rapid adoption in the medical imaging community due to its ability to measure the non-Gaussian property of water diffusion in biological tissues. Compared with the traditional diffusion tensor imaging (DTI), DKI can provide additional details about the underlying microstructural characteristics of neural tissues. It has shown promising results in studies on changes in gray matter and mild traumatic brain injuries, where DTI is often found to be inadequate. However, the highly detailed spatio-angular fields in DKI datasets present a special challenge for visualization. Traditional techniques that use glyphs are often inadequate for expressing subtle changes in the DKI fields. In this dissertation, we outline a systematic way to manage, analyze, and visualize spatio-angular fields using spherical harmonics lighting functions to facilitate insights into the micro-structural properties of the brain.

VISUAL COMPUTING TOOLS FOR STUDYING MICRO-SCALE DIFFUSION

by

Sujal Bista

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2014

Advisory Committee:
Professor Amitabh Varshney, Chair/Advisor
Professor Satyandra K. Gupta, Deans Representative
Professor David Mount
Professor Joseph F. JaJa
Professor Rao P. Gullapalli

Acknowledgments

I would like to thank my advisor Dr. Amitabh Varshney for encouraging me to join graduate school and giving me direction and support during this journey. With his encouragement, I have learned a great deal of knowledge about graphics and visualization, parallel computing, machine learning, and, most importantly, how to conduct research. It has been a delight to work with and learn from such an exceptional leader.

I would like to thank Dr. Satyandra K. Gupta for introducing and guiding me through the nanotweezers research project. I would also like to thank Dr. Jiachen Zhuo and Dr. Rao P. Gullapalli for giving me an opportunity to learn about medical imaging and for providing invaluable advice and direction. Dr. Joseph F. JaJa and Dr. David Mount, have provided me valuable advice and knowledge. I am thankful to all of my committee members.

I would like to thank Michael Pack for giving me an opportunity to work on the virtual earth project and for helping me sharpen my graphics programming skills.

I would like to thank my colleagues and lab-mates: Horace Ip, Rob Patro, Daehwan Kim, Sagar Chowdhury, André Maximo, Derek Juba, Philip Yang, Adil Yalçin, Ícaro Lins Leitão da Cunha, Eric Krokos, Greg Kramida, Patricia Sazama, and Hsueh-Chien Cheng for their support, friendship, helpful discussion, and for making graduate school life interesting.

I am thankful to my family and friends who have made this thesis possible.

Table of Contents

List of Tables v

List of Figures vi

1 Introduction 1

2 Using GPUs for Realtime Prediction of Optical Forces on Microsphere Ensembles 5
- 2.1 Introduction . . . 5
- 2.2 Related Work . . . 10
- 2.3 Approach . . . 14
  - 2.3.1 Ray-Object Intersection . . . 14
  - 2.3.2 Force Calculation . . . 15
  - 2.3.3 Force Calculation using Non-Negative Matrix Factorization . . . 17
  - 2.3.4 Force Integration . . . 20
- 2.4 Results and Discussion . . . 21
  - 2.4.1 Performance Comparison . . . 22
  - 2.4.2 Shadowing Phenomenon . . . 24
  - 2.4.3 Calibration and Validation . . . 27
- 2.5 Conclusion and Future Work . . . 32

3 Kinetic Depth Images: Flexible Generation of Depth Perception 33
- 3.1 Introduction . . . 33
- 3.2 Background . . . 36
- 3.3 Related Work . . . 36
- 3.4 Overview of our Approach . . . 39
- 3.5 Kinetic-Depth Effect Parameters . . . 40
- 3.6 Energy Map Computation . . . 42
- 3.7 Depth Mesh Generation . . . 45
  - 3.7.1 Scene Depth Approximation . . . 45
  - 3.7.2 Optimized depth compression . . . 46
  - 3.7.3 Perceptual re-mapping . . . 47
  - 3.7.4 Depth Mesh Enhancement . . . 50

3.8 Experiments . . . 52
3.8.1 Experiment I . . . 53
3.8.2 Experiment II . . . 56
3.9 Rendering Camera Motion . . . 57
3.10 Rendering and Interaction . . . 59
3.11 Results and Discussion . . . 60
3.11.1 Subjective Evaluation . . . 60
3.11.2 Limitations of our Approach . . . 63
3.12 Conclusions and Future Work . . . 64

4 Visualization of Brain Microstructure through Spherical Harmonics Illumination of Spatio-Angular Fields 65
4.1 Introduction . . . 65
4.2 Related Work . . . 68
4.3 Overview . . . 71
4.4 Background . . . 71
4.4.1 Diffusion Tensor Imaging . . . 71
4.4.2 Diffusion Kurtosis Imaging . . . 73
4.4.3 Spherical Harmonics . . . 74
4.5 Diffusion Kurtosis Imaging Data . . . 76
4.6 Approach . . . 77
4.6.1 Lighting . . . 77
4.6.2 Visualization and Interaction . . . 84
4.7 Case Study . . . 88
4.7.1 Case Study I . . . 88
4.7.2 Case Study II . . . 88
4.7.3 Case Study III . . . 89
4.7.4 Case Study IV . . . 90
4.8 Discussion . . . 91
4.9 Conclusion And Future Work . . . 93

5 Conclusions and Future Plans 95

Bibliography 99

Bibliography 100

List of Tables

2.1 The time in seconds taken by the various methods to compute total force exerted on a single microparticle performed 5000 times at different locations. 20
2.2 Here we show the comparison of precision between various methods rounded up to the nearest 4 digits. We take Ashkin's method as the reference and compute the relative error to compare other methods with an equal number of rays. As the number of rays increase, the relative error decreases in general but the computational cost increases. 21
2.3 The time taken (in seconds) by the various methods to compute total force exerted by a laser beam on 32 interacting microparticles computed 5000 times at different locations. It is interesting to note that when the number of rays is low, brute-force ray tracing is faster than the ray tracing method that uses a 3D grid data structure. This is due to the additional cost of creating and maintaining the data structure. 22

List of Figures

2.1 In an optical tweezer setup, a Gaussian laser beam is converged by a convex lens (objective lens of a microscope) to a focal point which is used for trapping microparticles. To create multiple optical traps, the laser beam is split into multiple-beams using a diffraction grating. Diagram courtesy of [1]. 6

2.2 An illustration of the optical tweezers system. A laser beam with a Gaussian-based intensity distribution is converged into a focal point with the help of a convex lens. The figure shows laser beam trapping microparticles at the focal point. 7

2.3 When an optical trap is placed close to a microparticle, it pulls the particle towards the focal point. The images above captured using the imaging device in the optical tweezers system show a microparticle moving into a trap. 8

2.4 A diagram showing the simplified ray-optics model for calculating the force. The incident ray is diverted from its original path when it interacts with the microparticle. This causes the ray to change its momentum. When the ray changes momentum due to the microparticle, equal and opposite force is applied to the microparticle. 13

2.5 An overview of the GPU pipeline. The properties of the laser and the 3D grid are saved into the constant GPU memory whereas the properties of the particles and the 3D grid cells are saved in the global GPU memory. These are used by the first GPU kernel that performs ray-object intersection and force per ray calculation. The output is written to a large global memory array. We then perform a parallel-prefix sum at the output. As the parallel-prefix sum adds up all the components together, Segmentation/Final force calculation kernel finds the proper segment boundaries for each component and subtracts necessary amount from the boundaries to compute the final result. 16

2.6 Pictorial view of the matrices that map discretized representation of incident ray angles to the force applied to the microparticle, the direction of the transmitted ray, and the position of the transmitted ray. The mapping is highly coherent which allows NMF to efficiently factorize each component of the matrix into two compact sized outer product matrices. Value of m used in our experiments is 4. 18
2.7 The final force contribution for each particle is calculated by subtracting values from the segment boundaries of an array that contains the result of the parallel-prefix sum. In this figure, we show how the final value of the scattering force is computed for a particle. 21
2.8 Here we show the time taken to compute the force exerted by a laser beam containing 32 rays 5000 times on a varying number of particles. We compare brute-force GPU ray tracing against GPU ray tracing with a 3D grid. As the number of particles increases, the use of a 3D grid data structure shows a clear advantage. 24
2.9 An illustration of the shadowing phenomenon. Figure (a) shows the focal point of three laser beams at location $(0.0, 0.0, 0.0)$, $(-1.0, 7.5, 0.0)$, and $(-1.0, 2.5, 0.0)$. Figure (b) shows the movement of a single particle from $(0.0, -4.0, 0.0)$ to $(0.0, 0.0, 0.0)$. The Figure (c) shows the movement of same particle when second particle is present at location $(-1.0, 5.5, 0.0)$. Finally, Figure (d) shows the difference in force experienced by the first bead caused by the shadowing phenomenon. 25
2.10 An illustration of the shadowing phenomenon similar to the previous figure. Figure (a) shows the movement of a single particle from $(-4.0, 0.0, 0.0)$ to $(0.0, 0.0, 0.0)$. The Figure (b) shows the movement of same particle when second particle is present at location $(-1.0, 5.5, 0.0)$. Finally, Figure (c) shows the difference in force experienced by the first bead caused by the shadowing phenomenon. 26
2.11 Here we show the arrangement of the microparticles in the upward and downward configuration. 28
2.12 Downward configuration with spacing 2.5 μm between the lower microparticles and laser moving with the velocity 22.4 μm/s. The two beads are trapped as the laser moves. The top row shows the captured video and the bottom row shows the simulated result. 29
2.13 Downward configuration with spacing 4.0 μm between lower two microparticles and laser moving with the velocity 22.4 μm/s. Only one bead is trapped as the laser moves. The top row shows the captured video and the bottom row shows the simulated result. 30
2.14 Here we show the comparison between the force calculated using our method and the force computed using stiffness. Here the focal point of the laser is located at the $(0.0, 0.0, 0.0)$ and we compute force by placing the microparticle along the Y-axis. Both forces are similar. The force computed using stiffness is an approximation but we validate our result since the stiffness value computed by Singer *et al.* [2] is calibrated. . . . 31

3.1 We first compute the optical flow and the depth map from the input images (generally a stereo pair). We then generate a triangulated depth mesh using the re-mapped depth map which is guided by human perception. We also create an energy map of the image using depth, centrality, and image saliency. By using the depth mesh and the energy map, we generate a high-quality animation to best experience the kinetic-depth effect. Our system has been informed by and validated through user studies. 34
3.2 The pivot point and the rotation axis of the scene are shown in both 3D space and the projection space. 40
3.3 Energy components. Here we show (a) the original image, (b) the depth map, (c) the saliency component, (d) the radial component, and (e) the final computed energy map. In these maps, yellow regions are associated with lower energy. We choose the pixel with the lowest energy to be the pivot point. 43
3.4 (a) the original image, the image set (b) is associated with the raw depth, and the image set (c) shows the depth used to generate final depth mesh after depth compression and perceptual enhancements. The depths in (b) and (c) are normalized. In each set we are showing the depth values and the optical flow after the camera motion to depict the value of depth remapping. 48
3.5 The image (a) shows the side view of the object being displayed in the experiment rendered with lighting for illustration purposes. The image (b) shows the stimulus shown to the participants. Without KDE motion, the structure is seen as a flat collection of points. 53
3.6 The figure shows the mean and standard deviation of the normalized depth perceived by subjects. When the objects are placed at various distances from the pivot point, the perceived velocity of the object changes. We show separate curves for each distance we tested and list the midpoint velocity. Between relative velocities of 0.2 to 1.25 degrees visual angle per second, participants perceived increased depth. For higher relative velocities, the perceived depth remained constant. 55
3.7 Here we show the total number of selections made when participants were asked if the depth perceived varied between the objects when positioning was different. If they observed a difference, they were asked to select the object with the greater depth. 57
3.8 An illustration of the different types of camera movements: the angular motion (a) where the camera swivels on a plane perpendicular to the rotation axis while looking at the pivot point, and the conical pendulum motion (b) where the camera rotates along a conical surface while looking at the pivot point. 58
3.9 Representation of the angular camera motion 58
3.10 The results of the subjective evaluation. Here we show the average user preference. 62

4.1 Using spherical harmonics lighting functions described in this chapter, we visualize spatio-angular fields. In this figure, we use the dataset of a patient suffering from traumatic brain injury. In traditional diffusion tensor imaging, the region around the injury does not exhibit a high contrast, as seen in the mean diffusion (Figure 4.1(a)) and fractional anisotropy images (Figure 4.1(b)). In Figure 4.1(c) and Figure 4.1(d), the detailed structure around the injury site can be seen, which provides vital information for investigators assessing the extent of injury. Figure 4.1(c) is generated using the novel planar visualization method, which incorporates shape information captured by diffusion kurtosis imaging of each data point through changes in color and intensity. Figure 4.1(d) is generated using the novel volume visualization method and conveys information about various shapes through the use of color, intensity, and opacity. 66

4.2 Visualization of a DKI spatio-angular fields using glyphs. This method demands a lot of computational power to render. Due to occlusion, the density of glyphs, a difference in orientation, changes in lighting, and surface irregularity of the glyphs, it is difficult to analyze and compare data visually. The visualization becomes even more cluttered when multiple planes must be shown. 69

4.3 Overview of the proposed method. By using MRI, a large number of diffusional readings are recorded. Domain-specific processing is then performed to compute values such as mean diffusion, fractional anisotropy, mean kurtosis, and diffusion/kurtosis tensors. Then, depending on the task and the complexity of the field, we select either a single or multiple spherical harmonics lighting functions. Finally, by combining the dynamic spherical harmonics lighting functions and the input spatio-angular field, the scene is rendered. The output is either a planar-rendered image, a volume-rendered image, or both. 72

4.4 Representative lighting functions used to analyze spatio-angular fields. The shape and the size of these spherical harmonics lighting functions can be altered to analyze different properties. The first function shown above, which is spherical, is used to compute the average value. The next four functions are used to compute single and multiple directional properties of the spatio-angular field. These lighting functions can be rotated or combined with other functions as desired. 78

4.5 The effect of using different lighting functions. We use various lighting functions on the same MRI and show different views of the structural properties of the underlying spatio-angular fields. The orientation of these lighting functions can be adjusted to investigate directional changes in the diffusion profile. In Figure 4.5(a), we use a spherical lighting function, which provides the mean strength of the field in all directions. In Figure 4.5(b), we use the directional lobe aligned with the y-axis. The figure shows the strength of the spatio-angular field in the y direction. Similarly, in Figure 4.5(c), we use the radial lobe. In Figure 4.5(d) and Figure 4.5(e), we use multiple lighting functions with different colors mapped to each axis. Color bars with a numerical range are shown on the side of the image to illustrate the color scale used. 79
4.6 The spherical harmonics lighting function allows investigators to easily study the shape of spatio-angular fields and get an appreciation of the local water diffusion environment. The spherical harmonics lighting function captures information about the shape of spatio-angular fields. Here we are using three directional spherical harmonics lighting functions and displaying the relation between them and the individual shapes that define each voxel. We use red, green, and blue colors for each spherical harmonics lighting function. The sub-images show the individual shapes of the kurtosis tensor. The variation in shape is captured by a spherical harmonics lighting function and can be observed by the change in the color. 81
4.7 The orientation of the lighting function can be global or local. Figure 4.7(a) shows the global orientation, where each data point is lighted with the lighting function (show in green glyph) in the same direction. Figure 4.7(b) shows the local orientation, where each data point is lit with the lighting function oriented towards the principal diffusion direction of the diffusion tensor. Often, there is a local coherence between neighboring data points. For example, a path might go through them. To visualize the properties along the path, a directional spherical harmonics light can be aligned with the desired direction for each data point separately. For Figure 4.7(c), the global lighting direction is used. For Figure 4.7(d), we align the light along the principal diffusion direction for each data point and display how the kurtosis values change. Kurtosis values are dominant in the direction orthogonal to the principal diffusion direction of each data point. Both Figure 4.7(c) and Figure 4.7(d) are rendered by using method described in Section 4.6.2. 83
4.8 By combining multiple directional spherical harmonics lighting functions, we can produce an overview of the spatio-angular field. In this image, we use kurtosis data to summarize the injured region. The first three images (red, green, and blue) reveal how the shape varies in the x-, y-, and z-axes, respectively. The final image is the summation. Notice how structural changes in the spherical harmonics lighting function capture different properties of the imaged tissue. 85

4.9 The directional kurtosis value of a normal subject and an injured patient using the spherical harmonics lighting tool. The coloring transforms from blue to green to red to denote the increasing value of kurtosis. In the injured patient, the region around the injury (seen in red) has a very high kurtosis value. By rotating the light, the response values of each data point are changed. When combined with a transfer function that has variations in opacity, investigators can study the magnitude and directional changes at the same time. 86
4.10 We visually compare different color maps, such as mean diffusion (Figure 4.10(a)) and fractional anisotropy (Figure 4.10(b)), with the image generated using our tool. Kurtosis values are very high around the injury region and are considered vital for assessing the extent of the injury. The image generated by our tool, Figure 4.10(c), incorporates structural information from the DKI data into the visualization, which is shown through the changes in color and intensity. Figure 4.10(d), generated using local spherical harmonics lighting, shows the entire volume. 89
4.11 We compare the MRI of a patient taken 10 days (left) and 6 months (right) after a traumatic brain injury. We use a radial spherical harmonics light to show the kurtosis value orthogonal to the principal diffusion direction of each data point. Despite a lack of anatomical changes in the patient, severe changes in kurtosis values, highlighted in red, are observed over time. 90
4.12 We compare the MRI of a patient taken 8 days (Figure 4.12(a) and Figure 4.12(c)) and 1 month (Figure 4.12(b) and Figure 4.12(d)) after a traumatic brain injury. In Figure 4.12(a) and Figure 4.12(b), we show volumetric rendering the fractional anisotropy value, and in Figure 4.12(c) and Figure 4.12(d) we show volume visualization using local spherical harmonics lighting. Changes in the kurtosis values (highlighted in red) over time are drastic compared to fractional anisotropy values. 91
4.13 As before, we visually compare different color maps (such as mean diffusion, fractional anisotropy, principal diffusion directional color map weighted by fractional anisotropy, and mean kurtosis) with the image generated using our tool. Kurtosis is very high around the tumor, as seen in Figure 4.13(d), Figure 4.13(e), and Figure 4.13(f). With the use of spherical harmonics lighting additional structural information can be seen in Figure 4.13(d). 92

Chapter 1: Introduction

In this dissertation we present several computational tools to aid in the study of diffusion at the micro-scale. When studying micro-scale objects, the diffusion process must be carefully considered. The random molecular movements that cause diffusion [3, 4] add stochasticity. These random motions happen in a timescale finer than 10^{-6} sec, which makes exploration, simulation, and visualization of microstructures challenging. To address these concerns, we present visual computing tools for two driving applications: one is the use of optical tweezers for nano-assembly, and the second is the study of the microstructure of the brain through diffusion kurtosis imaging.

In Chapter 2, we describe our work in force simulation on central processing units (CPU) and the graphics processing units (GPU) for the optical tweezers system, which uses laser beams to create optical traps that can hold and transport small particles. Optical trapping has been used in a number of applications, ranging from prototyping at the micro-scale to biological cell manipulation. The microparticles that are manipulated by the optical tweezers system are affected by the diffusion process, which introduces unpredictability when trapped. The random Brownian motion adds stochasticity and noise to the measurements. Successfully using optical tweezers requires predicting optical forces on the particle that is being trapped and transported. Reasonably accurate theory and

computational models exist for predicting optical forces on a single particle in the close vicinity of a Gaussian laser beam. However, in practice, the workspace includes multiple particles that are manipulated using individual optical traps. It has been experimentally shown that the presence of a particle can cast a shadow on a nearby particle and, hence, affect the optical forces acting on it. Computing optical forces in the presence of shadows in real-time is not feasible on CPUs. We introduce a ray-tracing-based application optimized for GPUs to calculate forces exerted by laser beams on microparticle ensembles in an optical tweezers system. When evaluating the force exerted by a laser beam on 32 interacting particles, our GPU-based approach is able to achieve a 66-fold speed-up compared to a single-core CPU implementation of the traditional Ashkin's approach and a 10-fold speed-up over the single-core CPU-based implementation of our approach. We also present an alternative way to calculate the force exerted by a laser by exploiting the coherence of mapping from the incident ray to the x, y, z components of the force and the transmitted ray through nonnegative matrix factorization (NMF). This method is useful when the ray's path within the microparticle cannot be easily computed using simple sphere-object intersections, which is possibly caused by the uneven density of the microparticle. We present an instance where the shadowing effect drastically changes the amount of force applied on a microparticle. We have validated the output of our simulation by comparing it with the observed behavior of the microparticles.

In Chapter 3 we discuss the use of kinetic depth effect (KDE) of facilitate the perception of the 3D structure of a scene using monocular displays. KDE serves a very important role when other forms of monocular depth cues, such as shading, relative size, occlusion, perspective relative, or texture gradient, are either absent or do not provide

sufficient depth information. This is often the case when visualizing simulation of the optical tweezers system. To experience KDE, we present a systematic approach to create smoothly-varying images from a pair of photographs to facilitate the enhanced awareness of the depth structure of a given scene. Since our system does not rely on sophisticated display technologies, such as stereoscopy or auto-stereoscopy, for depth awareness, it (a) is inexpensive and widely accessible, (b) does not suffer from vergence-accommodation fatigue, and (c) works entirely with monocular depth cues. The challenge in creating kinetic depth images is determining the proper parameters, such as rotation amount, rotation amplitude, the fixation point, and depth re-mapping, which are required to create smoothly-varying images from a pair of photographs. They are vital in facilitating enhanced awareness of the depth structure of a given scene while reducing motion-based artifacts and distractions. Our approach enhances the depth awareness by optimizing across a number of features, including depth perception, optical flow, saliency, centrality, and disocclusion artifacts. We report the results of user studies that examine the relationship between depth perception, relative velocity, spatial perspective effects, and the positioning of the pivot point when generating kinetic depth images. We also present a novel depth re-mapping method guided by perceptual relationships based on the results of our user study. We validate our system by presenting a user study that compares the output quality of our proposed method against other existing alternatives on a wide range of images.

In Chapter 4, we present the use of spherical harmonics based illumination to show the structure of dense spatio-angular fields. We use our tool to visualize diffusion kurtosis imaging (DKI) datasets to improve the understanding of the brain's microstructure.

DKI is gaining rapid adoption in the medical imaging community due to its ability to measure the non-Gaussian property of water diffusion in biological tissues. Compared with traditional diffusion tensor imaging (DTI), DKI can provide additional details about the underlying microstructural characteristics of neural tissues. It has shown promising results in studies on changes in gray matter and mild traumatic brain injuries where DTI is often found to be inadequate. The DKI dataset, which contains spatio-angular fields, is difficult to visualize. Glyph-based visualization techniques are commonly used; however, due to changes in orientation, lighting, and occlusion, visually analyzing the data can be very difficult. In this chapter, we present a systematic way to manage, analyze, and visualize spatio-angular fields using spherical harmonics lighting functions to facilitate insights into the structural properties of advanced magnetic resonance imaging (MRI)-based datasets.

Chapter 2: Using GPUs for Realtime Prediction of Optical Forces on Microsphere Ensembles

2.1 Introduction

An optical tweezers system is a scientific instrument that uses light to manipulate micron-sized particles. Ashkin first introduced the system in 1986 [5]. Since then scientists have been using this system to manipulate and study microparticles such as dielectric spheres, cells, DNA, bacteria, and virus. They are often used in creating assembly of micro- and nano-scaled components to make a functional device due to the extensive range of positioning and orienting capabilities of the system [6]. Additionally, optical tweezers systems can be used to manipulate cells in a controlled manner without causing them any damage [7–11].

The optical tweezers system is composed of a very powerful laser beam that has a Gaussian-based intensity distribution and a convex lens that focuses the laser beam onto the focal point as shown in Figure 2.1. This focused laser beam is used to move microparticles that are submerged in the fluid. When the microparticles are larger than the wavelength of the light used in the laser beam, the ray optics model is used to define the behavior of the optical tweezers system [5]. The laser beam is decomposed into a

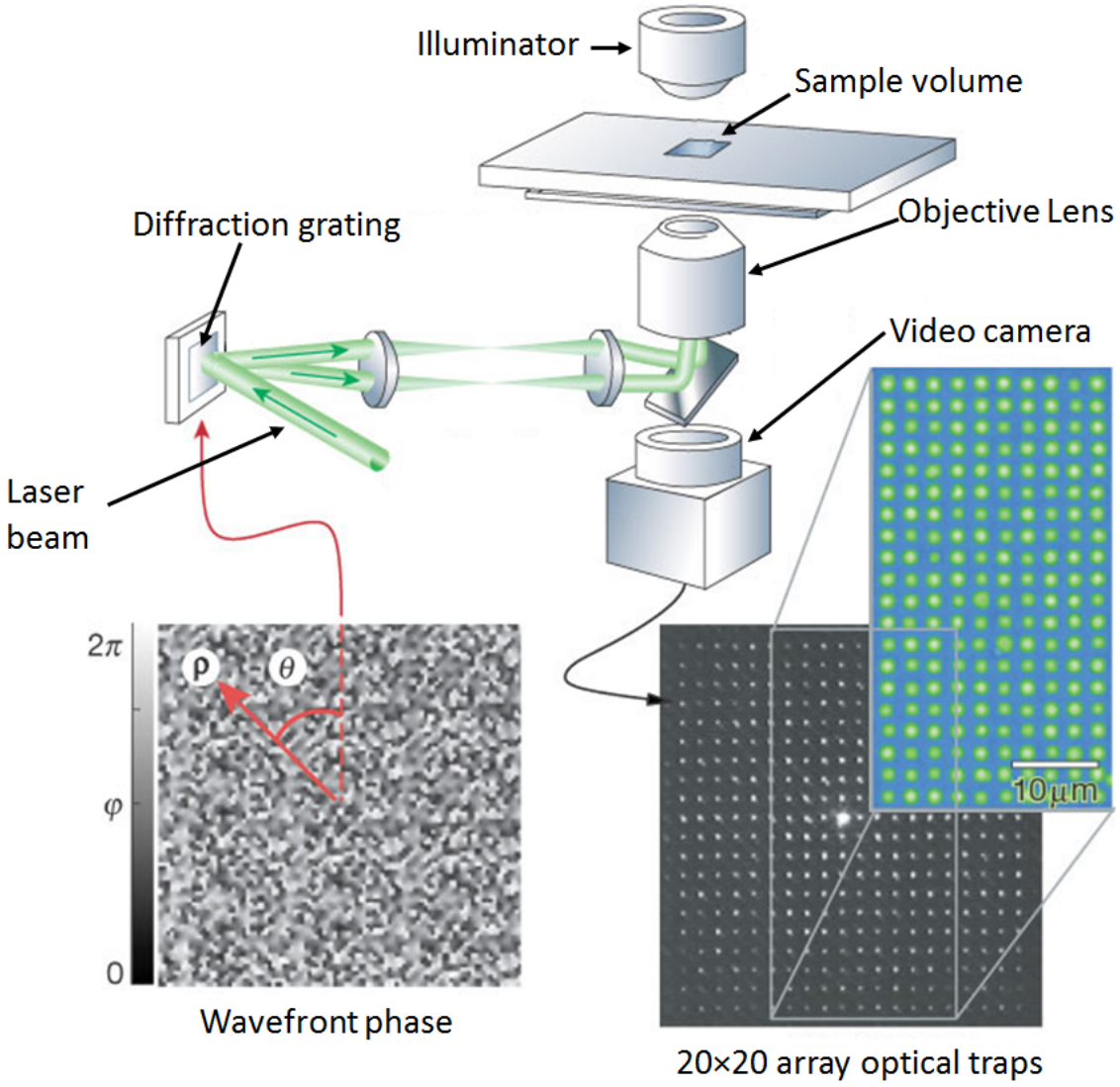


Figure 2.1: In an optical tweezer setup, a Gaussian laser beam is converged by a convex lens (objective lens of a microscope) to a focal point which is used for trapping microparticles. To create multiple optical traps, the laser beam is split into multiple-beams using a diffraction grating. Diagram courtesy of [1].

bundle of rays, each carrying a photon. When these rays interact with the microparticles, they get reflected and refracted. As each ray consists of a photon, the change in the momentum gives rise to the optical force that is exerted on the microparticles. This force is used by the optical tweezers system to trap and move the microparticles. Figure 2.2 shows an illustration of a microparticle getting trapped. Figure 2.3 shows a series of

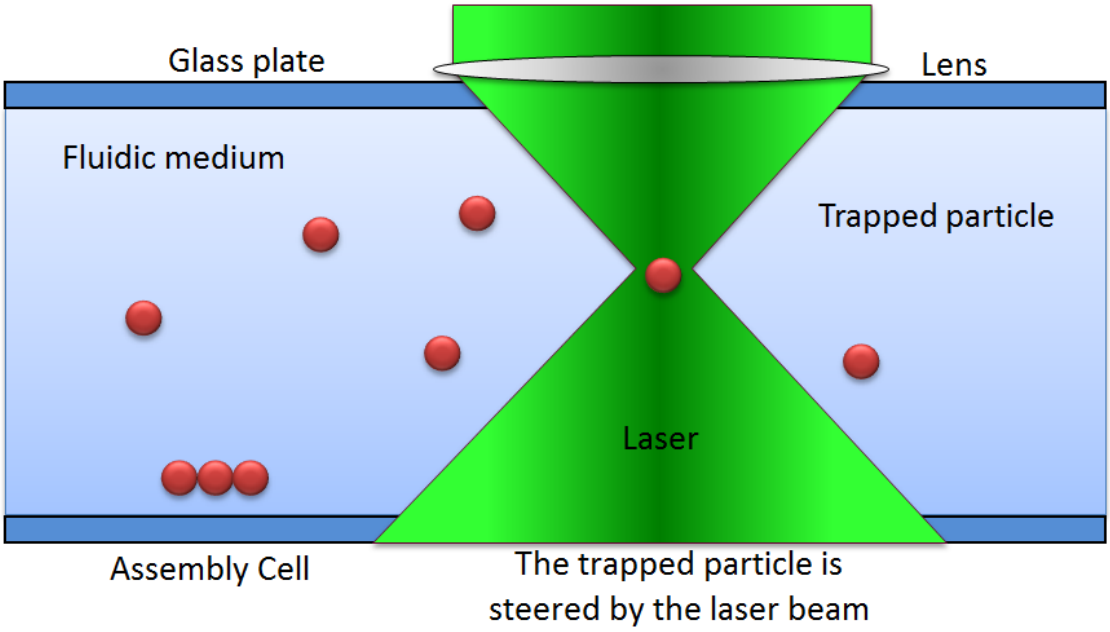


Figure 2.2: An illustration of the optical tweezers system. A laser beam with a Gaussian-based intensity distribution is converged into a focal point with the help of a convex lens. The figure shows laser beam trapping microparticles at the focal point.

images captured through the imaging device in the optical tweezers system in our lab showing a microparticle (silica bead) getting trapped. An optical trap is placed close to the microparticle which exerts a strong gradient force that pulls the particle towards the focal point.

Simulation plays an important role in understanding the optical tweezers system. To manipulate microparticles precisely, the force exerted by the laser has to be known; this is studied by performing simulations. The force calculation is a computationally intensive task because it includes simulation of the Brownian motion of the fluid-suspended microparticles requiring simulation fidelity finer than 10^{-6} sec. The popular approach to overcome this timing constraint is to use a pre-computed force look-up table to study simulation as done by Banerjee *et al.* [6]. Reasonably accurate theory and computational

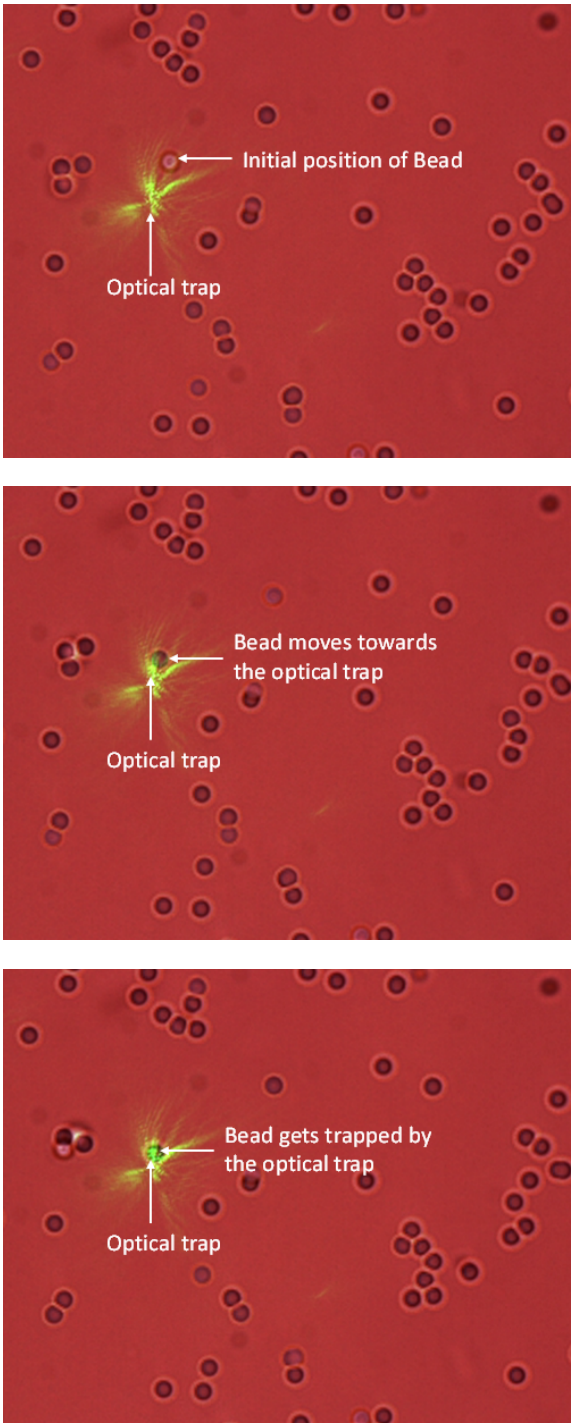


Figure 2.3: When an optical trap is placed close to a microparticle, it pulls the particle towards the focal point. The images above captured using the imaging device in the optical tweezers system show a microparticle moving into a trap.

models exist for predicting optical forces on a single particle in the close vicinity of the Gaussian laser beam. However, in practice the workspace includes multiple particles that are manipulated using individual optical traps. Experiments have shown that the presence of a particle can cast a shadow on a neighboring particle and hence affect the optical forces acting on it. When microparticles are closely placed under several laser beams, the rays get reflected and refracted which introduces secondary forces that affect the trapping. This behavior is often referred as shadowing phenomenon. It occasionally causes trapped microparticles to escape or an unwanted microparticle to jump into the trap. Studying this phenomenon is vital for scientists who are using the optical tweezers system for micro assembly or path planning [12–17].

In this chapter, we present an optimized GPU-based ray tracing application to calculate the force exerted by the laser beams on the microparticles to study the shadowing phenomenon. Our program is capable of computing the forces exerted by laser beams on multiple microparticles (up to 32) at more than 100Hz which is faster than the rate at which a typical optical device would image/monitor the microparticles. We are able to calculate the interaction between the lasers and several microparticles to study the shadowing phenomenon vital for understanding optical trapping. When evaluating the force exerted by a laser beam on 32 interacting particles, our GPU-based approach is able to get approximately a 66 times speed up compared to a single core CPU implementation of traditional Ashkin's approach and 10 times speedup over our approach's CPU implementation. In this chapter, we also talk about several choices we made while developing this application and compare them in terms of performance and precision. We also present an alternative way to calculate force exerted by the laser that exploits coherence of the

mapping from incident ray to the x, y, z components of the force and the transmitted ray by using non-negative matrix factorization (NMF). This method is useful when ray's path within the microparticle cannot be easily computed by simple sphere-object intersections (possibly caused by uneven density of the microparticle). We also present an instance where the shadowing effect drastically changes the amount of force applied on a microparticle.

2.2 Related Work

Powerful lasers are used to manipulate microparticles in an optical tweezers system. This was first introduced by Ashkin *et al.* [18] where a single-beam gradient force was used to trap micro- and nano-sized dielectric particles. Ashkin later introduced a geometric ray-optics model that is used to compute trapping forces created by a laser acting on microparticles much larger than the wavelength of light [5]. Though the equation Ashkin used is fairly optimized as it computes scattering and gradient forces based only on the incident angle and the radial position of the ray, it only works with rigid spherical objects and cannot be used directly to study interaction between several beams and microparticles. Our work is focused on calculating forces using our GPU-based ray tracing algorithm which provides both speed and flexibility needed to study shadowing phenomenon.

One of the biggest challenges in simulating the optical tweezers system is efficiently performing calculation of the laser force calculation. Since the microparticles are influenced by the Brownian motion, simulations have to be done at a time scale much smaller

than a microsecond. Banerjee *et al.* [6] introduced a framework where offline simulation is used to pre-compute data at discrete points and is later used to perform fast and accurate calculation of dynamic trapping probability estimates at any arbitrary point in 3D. This approach cannot accurately compute the effect when laser interacts with several nanoparticles. We focus on calculating the force quickly on dynamic microparticles so that the interactions of laser with several microparticles can be accurately simulated to study the shadowing phenomenon.

Bianchi and Leonardo [19] use GPUs to perform optical manipulation using holograms in real-time. They achieved speedups of $45X$ and $350X$ over CPU on their super position algorithm (SR) and Gerchberg-Saxton weighted algorithm (GSW) respectively. The speedup helped them to perform interactive micromanipulation. Balijepalli *et al.* [20] and Patro *et al.* [21] have used GPUs to compute trapping probabilities and have gotten significant speedups. Our approach in this chapter can work on CPUs and GPUs. We perform ray tracing to compute the force exerted by the laser as it interacts with several microparticles.

Sraj *et al.* [22] used dynamic ray tracing to deduce the optical force on the surface of the deformable cell from which they calculate stress distribution. Rather than using the rigid spheres as an approximated shape of the cell, they perform force calculation on the actual cell. They show that the shape of the cell strongly influences how the optical force stretches and deforms them. They also highlight that the applied optical forces change drastically when the cells are deformed. We focus our efforts on reducing the amount of time required to compute the exerted force. We perform our calculation on rigid microparticles and study how optical forces change when laser interacts with several

nanoparticles that are closely interacting with each other.

Zhou *et al.* [23] have introduced a force calculating model that uses ray tracing based on spatial analytic geometry. In our ray-tracing-based approach we perform GPU-based optimizations and calculate interaction of laser beams with multiple particles efficiently which is critical to studying the shadowing phenomenon. We also provide an alternative way of computing the forces using NMF.

Using GPUs to accelerate computationally expensive algorithms is gaining a strong interest in the scientific and gaming community. Early work done by Harris *et al.* [24] used GPUs to perform visual simulation of fluids, clouds, and smoke. They mapped some basic operators (like heat and Laplace) on the GPU and used these operators to accelerate the simulation. They performed their calculation on the GPU using programmable shaders before general languages for GPU like CUDA, DirectX Compute, and OpenCL became prominent. Considerable advancements in physically-based simulation have been made recently due to their application in games and graphics [25]. In particular, fluid simulations on GPUs have gained significant momentum [26–29]. Recently, Phillips *et al.* [30] used a cluster of GPUs to accelerate solver for 2D compressible Euler equation and MBFLO solvers. Using a cluster of 16 GPUs they achieve speedups of 496X and 88X on their Euler and MBFLO solvers. The trend of using GPUs to accelerate existing algorithms is growing. In our work, we use GPU-based accleration of ray tracing to compute the force exerted by the optical tweezers on the microparticles.

Carr *et al.* [31] make a persuasive case for the use of GPUs for computing ray-triangle intersections efficiently by using pixel shaders. Purcell *et al.* [32] mapped the complete ray tracing algorithm to the GPUs, using different pixel shaders for creating

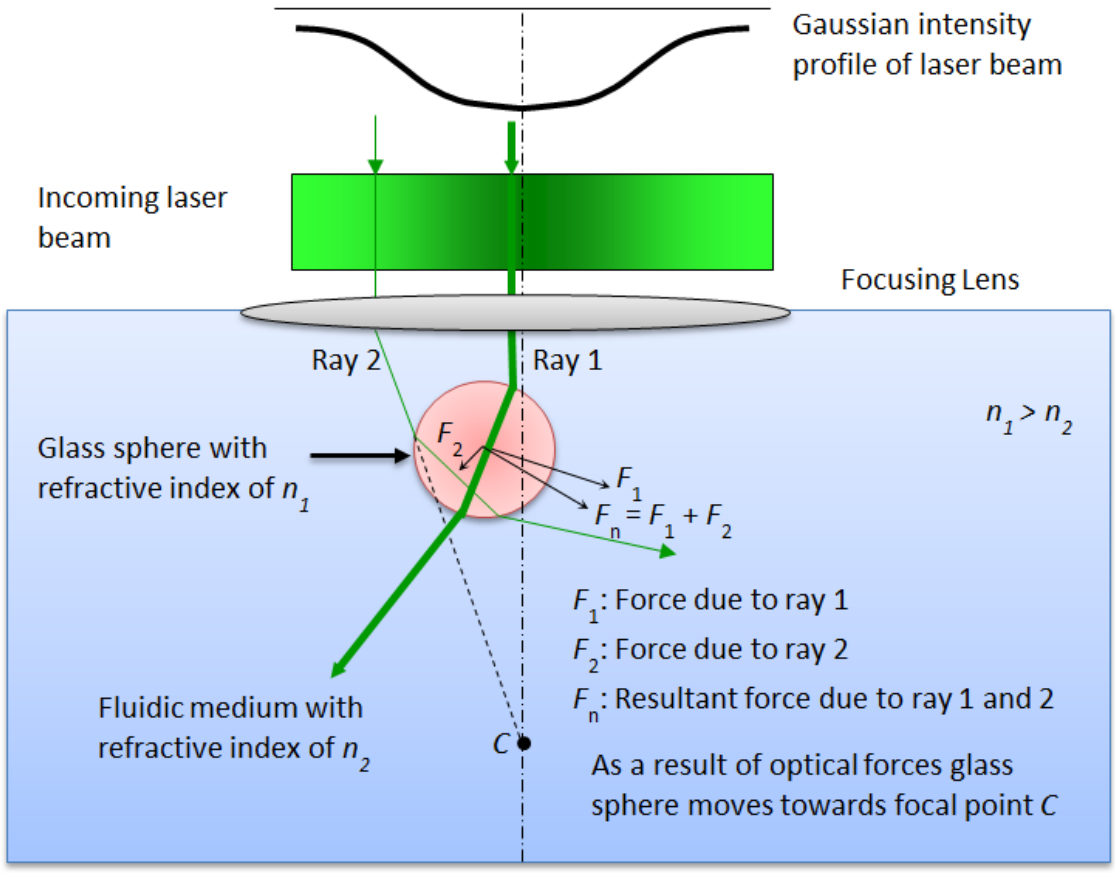


Figure 2.4: A diagram showing the simplified ray-optics model for calculating the force. The incident ray is diverted from its original path when it interacts with the microparticle. This causes the ray to change its momentum. When the ray changes momentum due to the microparticle, equal and opposite force is applied to the microparticle.

rays, traversing rays, intersecting rays with triangles, and illumination calculations. Our ray tracing program to NVIDIA's Optix but rather than calculating color per pixel we compute the force exerted by the laser beam on the microparticles and perform integration. Also, the density of rays and the paths taken by the rays used in our calculation are different from the ones used by a typical ray tracing program that uses a pinhole camera model.

2.3 Approach

A simplified ray-optics model for calculating the force is shown in Figure 2.4. Due to the change in the index of refraction between the fluid and the microparticle, the incident ray is diverted from its original path as it goes through the microparticle. This causes the ray to change its momentum. When the ray changes momentum, equal and opposite force is applied to the microparticle. We calculate force contributed by each ray for each particle. After the contribution of each ray is calculated, integration is done to find the total force. We divide the entire force calculation process into several steps described below.

2.3.1 Ray-Object Intersection

We compute ray-object intersection on the GPU using a 3D-grid-based data structure. We choose a uniform grid-based data structure over BSP, kDTree, and Octree because creating, updating, and ray traversing operations are faster due to the constant time access to the cells and the use of the efficient 3D-DDA algorithm for ray-traversal [32,33]. In our application, the grid-based data structure is created on the CPU and sent to the GPU memory every frame. In the optical tweezers system, the number of particles monitored in the experiments is often less than 64, so we create and update the data structure on the CPU. Once the grid data is transferred to the GPU, we perform ray-object intersection using a GPU Kernel. The ray-object intersection is highly parallelizable and a significant performance gain is achieved by using the massively parallel cores of a GPU as compared to a single core of a CPU. At first we considered using Optix for ray tracing. However

we soon realized that we needed a ray tracer that was more flexible to meet our memory mapping needs, easily integrable with the further steps in the force calculation pipeline, and incurred less overhead. As Optix is a general-purpose ray tracing software made for rendering, we decided to develop our own dedicated GPU-based program that is highly specialized for force calculation.

The laser beams are decomposed into R_N rays. Each ray is mapped to a thread in the CUDA kernel and all R_N threads are launched at the same time. We save the attributes (such as position, radius) of the microparticles and the 3D grid data in the global GPU memory whereas the properties of the 3D grid and the laser beam are saved in the constant GPU memory as shown in Figure 2.5. Every thread traces the path of a ray independently.

2.3.2 Force Calculation

When a ray intersects a microparticle, we compute the reflected, refracted, and the final transmitted ray by performing basic intersections. Using these rays and the properties of the microparticle, we first compute only the magnitude of the scattering and the gradient force using the equation described by Ashkin [5].

$$F_s = \frac{n_1 P}{c}\left\{1 + R\cos(2\theta) - \frac{T^2[\cos(2\theta - 2r) + R\cos(2\theta)]}{1 + R^2 + 2R\cos(2r)}\right\}$$

$$F_g = \frac{n_1 P}{c}\left\{R\sin(2\theta) - \frac{T^2[\sin(2\theta - 2r) + R\sin(2\theta)]}{1 + R^2 + 2R\cos(2r)}\right\}$$

where n_1 is the index of refraction of the incident medium, c is the speed of light, P is the incident power of the ray, R is the Fresnel reflection coefficient, T is the Fresnel

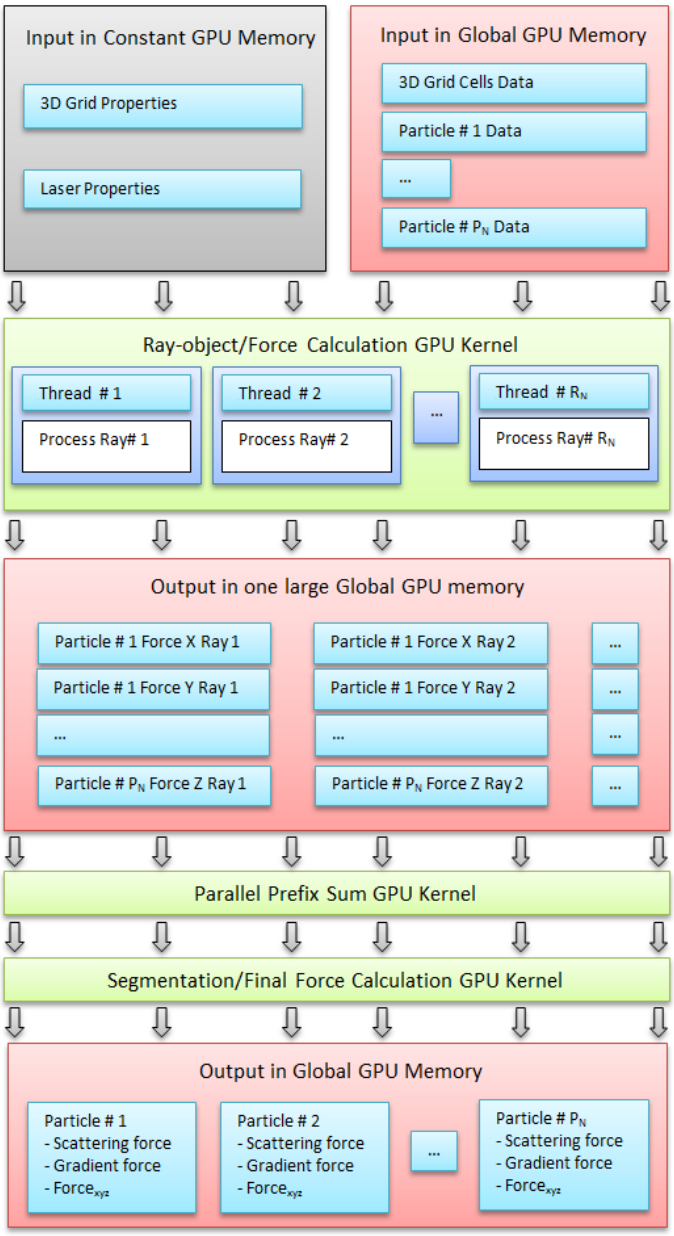
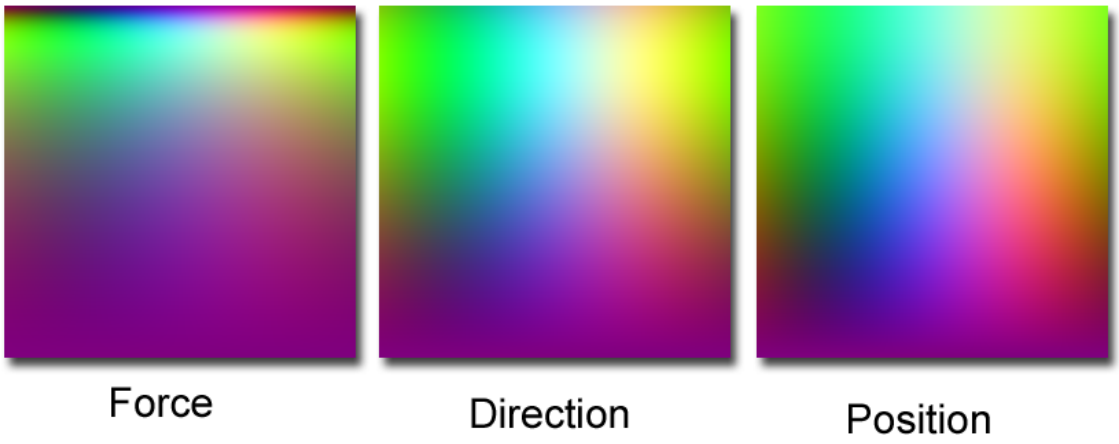


Figure 2.5: An overview of the GPU pipeline. The properties of the laser and the 3D grid are saved into the constant GPU memory whereas the properties of the particles and the 3D grid cells are saved in the global GPU memory. These are used by the first GPU kernel that performs ray-object intersection and force per ray calculation. The output is written to a large global memory array. We then perform a parallel-prefix sum at the output. As the parallel-prefix sum adds up all the components together, Segmentation/Final force calculation kernel finds the proper segment boundaries for each component and subtracts necessary amount from the boundaries to compute the final result.

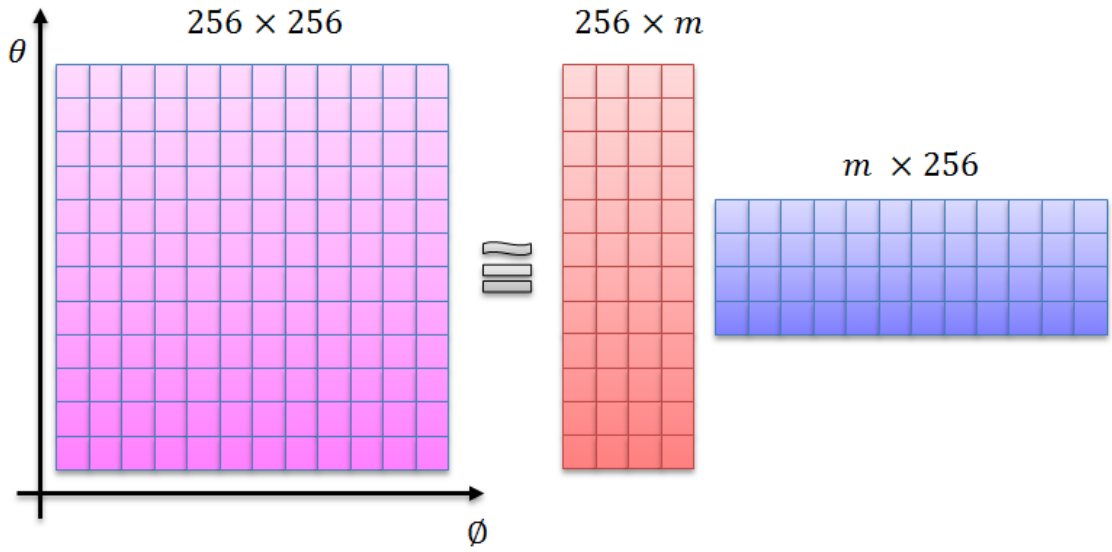
transmission coefficient, θ is the angle of incidence, and r is the angle of refraction. Then we compute the direction of the scattering and the gradient forces directly from the vectors obtained from ray tracing. The direction of the ray is the same as the scattering direction. For the gradient direction, we use the scattering direction's orthogonal component that lies on the plane formed by the center of the particle, the point where the ray intersects the particle, and the intersection of the ray with the horizontal plane as described in [5, 23]. Now using the computed magnitude and the direction, we find the scattering and gradient forces. These forces are combined to calculate the total force exerted by the ray. The total force is then saved in the GPU memory. The transmitted ray is further traced to find the intersection of the ray with other particles and the steps described above are repeated as needed.

2.3.3 Force Calculation using Non-Negative Matrix Factorization

As an alternative to calculating the force independently for each ray, we also use non-negative matrix factorization (NMF) to take advantage of the coherence between the input rays, the force exerted, and the transmitted rays. The idea is to rotate the ray-object intersection point so that it lines up with the axis-aligned pole of the microparticle. Then we represent the ray by its spherical coordinates. We discretize the angles, compute the force exerted, and the outgoing transmitted ray for each discrete set of angles. This provides us a look-up map that is used to convert the incident ray angle to the exerted force, outgoing ray position, and direction directly. This look-up map can be very large and is generated for each component of the force and the outgoing ray. We have observed



(a) Matrices used in NMF



(b) Non-negative matrix factorization

Figure 2.6: Pictorial view of the matrices that map discretized representation of incident ray angles to the force applied to the microparticle, the direction of the transmitted ray, and the position of the transmitted ray. The mapping is highly coherent which allows NMF to efficiently factorize each component of the matrix into two compact sized outer product matrices. Value of m used in our experiments is 4.

that there is a high coherence between the mapping of the exerted force and the transmitted rays as shown in Figure 2.6(a). To take advantage of this coherence, we use non-negative matrix factorization (NMF) to factorize each mapping matrix into two compact matrices. Lawrence *et al.* [34] have applied NMF to decompose the re-parameterized BRDF matrix

that uses the half angle formulation. Here, we use NMF to compact the mapping matrix as shown in Figure 2.6(b).

Once the matrices are saved, computing the force exerted by a ray is straightforward. When a ray intersects a sphere, we first compute the intersection position and the incident direction. We rotate the ray so the intersection position aligns with the pole of the microparticles. We then convert the incident ray into spherical coordinates. Using these angles, we multiply the appropriate rows and columns of the matrices generated by the NMF to get the mapping for each component of the force and the transmitted ray. Once we have the components, we apply inverse of the rotation that we applied to the incident ray to then calculate final values.

By using this alternative method to compute the force, we can take advantage of the coherence of the mapping. However, this method suffers from performance and precision issues compared to the ray tracing approach. The performance loss is caused by the matrix multiplication to rotate the vectors and the cost of multiplying a row and a column to get values for each component of the force and the transmitted ray. The performance results are shown in Table 2.1. The precision loss is created by discretization of the input angles and the compact factorization created by NMF. Due to these reasons, we use regular Ashkin's equation to calculate force using ray tracing when micro-sphere is used. However we believe that the NMF method can be useful when the microparticle has an uneven density which makes computation of the path a ray travels within the particle both difficult and computationally expensive. Such cases arise when computing the force applied to cells or non-homogeneous micro-spheres with non-uniform density distribution.

Method	Number of Rays					
	8^2	16^2	32^2	64^2	128^2	256^2
Ashkin (Float)	0.0759	0.3558	1.2708	5.0548	20.2793	81.7446
Ashkin (Double)	0.0762	0.3705	1.3399	5.3316	21.5276	86.5138
CPU Ray (Float)	0.0807	0.3389	1.4369	5.4946	22.1243	88.9347
CPU Ray (Double)	0.0852	0.3529	1.4268	5.7644	22.8563	92.5199
GPU NMF (Float)	0.9592	0.9589	0.9826	1.1923	2.0615	5.4888
GPU Ray (Float)	0.7132	0.8745	0.8337	0.9007	1.2058	2.3813

Table 2.1: The time in seconds taken by the various methods to compute total force exerted on a single microparticle performed 5000 times at different locations.

2.3.4 Force Integration

Once the CUDA kernel to compute the force for each ray is completed, we compute the net force applied on each particle by performing integration of the force field over the surface of the microparticle as done by Banerjee *et al.* [6]. We also perform this calculation on the GPU to save data transfer latency. At this stage of the pipeline, we have the force exerted by each ray in the GPU memory. Each component of the force for each particle is grouped and saved in a different part of a single large memory array. For example, all the scattering force exerted on a particle is saved in the first block followed by the gradient force and then followed by the entries of the second particle as shown in Figure 2.5. We perform a parallel-prefix sum on this large array. This will scan all the components of the force for all particles together. Since the number of particles and the individual values of the force components used in the simulation are small, we do not suffer overflow error while performing the parallel-prefix sum. We then execute another CUDA kernel that segments and outputs the final force contribution for each particle by subtracting appropriate entries from the segment boundaries of each component as shown in Figure 2.7. This final step performs extremely well on the GPU because the output of

the previous step is large and so transferring it to the CPU will incur a large latency. By calculating the final force contribution directly on the GPU, we only need to read back the several components per particle.

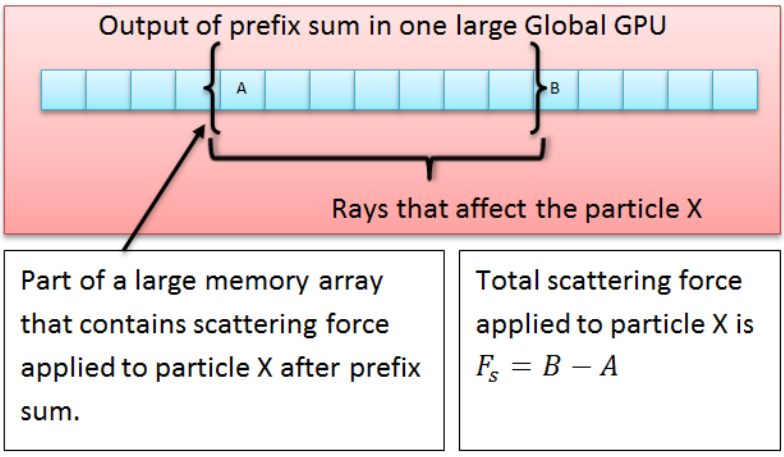


Figure 2.7: The final force contribution for each particle is calculated by subtracting values from the segment boundaries of an array that contains the result of the parallel-prefix sum. In this figure, we show how the final value of the scattering force is computed for a particle.

	Number of Rays						
Method	8^2	16^2	32^2	64^2	128^2	256^2	512^2
GPU NMF (Float)	0.0068	0.0047	0.0034	0.0028	0.0035	0.0025	0.0032
CPU Ray (Double)	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
CPU Ray (Float)	0.0005	0.0001	0.0001	0.0001	0.0002	0.0001	0.0001
GPU Ray (Float)	0.0005	0.0006	0.0005	0.0005	0.0005	0.0005	0.0005

Table 2.2: Here we show the comparison of precision between various methods rounded up to the nearest 4 digits. We take Ashkin's method as the reference and compute the relative error to compare other methods with an equal number of rays. As the number of rays increase, the relative error decreases in general but the computational cost increases.

2.4 Results and Discussion

We have implemented our system in C++. We use the CUDA API for the GPU-based ray tracing. For all of our experiments, we use Windows 7 64-bit machine with

Method	Number of Rays 8^2	16^2	32^2	64^2	128^2	256^2
Ashkin (Float)	1.88	7.77	31.51	128.13	515.13	2081.60
Ashkin (Double)	1.79	7.75	32.09	129.21	519.88	2101.70
CPU Ray (Float)	0.29	1.24	5.14	21.49	86.41	346.26
CPU Ray (Double)	0.31	1.30	5.95	23.81	95.11	379.29
CPU Ray with 3D Grid (Double)	0.38	1.34	5.78	22.85	90.72	360.80
GPU NMF (Float)	1.30	2.04	3.58	9.10	30.77	116.54
GPU Ray (Float)	1.26	1.61	1.98	3.75	9.91	33.39
GPU Ray with 3D Grid (Float)	1.88	1.86	2.26	3.69	9.45	31.50

Table 2.3: The time taken (in seconds) by the various methods to compute total force exerted by a laser beam on 32 interacting microparticles computed 5000 times at different locations. It is interesting to note that when the number of rays is low, brute-force ray tracing is faster than the ray tracing method that uses a 3D grid data structure. This is due to the additional cost of creating and maintaining the data structure.

Intel I5-750 2.66 GHz processor, NVIDIA GeForce 470 GTX GPU, and 8 GB of RAM.

2.4.1 Performance Comparison

We first show the performance gains achieved by using our GPU-based method. In our first set of experiments, we use rigid microparticles and record the amount of time it takes to compute the force. Our performance results are shown in Table 2.1 and Table 2.3. We compare the timings of various methods: Ashkin's traditional, CPU-based ray tracing, GPU-based method that uses NMF, and GPU-based ray tracing methods using single and double precision floating-point arithmetic. For the first experiment, we performed force calculations on a single microparticle 5000 times placed at different locations around the focal point of the laser beam. We also varied the number of rays that are used to describe the laser beam. As shown in Table 2.1, when only one microparticle and 256^2 rays are used to represent the laser beam, the GPU-based force calculation is about 34 times faster than Ashkin's method.

We compare precision among CPU- and GPU-based implementations of our approaches against CPU-based Ashkin's method using equal number of rays and double-precision floating-point arithmetic. We perform several comparisons by varying the number of rays to represent the laser beam. The results are shown in Table 2.2. In general, the relative error decreases as the number of rays increases. For NMF based computation, the relative error decreases at first and then fluctuates slightly as we increase the number of rays. This is because we discretized input angles while creating the mapping table. Due to this, increasing the number of rays while keeping the size of the mapping table constant, can increase the amount of error. For regular computations, 32^2 is an ideal number of rays to use to represent the laser as both the relative error and the computation cost are low.

For the second experiment, we performed force calculations using a laser beam and 32 interacting microparticles computed 5000 times placed at different locations. The ray tracing methods can capture the interaction of a laser with multiple particles while Ashkin's traditional method can only capture interaction of the laser with a particle at a time ignoring the shadowing effects. For the second experiment, we also show the performance difference from the use of a spatial data structure while doing ray tracing. As shown in Table 2.3, GPU-based force calculation that uses grid based data structure is about a 66 times faster than traditional Ashkin's method and about 10 times faster than its CPU-based ray tracing analog when 256^2 rays are used to represent the laser beam. As shown in Figure 2.8, when the numbers of rays or particles increases, the 3D grid performs better than the brute-force ray tracing method. This is generally because of the overhead of creating, updating, and transferring the grid to the GPU.

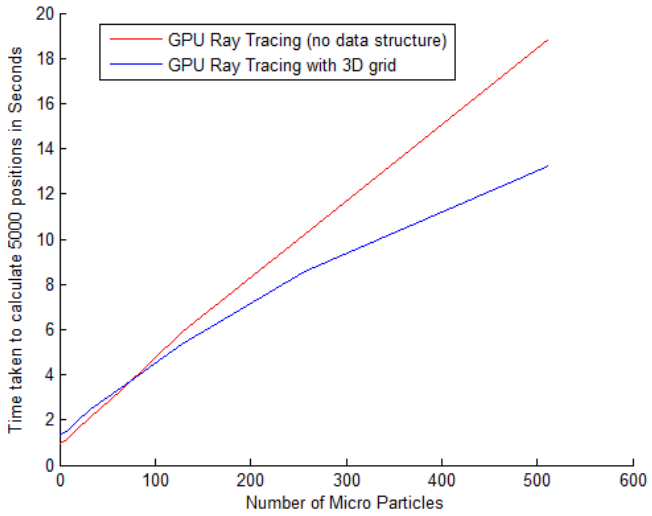
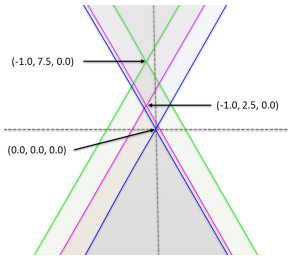


Figure 2.8: Here we show the time taken to compute the force exerted by a laser beam containing 32 rays 5000 times on a varying number of particles. We compare brute-force GPU ray tracing against GPU ray tracing with a 3D grid. As the number of particles increases, the use of a 3D grid data structure shows a clear advantage.

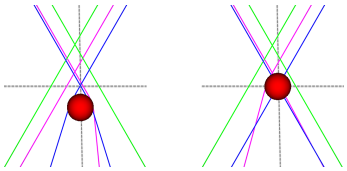
2.4.2 Shadowing Phenomenon

In the traditional ray-tracing community, the phenomenon of multiple refractions we are simulating would be referred to as the second and higher-order refractions. However, since this is referred to as the *shadowing phenomenon* by the optical tweezers community, this is the term we shall use here.

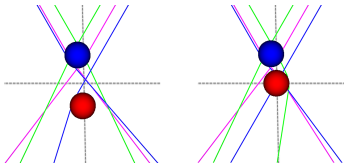
We use two microparticles for these experiments. The first microparticle moves along a path. The second microparticle is stationary and is gripped by two laser beams with one focal point above and the other below the microparticle. The rays that are incident on the second microparticle get refracted which can influence the number of rays that interact with the first microparticle. Thus the presence of the second microparticle causes a change in the amount of force being applied to the first microparticle. We show



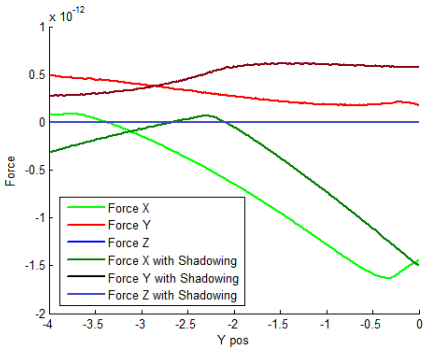
(a) Focal point of three laser beams



(b) Single microparticle moving in Y axis



(c) Multiple microparticle with one bead moving in Y axis

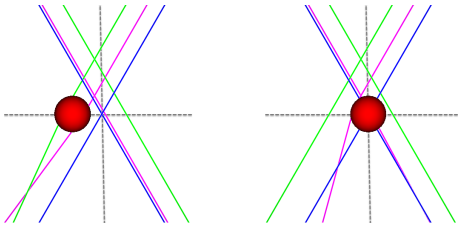


(d) Force comparision

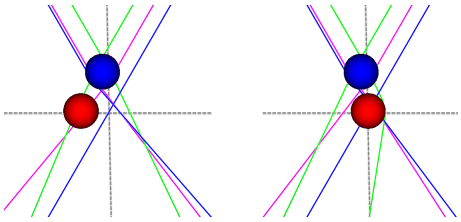
Figure 2.9: An illustration of the shadowing phenomenon. Figure (a) shows the focal point of three laser beams at location $(0.0, 0.0, 0.0)$, $(-1.0, 7.5, 0.0)$, and $(-1.0, 2.5, 0.0)$. Figure (b) shows the movement of a single particle from $(0.0, -4.0, 0.0)$ to $(0.0, 0.0, 0.0)$. The Figure (c) shows the movement of same particle when second particle is present at location $(-1.0, 5.5, 0.0)$. Finally, Figure (d) shows the difference in force experienced by the first bead caused by the shadowing phenomenon.

this change through simulations.

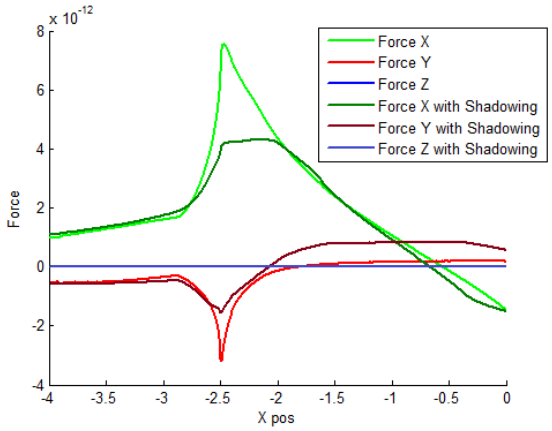
In the first experiment, we simulate a single silica bead of size 5 µm. We use three downward pointing Gaussian laser beams focused at locations $(0.0, 0.0, 0.0)$, $(-1.0, 7.5, 0.0)$, and $(-1.0, 2.5, 0.0)$, respectively. A bead is placed at $(0.0, -4.0, 0.0)$ and it slowly moves



(a) Single microparticle moving in X axis



(b) Multiple microparticle with one bead moving in X axis



(c) Force comparision

Figure 2.10: An illustration of the shadowing phenomenon similar to the previous figure. Figure (a) shows the movement of a single particle from $(-4.0, 0.0, 0.0)$ to $(0.0, 0.0, 0.0)$. The Figure (b) shows the movement of same particle when second particle is present at location $(-1.0, 5.5, 0.0)$. Finally, Figure (c) shows the difference in force experienced by the first bead caused by the shadowing phenomenon.

to $(0.0, 0.0, 0.0)$. In Figure 2.9, we show the force experienced by the bead as it goes from $(0.0, -4.0, 0.0)$ to $(0.0, 0.0, 0.0)$.

Now to show the shadowing phenomenon, we add an extra bead at location $(-1.0, 5.5, 0.0)$ in the setup described above. This bead acts like a lens and changes the direction of the rays from the lasers. This causes the first bead to experience force from secondary rays. We compute the force experienced by the bead as it goes from $(0.0, -4.0, 0.0)$ to $(0.0, 0.0, 0.0)$ when the shadowing phenomenon is occurring. In Figure 2.9, we show the difference in the amount of force experienced by the first microparticle. This change adds instability and weakens the optical traps. We next repeat the experiment but this time move the bead from $(-4.0, 0.0, 0.0)$ to $(0.0, 0.0, 0.0)$. Figure 2.10 shows the result of force calculation with and without the shadow phenomenon.

In both experiments, shadowing effects change force applied significantly. This can change the behavior of the optical traps. Experimentally validating the results of simulations is challenging. There is no direct way to measure force. The force needs to be inferred from the observed motion. This requires a high speed image capture, accounting for the Brownian motion, and accounting for image blurring due to motion in the z-direction. We are currently in the process of designing experiments to record particle trajectories in the presence and absence of shadowing phenomena.

2.4.3 Calibration and Validation

We have calibrated our system by using multiple recorded videos of the holographic optical tweezers system trapping a single freely moving microparticle. The system is

calibrated by estimating the laser power at the objective lens by trapping a freely diffusing particle and observing its velocity as it falls into the trap. We used estimated laser power for our simulation. We added a repulsive force acting upward in the positive Y direction in the simulation to model the electrostatic force from the surface that prevents the beads from falling down to the surface. The distance of each microparticle from the bottom of the surface can vary and they can all have different charges, resulting in different amounts of repulsive force. We estimated the repulsive force based on the observed behavior. Using the estimated power of the laser and the estimated repulsive force, we were able to simulate the observed behavior of the microparticles.

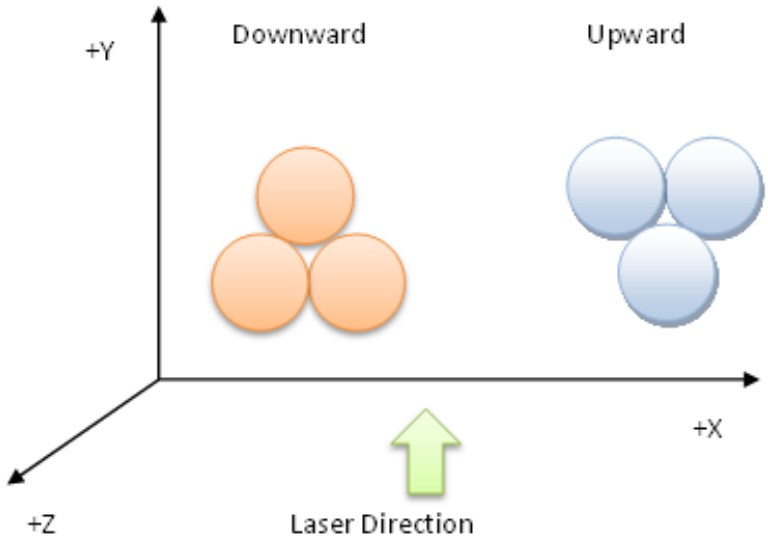


Figure 2.11: Here we show the arrangement of the microparticles in the upward and downward configuration.

Using our calibrated system we analyze the stability of gripper configurations reported in [9]. We are able to successfully predict which microparticles in a particular gripper formation are more likely to jump out of the optical traps due to the shadowing

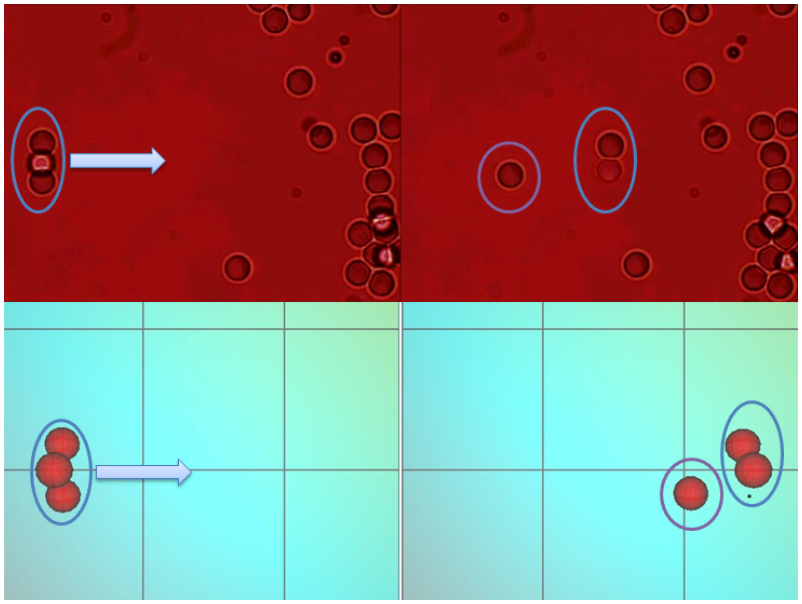


Figure 2.12: Downward configuration with spacing 2.5 µm between the lower microparticles and laser moving with the velocity 22.4 µm/s. The two beads are trapped as the laser moves. The top row shows the captured video and the bottom row shows the simulated result.

effect when the gripper moves faster than the allowable speed. We consider two different gripper configurations for our analysis. The first configuration has two microparticles trapped above the image plane while the third microparticle is trapped below. In the second configuration, two microparticles are trapped below the image plane while the third one is slightly above. Both configurations are shown in Figure 2.11. We considered three different gripper geometries for each type of gripper configuration. For the X-Z distance between microparticles, we use 0.0, 2.5, and 4.0 µm. We analyze the gripper stability for three different velocities, namely 3.8, 15.0, and 22.4 µm/s.

In the upward configuration, all three particles are moved successfully and the configuration is stable. Our simulator is able to accurately reproduce the behavior of the configuration observed in physical experiments

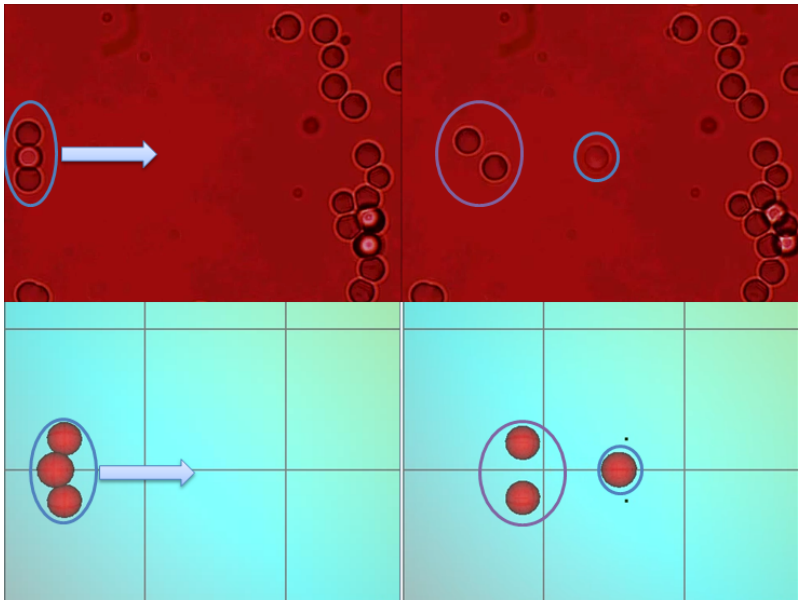


Figure 2.13: Downward configuration with spacing 4.0 µm between lower two microparticles and laser moving with the velocity 22.4 µm/s. Only one bead is trapped as the laser moves. The top row shows the captured video and the bottom row shows the simulated result.

Downward configuration where the lower beads are in contact with each other can be stably held while transporting at laser trap velocities of 3.8, 15.0, and 22.4 µm/s. On the other hand, the configuration where the beads are 2.5 µm apart is stable only at lower velocities of 3.8 and 15.0 µm/s. One of the lower beads falls behind while transporting with 22.4 µm/s and hence the configuration breaks down (see Figure 2.12). However, in case of the configurations with the lower beads separated by 4 µm, only the upper bead can be steered by the laser at 22.4 µm/s (see Figure 2.13). The configuration remains stable while transporting with the lower velocities (3.8 and 15.0 µm/s). Our simulation is able to successfully predict the breaking point for the configurations (see Figure 2.12, Figure 2.13).

We have also compared and validated force calculated using our method against the

force computed using stiffness $F = -kd$, where k is the stiffness and d is the displacement distance between the center of the microparticle and the focal point of the laser. Here, we use the stiffness value computed by Singer *et al.* [2] for a microparticle of diameter 5 μm and the laser with power 0.05 watts. In Figure 2.14, we show the force computed by both methods when the microparticle is placed at various distances from the focal point. We have found the values to be comparable. Although the force computed using stiffness is an approximation, we have compared and validated our result because the stiffness value computed by Singer *et al.* [2] is calibrated.

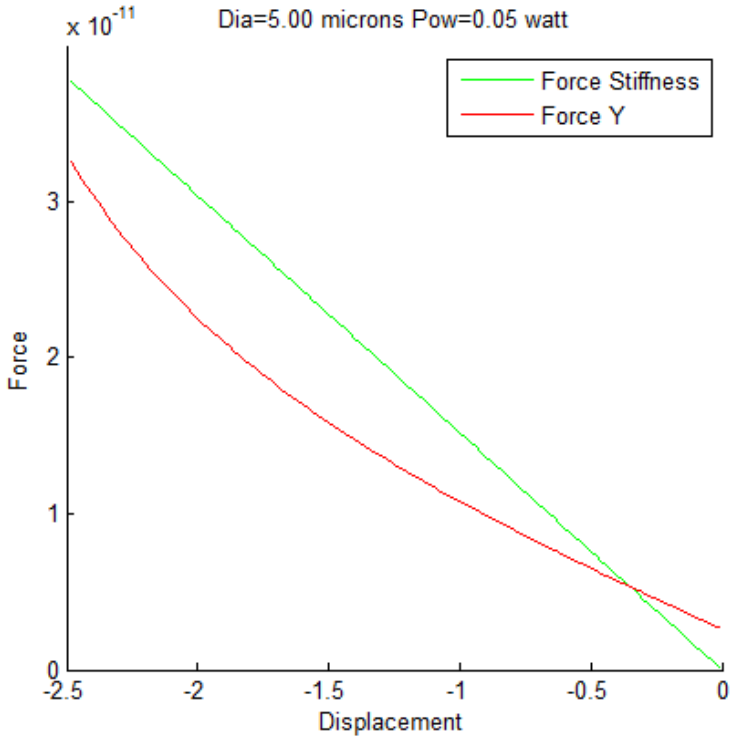


Figure 2.14: Here we show the comparison between the force calculated using our method and the force computed using stiffness. Here the focal point of the laser is located at the $(0.0, 0.0, 0.0)$ and we compute force by placing the microparticle along the Y-axis. Both forces are similar. The force computed using stiffness is an approximation but we validate our result since the stiffness value computed by Singer *et al.* [2] is calibrated.

2.5 Conclusion and Future Work

The GPU-based system we have presented in this chapter computes the forces when laser beams interact with multiple microparticles and allow a scientist to study the shadowing phenomenon. Studying these phenomenon in real-time is vital as it allows efficient planning required for trapping and manipulating microparticles. When evaluating the force exerted by a laser beam on 32 interacting particles, our GPU-based application is able to get approximately a 66-fold speed up compared to the single core CPU implementation of traditional Ashkin's approach and 10-fold speedup over our approach's serial CPU implementation. We have also presented an alternative way to calculate the force exerted by the laser that exploits the coherence of the mapping from the incident ray to the components of force and the transmitted ray by using non-negative matrix factorization (NMF).

In future, we plan to investigate our computational model by performing tests on scenarios that can be validated experimentally. Currently every time step is computed independently. Computing the force over a few time steps by efficiently processing the incremental changes might provide further speedup.

Chapter 3: Kinetic Depth Images: Flexible Generation of Depth Perception

3.1 Introduction

The kinetic-depth effect (KDE) is the perception of the three-dimensional structure of a scene resulting from a rotating motion. First defined by Wallach and O'Connell [35], the kinetic-depth effect has been used widely. Images that exhibit KDE are commonly found online. These images are variously called Wiggle images, Piku-Piku, Flip images, animated stereo, and GIF 3D. We prefer to use the term *Kinetic Depth Images* (KDI), as they use the KDE to give the perception of depth. Last year, Gizmodo organized an online competition to reward the KDI submission that best provided a sense of depth [36]. Recently, the New York Public Library published a collection of animated 3D images online [37]. Flickr has a large number of groups that discuss and post animated 3D stereo images. Also, online community-based galleries that focus on KDI can be found at the Start3D website [38]. With the increasing availability of stereo and lightfield cameras (such as the Lytro), the use of the KDE to express depth on ordinary displays is rising rapidly.

Although the basic form of the KDE is trivial to implement for a virtual environ-

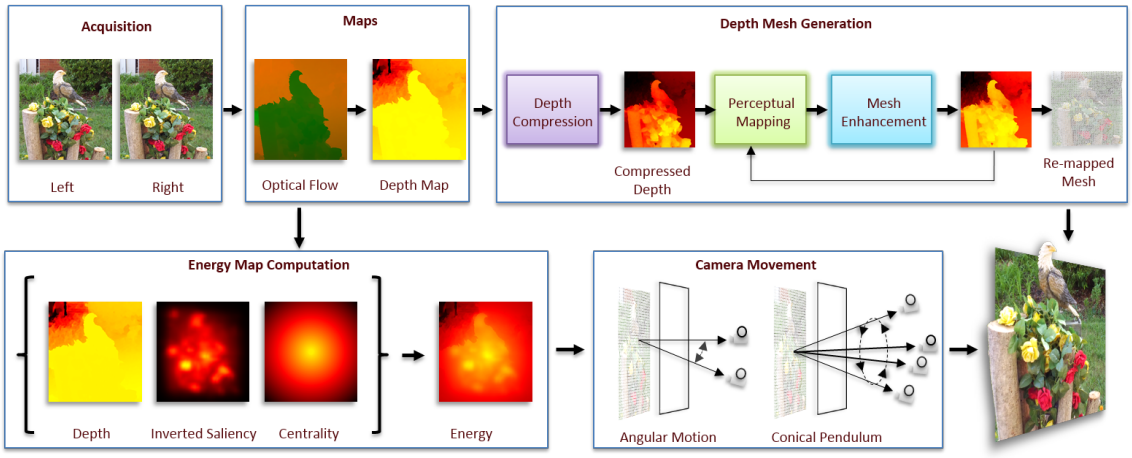


Figure 3.1: We first compute the optical flow and the depth map from the input images (generally a stereo pair). We then generate a triangulated depth mesh using the re-mapped depth map which is guided by human perception. We also create an energy map of the image using depth, centrality, and image saliency. By using the depth mesh and the energy map, we generate a high-quality animation to best experience the kinetic-depth effect. Our system has been informed by and validated through user studies.

ment where the 3D geometry and lighting is known, excessive motion and reduced depth perception mar the visual experience unless proper care is taken. In fact, accomplishing high-quality KDE from a pair of photographs is not trivial and has several interesting nuances, as outlined in this chapter, which should be useful for any practitioner wishing to use this technique for facilitating depth awareness.

In this chapter we describe an algorithm and its accompanying system, for facilitating depth awareness through KDE using only a pair of photographs or images.

The use of the KDE has several advantages over the use of stereoscopic and autostereoscopic displays: (i) the KDE provides monocular depth cues that allow us to experience depth, even with one eye, which accommodates people who suffer from various monocular disorders, (ii) the depth is perceived due to the rotation of the object and does not require any special device (glasses or lenses) to view; this works with any display

and can be easily shared online, (iii) unlike stereoscopic and auto-stereoscopic displays, KDE animations do not suffer from vergence-accommodation conflicts, and (iv) depth perception achieved by the KDE can exceed that of the binocular depth perception as the disparity provided by binocular vision is limited compared to the angular rotation that can be produced by KDE [39, 40].

Although there are a large number of KDE based images online, most of them are manually created by the artist, require tedious user input to create them, or suffer from visual artifacts introduced by the automated systems used to create them. Manual creation of these images may seem simple, but it is difficult for average users to make their own high quality KDE based images. A lot of KDI suffer from artifacts caused by abrupt changes in motion, color, or intensity, as well as alignment errors and excessive motion.

Our contributions in this chapter are:

1. Given a stereo image or an image/depth pair as an input, we automatically optimize a KDE based image sequence by taking human depth perception, optical flow, saliency, radial components, and disocclusion artifacts into consideration;
2. We report the results of a user study that examines the relationship between depth perception, relative velocity, spatial perspective effects, and the positioning of the pivot point when generating KDI;
3. We present a novel depth re-mapping method guided by image saliency and the perceptual relationship found in our user study to reduce excessive motion in KDI.

3.2 Background

The kinetic depth effect (KDE) is defined as the perception of the three-dimensional structural form of an object when viewing it in rotational motion [35, 39]. As an observer moves, the nearby objects are seen from different angles. To experience KDE, the optical-flow-driven motion cue has been found to be very important [41–43] and can be experienced from just two photographs taken from different views [44, 45]. Another effect that gives the perception of depth is the stereo kinetic effect (SKE). Although SKE and KDE both give the perception of depth, they are quite different. SKE gives a perception of depth when viewing a 2D pattern as it is rotated in the view plane (fronto-parallel) [46], whereas in KDE the depth perception arises from rotating the object along an axis. Motion parallax is another monocular depth cue that is experienced through the relative motion of the near and far objects [39]. Generally, KDE is associated with the rotational viewing of objects whereas motion parallax is associated with the translational viewing of objects. When an observer fixates on an object and makes small rotational or translational motions, both KDE and motion parallax are similar [39].

3.3 Related Work

Although a few tools are available online to generate the KDE from image pairs, we found them to be largely ad-hoc techniques with variable quality. We have not come across any previous work in the technical literature that systematically identifies the various competing constraints in building such tools to achieve an aesthetically superior view-

ing result. We next give an overview of the various online tools we have come across on the Internet.

The New York Public Library has an online tool, *Stereogranimator*, that converts stereograms into animated 3D images employing user input to align images [37]. The tool allows users to manually rotate and translate stereo images to align them and change animation speed to create animated GIF images that switch between images. With carefully acquired stereo images and with proper user input, the output of this tool can produce good results. Otherwise the output may contain artifacts created by the abrupt changes in motion, color, or intensity, or through alignment errors.

Wiggle images follow the same principle and are created by taking a pair of stereo images (or a sequence of images) and flipping between them. With careful consideration during the acquisition stage and a very precise manual alignment during the animation generation stage, a reasonably good quality effect can be achieved. However, the process is tedious and requires significant skill in photography and image processing. Rather than relying on the number and quality of input images, we use a judicious mix of computer vision and rendering algorithms to create high-quality animations needed to experience the KDE from a pair of images.

Piku-Piku images from Start3D (`www.start3d.com`) provides an online tool for generating KDI. A user uploads an image pair to the Start3D server, which then generates a sequence of intermediate images and provides a hyperlink to view the resulting images on its server [38]. To the best of our knowledge, the underlying algorithm for generation of Piku-Piku images, their processing, and their viewing has not been published. A careful study of the animation created by Piku-Piku images shows that the intermediate frames

between an input pair of stereo images are computed as a translation from one input image to other. This method does not produce a good result if the input set of stereo images does not happen to fall on the path suited for the KDE. Blending artifacts are often seen and if the stereo images are not carefully acquired (for example, if shear or slant is present in the stereo pair), then the output is often of a poor quality.

Another similar tool is the *Stereo Tracer* created by Triaxes (`www.triaxes.com`). This uses an image-depth pair or a stereo-image pair to generate intermediate views to create animation for producing the KDE. To secure a good output tedious manual adjustment is required. These adjustments include (a) depth map adjustment, (b) image alignment, and (c) manipulation of parameters that control parallax and the plane of zero parallax.

Recently, Lytro Camera announced a perspective shift feature on their camera application [47]. Based on user input, they allow users to change the camera perspective computed using the captured light-field data. To our knowledge, the underlying algorithm has not been published.

Zheng *et al.* [48] presented a method that focuses on automatically creating a cinematic effect that exhibits the parallax effect. They reconstruct the scene from a series of images, fill the occluded regions, and render the scene using a camera path that attempts to maximize the parallax effect while taking into account occluded regions. Rather than trying to recreate cinematic effect, we try to maximize depth perception based on KDE while reducing motion induced artifacts. KDE uses rotational motion that is different from the cinematic effects presented by Zheng *et al.* [48].

None of the above approaches seem to take into account the depth perception, im-

age saliency, identification of the best rotation axis, identification of good pivot points for fixation, or depth re-mapping to generate high-quality KDE that we have used in our approach. Another significant departure in our approach has been the decoupling of the rendering camera from the acquisition camera. This has allowed us significantly greater freedom in using the standard angular camera motion with substantially fewer visual artifacts as well as using previously unexplored camera motions to achieve the KDE.

3.4 Overview of our Approach

The input to our system is a pair of images that can generate an approximate depth map. It also works with an image and a depth-map pair of the scene available from a camera coupled with a depth sensor such as Microsoft Kinect.

We will refer to the cameras used in taking the initial images as *input cameras* and the cameras used for generating the animation (to simulate the KDE) as *rendering cameras*. Our system is able to generate this animation through a series of steps as shown in Figure 3.1. First, from the input stereo image pair we compute the optical flow and a depth map. After that we compute parameters needed to generate KDE by taking into account depth, centrality, saliency, and optical flow. Depth perceived by kinetic motion depends on the relative velocity. Using the result of the user study (which allows us to map velocities to perceived depth) and image saliency, we re-map the depth and create a depth mesh. This remapping reduces total motion while enhancing the depth perception. Finally, we visualize the depth mesh using any desirable rendering camera motions (such as angular or conical-pendulum) to generate the animation needed to experience the KDE.

Details of our approach are described in the following sections.

3.5 Kinetic-Depth Effect Parameters

We generate KDI by calculating proper values for pivot point, rotation axis (shown in Figure 3.2), magnitude of angular rotation, and frequency. The pivot point is the look-at point of the rendering cameras as they move about the rotation axis. In this chapter, we use two different types of camera motion and each type uses the rotation axis in a slightly different way. We discuss this further in Section 3.9. The pivot point is also a position in the image through which the rotation axis passes. This results in minimal optical flow near the region around the pivot point when the final animation is created.

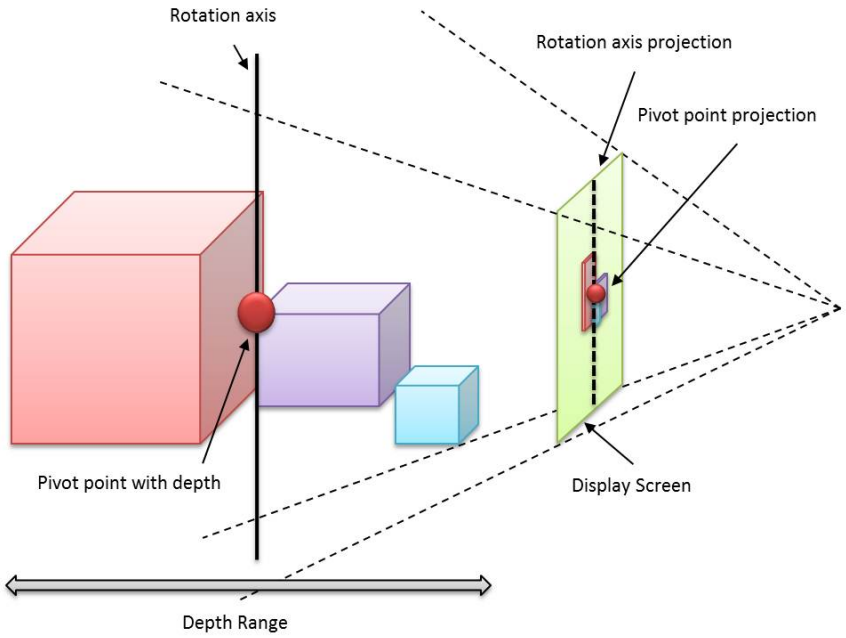


Figure 3.2: The pivot point and the rotation axis of the scene are shown in both 3D space and the projection space.

To create an effective kinetic-depth experience, we need to determine the magni-

tude of angular rotation and frequency. Wallach and O'Connell [35] used an angle of 42 degrees at the rate of 1.5 cycles per second. Epstein [49] used angles of 15 degrees and higher in his experiment and reported that 15 degrees is sufficient to perceive the KDE. His experiment used the projection of a shadow with no visible shading and texture. Since we are using typical natural stereo images with full texture, we have found that rotation as small as 0.5 degrees about the pivot point is sufficient. This is consistent with human vision, where an average intra-ocular distance is 6.25 cm, which gives about 4 degrees separation at a fixation point 1 meter away. For objects farther away seperation is much less. In our results, we perform rotations between 0.5 to 2 degrees around the pivot point. The frequency of rotation used in our system is 2 cycles per second since this has been reported as the optimum temporal frequency by Nakayama and Tyler [50], Caelli [51], and Rogers and Grams [52] .

Although we keep the rotation at the pivot point small, objects that are close or far might exhibit much higher movement, depending on the scene. According to Ujike *et al.* [53], 30 to 60 degrees/sec on each axis produces the highest motion sickness; roll produces more sickness than pitch and yaw. Taking this into account, we keep the maximum rotation over the entire scene low and keep the change in the vertical axis to a minimum.

The positioning of the pivot point, frequency of rotation, magnitude of angular rotation, and the scene depth range all directly affect the velocity of the stimulus moving on the screen. Gibson *et al.* [42] and Rogers and Grams [43] have shown that depth perceived by kinetic motion depends on the relative motion. Although some increase in relative motion enhances the perception of depth, excess motion causes motion sickness and reduces the ability to smoothly track objects. Robinson *et al.* [54] showed that

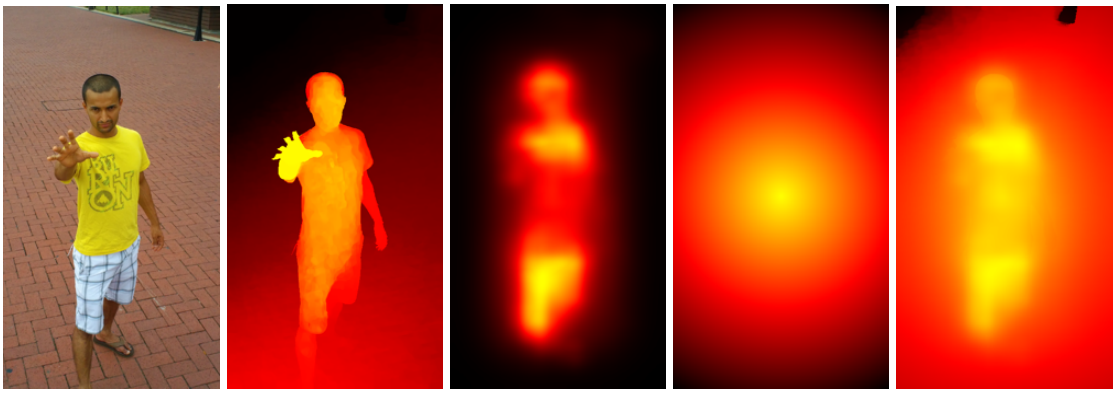
while tracking objects with velocity of 5 degrees per second, the human eye took about 0.22 seconds to reach the maximum velocity and about 0.48 seconds to reach the target steady-state velocity. In our experiment, the frequency of rotation is 2 or more cycles, and therefore to maximize KDE and at the same time reduce unwanted motion based artifacts, we keep motion below 5 degrees per second. Using depth re-mapping based on results obtained in Section 3.8, we minimize motion artifacts while maximizing the perception of depth.

3.6 Energy Map Computation

As the pivot point experiences minimal optical flow, it is preferable to locate the pivot point at a salient region of the scene. If the pivot point salient region has text or a face, it becomes easier to read or identify it when the region is not exhibiting high optical flow due to the rendering camera movement. It is also preferable to have the pivot point in the middle of the scene to minimize the optical flow at the scene boundaries. If the pivot point is chosen at a scene boundary (as an extreme example), the opposite end of the scene will exhibit excessive motion that can create visual discomfort and can also give rise to significant disocclusion artifacts.

To quantify and incorporate such considerations, we determine the placement of the pivot point by computing an energy map. We seek a global minimum on the energymap to determine the kinetic-depth parameters:

$$E(x,y) = E_d(x,y) + E_s(x,y) + E_r(x,y),$$



(a) (b) (c) (d) (e)

Figure 3.3: Energy components. Here we show (a) the original image, (b) the depth map, (c) the saliency component, (d) the radial component, and (e) the final computed energy map. In these maps, yellow regions are associated with lower energy. We choose the pixel with the lowest energy to be the pivot point.

where (x,y) refers to the position of the pivot point in the reference image (projection space) and $E_r(x,y)$, $E_d(x,y)$ and $E_s(x,y)$ are the radial, depth, and saliency energy functions, respectively. Each energy function component is further discussed below.

Depth Energy: We use the depth energy to express a preference for positioning the pivot point on regions that are close to the middle of the depth map. This minimizes the visibility of the occluded regions during the rendering camera movements and also lowers the amount of motion when the final animation is generated. We obtain the depth map of a given image set by calculating its optical flow using Sun *et al.*'s [55] algorithm. Details about depth map calculation are described in Section 3.7. After calculating the depth map we can calculate the depth energy as: $E_d(x,y) = w_d \left\| P_d(x,y) - D_m \right\|_2$, where w_d is the weight given to the depth component, $P_d(x,y)$ refers to the depth value of the pixel at (x,y) and D_m is the median depth of the scene. The second image in Figure 3.3 illustrates the depth map used in $P_d(x,y)$.

Saliency Energy: As mentioned earlier, we desire to position the pivot point on a salient region of the scene. These salient regions are different in color, orientation, and intensity from their neighbors and can be found using the image saliency algorithm proposed by Itti *et al.* [56]. The saliency map represents high saliency regions with high values. When calculating the saliency energy of the pivot point, we invert the saliency values so that the most salient regions are represented with the lowest values. Our equation for calculating the saliency energy is: $E_s(x,y) = w_s[1 - P_s(x,y)]$, where w_s is the weight given to the saliency component and P_s refers to the saliency value of the pixel. The third image in Figure 3.1 illustrates the saliency map used in $P_s(x,y)$.

Radial Energy: To express a preference for a centralized pivot point, we use a radial energy function as one of the energy components. Essentially, the closer the point is to the center, the less radial energy is associated with it. The radial energy is calculated as: $E_r(x,y) = w_r P_r(x,y)$, where w_r is the weight given to the radial component and P_r refers to the radial value of the pixel defined by the Euclidean distance between the point and the image center. The fourth image in Figure 3.3 shows an example of the radial map.

The radial component $P_r(x,y)$ depends upon the dimensions of the image, but the saliency component $P_s(x,y)$ and the depth component $P_d(x,y)$ depend upon the scene. To take these factors into account, we use weights while calculating the energy function. Assigning a higher w_s term will increase the importance of the salient regions, whereas higher w_d and w_r will give greater priority to the image center and lower the total optical flow between frames. Figure 3.3 shows an example of energy map calculation. We compute the energy of all the pixels and find the position (x,y) that has the lowest energy. Then the pivot point in $2.5D$ is defined by the coordinate $(x,y,P_d(x,y))$.

3.7 Depth Mesh Generation

We generate the depth mesh by first approximating the scene depth, followed by an optimized compression. Then, we perform re-mapping of the depth mesh by taking perception into account. Finally, we enhance the depth mesh to so that the motion is constrained to a desired range and disocclusion artifacts are minimized.

3.7.1 Scene Depth Approximation

Generally the number of input images or photographs is smaller than the desired number of views. Also, the parameters used by the rendering cameras are often different from the input cameras. Due to these reasons, a number of additional intermediate frames need to be generated. There are numerous ways of generating intermediate frames, such as basic interpolation between input images, flow field interpolation [57], image based rendering [58, 58, 59], and structure approximation to create depth mesh [60–63].

In our approach, we use a depth-image-based rendering to generate high-quality intermediate frames needed for achieving the KDE. This is computationally less demanding than a full-fledged Structure from Motion and allows sufficient flexibility in the choice of the rendering camera parameters. For every pair of images, we first calculate the depth map. Although there are numerous methods to compute the depth map, we decided to approximate depth based on the optical flow between the input image pair by using the inverse relation defined as: $d = f\frac{t}{o}$, where d is the depth, f is the focal length, t is the distance between the camera positions, and o is the optical flow of the pixel . Since we do not know the camera parameters f and t, we recover depth up to a projective transformation.

3.7.2 Optimized depth compression

Depth range on a raw depth map is usually very high and contains a lot of artifacts. Directly using a raw depth map will cause excessive motion that is visually disconcerting. One naïve solution to this problem is to perform simple scaling of the depth map to fit a certain range. However, this will compress both important and unimportant regions of the scene uniformly. We want the salient regions of the scene to occupy a larger share of the depth range while compressing non-salient regions and artifacts. An analogous problem arises in stereo viewing and relief mapping, where disparity or depth compression is needed [64, 65]. In KDI, compression requires global consistency of the scene depth as the depth mesh is viewed from different angles; otherwise the inconsistencies will be easily visible. The disparity histogram and saliency map have been used by Lang *et al.* [64] to compress disparity for stereo viewing. In the slightly different problem of video retargeting, manual constraints have been used to enforce feature preservation while compressing the frames [66].

We perform depth compression based on the image saliency and use cues from carefully chosen edges to enforce feature preservation. To compress depth, we first divide the depth range into k intervals $\{r_0, r_1, ...r_{k-1}\}$. Then we compute the histogram of the saliency weighted depth values $\{h_0, h_1, ...h_{k-1}\}$ similar to [64]. Using the saliency weighted histogram values, we determine the size of each interval

$$s_x = \min\left(\frac{h_x}{g * max(h_0, h_1, ...h_k)}, 1.0\right),$$

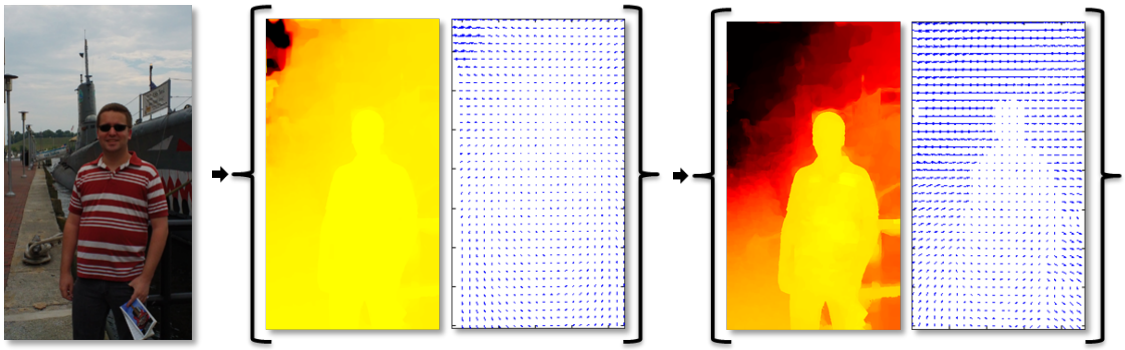
where s_x is the size of the interval x after using saliency compression, and g is a constant (in this chapter we use $g = 0.4$) which gives extra rigidity to salient depth intervals. This mapping compresses the intervals that are less salient.

Only using saliency to compress the depth map will cause features, such as lines and curves, to change. Some constraints have to be placed in order to preserve features. We use information of carefully chosen edges to enforce loose feature preservation. We first calculate edges in the image using Canny edge detection [67]. If needed, more sophisticated edge detectors can be easily integrated. We filter edges that are very short in length to remove noise. After that, we perform additional filtering to remove edges that lie at the border region of objects at various depths. This is done by removing edges with very high gradient on their depth values. This step is important because the depth map contains errors/artifacts and the depth approximation close to the depth boundaries is less reliable. Edges that are left after filtering will be longer and will have more reliable depth values. These edges are used for feature preservation. We find the depth interval associated with each of these filtered edges and uniformly distribute depth among them as $s_x^* = \frac{\sum_{n=i}^{j} s_n}{j-i}$, where s_x^* is the new size of the depth interval x after using saliency compression with feature preservation; i and j are the depth interval associated with a filtered edge and $x \in [i, j]$.

3.7.3 Perceptual re-mapping

Perceptual Maps: For a given camera motion, the relative velocity is the difference in velocities when the objects are projected onto the display screen, and studies

have shown that relative velocity is an important motion cue to experience KDE [41–43]. Using the results from the user study as described in Section 3.8.1, we compute a set of curves that are used to map depth perception to relative velocity and vice versa. We generated two sets of curves to approximate the results from the user study as shown in Figure 3.6. The first set of curves maps the users' normalized depth perception to the relative velocities(NDP2RV map) used in the study and the second set maps the relative velocities to normalized depth perception (RV2NDP map). The curves within each set show how the relation between relative velocity and depth perception changes as the midpoint velocity is changed. This mapping gives an empirical relationship between the normalized depth perception and the relative velocity.



(a) (b) (c)

Figure 3.4: (a) the original image, the image set (b) is associated with the raw depth, and the image set (c) shows the depth used to generate final depth mesh after depth compression and perceptual enhancements. The depths in (b) and (c) are normalized. In each set we are showing the depth values and the optical flow after the camera motion to depict the value of depth remapping.

Mesh Re-mapping: We generate the re-mapped depth mesh by taking depth perception into account. Since we would like the viewers to perceive depth that is close to the 3D structure we present, we use the relative depth (separation) computed from the

compressed depth mesh M_{Comp} as an input for the NDP2RV map to generate a relative velocity map needed to perceive the desired depth. This step is done for each vertex v in M_{Comp}. For each v, let V_n be the set of its four connected neighbors such that $V_n \subseteq M_{Comp}$. We created four vertex-neighbor pairs (v,w) where w is in the set V_n. Then for each pair (v,w), we compute the relative depth $R_d(v,w)$ between them. The relative depth is scaled so that it is within the maximum depth range allowed in the scene. Initially, maximum depth range is initialized to the depth range that results in a relative velocity that is equal to the maximum desirable relative velocity which is explained in Section 3.8.1. The scaled depth is then used as an input for the NDP2RV map to get the relative velocity $V_p(v,w)$ between the pair. We call $V_p(v,w)$ a perceptual relative velocity because it is based on the perceptual relationships explained earlier. The $V_p(v,w)$ between a vertex pair is necessary to perceive the relative depth $R_d(v,w)$ between them. In other words, to perceive a depth that is equal to $R_d(v,w)$ between a vertex pair, we need the relative velocity between them to be equal to $V_p(v,w)$.

Using the perceptual relative velocities between vertices, we can compute their separation (we call this perceptual separation $S_p(v,w)$). We move one of the vertices in (v,w) along the line of the view point and find at which separation $S_p(v,w)$ the desired $V_p(v,w)$ is achieved. The relative velocity between a pair of vertices is computed by projecting each vertex on the display screen using a standard projection matrix, then calculating the instantaneous velocity for each vertex, and finally finding the difference between the velocities. This process is accelerated by a binary search algorithm. Since this step is performed by only using local data on the distances between neighbors (v,w), an extra step to make a globally consistent mesh is required. This is done by minimizing a $1D$

problem. We first discretize depth of the scene into 255 bins. Then for each (v,w), we find associated discretized depths d_v and d_w based on their depth in M_{Comp}. We then add a link between d_v and d_w that specifies the $S_p(v,w)$. Since the $S_p(v,w)$ is locally consistent, some links will try to contract the distance between the discretized depths while some will try to expand them. This process can be pictured as a spring mesh where the links added will act like a spring and the discretized depths are the places where the springs are attached. We define $d(v,w) = \mid d_v - d_w \mid$. For all (v,w), we find $d(v,w)$ that minimizes $\sum \left\| d(v,w) - S_p(v,w) \right\|_2$ to perceptually re-map the depth mesh M_P to make it more globally consistent.

3.7.4 Depth Mesh Enhancement

Depth values after compression and perceptual re-mapping have to be readjusted for a pleasant viewing experience. Depending on the rendering parameters and the depth range of the scene, depth mesh can be compacted or expanded to allow enhanced depth perception. Also, depth has to be adjusted to reduce disocclusion artifacts. In Section 3.8.1 we mention that when using KDE, we have a limited perception of depth, which is highly dependent on the relative velocity. The relative velocity of objects depends on the relative distance between them, the placement of the pivot point, and the camera movement. However, there is a tradeoff between the amount of motion in the scene due to relative velocity and perception of depth. The results of Section 3.8.1, give us the maximum desirable relative velocity, taking into account the tradeoff. Using the maximum desirable relative velocity, perceptual depth mesh, and placement of the pivot,

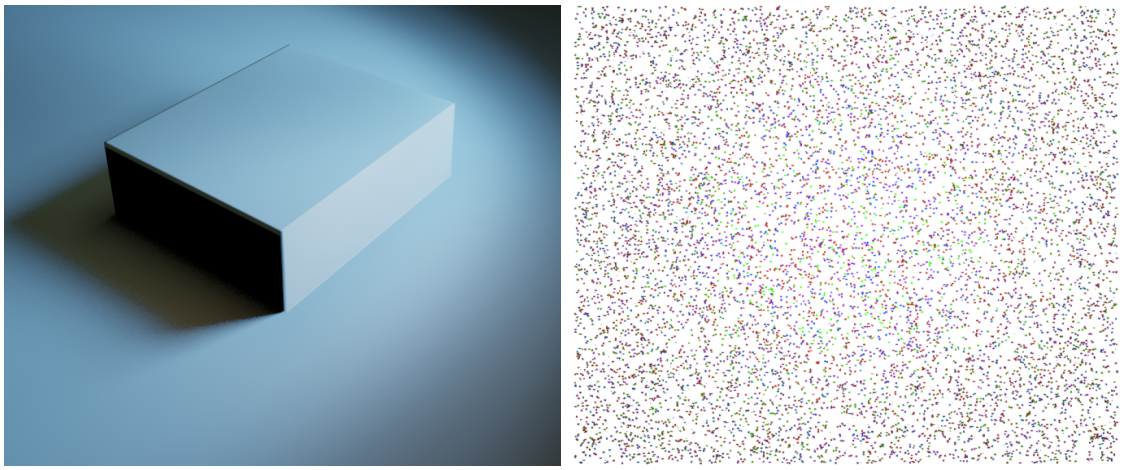
we find the maximum depth range D_{Max} allowed in the scene. This is done by searching for the depth range that results in the desirable relative velocity. This process is accelerated by a binary search algorithm.

Once we find the maximum depth range allowed, we focus our attention on finding the depth range that minimizes disocclusion artifacts. In between vertices, we estimate disocclusion by computing relative separation in pixel units between neighboring vertices of the depth mesh when the camera movement needed for KDI is performed. However, when colors between the vertices are the same, disocclusion is not visible and we can ignore disocclusion between these vertices. To approximate the disocclusion of an entire scene, we take the mean of n highest disocclusion estimates between vertices whose color is perceptually different. When the depth range is small, disocclusion artifacts as well as the perception of depth is reduced. So to find a proper balance, we compute teh depth range by $D_{Final} = \max\left(D_{Opt}, D_{Max} * .5\right)$, where D_{Opt} is the depth range that has n highest disocclusion estimates that are less than a user specified threshold (here we use threshold value of 2). D_{Max} is the maximum desirable relative velocity observed in Section 3.8.1. If the difference between D_{Final} and the depth range of M_P is small, we simply scale the depth range of M_P to match D_{Final}. However, if the depth range is large, we modify the maximum depth range allowed in the scene and start the perceptual re-mapping process again. In Figure 3.4, we show a comparison between the raw depth and the depth used to generate final depth mesh after applying all the optimizations stated here.

3.8 Experiments

We performed experiments to determine the relationship between velocities, positioning, and depth perception. Numerous studies have been carried out to determine what causes motion-based depth perception and its implications on segmentation, depth ordering, and depth magnitude estimation [39, 42, 43, 45, 50, 52, 68–70]. Most of them study motion in isolation and focus only on relative velocities. Since we generate KDE based on stereo images, for us the relationships between depth perception, spatial perspective, and the positioning of the pivot point in the scene is very important. Studying the totality of these effects is different from studying just the isolated motion because: (a) the background and foreground of an object placed at the pivot point move in opposite directions, while for an object distant from the pivot point they move in the same direction, (b) the average velocities of the objects placed at the pivot point are much lower than for objects that are distant from the pivot point, (c) the relative velocity between the foreground and the background of an object decreases considerably when receding from both the view point and the pivot point due to the perspective effect, and finally (d) the pivot point has no motion. These factors influence our depth perception generated by KDE. To our knowledge, there has been no attempt to examine the relationship among these factors and KDE.

We recruited volunteers from our campus and the local community for two experiments. All volunteers had normal or corrected-to-normal vision and were able to perceive KDE. The experiments were performed by following the university IRB guidelines and the participants were compensated nominally for their time and effort. The experiments



(a) (b)

Figure 3.5: The image (a) shows the side view of the object being displayed in the experiment rendered with lighting for illustration purposes. The image (b) shows the stimulus shown to the participants. Without KDE motion, the structure is seen as a flat collection of points.

were conducted in a brightly lit lab using a Dell monitor 2407WFPH with pixel pitch 0.270 mm. The distance between the participant's eye and the monitor was around 0.75 meters, which is within the OSHA recommended range [71]. In both studies, we showed stimuli, which subtended around 20×20 degrees visual angle, made of randomly generated dots with various depths. When motion was not present, the randomly generated dots were perceived as a flat fronto-parallel plane (Figure 3.5(b)).

3.8.1 Experiment I

The first experiment was done to study the relationship between relative velocity, depth perception, and the positioning of the pivot point. In this experiment, we rendered objects with various depths using randomly generated dots. The objects were composed of a plane with a box either going in or coming out as shown in Figure 3.5. The varia-

tion of depth between the front and the back of the object changes the relative velocities between them. The objects were placed at various distances from the pivot point, which changes rotational velocity. The objects close to the pivot point experience less velocity. The positioning of the pivot point, scene depth, and spatial perspective affects the relative velocity of pixels on the projected screen. For the study, the relative velocity between the foreground and the background was in the range 0.02 to 2.5 degrees visual angle per second. In this study, 11 volunteers participated.

Method: Each participant performed 40 trials with 3 second intervals between trials. There was no restriction on time. Participants were asked to estimate the size of the stimulus using a slider.

Result: We found that the estimation of the perceived depth between subjects varied dramatically. This is consistent with the previous study [39], where the researchers reported a high variation in perceived depth. However, when depth values are normalized per subject, a distinct pattern emerged. We computed the average of the normalized depth for all subjects for each distance from the pivot point, as shown in Figure 3.6.The participants perceived increased depth between relative velocities of 0.2 to 1.25 degrees visual angle per second; for higher relative velocities, the perceived depth remained constant. Thus, we consider 1.25 as the maximum desirable relative velocity. This is taken into consideration while generating the depth mesh. We performed a two-way analysis of variance (ANOVA) between the distance from the pivot point and the relative velocity in the field of view (view angle). The results of ANOVA for the distance from the pivot point are $(F(3,400) = 3.16, P = 0.0248)$, the relative velocity are $(F(9,400) = 108.94, P < 0.0001)$, and the interactions between them are $(F(27,400) = 0.96, P = 0.5208)$. These

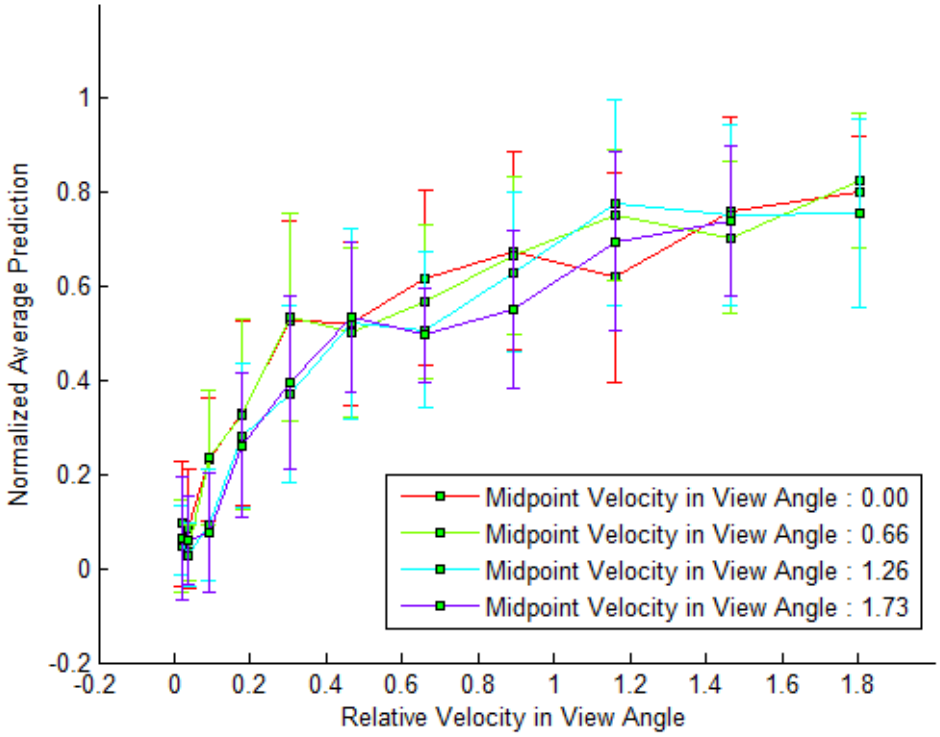


Figure 3.6: The figure shows the mean and standard deviation of the normalized depth perceived by subjects. When the objects are placed at various distances from the pivot point, the perceived velocity of the object changes. We show separate curves for each distance we tested and list the midpoint velocity. Between relative velocities of 0.2 to 1.25 degrees visual angle per second, participants perceived increased depth. For higher relative velocities, the perceived depth remained constant.

values indicate that both the distance from the pivot point and the relative velocity affect the perceived depth, but there is no evidence of an interaction between the two. In Section 3.7 we use this data to approximate normalized depth perception from the relative velocity. Please look at the accompanying video for a visual explanation of the results.

3.8.2 Experiment II

For the second study, we wanted to examine if placing objects at the pivot point (with part of the object in front and another part behind the pivot point) would have any impact on the depth perception even when the relative velocities are the same. The setup of the second experiment was the same as the first one. In this experiment, we rendered two objects side by side. The first object was placed at the pivot point, and therefore its foreground and background moved in opposite directions. The second object was placed at various distances from the pivot point. Both objects had the same relative velocity between their foreground and the background components. In this study, 6 volunteers participated.

Method: Each participant performed 44 trials with 3-second intervals between trials. Each participant was asked if they perceived the two objects at different depths and, if so, they were asked to select the object with the greater depth. The order of the images shown was randomized. Choices given to participants were: 1) No depth was perceived, 2) No preference, 3) Left Image, and 4) Right Image. These choices were randomly mapped to the rendering of the object placed at different locations.

Result: For relative velocities between 0.1 to 1.0 degrees visual angle per second, subjects perceived greater depth for the object placed at the pivot point. For higher velocities, the difference was reduced. For velocities less than 0.1 degrees visual angle per second, subjects had difficulty perceiving depth as shown in Figure 3.7. This variation in perception might be due to the very low average velocities around the pivot point compared to the velocities of other regions, even though the relative velocities are the same.

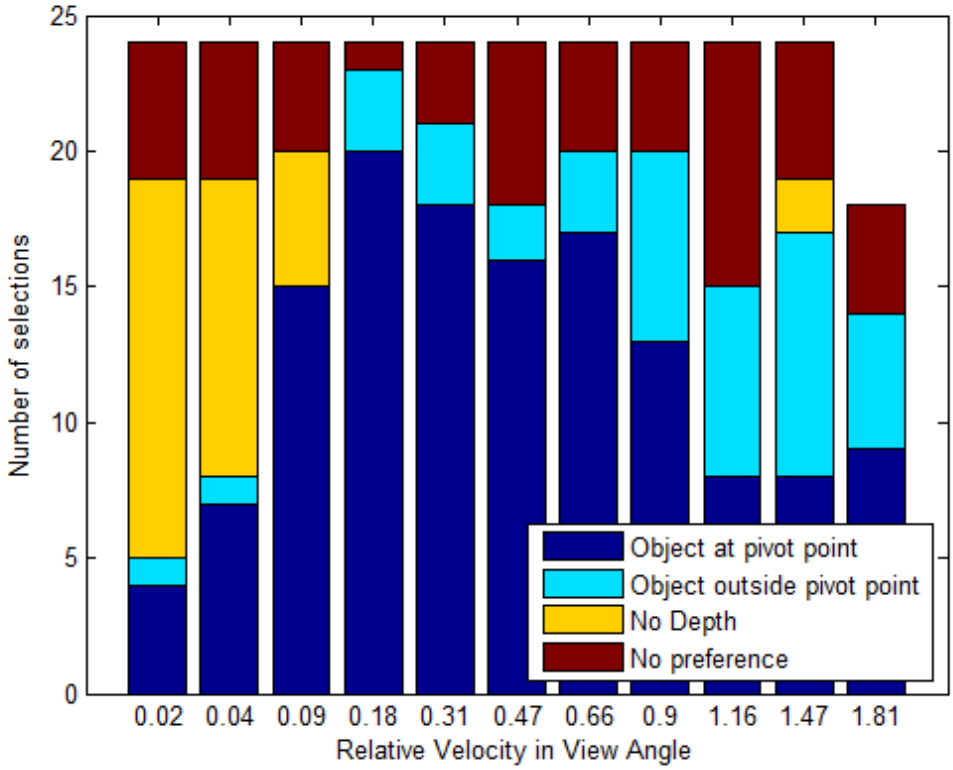


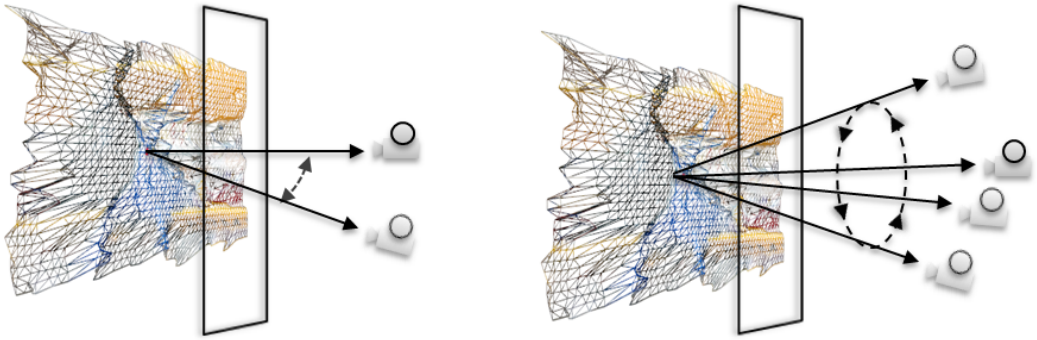
Figure 3.7: Here we show the total number of selections made when participants were asked if the depth perceived varied between the objects when positioning was different. If they observed a difference, they were asked to select the object with the greater depth.

These results highlight the importance of choosing a proper pivot point in the scene. The accompanying video shows an example of the comparison.

3.9 Rendering Camera Motion

We make use of the pivot point and the rotation axis calculated earlier to compute the rendering camera motion in two ways: angular motion and conical pendulum motion.

Angular motion to experience the KDE involves rotating the camera on a plane perpendicular to the rotation axis while looking at the pivot point as illustrated in Fig-



(a) Angular Motion (b) Conical Pendulum Motion

Figure 3.8: An illustration of the different types of camera movements: the angular motion (a) where the camera swivels on a plane perpendicular to the rotation axis while looking at the pivot point, and the conical pendulum motion (b) where the camera rotates along a conical surface while looking at the pivot point.

ure 3.8(a). The angular motion is close to most of the wiggle stereo images found online. However, rather than just flipping between two images and only relying on the sequence of images that were captured by the camera, we generate the intermediate views by rotating the rendering camera positions along an arc subtending a fixed angle at the salient pivot position. Figure 3.9 shows our results using the angular motion.

Figure 3.9: Representation of the angular camera motion

Conical pendulum motion involves rotating the camera along a circle on a plane

parallel to the view-plane, as the vector from the camera to the look-at pivot point traces out a cone. Figure 3.8(b) illustrates the conical pendulum motion performed by the camera.

3.10 Rendering and Interaction

The depth mesh, along with the camera motion described earlier, is used for rendering the scene. The depth mesh is kept static while the rendering camera moves to generate the KDE. Since we compute the depth mesh, our method allows us to add virtual objects or different layers into the scene at user-specified depth locations. We can therefore use the same camera parameters to render additional geometry at the desired locations while also rendering the depth mesh. This makes the kinetic-depth motion consistent for both the depth mesh as well as the additionally-added geometry. Currently we do not attempt to make lighting and shading seamless and consistent between the virtual object and the scene, but it would be an interesting exercise to attempt to achieve it by estimating the lighting parameters or by using methods such as Poisson Image Editing [72].

Although we compute the pivot point and the rotation axis automatically as a default, our system also allows the user to change these features as desired. This allows the user to customize the output of the kinetic-depth movement according to their needs. As mentioned earlier, the area around the pivot point has less motion. If there is more than one region that has low energy in a scene, being able to move the pivot point to various locations is critical.

3.11 Results and Discussion

We have implemented our system in C++ and Matlab. For all of our experiments, we use a Windows 7 64-bit machine with an Intel Core i5 2.67 GHz processor, an NVIDIA GeForce 470 GTX GPU, and 8 GB of RAM. We calculate image saliency using the graph-based visual saliency algorithm of Harel *et al.* [73] implemented in MatLab and we have used Sun *et al.*'s [55] code for optical flow calculations.

3.11.1 Subjective Evaluation

In order to evaluate the perceptual quality of the generated KDI images, we conducted a user study with 11 subjects who also participated in the earlier user study. Each subject performed 4 different tests to compare between 1) Wiggle 3D and our proposed method, 2) the naïve method and our proposed method, 3) Piku-Piku and our proposed method, and 4) angular and conical camera motion. We presented 10 examples for each test selected randomly from a larger collection of examples. Each example showed two images side by side. Subjects were asked which image they preferred by taking into account perceived depth and motion. They were asked to choose between left image, right image, and no preference. The subjects did not know which images were generated using our method.

Naïve vs. our method: The naïve method uses a scaled depth mesh (the magnitude of raw depth range scaled to the same range as our method) with the pivot point selected at the middle of the scene. The same camera motion in rendering was used for both the naïve method and our method. Out of 110 examples, subjects selected the naïve method 21

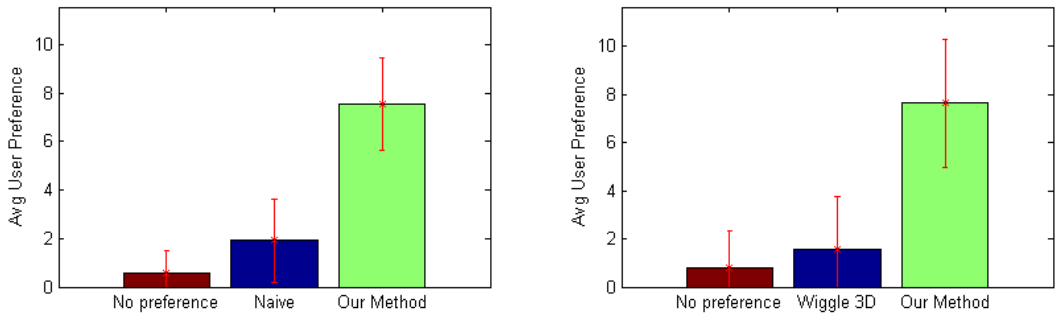
times, our method 83 times and had no preference 6 times. In Figure 3.10(a), we show the user preference per subject. We performed ANOVA with $(F(1,20) = 53.24; P < 0.0001)$ which shows that the difference is significant.

Wiggle 3D vs. our method: Wiggle 3D images were manually created using our tool. The zero plane position is selected to coincide with the most salient part of the scene usually located around the central region of the scene. Out of 110 examples, subjects selected Wiggle 3D images 17 times, our method 84 times and had no preference 9 times. In Figure 3.10(b), we show the user preference results per subject. We performed ANOVA to analyze the difference between user selection of Wiggle 3D and our method. We got $(F(1,20) = 34.21; P < 0.0001)$, which indicates that the difference is significant.

Piku-Piku vs. our method: Here we compared our method with Piku-Piku images from Start3D. Out of 110 examples, subjects selected Piku-Piku images 8 times, our method 87 times and had no preference 15 times. In Figure 3.10(c), we show the selection results according to each type of camera motion per subject. We performed ANOVA with $(F(1,20) = 55.04; P < 0.0001)$ which indicates that the difference is significant.

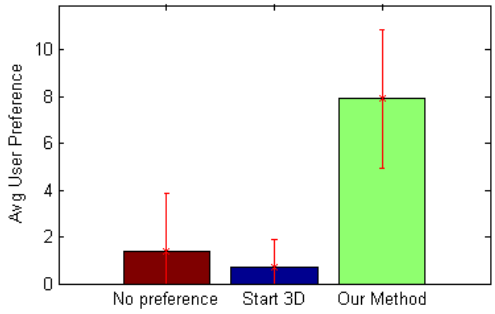
Angular vs. conical motion: We have compared images generated by two camera motions to find which method subjects preferred. Out of 110 examples, subjects selected angular motion 51 times, conical motion 52 times and had no preference 7 times. In Figure 3.10(d), we show the selection results for each type of camera motion per subject. We performed ANOVA with $(F(1,20) = 0.02; P = 0.8987)$, which indicates that the difference is not significant.

Discussion: We took a detailed look at the examples where the subjects chose other alternative methods over our method. In most of these examples, we found that the

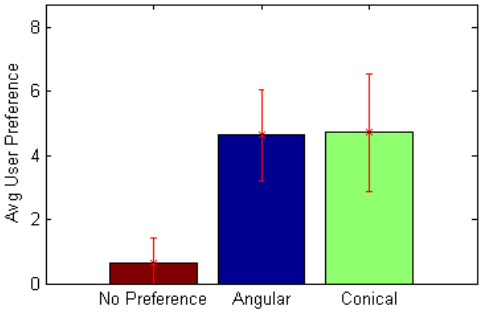


(a) Naïve vs. our method

(b) Wiggle 3D vs. our method



(c) Piku-Piku vs. our method



(d) Angular vs. conical motion

Figure 3.10: The results of the subjective evaluation. Here we show the average user preference.

depth range was low and the salient objects were already in the middle of the scene. In these examples, all of these methods generated comparable output. Also, both our method and Piku-Piku images rely on the computed depth maps. Any error in depth maps will affect the final output.

Wiggle3D and Piku-Piku images are highly dependent on the input pair of stereo images to generate the final rendering. They either switch or generate intermediate frames between the input images. Based on the angle of the input camera, the images may contain scaling or shearing. The perception of depth is reduced when noise, scaling, or shearing appears in the animated images. This can be easily observed when watching the

animation. However, these subtle artifacts are difficult to show using the vector plot, so we would like to request that the reviewers look at the accompanying video.

3.11.2 Limitations of our Approach

Although the use of the KDE to help understand the three-dimensional structure of a scene is valuable, it also has some disadvantages. First, adding motion could be visually distracting and has the potential to induce motion-sickness. Although this can be considerably reduced by making the camera motion small and smooth and by depth re-mapping, it cannot be completely eliminated. Second, the depth perceived by the KDE is not the same as the depth perceived by binocular disparity. In fact, Durgin *et al.* [39] have shown in their experiments that depth judgments based on binocular disparity are more accurate compared to depth judgments based on the KDE or motion parallax. However, binocular disparity perceived using stereoscopic displays has its own disadvantages such as the vergence and accommodation conflicts, and visual fatigue [74]. To reduce the visual fatigue in stereo displays, various techniques that compress the depth are also used [74, 75], and they too are likely to reduce the accuracy of the depth judgments. Third, our algorithm relies on computing approximated depth maps from a set of stereo images. When the depth map calculation has a significant error, the quality of our animation could also suffer. Thus, both ways of looking at 3D scenes have their own advantages and disadvantages. However, the use of the KDE is simpler and does not require any special devices.

3.12 Conclusions and Future Work

We have presented a method to automatically create smoothly animated images necessary to experience the KDE. Given a stereo image pair or an image-depth map pair as input, we have presented an approach to automatically generate animation that exhibits the KDE. Our approach allows decoupling of the input cameras from the rendering cameras to give us greater flexibility in defining multiple rendering camera motions. We have used two different ways of performing rendering camera motions (angular and conic pendulum motion) to view the resulting scene that minimize visual artifacts by taking into account depth perception, image saliency, occlusions, depth differences and user-specified pivot points. We have reported the results of user studies that examine the relationship between depth perception, relative velocity, spatial perspective effect, and positioning of the pivot point when generating KDI. We have presented a novel depth re-mapping method guided by perceptual relations based on the results of our user study. And finally, we have presented a subjective evaluation of our method by comparing it against other existing alternatives on a wide range of images.

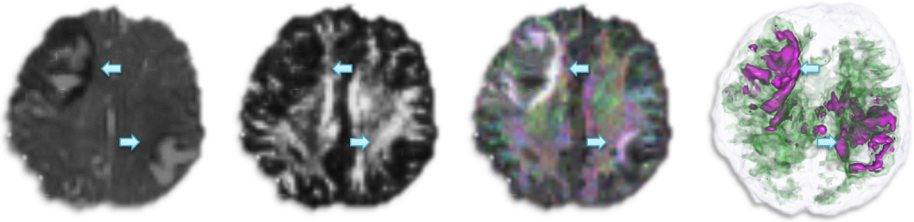
One future direction that can be taken in the research is to explore the usage of KDE on modern 3D games to enhance the perception of the 3D structures. KDI may very well become a popular 3D previewing tool for scientific visualization applications where depth awareness is important (such as stereo microscopy) and for consumer-entertainment applications including technology-mediated social community sites.

Chapter 4: Visualization of Brain Microstructure through Spherical Harmonics Illumination of Spatio-Angular Fields

4.1 Introduction

Spatio-angular fields are defined by a high angular resolution data defined in 3D space. These fields are commonly used in physics and chemistry when analyzing gravity [76, 77], heat transfer [78], geomagnetism [79], seismology [80], electric fields [81], and atoms' electron distributions [82]. In radiology, the diffusion patterns measured by magnetic resonance imaging (MRI) are also expressed in spatio-angular fields. Although commonly encountered, the visualization of spatio-angular fields has been barely explored in the visualization community.

MRI is a non-invasive tool that uses powerful magnetic fields to image biological tissues. Using MRI, one can detect the pattern of the diffusion of water movement using diffusion tensor imaging (DTI). DTI is effective in measuring the dominant direction of water diffusion in tissues and is widely used in studying white matter tracts in the brain. However, the traditional DTI technique is limited because the tensor estimation is based on the assumption that water diffusion patterns follow a Gaussian distribution. To measure the degree of the diffusional non-Gaussianity of water molecules in biological



(a) (b) (c) (d)

Figure 4.1: Using spherical harmonics lighting functions described in this chapter, we visualize spatio-angular fields. In this figure, we use the dataset of a patient suffering from traumatic brain injury. In traditional diffusion tensor imaging, the region around the injury does not exhibit a high contrast, as seen in the mean diffusion (Figure 4.1(a)) and fractional anisotropy images (Figure 4.1(b)). In Figure 4.1(c) and Figure 4.1(d), the detailed structure around the injury site can be seen, which provides vital information for investigators assessing the extent of injury. Figure 4.1(c) is generated using the novel planar visualization method, which incorporates shape information captured by diffusion kurtosis imaging of each data point through changes in color and intensity. Figure 4.1(d) is generated using the novel volume visualization method and conveys information about various shapes through the use of color, intensity, and opacity.

tissues, Jensen and Helpern [83] introduced diffusion kurtosis imaging (DKI). DKI has gained attention in the medical imaging community because of its ability to provide a more detailed structure of underlying tissues and because it shows promising applications in detecting tissue micro-structural changes caused by mild traumatic brain injuries and other neurological diseases [84]. In DKI, kurtosis tensor is computed for each imaged voxel, which collectively defines a spatio-angular field.

Spatio-angular fields present in DKI are challenging to visualize. In these datasets, each point has a unique shape defined by its directional data. The shape of the spatio-angular field is highly irregular in DKI because the fourth-order kurtosis tensor represents a more complex structure compared to the second-order diffusion tensor used in DTI. Statistical summarization of the shape of the datasets is a typical approach used for DTI datasets that might overlook important subtle details if used in DKI. Using glyphs

for visualization is another common approach used in various other imaging techniques [85–88]; however, because of the high variation of spatio-angular fields in DKI, it is difficult to compare different shapes due to occlusion, lighting, glyph density, and orientation. As the number of glyphs increases, the cost of rendering also increases; furthermore, the high visual density of glyphs makes them visually challenging to study. The highly irregular nature of these shapes limits the information conveyed by visualization if simplified glyphs are used.

To overcome these challenges, we propose the use of spherical harmonics lighting to visualize spatio-angular fields. We approximate spatio-angular fields using spherical harmonics basis functions. Then, using several spherical harmonics lighting functions, we light the spatio-angular fields, expressing the underlying structure. This gives investigators a quick and dynamic way to visually explore and analyze any spatio-angular field. In this chapter, we provide a systematic way to manage, analyze, and visualize spatio-angular fields (such as the shape of diffusion and kurtosis tensors used in DKI datasets) to facilitate insights into the micro-structural properties of the imaged volume. Our contributions are:

1. We apply spherical harmonics lighting functions to allow researchers to explore, visualize, and analyze the subtle changes in the shape of spatio-angular fields.

2. We incorporate 3D volume rendering by mapping spherical harmonics light integration of the spatio-angular field into the color and opacity to visualize structural properties present in these datasets.

3. We provide examples of case studies in traumatic brain injuries and cancer, where

we apply our method to study the shape of diffusion and kurtosis tensors.

4.2 Related Work

Numerous studies and literature reviews have been conducted on how 3D structures (glyphs) are used to study, analyze, and segment tissues using various types of MRI.

Kindlmann [89] and Ennis *et al.* [85] used superquadric glyphs to visualize the tensor field and applied it on DTI datasets. These perceptually motivated shapes simplify the visualization process. However, when the shape of the spatio-angular fields is highly irregular (as in DKI datasets), the effectiveness of these glyphs is limited.

Prckovska *et al.* [86] introduced a hybrid approach to visualize the structure of diffusion. They performed semi-automatic human-assisted classification of diffusion structures to separate different diffusion models, such as isotropic gray, anisotropic Gaussian, and non-Gaussian areas. Then, based on the classification, they used ellipsoids to display a simple diffusion shape and ray-traced spherical harmonics glyphs to highlight the complex structures.

Almsick *et al.* [88] presented a ray-casting-based approach to display a large number of spherical harmonics glyphs that are used to visualize high angular resolution diffusion imaging data. Their hybrid CPU- and GPU- based approach was able to interactively display large amount of glyphs in interactive frame rates.

Similar glyph-based methods have been used by Lu et al. [90] and Lazar *et al.* [91] to visualize DKI. Lu *et al.* [90] used spherical harmonics basis to analyze the DKI dataset. They performed harmonic analysis of the first three bands and used coefficient summa-

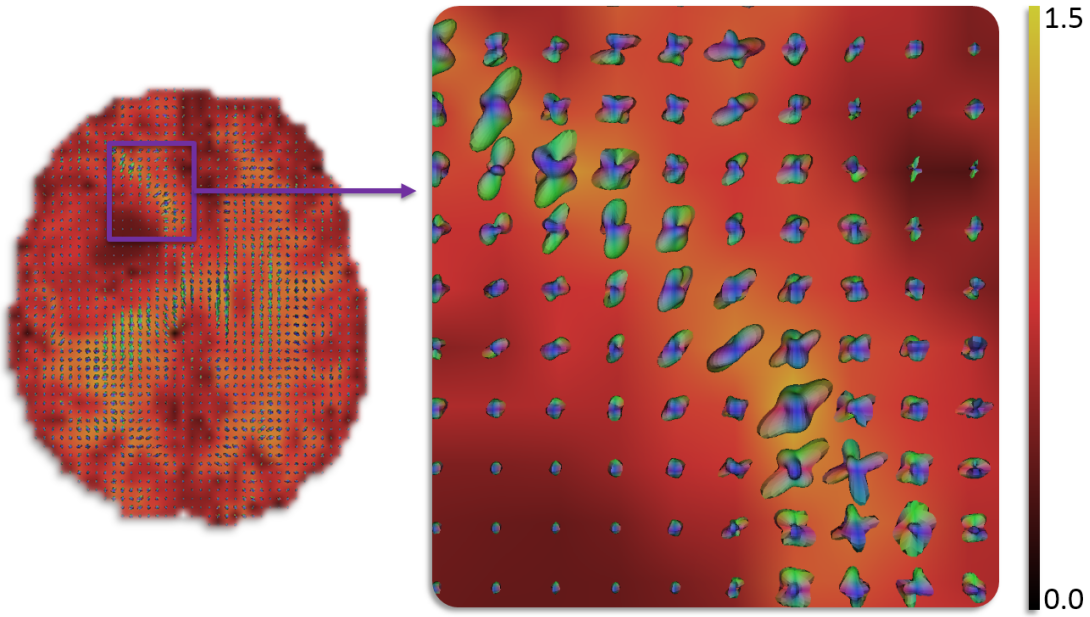


Figure 4.2: Visualization of a DKI spatio-angular fields using glyphs. This method demands a lot of computational power to render. Due to occlusion, the density of glyphs, a difference in orientation, changes in lighting, and surface irregularity of the glyphs, it is difficult to analyze and compare data visually. The visualization becomes even more cluttered when multiple planes must be shown.

tion to describe the rotationally invariant property of each band. Then they segmented white matter, gray matter, and fiber. In their paper, they used glyph and image-based visualization to study the DKI dataset. Lazar *et al.* [91] presented equations to approximate the orientation distribution function from DKI. For visualization, they also used glyphs. Both of these studies were focused on DKI analysis. Our technique is focused more on the visualization aspects, and we use a DKI dataset to visualize the diffusion kurtosis tensor. Our tool is also flexible in that it can incorporate the orientation distribution function presented by Lazar *et al.* [91].

Using glyphs with a reference image in the background has often been successfully used to visualize spatio-angular fields [85–89] such as DTI. However, for irregular and

complicated shapes present in DKI, these glyphs demand a lot more computational power to render and are difficult to visually study. This is because 1) part of the glyphs are always occluded, 2) the density of the glyphs shown in the screen can be very high and will require the proper management of glyph size and zoom level, 3) there are changes in orientation and lighting between neighboring glyphs, 4) and the surface of the spatio-angular field can be highly irregular. When multiple planes must be shown, which is often the case in volumetric datasets, the visualization can become very cluttered. Although the proposed method supports glyph-based visualization (Figure 4.2), we primarily provide viewing mechanisms based on spherical harmonics lighting.

Schussman and Ma [92] used anisotropic volume rendering to visualize dense lines. By taking viewpoint into consideration, they accumulated the average color and opacity contribution for every voxel and stored its spherical harmonics representation. While rendering, they converted anisotropic representation into standard voxels by evaluating spherical harmonics approximation. In our work, we use various lighting functions (which can be complicated shapes themselves) and their combinations to study the shape of spatio-angular fields. These lighting functions allow investigators to study irregular fields, such as DKI datasets, in a variety of different ways.

Volume rendering is widely used to visualize MRI datasets. Extensive work has been done to improve visualization by using advanced shading techniques, multiple depth cues, transfer functions, and global illumination [93–103]. For multiple lighting designs in volume rendering, Tao *et al.* [101] used an automatic lighting design based on the structure of the scene by measuring the structural changes in the images. In recent work by Zhang and Ma [103], a three point lighting system was used to enhance the depth and

the shape of the volume. These studies on volume rendering contain remarkable ways to enhance the visualization of the volume being displayed. Our work focuses on capturing the shape of each data point in the spatio-angular field and, with the aid of spherical harmonics lighting functions, easily visualizing them through multiple rendering techniques, including volume rendering.

4.3 Overview

The proposed method takes spatio-angular fields as an input and converts them into spherical harmonics fields using spherical harmonics basis functions. Depending on the task and the complexity of the field, we either choose to configure a single or multiple spherical harmonics lighting functions. Finally, by combining classified segments, the dynamic spherical harmonics lighting functions, and the input spatio-angular field, we render the scene. We provide two modes to view the final output using either planar or volume rendering. An overview of the system is shown in Figure 4.3.

4.4 Background

4.4.1 Diffusion Tensor Imaging

DTI considers the diffusion process to be Gaussian. For each gradient direction, the diffusion-weighted signal is approximated by the Taylor series expansion [4] given by the equation

$$ln\left[S\left(g,b\right)\right] = ln\left[S_0\right] - bD_{app}\left(g\right) + O\left(b^2\right),$$

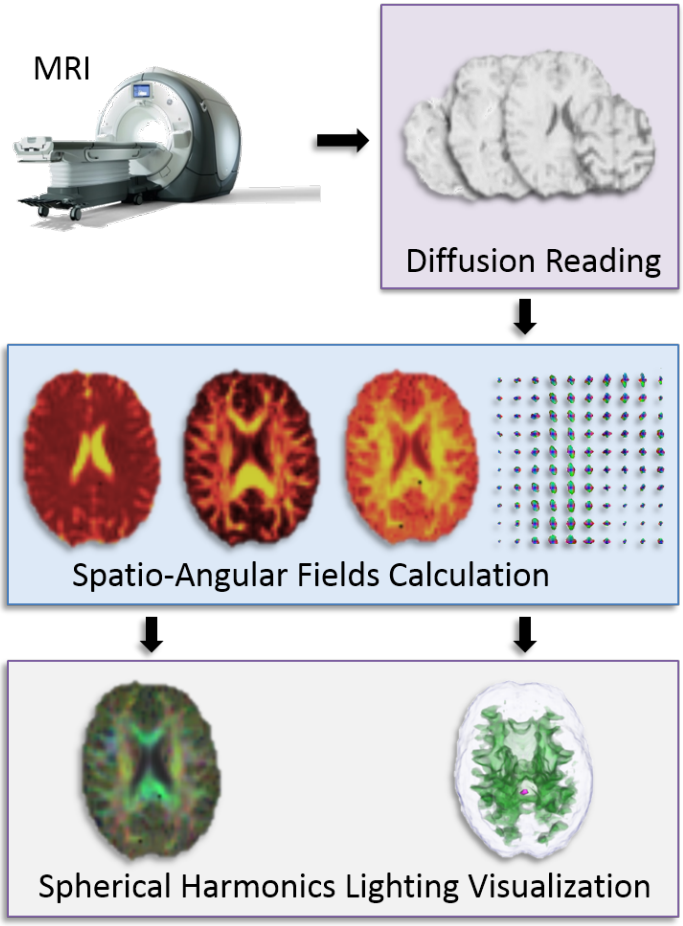


Figure 4.3: Overview of the proposed method. By using MRI, a large number of diffusional readings are recorded. Domain-specific processing is then performed to compute values such as mean diffusion, fractional anisotropy, mean kurtosis, and diffusion/kurtosis tensors. Then, depending on the task and the complexity of the field, we select either a single or multiple spherical harmonics lighting functions. Finally, by combining the dynamic spherical harmonics lighting functions and the input spatio-angular field, the scene is rendered. The output is either a planar-rendered image, a volume-rendered image, or both.

$$D_{app}(g) = \sum_{i=1}^{3}\sum_{j=1}^{3} g_i g_j D_{ij},$$

where g is a diffusion gradient, b is the MRI acquisition parameter b-value expressed in s/mm^2, S_0 is the signal without diffusion weighting, D_{ij} is the element of the diffusion tensor, and D_{app} is the apparent diffusion coefficient. The diffusion tensor is a second

order symmetric tensor with six independent elements. In DTI, the diffusion tensor is calculated for each voxel. The direction of the dominant eigenvector of the diffusion tensor is often used in fiber tracking.

4.4.2 Diffusion Kurtosis Imaging

The non-Gaussian property of water diffusion is measured in DKI. There are limitations in the traditional DTI technique because the tensor estimation is based on the assumption that water diffusion patterns follow a Gaussian distribution. While this may be true over larger diffusion time scales, it is less effective when the diffusion time is shortened, or in other words, when exploring even shorter diffusion distances of the water molecule. The measurement of diffusion over shorter time periods sensitizes the technique to the local diffusion heterogeneity reflective of the tissue micro-environment. This diffusion heterogeneity tends to make the probability distribution function for water diffusion non-Gaussian, suggesting that the normal way of obtaining the diffusion tensor which assumes Gaussian distribution, is no longer valid [104]. To measure the degree of the diffusional non-Gaussianity of water molecules in biological tissues, Jensen and Helpern [83] introduced DKI. Data acquisition needs are much larger in DKI compared to acquisition in DTI. The kurtosis tensor is often computed using data from 30 diffusional directions using at least two non-zero diffusion sensitivities. Common b-values used in DKI acquisition are 0, 1000, and $2000 s/mm^2$, and the scan time can be as long as 10min. Other forms of higher order diffusion-weighted imaging techniques, such as high angular resolution diffusion imaging or diffusion spectrum imaging, use a much higher

number of diffusional direction and b-values and, hence, are less clinically practical as they take a considerably longer time to scan. To measure the non-Gaussian property of the water diffusion, the Taylor series equation is further expanded [83, 104]. From the diffusional measurements in DKI, a fourth order diffusion kurtosis tensor is calculated by using the equation described by Jensen and Halpern [104].

$$ln\left[S\left(g,b\right)\right] = ln\left[S_0\right] - bD_{app}\left(g\right) + \frac{1}{6}b^2 D_{app}\left(g\right)^2 K_{app}\left(g\right) + O\left(b^3\right),$$

$$K_{app}\left(g\right) = \frac{1}{D_{app}\left(g\right)^2}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3} g_i g_j g_k g_l K_{ijkl},$$

$$K_{ijkl} = MD^2 W_{ijkl},$$

where MD is the mean diffusivity, K_{app} is the apparent kurtosis, and W_{ijkl} is the element of kurtosis tensor. The kurtosis tensor is a symmetric fourth order tensor with 15 independent elements. In full form, the signal in each gradient direction is described by

$$ln\left[S\left(g,b\right)\right] = ln\left[S_0\right] - b\sum_{i=1}^{3}\sum_{j=1}^{3} g_i g_j D_{ij} + \frac{1}{6}b^2\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3} g_i g_j g_k g_l K_{ijkl},$$

4.4.3 Spherical Harmonics

In our method, visualization of the spatio-angular fields is performed by using the spherical harmonics lighting functions. Spherical harmonics are basis functions that are used to represent and reconstruct any function on the surface of a unit sphere. Fourier functions are defined on the circle, whereas spherical harmonics are defined over the

surface of a sphere [105]. In visualization and graphics, spherical harmonics are used for lighting scenes with low frequency lights. It is a fast technique often used to approximate a computationally complex physical process, such as subsurface scattering and global illumination [100, 103, 106–109].

Spherical harmonics are ortho-normal functions defined by

$$Y_l^m(\theta,\phi) = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi},$$

where l is the band index, m is the order, P_l^m is an associated Legendre polynomial, and (θ,ϕ) is the representation of the direction vector in the spherical coordinate. Since the values used to define spatio-angular fields are positive and real, we use real-valued spherical harmonics.

To convert the function $f(\theta,\phi)$ into spherical harmonics basis, spherical harmonics coefficients a_l^m are approximated using the equation

$$a_l^m = \int_s f(\theta,\phi) Y_l^m(\theta,\phi) ds,$$

Once the coefficients are computed, the function $f(\theta,\phi)$ can be approximated by using the equation

$$f(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} a_l^m Y_l^m(\theta,\phi),$$

One key advantage of spherical harmonics representation is that integrating two functions over the sphere can be approximated easily by performing a dot product of their spherical harmonics coefficients [107, 110].

$$\int U(s) \times V(s) ds = \sum_{i=0}^{l^2} u_i(s) \times v_i(s),$$

where U and V are two functions defined on the surface of a sphere, and $u(s)$ and $v(s)$ are their spherical harmonics coefficients.

4.5 Diffusion Kurtosis Imaging Data

Imaging was performed using a 3T Siemens Tim Trio Scanner (Siemens Medical Solutions; Erlangen, Germany). Diffusion weighted images were obtained with $b = 1000, 2000 s/mm^2$ at 30 directions, together with 4 b_0 images, in-plane resolution $= 2.7 mm^2$, echo time/time repetition $= 101 ms/6000 ms$ at a slice thickness of $2.7 mm$ with two averages. DKI reconstruction was then carried out on each voxel using a MATLAB program as described by Zhuo *et al.* [84].

After diffusion and kurtosis tensors are computed, we represent the shape of these tensors by using spherical harmonics approximation. We use D_{app} and K_{app} to find the shape represented by the diffusion and the kurtosis tensor, respectively. The shape is then transformed into a spherical harmonics basis by computing spherical harmonics coefficients a_l^m to approximate the shapes of both the diffusion and kurtosis tensor. The shape of the diffusion tensor is simpler than the kurtosis tensor. For kurtosis, we use up to five bands to represent the shape; however, as described by Lu *et al.* [90], bands 2 and 4 do not contain any information, as the structure of the tensors are symmetric. Using a greater number of bands (> 5) is considered less beneficial, as high frequency data in both tensors contains more noise, as discussed in [90]. Once data is transformed to spherical

harmonics basis for each shape in a voxel, we have up to 15 spherical harmonics coefficients (there are 25 coefficients in total, but bands 2 and 4 are not used). These coefficients capture the shape, magnitude, and direction of the tensors. This approximation, using 15 spherical harmonics coefficients, is used for exploration and visualization.

4.6 Approach

Our method can be divided into two major steps: lighting function selection and visualization. Details of each step are described in the following sections.

4.6.1 Lighting

The key idea of our approach is to use the spherical harmonics lighting functions to visualize the spatio-angular fields. We define spherical harmonics lighting functions as basic shapes expressed in spherical harmonics basis that define different query functions researchers are interested in. The shape, size, orientation, and combination of the lighting functions defines the output of the visualization. By making changes in these attributes, one can study a variety of characteristics expressed in the spherical harmonics fields that may be representative of the local tissue microstructure.

Lighting spatio-angular fields: Spherical harmonics lighting is used to explore the directional strength of the spatio-angular field. In Section 4.4.3, we explained how spherical harmonics is used for lighting in visualization and graphics. We extend the application of spherical harmonics to visualize spatio-angular fields. Having expressed lighting functions and the volume field in spherical harmonics basis, we compute the

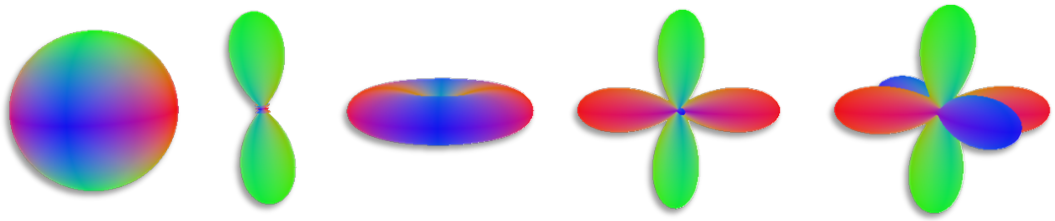


Figure 4.4: Representative lighting functions used to analyze spatio-angular fields. The shape and the size of these spherical harmonics lighting functions can be altered to analyze different properties. The first function shown above, which is spherical, is used to compute the average value. The next four functions are used to compute single and multiple directional properties of the spatio-angular field. These lighting functions can be rotated or combined with other functions as desired.

light response value R for each light p using the equation $R_p = k \times \sum_{i=0}^{l^2} u_i(s) * v_i(s)$, where $u(s)$ and $v(s)$ are their spherical harmonics coefficients of the lighting function and a data point in the spatio-angular field, respectively. k is used to scale the light response value in order to map it to the visual range ($[0-1]$) used in rendering.

Shape and Size: The shape of a spherical harmonics lighting function is one of the main attributes that determines how a spatio-angular field is visualized. Some common shapes are spherical, directional lobe, cross-sectional lobes, and radial lobe, as shown in Figure 4.4. The shape can be roughly divided into two parts, the directional component and its angular strength. The directional component expresses the incident path of the light, which is used to explore the strength of the spatio-angular field in the given direction (in DKI dataset, the strength of the spatio-angular field is the apparent kurtosis value). The angular strength corresponds to the area of the light. The range of angular strength is $0-360$ degrees. When the angular strength of the spherical harmonics lighting function is set to maximum, a spherical shape is formed. Lighting the spatio-angular field at this setting will give the mean strength of the field in all directions shown in Figure 4.5(a). By

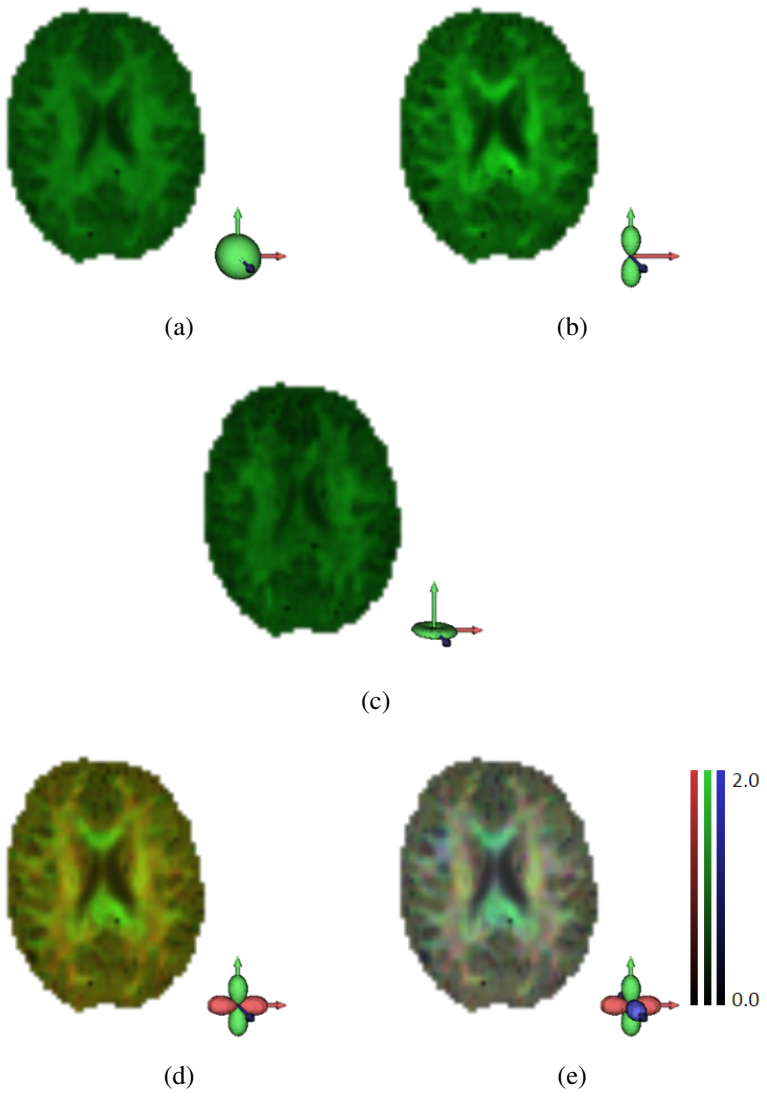


Figure 4.5: The effect of using different lighting functions. We use various lighting functions on the same MRI and show different views of the structural properties of the underlying spatio-angular fields. The orientation of these lighting functions can be adjusted to investigate directional changes in the diffusion profile. In Figure 4.5(a), we use a spherical lighting function, which provides the mean strength of the field in all directions. In Figure 4.5(b), we use the directional lobe aligned with the y-axis. The figure shows the strength of the spatio-angular field in the y direction. Similarly, in Figure 4.5(c), we use the radial lobe. In Figure 4.5(d) and Figure 4.5(e), we use multiple lighting functions with different colors mapped to each axis. Color bars with a numerical range are shown on the side of the image to illustrate the color scale used.

using a small value (for example, 30 degrees) for angular strength, we are able to study the strength of the spatio-angular field in any given direction shown in Figure 4.5(b). Spherical harmonics lighting with a radial-shaped lobe has a very useful property. It can be used to find the strength of values orthogonal to any given direction. This is beneficial in studying kurtosis tensor, as kurtosis values are very high in the direction orthogonal to the principle diffusion direction as shown in Figure 4.5(c). Taken together, the variations of the lighting functions provide a means to probe the tissue microstructure. Most normal tissues in the brain are generally characterized as gray matter, white matter, and cerebrospinal fluid. The spherical harmonics lighting allows researchers to further probe local tissue microstructure. An injured region of the brain may result in spatio-angular field of several shapes and sizes. In such cases the heterogeneity shown by the spherical harmonics lighting method can highlight of the local changes in the tissue microenvironment compared to a similar tissue in another region of the brain. In Figure 4.5, we show effect of using different lighting functions.

All of the lighting functions are expressed in spherical harmonics basis. For directional and radial lights, we define functions (F_{dir} and F_{rad}) on the surface of the sphere given by equation

$$F_{dir}(g) = \begin{cases} 1 & \text{if } (g \cdot g_{light}) > cos(a * .5), \\ 0 & \text{otherwise.} \end{cases}$$

$$a_{rad} = \begin{cases} 90 - a * .5 & \text{if } (90 - a * .5) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

$$F_{rad}(g) = \begin{cases} 1 & \text{if } ((g \cdot g_{light}) < cos(a_{rad}), \\ 0 & \text{otherwise.} \end{cases}$$

where a is angular strength, g_{light} is the direction of the light, g is unit direction. For symmetric lighting function, absolute value of the dot product is taken.

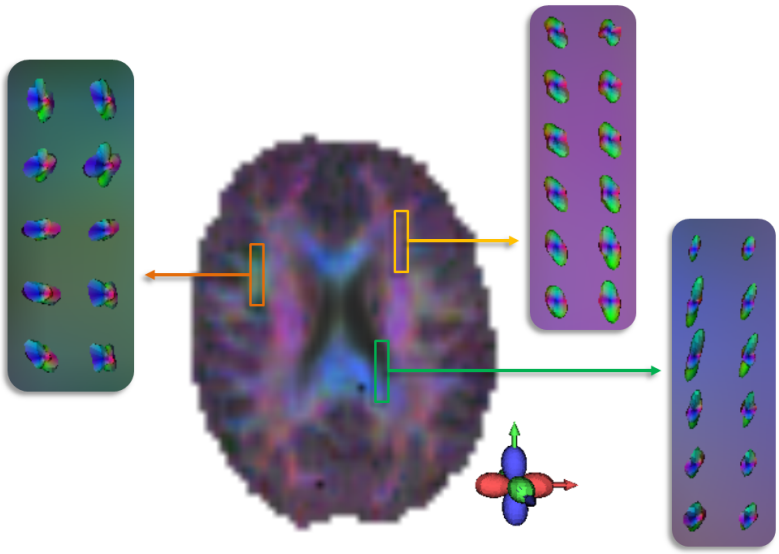


Figure 4.6: The spherical harmonics lighting function allows investigators to easily study the shape of spatio-angular fields and get an appreciation of the local water diffusion environment. The spherical harmonics lighting function captures information about the shape of spatio-angular fields. Here we are using three directional spherical harmonics lighting functions and displaying the relation between them and the individual shapes that define each voxel. We use red, green, and blue colors for each spherical harmonics lighting function. The sub-images show the individual shapes of the kurtosis tensor. The variation in shape is captured by a spherical harmonics lighting function and can be observed by the change in the color.

Global Orientation: The direction of the spherical harmonics lighting function allows us to guage the strength of a spatio-angular field in any desired direction. One can dynamically rotate the spherical harmonics lighting function to examine how the field is changing or how heterogeneous the environment is to water diffusion. By combining multiple lighting directions, studying multiple aggregated directions at any given time is

possible. This is often desired when the individual shapes in the spatio-angular fields have some known relation, such as symmetry. The adjustable lighting orientation will also allow researchers to probe the directional changes of the diffusion profile, which can be especially useful when studying complex tissue microstructure (e.g. regions around lesions).

Local Orientation: Information about local coherences can be utilized by spherical harmonics lighting to provide visualization of local properties. We applied this local adjustment to both DTI and DKI datasets. The diffusion values are dominant in the principal diffusion direction for each voxel and can be connected together to form a path if the fractional anisotropy (FA) value is also high. When the direction of the lighting function is aligned with the principal diffusion direction, diffusion and kurtosis values along the path can be visualized. In other words, it is quite possible to follow a fiber tract. For this visualization, we align the z-axis of the spherical harmonics lighting function with the first eigenvector of the diffusion tensor for every voxel, as shown in Figure 4.7. This will cause the direction of the light to be different for every voxel that is along the local coherences between the voxels. By interacting with the application, users are able to determine how the values change within the path. Figure 4.7 shows this application on the kurtosis datasets.

Multiple Lighting: Multiple simple shapes can be combined to create a complex lighting function. Multiple lighting functions allow the researchers to study the strength of a spatio-angular field on different shapes separately. This is slightly different from having multiple directional lobes in one lighting function. Using multiple lighting functions results in a separate light response value R_p for each light, whereas using one lighting

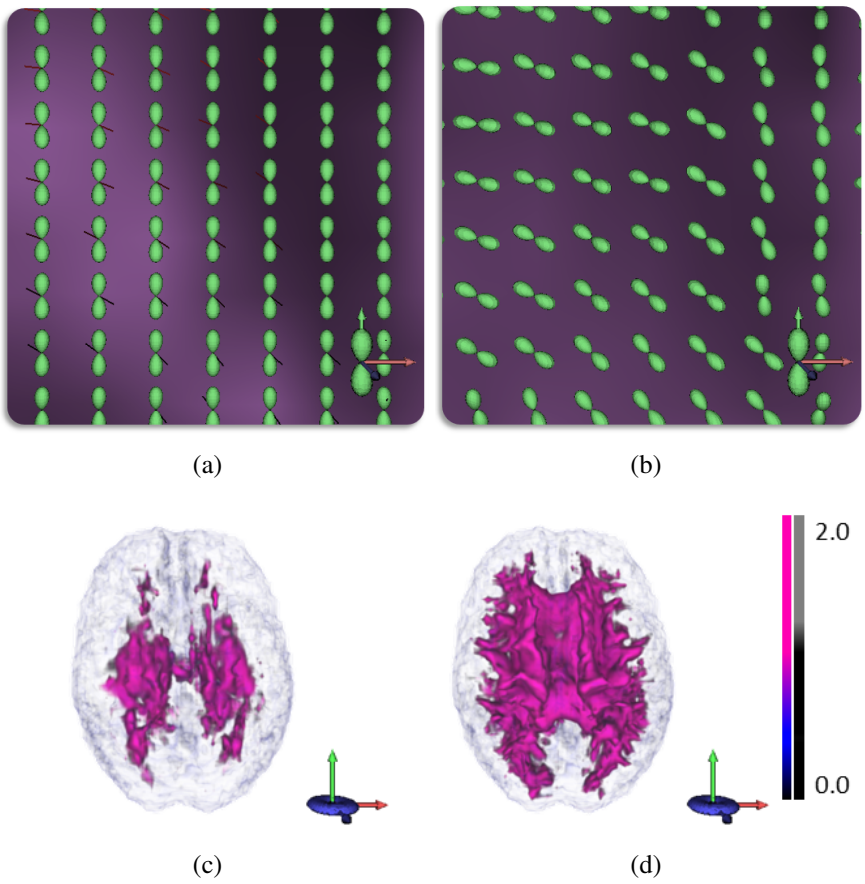


Figure 4.7: The orientation of the lighting function can be global or local. Figure 4.7(a) shows the global orientation, where each data point is lighted with the lighting function (show in green glyph) in the same direction. Figure 4.7(b) shows the local orientation, where each data point is lit with the lighting function oriented towards the principal diffusion direction of the diffusion tensor. Often, there is a local coherence between neighboring data points. For example, a path might go through them. To visualize the properties along the path, a directional spherical harmonics light can be aligned with the desired direction for each data point separately. For Figure 4.7(c), the global lighting direction is used. For Figure 4.7(d), we align the light along the principal diffusion direction for each data point and display how the kurtosis values change. Kurtosis values are dominant in the direction orthogonal to the principal diffusion direction of each data point. Both Figure 4.7(c) and Figure 4.7(d) are rendered by using method described in Section 4.6.2.

function with multiple directional lobes provides an aggregated view of the tissue microstructure. The lighting combination has a large number of applications. For example, when we applied three orthogonal lighting directions to the DKI dataset, we were able

to determine both the magnitude and directional variation present in the spatio-angular field. The result is shown in Figure 4.6. Here we used red, green, and blue colors for each spherical harmonics lighting function. The right sub-images show the individual shapes of the kurtosis tensor. Notice how the change in shape can be observed by the change in the color. In Figure 4.8, we demonstrate the contribution of individual lights in the brain of a patient who suffered a traumatic brain injury, along with the aggregated final image.

The principal diffusion direction points to the direction of the white matter axons. Probing into the directional kurtosis tensor in this direction offers information about the direction of the diffusion restriction. Probing along the two minor diffusion directions will also provide complementary information regarding diffusion. Hence, the combination of the two can potentially provide powerful information such as the general status of the axons and the degree of myelination or loss in myelination of the axons. Such information can provide extremely useful insights not only regarding the status of the tissue microenvironment, but also can be useful when evaluating the therapeutic efficacy of new drugs.

4.6.2 Visualization and Interaction

We use two methods to visualize the spatio-angular fields.

Planar Visualization: In this mode of visualization, only one plane of the spatio-angular field is shown at a time. For each data point in the plane, we apply the spherical harmonics lighting by computing the light response value R_p and then mapping it to the color space using a transfer function. After that, the color contribution of all the

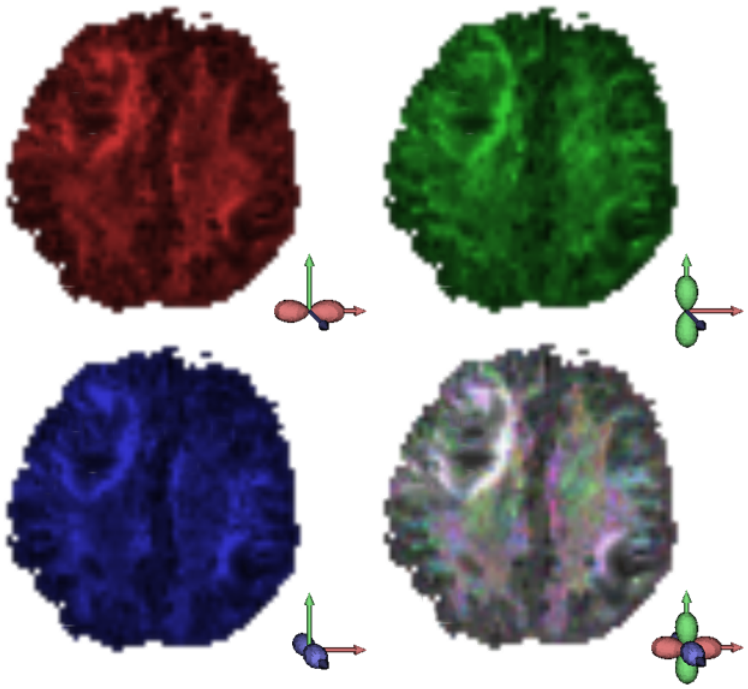
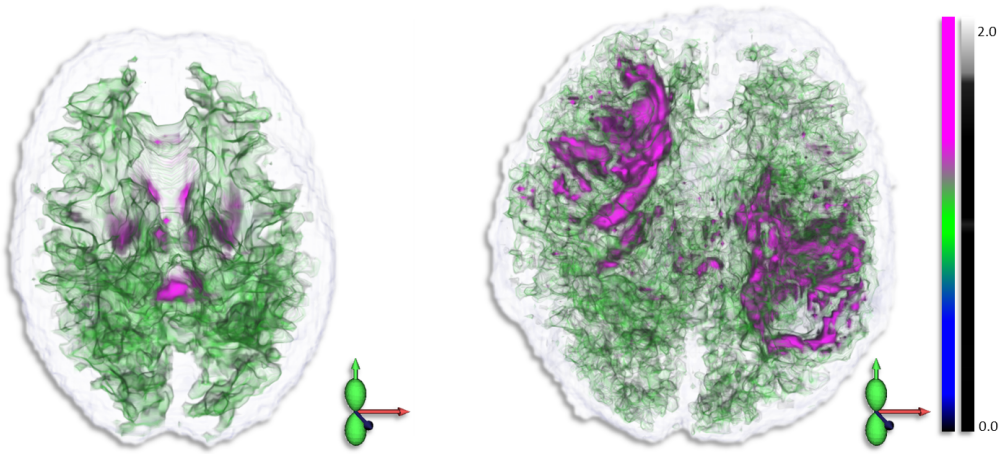


Figure 4.8: By combining multiple directional spherical harmonics lighting functions, we can produce an overview of the spatio-angular field. In this image, we use kurtosis data to summarize the injured region. The first three images (red, green, and blue) reveal how the shape varies in the x-, y-, and z-axes, respectively. The final image is the summation. Notice how structural changes in the spherical harmonics lighting function capture different properties of the imaged tissue.

lights is integrated together to compute the final color C of the voxel using the equation $C = \sum_i T_i(R_i)$, where T is the transfer function that maps light response value to a color value. Examples of this mode of visualization are shown in Figure 4.6 and Figure 4.8. It incorporates the shape information of each visible data point in spatio-angular fields through changes in color and intensity.

Volume Visualization: In this mode of visualization, we incorporate our method with a widely used volume rendering technique to show the entire dataset. Volume visualization allows researchers to analyze and visualize scalar fields with multiple layers at



(a) Normal Subject (b) Injured Patient

Figure 4.9: The directional kurtosis value of a normal subject and an injured patient using the spherical harmonics lighting tool. The coloring transforms from blue to green to red to denote the increasing value of kurtosis. In the injured patient, the region around the injury (seen in red) has a very high kurtosis value. By rotating the light, the response values of each data point are changed. When combined with a transfer function that has variations in opacity, investigators can study the magnitude and directional changes at the same time.

the same time, which has the potential to provide a complete picture of the dataset.

Dynamic light functions are used to map spatio-angular data to the surface of the volume. Typical volume rendering is done in scalar data. To perform volume visualization of a spatio-angular field, we first apply spherical harmonics lighting to each data point like we described in the planar visualization mode. Depending on the number of lights used, we may compute multiple light response values R_i. These values can be used to directly perform the transfer function lookup for each light separately. However, to lower the number of texture lookups, we compute the mean light response, which is used for performing transfer function lookup. The transfer function lookup obtains information about opacity and color value. When the lighting function or just the orientation of the lighting function is changed, the light response value is also changed, as different

directions of the spatio-angular fields are probed by the spherical harmonics light. This results in a different transfer function lookup, which will cause the surface of the volume to change. In doing so, we map the structural property of the spatio-angular field to the surface of the volume. This can be seen in Figure 4.7(c) and Figure 4.7(d) where we use global and local orientation, respectively, to display the underlying structure from different views.

The opacity of each voxel is directly determined by the result of the transfer function lookup. However, to compute the color value, we have multiple options. The first option is to directly use the color from the transfer function lookup, which is typical in volume rendering (using the mean light response). The second option is to map the response value for each spherical harmonics light to color the space and integrate them together to compute the final color. This coloring mode will produce a smooth diffuse lighting value that comes naturally with spherical harmonics lighting. However, to improve visualization, we enhance the lighting by incorporating monocular depth cues, such as shadows, ambient occlusion, edge enhancement, etc.

In Figure 4.9, we show the volume rendering of both a normal subject and an injured patient. Here, kurtosis values are differentiated based on color. The regions with injury in the volume rendered images are clearly depicted and represent the extreme kurtosis values, and the location of these high kurtosis values are consistent with the coup and contra-coup injury pattern typically seen in traumatic brain injury patients.

Interaction: For user interaction, we use mouse-based input and screen-based sliders. Using these input widgets, users can change the direction, size, and shape of the lighting function.

4.7 Case Study

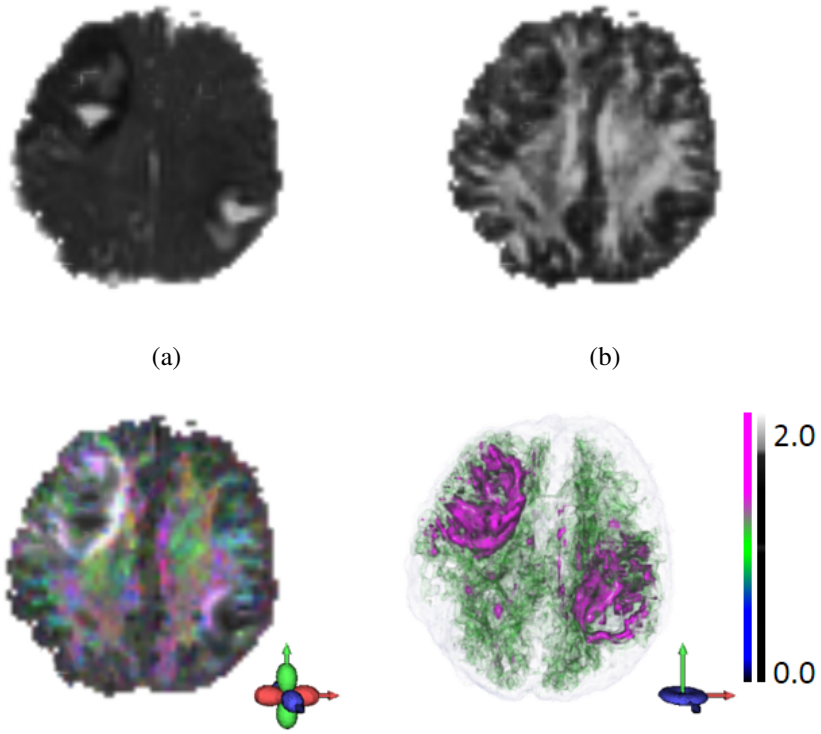
Clinicians use several different images for the diagnosis of injuries and diseases of the brain. Mean diffusion, fractional anisotropy, and principal diffusion directional color maps are used to study the Gaussian behavior of the diffusion with DTI. With DKI, an additional mean kurtosis map is used. The kurtosis tensor is a complex structure. Our tool allows researchers to study such complex shapes easily. We will now examine several case studies.

4.7.1 Case Study I

In Figure 4.10, we visually compare these different maps with the image generated using our tool. We have demonstrated that the region around the injury, which has very high kurtosis values can be visualized easily (Figure 4.10(c)). Furthermore, images depicting the structural information can also be generated using our tool (Figure 4.10(c) and Figure 4.10(d)). By interacting with the lighting functions, investigators can appreciate the local micro-structural information in any direction.

4.7.2 Case Study II

Another injury case is shown in Figure 4.11, where the injury is subtler, with no obvious lesions in the patient. However, the patient has sustained post-concussive symptoms, declined cognitive function, and brain atrophy. Using our tool, we examine the shape of the kurtosis tensor that is orthogonal to the principal diffusion direction of each data point. It is well-known that kurtosis is higher in these areas. In Figure 4.11, we show



(a) (b)

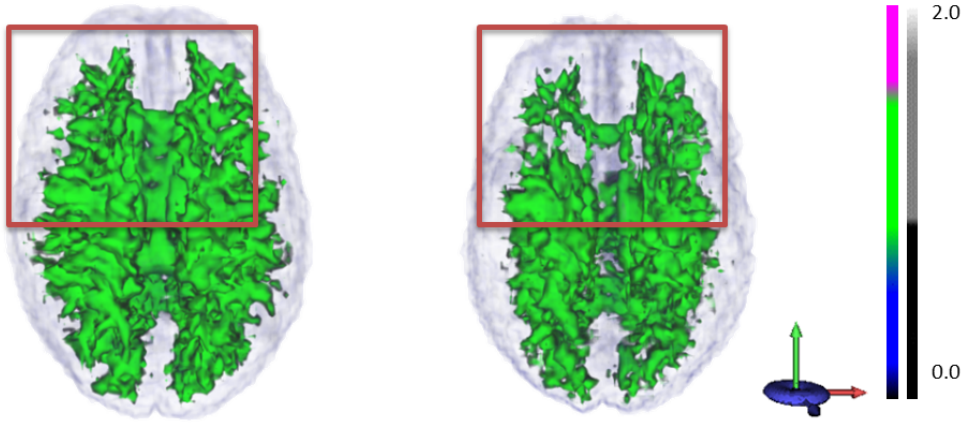
(c) (d)

Figure 4.10: We visually compare different color maps, such as mean diffusion (Figure 4.10(a)) and fractional anisotropy (Figure 4.10(b)), with the image generated using our tool. Kurtosis values are very high around the injury region and are considered vital for assessing the extent of the injury. The image generated by our tool, Figure 4.10(c), incorporates structural information from the DKI data into the visualization, which is shown through the changes in color and intensity. Figure 4.10(d), generated using local spherical harmonics lighting, shows the entire volume.

the difference observed in kurtosis values with the loss of white matter integrity, which is inline with the patient's clinical status.

4.7.3 Case Study III

In the injury case shown in Figure 4.12, the patient has sustained frontal lobe damage. The patient showed impressive recovery at a one month follow-up after the injury. The rendering of the kurtosis tensor with local spherical harmonics lighting showed im-



(a) (b)

Figure 4.11: We compare the MRI of a patient taken 10 days (left) and 6 months (right) after a traumatic brain injury. We use a radial spherical harmonics light to show the kurtosis value orthogonal to the principal diffusion direction of each data point. Despite a lack of anatomical changes in the patient, severe changes in kurtosis values, highlighted in red, are observed over time.

provement of the frontal lobe white matter structure, which was otherwise missed by the diffusion tensor.

4.7.4 Case Study IV

In this case, we look into the MRI of a patient with a tumor. High kurtosis values are observed around the tumor region, which were not visible in mean diffusion, fractional anisotropy, or principal diffusion directional color map weighted by fractional anisotropy. With the use of spherical harmonics lighting, additional structural information can be seen, which is shown in Figure 4.13(d).

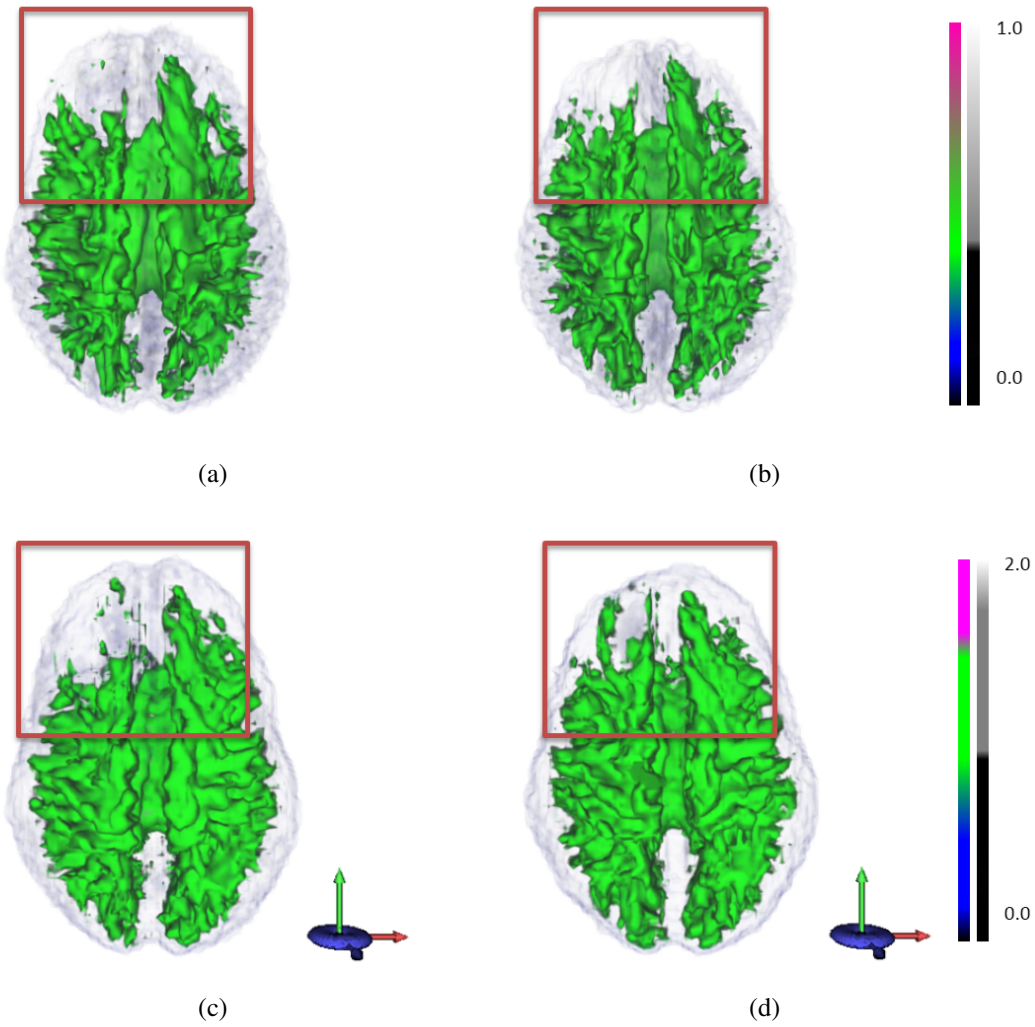


Figure 4.12: We compare the MRI of a patient taken 8 days (Figure 4.12(a) and Figure 4.12(c)) and 1 month (Figure 4.12(b) and Figure 4.12(d)) after a traumatic brain injury. In Figure 4.12(a) and Figure 4.12(b), we show volumetric rendering the fractional anisotropy value, and in Figure 4.12(c) and Figure 4.12(d) we show volume visualization using local spherical harmonics lighting. Changes in the kurtosis values (highlighted in red) over time are drastic compared to fractional anisotropy values.

4.8 Discussion

By using the lighting functions, the directional information of any complex shape can be observed. Key advantages of using the spherical harmonics lighting function in-

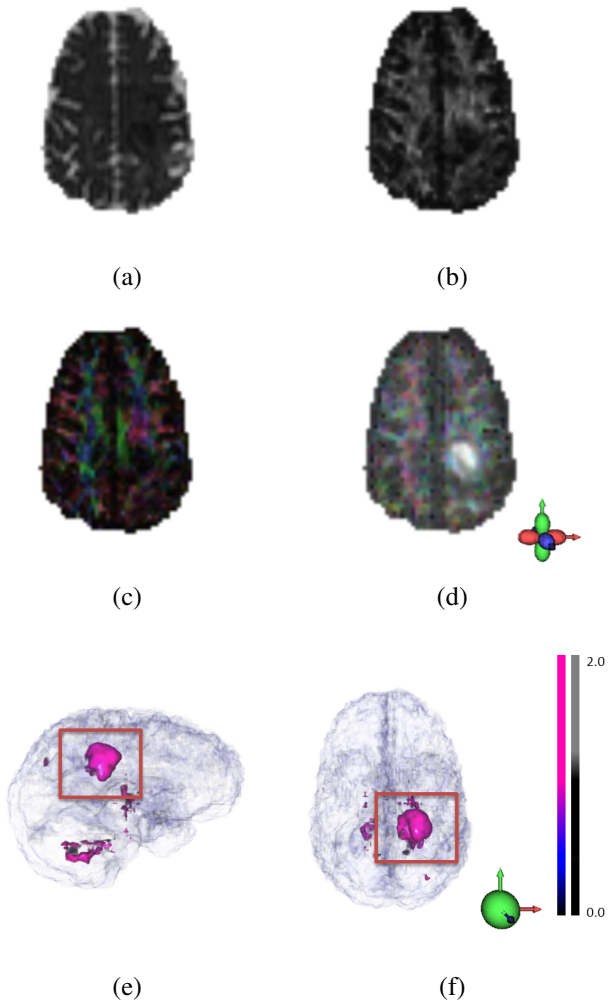


Figure 4.13: As before, we visually compare different color maps (such as mean diffusion, fractional anisotropy, principal diffusion directional color map weighted by fractional anisotropy, and mean kurtosis) with the image generated using our tool. Kurtosis is very high around the tumor, as seen in Figure 4.13(d), Figure 4.13(e), and Figure 4.13(f). With the use of spherical harmonics lighting additional structural information can be seen in Figure 4.13(d).

clude allowing us to a) analyze a spatio-angular field in any desired direction with any desired function, b) interactively explore the spherical harmonics field, c) easily compare values in any given direction without occlusion, scale, and density problems, which often occur when glyphs are used, d) effortlessly locate local changes, e) easily aligns in with the extensively researched direct volume rendering for visualization, and f) customize for

various datasets with only minor changes.

In DTI, the direction of the principal diffusion in regions with high fractional anisotropy is very important. In a study by Schwartzmann *et al.* [111], the difference in principal diffusion direction between good and poor readers was observed. Kurtosis tensors are more complex than diffusion tensors. Directional information that can be shown easily through spherical harmonics lighting can be a vital tool for radiologists. As already illustrated in Lazar *et al.*'s paper [91], the shape of the kurtosis tensor provides information about crossing fibers within a voxel, which is not depicted by the diffusion tensor. In more complex tissue environments (e.g. surrounding lesions in the brain in Figure 4.9), the kurtosis tensor is able to depict a unique shape and, may be reflective of the direction of tissue scaring, while the diffusion tensor only indicates non-directional restriction (low fractional anisotropy and low mean diffusion). These unique shapes can be studied by using various spherical harmonics lighting functions.

Although multiple lighting functions can be used to summarize the spatio-angular field, spherical harmonics lighting is designed to be an interactive tool. Interaction (by rotating the lighting functions) is required to exploit full power of the spherical harmonics lighting, whereas glyph-based visualization can be static especially when the number of glyphs is low. Both of these methods can be used together as they complement each other.

4.9 Conclusion And Future Work

We presented the use of spherical harmonics lighting functions, which have the potential to assist investigators in visually exploring and analyzing any spatio-angular

fields that are approximated in spherical harmonics basis function. Our approach includes spherical harmonics based lighting and visualization. Our dynamic approach allows researchers to gather information about the shape, magnitude, and direction for each data point of a spatio-angular volume field, which was not possible with previous methods. We have applied this method to state-of-the-art diffusion kurtosis imaging, which can benefit from visualization of the brain's complex microstructure.

In the future, we hope to extend the utility of our tool to various disease processes involving the human brain to study inflammation and neurodegeneration. We also hope to explore other spherical basis functions, such as spherical wavelets.

Chapter 5: Conclusions and Future Plans

In this dissertation, we have presented visual computing tools to study diffusion at the micro-scale for two driving applications. These applications are grounded in real-world uses with potentially significant impact and require high-throughput processing and visual computing tools. We have first described a visual computing tool to predict optical forces on microsphere ensembles in an optical tweezers system using CPUs and GPUs. The microparticles that are manipulated by the optical tweezers system are affected by the diffusion process, which introduces stochasticity when they are trapped. The random Brownian motion of fluid-suspended microparticles requires simulation fidelity finer than a microsecond, which demands an enormous amount of computation. Our high-performance visual computing system estimates the forces when laser beams interact with multiple microparticles and allows scientists to study the shadowing phenomenon. Studying this phenomenon in real-time is vital as it allows efficient planning required for trapping and manipulating microparticles. Our GPU-based implementation was approximately 66 times faster than traditional Ashkin's approach that was implemented in single-core CPU. When we compared our GPU-based approach with our CPU-based counterpart, GPU-based approach was approximately 10 times faster. We also presented the use of nonnegative matrix factorization to calculate the force which ex-

ploits the coherence of mapping from the incident ray to the components of force and the transmitted ray.

We have next presented the use of kinetic depth effect to facilitate the perception of the 3D structure of a scene, using a flexible way to generate and experience KDE from a pair of stereo images. Our technique provides a way to convey depth and is useful for visualizing the participating medium and for differentiating between different layers. It serves a very important role when other forms of monocular depth cues, such as shading, relative size, occlusion, spatial perspective, or texture gradient, are either absent or do not provide sufficient depth information. This situation occurs in the optical tweezers system where numerous microparticles are suspended in a fluid medium. Due to insufficient depth cues, microparticles can appear to be on the same depth plane even when there is a considerable variability in depth. This can create confusion and misperception when visually observed. KDE overcomes this problem, as it is very effective in enabling the depth perception of a scene. We have also described how we automatically created smoothly-animated images necessary to experience the kinetic depth effect. Given a stereo image pair or an image-depth map pair as input, we have presented an approach to automatically generate animation that exhibits the kinetic depth effect. Our approach allows the decoupling of the input cameras from the rendering cameras to give us greater flexibility in defining multiple rendering camera motions. We have used two different ways of performing rendering camera motions (angular and conic pendulum motion) to view the resulting scene that minimizes visual artifacts by taking into account depth perception, image saliency, occlusions, depth differences, and user-specified pivot points. We have reported the results of user studies that examine the relationship between depth percep-

tion, relative velocity, spatial perspective effect, and the positioning of the pivot point when generating kinetic depth images. Based on this study, we have presented a novel depth re-mapping method. We have also reported an evaluation of our method through a user study that compares with other existing alternatives on a wide range of images.

We have presented the use of spherical harmonics lighting functions to visually explore and analyze any spatio-angular fields that are approximated using spherical harmonics basis functions. Our approach includes spherical-harmonics-based lighting and visualization. Our dynamic spherical harmonic illumination allows researchers to gather information about the shape, magnitude, and direction for each data point of a spatio-angular volume field, which was not possible with previous methods. We have applied this method to state-of-the-art diffusion kurtosis imaging, which can benefit from the visualization of the brain's complex microstructure. We have presented several case studies where visualization of the kurtosis dataset using our tool produced a better diagnosis of injured regions, which would not have been possible with traditional diffusion tensor imaging.

Future Work

In force calculation of optical tweezers system, one future consideration is computing the force over several time steps by efficiently processing incremental changes. This can provide further speedup since we currently compute every time step independently.

In KDI, one future avenue to explore is the usage of KDE in 3D games to enhance the perception of 3D structures. We also would like to apply our tool to enhance the depth awareness of scientific applications (such as stereo microscopy) and for consumer-entertainment applications, including technology-mediated social community sites.

Finally, we plan to extend the use of our spherical harmonic lighting-based visualization tool to study various disease processes involving the human brain to study inflammation and neurodegeneration. We also plan to apply our visual tool to non-medical datasets. We also would like to explore other spherical basis functions, such as spherical wavelets.

List of Publications

[1] Sujal Bista, Jiachen Zhuo, Rao P. Gullapalli, and Amitabh Varshney. Visualization of Brain Microstructure through Spherical Harmonics Illumination of Spatio-Angular Fields. *Under Review IEEE Transactions on Visualization and Computer Graphics 2014*, 2014.

[2] Sujal Bista, Ícaro Lins Leitão da Cunha, and Amitabh Varshney. Kinetic Depth Images: Flexible Generation of Depth Perception. *In preparation for submission to ACM TAP*, 2014.

[3] Sujal Bista, Sagar Chowdhury, Satyandra K Gupta, and Amitabh Varshney. Using GPUs for Realtime Prediction of Optical Forces on Microsphere Ensembles. *Journal of Computing and Information Science in Engineering*, 13(3):031002, 2013.

[4] Sujal Bista, Sagar Chowdhury, Satyandra. K. Gupta, and Amitabh Varshney. Using GPUs for Realtime Prediction of Optical Forces on Microsphere Ensembles. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE*, pages DETC2012–71236. American Society of Mechanical Engineers, 2012 **[Best Paper Award]**.

[5] R. Patro, J. P. Dickerson, Sujal Bista, Satyandra K. Gupta, and A. Varshney. Speeding Up Particle Trajectory Simulations under Moving Force Fields using GPUs. *ASME Journal of Computing and Information Science in Engineering*, 12(2):021006, 2012.

[6] R. Patro, Cheuk Yiu Ip, Sujal Bista, S.S. Cho, D. Thirumalai, and A. Varshney. MDMap: A system for data-driven layout and exploration of molecular dynamics simulations. In *IEEE Symposium on Biological Data Visualization*, pages 111 – 118, 2011.

[7] R. Patro, C. Y. Ip, Sujal Bista, and A. Varshney. Social Snapshot: A System for Temporally Coupled Social Photography. *IEEE Computer Graphics & Applications*, 31(1):74 –84, 2011.

[8] Sujal Bista and Amitabh Varshney. Global contours. *UMIACS Technical Report*, CS-TR-4957, 2010/05/05/ 2010.

Bibliography

[1] David G. Grier. A revolution in optical manipulation. *Nature*, 424:810–816, August 2003.

[2] W. Singer, S. Bernet, and M. Ritsch-Marte. 3d-force calibration of optical tweezers for mechanical stimulation of surfactant-releasing lung cells. *Laser physics*, 11(11):1217–1223, 2001. eng.

[3] Wilhelm Jost. *Diffusion in solids, liquids, gases*. Physical chemistry. Academic Press, 1960.

[4] Derek K. Jones. *Diffusion MRI theory, methods, and applications*. Oxford University Press, USA, 2011.

[5] Arthur Ashkin. Forces of a single-beam gradient laser trap on a dielectric sphere in the ray optics regime. *Biophysical Journal*, 61:569–582, February 1992.

[6] A. G. Banerjee, A. Balijepalli, Satyandra K. Gupta, and T. W. LeBrun. Generating Simplified Trapping Probability Models From Simulation of Optical Tweezers System. *Journal of Computing and Information Science in Engineering*, 9:021003, 2009.

[7] B. Koss, S. Chowdhury, T. Aabo, W. Losert, and Satyandra K. Gupta. Indirect optical gripping with triplet traps. *J. Opt. Soc. America B*, 28(5):982–985, Apr. 2011.

[8] A. G. Banerjee, S. Chowdhury, W. Losert, and Satyandra K. Gupta. Survey on indirect optical manipulation of cells, nucleic acids, and motor proteins. *J. Biomed. Opt.*, 16(5), May 2011.

[9] S. Chowdhury, P. Svec, Chenlu Wang, W. Losert, and S.K. Gupta. Gripper synthesis for indirect manipulation of cells using holographic optical tweezers. In *IEEE International Conference on Robotics and Automation*, pages 2749–2754, may 2012.

[10] Sagar Chowdhury, Atul Thakur, Chenlu Wang, Petr Svec, Wolfgang Losert, and Satyandra K. Gupta. Automated indirect transport of biological cells with optical tweezers using planar gripper formations. In *IEEE Int. Conf. Automat. Sci. Eng.*, 2012.

[11] Atul Thakur, Sagar Chowdhury, Chenlu Wang, Petr Svec, Wolfgang Losert, and Satyandra K. Gupta. Automated indirect optical micromanipulation of biological cells using indirect pushing for minimizing photo-damage. In *Proc. ASME Int. Des. Eng. Tech. Conf. & Comp. Inf. Eng. Conf.*, 2012.

[12] Ashis Gopal Banerjee, Andrew Pomerance, Wolfgang Losert, and Satyandra K. Gupta. Developing a Stochastic Dynamic Programming Framework for Optical Tweezer-Based Automated Particle Transport Operations. *IEEE Trans. Autom. Sci. Eng.* , 7(2):218–227, Apr. 2010.

[13] Ashis Gopal Banerjee, Sagar Chowdhury, Wolfgang Losert, and Satyandra K. Gupta. Real-time path planning for coordinated transport of multiple particles using optical tweezers. *IEEE Trans. Automat. Sci. Eng.*, 9(4):669 –678, Oct. 2012.

[14] S. Chowdhury, P. Svec, C. Wang, K. Seale, J. P. Wikswo, W. Losert, and Satyandra K. Gupta. Investigation of automated cell manipulation in optical tweezers-assisted microfluidic chamber using simulations. In *Proc. ASME Int. Des. Eng. Tech. Conf. & Comp. Inf. Eng. Conf.*, 2011.

[15] S. Chowdhury, P. Svec, C. Wang, K. Seale, J. P. Wikswo, W. Losert, and S. K. Gupta. Automated cell transport in optical tweezers-assisted microfluidic chambers. *IEEE Trans. Automat. Sci. Eng.*, 2012. Accepted for publication.

[16] Sagar Chowdhury, Atul Thakur, Chenlu Wang, Petr Svec, Wolfgang Losert, and Satyandra K. Gupta. Automated indirect manipulation of irregular shaped cells with optical tweezers for studying collective cell migration. In *IEEE Int. Conf. Robot. Automat.*, 2013. Accepted for publication.

[17] Sagar Chowdhury, Atul Thakur, Chenlu Wang, Petr Svec, Wolfgang Losert, and Satyandra K. Gupta. Automated manipulation of biological cells using gripper formations controlled by optical tweezers. *IEEE Trans. Automat. Sci. Eng.*, 2013. Conditionally accepted for publication.

[18] A. Ashkin, J. M. Dziedzic, J. E. Bjorkholm, and Steven Chu. Observation of a single-beam gradient force optical trap for dielectric particles. *Optics Letters*, 11(5):288–290, May 1986.

[19] S. Bianchi and R. Di Leonardo. Real-time optical micro-manipulation using optimized holograms generated on the GPU. *Computer Physics Communications*, 181(8):1444–1448, 2010.

[20] A. Balijepalli, T.W. LeBrun, and Satyandra K. Gupta. Stochastic Simulations With Graphics Hardware: Characterization of Accuracy and Performance. *Journal of Computing and Information Science in Engineering*, 10:011010, 2010.

[21] R. Patro, J. P. Dickerson, S. Bista, Satyandra K. Gupta, and A. Varshney. Speeding Up Particle Trajectory Simulations under Moving Force Fields using GPUs. *ASME Journal of Computing and Information Science in Engineering*, 12(2):021006, 2012.

[22] Ihab Sraj, Alex C. Szatmary, David W. M. Marr, and Charles D. Eggleton. Dynamic ray tracing for modeling optical cell manipulation. *Opt. Express*, 18(16):16702–16714, Aug 2010.

[23] Jin-Hua Zhou, Hong-Liang Ren, Jun Cai, and Yin-Mei Li. Ray-tracing methodology: application of spatial analytic geometry in the ray-optic model of optical tweezers. *Applied Optics*, 47, 2008.

[24] Mark J. Harris, Greg Coombe, Thorsten Scheuermann, and Anselmo Lastra. Physically-based visual simulation on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '02, pages 109–118, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[25] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Tim Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[26] Mark Harris. Fast fluid dynamics simulation on the GPU. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 220, New York, NY, USA, 2005. ACM.

[27] Wei Li, Xiaoming Wei, and Arie E. Kaufman. Implementing lattice boltzmann computation on graphics hardware. *The Visual Computer*, 19(7-8):444–456, 2003.

[28] Youquan Liu, Xuehui Liu, and Enhua Wu. Real-time 3D fluid simulation on GPU with complex obstacles. In *Pacific Conference on Computer Graphics and Applications*, pages 247–256. IEEE Computer Society, 2004.

[29] Xiaoming Wei, Ye Zhao, Zhe Fan, Wei Li, Feng Qiu, Suzanne Yoakum-Stover, and Arie E. Kaufman. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):719–729, 2004.

[30] Everett H. Phillips, Yao Zhang, Roger L. Davis, and John D. Owens. Rapid aerodynamic performance prediction on a cluster of graphics processing units. In *AIAA Aerospace Sciences Meeting*, number AIAA 2009-565, January 2009.

[31] Nathan A. Carr, Jared Hoberock, Keenan Crane, and John C. Hart. Fast GPU ray tracing of dynamic meshes using geometry images. In *Graphics Interface*, pages 203–209. Canadian Human-Computer Communications Society, 2006.

[32] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002.

[33] Akira Fujimoto, Takayuki Tanaka, and Kansei Iwata. Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications*, 6:16–26, 1986.

[34] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics*, 23:496–505, 2004.

[35] Hans Wallach and D. N. O'Connell. The kinetic depth effect. *Journal of Experimental Psychology*, 45:205–217, 1953.

[36] Mark Wilson. Shooting Challenge: Wiggle 3D. www.gizmodo.com/5895289/shooting-challenge-wiggle-3d, January 2012.

[37] Joshua Heineman. Stereogranimator. http://stereo.nypl.org/, Jan 2012.

[38] Colin Davidson. Piku-Piku. www.start3d.com, Jan 2012.

[39] Frank H. Durgin, Dennis R. Proffitt, Thomas J. Olson, and Karen S. Reinke. Comparing depth from motion with depth from binocular disparity. *Journal of Experimental Psychology-human Perception and Performance*, 21:679–699, 1995.

[40] Shojiro Nagata. Pictorial communication in virtual and real environments. chapter How to reinforce perception of depth in single two-dimensional pictures, pages 527–545. Taylor & Francis, Inc., Bristol, PA, USA, 1991.

[41] George Sperling, Michael S. Landy, Barbara A. Dosher, and Mark E. Perkins. Kinetic depth effect and identification of shape. *Journal of Experimental Psychology-human Perception and Performance*, 15:826–840, 1989.

[42] E. J. Gibson, J. J. Gibson, O. W. Smith, and H. Flock. Motion parallax as a determinant of perceived depth. *Journal of Experimental Psychology*, 58(1):40–51, 1959.

[43] Brian Rogers and Maureen Graham. Motion parallax as an independent cue for depth perception. *Perception*, 8:125–134, 1979.

[44] Barbara A. Dosher, Michael S. Landy, and Geqrge Sperling. Kinetic depth effect and optic flow–I. 3D shape from Fourier motion. *Vision Research*, 29:1789–1813, 1989.

[45] Michael S. Landy, Barbara A. Dosher, George Sperling, and Mark E. Perkins. The kinetic depth effect and optic flow-ii. first- and second-order motion. *Vision Research*, 31(5):859 – 876, 1991.

[46] Dennis R. Proffitt, Irvin Rock, Heiko Hecht, and Jim Schubert. Stereokinetic effect and its relation to the kinetic depth effect. *Journal of Experimental Psychology-human Perception and Performance*, 18:3–21, 1992.

[47] Lytro. Perspective shift. www.lytro.com/camera#perspective_shift, January 2013.

[48] Ke Colin Zheng, R. Alex Colburn, Aseem Agarwala, Maneesh Agrawala, David Salesin, Brian Lee Curless, and Michael F. Cohen. Parallax photography: creating 3D cinematic effects from stills. In *Graphics Interface*, pages 111–118, 2009.

[49] W. Epstein. Perceptual invariance in the kinetic depth-effect. *The American Journal of Psychology*, 78(2):301–303, 1965.

[50] K. Nakayama and C. W. Tyler. Psychophysical isolation of movement sensitivity by removal of familiar position cues. *Vision Research*, 21:427–433, 1981.

[51] Terry Caelli. On the perception of some geometric properties of rotating three dimensional objects. *Biological Cybernetics*, 33:29–37, 1979.

[52] B. Rogers and M. Graham. Similarities between motion parallax and stereopsis in human depth perception. *Vision Research*, 22:261–270, 1982.

[53] H. Ujike, T. Yokoi, and S. Saida. Effects of virtual body motion on visually-induced motion sickness. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2004.

[54] D.A. Robinson, J.L. Gordon, and S.E. Gordon. A model of the smooth pursuit eye movement system. *Biological Cybernetics*, 55:43–57, 1986.

[55] Deqing Sun, S. Roth, and M.J. Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432 –2439, june 2010.

[56] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *PAMI, IEEE Transactions on*, 20(11):1254 –1259, nov 1998.

[57] C. Lawrence Zitnick, Nebojsa Jojic, and Sing Bing Kang. Consistent Segmentation for Optical Flow Estimation. In *International Conference on Computer Vision*, volume 2, pages 1308–1315, 2005.

[58] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432. ACM, 2001.

[59] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Transactions on Graphics*, 32, 2013. to be presented at SIGGRAPH 2013.

[60] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. In *International Conference on Computer Vision*, pages 72–79, 2009.

[61] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. 2000.

[62] Robert Patro, Cheuk Yiu Ip, Sujal Bista, and Amitabh Varshney. Social Snapshot: A System for Temporally Coupled Social Photography. *IEEE Computer Graphics and Applications*, 31:74–84, 2011.

[63] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH Conference Proc.*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[64] Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus H. Gross. Nonlinear disparity mapping for stereoscopic 3D. *ACM Transactions on Graphics*, 29, 2010.

[65] Tim Weyrich, Jia Deng, Connelly Barnes, Szymon Rusinkiewicz, and Adam Finkelstein. Digital bas-relief from 3d scenes. *ACM Trans. Graph.*, 26(3), July 2007.

[66] Philipp Krähenbühl, Manuel Lang, Alexander Hornung, and Markus Gross. A system for retargeting of streaming video. *ACM Trans. Graph.*, 28(5):126:1–126:10, December 2009.

[67] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[68] Mika E Ono, Josée Rivest, and Hiroshi Ono. Depth perception as a function of motion parallax and absolute-distance information. *Journal of Experimental Psychology-human Perception and Performance*, 12:331–337, 1986.

[69] Dhanraj Vishwanath and Paul B. Hibbard. Seeing in 3-D With Just One Eye: Stereopsis Without Binocular Vision. *Psychological Science*, 24(9):1673–1685, 2013.

[70] A. Yoonessi and C. Baker. Contribution of motion parallax to depth ordering, depth magnitude and segmentation. *Journal of Vision*, 10:1194–1194, 2011.

[71] US. Dept. of Labor. Occuptional saftey and health administartion. www.osha.gov/SLTC/etools/computerworkstations/components_monitors.html, January 2013.

[72] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22:313–318, 2003.

[73] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *Advances in Neural Information Processing Systems 19*, pages 545–552. MIT Press, 2007.

[74] D. M. Hoffman, A. R. Girshick, K. Akeley, and M. S. Banks. Vergence-accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision*, 8:33–33, 2008.

[75] Stereographics. *The Stereographics Developer's Handbook - Background on Creating Images for CrystalEyes and SimulEyes*. Stereographics Corporation, 1997.

[76] G. Balmino, B. Moynot, and N. Vals. Gravity field model of Mars in spherical harmonics up to degree and order eighteen. *Journal of Geophysical Research: Solid Earth*, 87(B12):9735–9746, 1982.

[77] Michael Gerstl. Computing the Earth Gravity Field with Spherical Harmonics. In MichaelH. Breitner, Georg Denk, and Peter Rentrop, editors, *From Nano to Space*, pages 277–294. Springer Berlin Heidelberg, 2008.

[78] Henry N. Pollack, Suzanne J. Hurter, and Jeffrey R. Johnson. Heat flow from the Earth's interior: Analysis of the global data set. *Reviews of Geophysics*, 31(3):267–280, 1993.

[79] D. R. Barraclough. Spherical harmonic models of the geomagnetic field. *Geomagnetic Bulletin*, 8, 1978.

[80] Jean-Paul Montagner. Can seismology tell us anything about convection in the mantle? *Reviews of Geophysics*, 32:115–137, 1994.

[81] D. R. Weimer. Models of high-latitude electric potentials derived with a least error fit of spherical harmonic coefficients. *Journal of Geophysical Research*, 100:19595–19607, 1995.

[82] R.C. Haddon, L.E. Brus, and Krishnan Raghavachari. Electronic structure and bonding in icosahedral {C60}. *Chemical Physics Letters*, 125(56):459 – 464, 1986.

[83] JH Jensen and JA Helpern. Quantifying non-Gaussian water diffusion by means of pulsed-field-gradient MRI. In *Proceedings of the 11th Annual Meeting of ISMRM*, volume 2154, 2003.

[84] Jiachen Zhuo, Su Xu, Julie L. Proctor, Roger J. Mullins, Jonathan Z. Simon, Gary Fiskum, and Rao P. Gullapalli. Diffusion kurtosis as an in vivo imaging marker for reactive astrogliosis in traumatic brain injury. *NeuroImage*, 59(1):467 – 477, 2012.

[85] Daniel B. Ennis, Gordon Kindlman, Ignacio Rodriguez, Patrick A. Helm, and Elliot R. McVeigh. Visualization of tensor fields using superquadric glyphs. *Magnetic Resonance in Medicine*, 53(1):169–176, 2005.

[86] V. Prckovska, T. H J M Peeters, M. Van Almsick, B. ter Haar Romeny, and A. Vilanova i Bartroli. Fused DTI/HARDI Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(10):1407–1419, 2011.

[87] Thomas Schultz and Gordon L. Kindlmann. Superquadric Glyphs for Symmetric Second-Order Tensors. *IEEE Transactions on Visualization and Computer Graphics*, 16:1595–1604, 2010.

[88] M. Van Almsick, T. H J M Peeters, V. Prckovska, A. Vilanova, and B. ter Haar Romeny. GPU-Based Ray-Casting of Spherical Functions Applied to High Angular Resolution Diffusion Imaging. *Visualization and Computer Graphics, IEEE Transactions on*, 17(5):612–625, 2011.

[89] Gordon L. Kindlmann. Superquadric tensor glyphs. In *Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 147–154, 2004.

[90] Hanzhang Lu, Jens H. Jensen, Anita Ramani, and Joseph A. Helpern. Three-dimensional characterization of non-gaussian water diffusion in humans using diffusion kurtosis imaging. *NMR in Biomedicine*, 19(2):236–247, 2006.

[91] Mariana Lazar, Jens H. Jensen, Liang Xuan, and Joseph A. Helpern. Estimation of the orientation distribution function from diffusional kurtosis imaging. *Magnetic Resonance in Medicine*, 60:774–781, 2008.

[92] Greg Schussman and Kwan-Liu Ma. Anisotropic volume rendering for extremely dense, thin line data. In *Proceedings of the conference on Visualization'04*, pages 107–114. IEEE Computer Society, 2004.

[93] Carlos Correa and Kwan-Liu Ma. Size-based transfer functions: A new volume exploration technique. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1380–1387, 2008.

[94] Runzhen Huang and Kwan-Liu Ma. RGVis: Region growing based techniques for volume visualization. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 355–363. IEEE, 2003.

[95] Cheuk Yiu Ip, Amitabh Varshney, and Joseph JaJa. Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2355–2363, 2012.

[96] Gordon L. Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Volume Visualization and Graphics*, pages 79–86, 1998.

[97] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 8(3):270–285, Jul 2002.

[98] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8:29–37, 1988.

[99] Eric B Lum and Kwan-Liu Ma. Lighting transfer functions using gradient aligned sampling. In *Proceedings of the conference on Visualization'04*, pages 289–296. IEEE Computer Society, 2004.

[100] Philipp Schlegel, Maxim Makhinya, and Renato Pajarola. Extinction-Based Shading and Illumination in GPU Volume Ray-Casting. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1795–1802, 2011.

[101] Yubo Tao, Hai Lin, Hujun Bao, Feng Dong, and Gordon Clapworthy. Structure-aware viewpoint selection for volume visualization. In *Visualization Symposium, Pacific Asia-Pacific*, pages 193–200, 2009.

[102] A. Tikhonova, C. D. Correa, and K.-L. Ma. An exploratory technique for coherent visualization of time-varying volume data. *Computer Graphics Forum*, 29(3):783–792, 2010.

[103] Yubo Zhang and Kwan-Liu Ma. Lighting Design for Globally Illuminated Volume Rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2946–2955, 2013.

[104] Jens H. Jensen and Joseph A. Helpern. MRI quantification of non-Gaussian water diffusion by kurtosis analysis. *NMR in Biomedicine*, 23(7):698–710, 2010.

[105] Martin J. Mohlenkamp. A Fast Transform for Spherical Harmonics. *Journal of Fourier Analysis and Applications*, 1997.

[106] Nathan A Carr, Jesse D Hall, and John C Hart. GPU algorithms for radiosity and subsurface scattering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 51–59. Eurographics Association, 2003.

[107] Jan Kautz, John Snyder, and Peter-Pike J. Sloan. Fast Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics. In *Eurographics Symposium on Rendering/Eurographics Workshop on Rendering Techniques*, pages 291–296, 2002.

[108] Françis X Sillion, James R Arvo, Stephen H Westin, and Donald P Greenberg. A global illumination solution for general reflectance distributions. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 187–196. ACM, 1991.

[109] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 527–536. ACM, 2002.

[110] Brian Cabral, Nelson L. Max, and Rebecca Springmeyer. Bidirectional reflection functions from surface bump maps. *ACM Siggraph Computer Graphics*, 21:273–281, 1987.

[111] Armin Schwartzman, Robert F Dougherty, and Jonathan E Taylor. Cross-subject comparison of principal diffusion direction maps. *Magnetic Resonance in Medicine*, 53(6):1423–1431, 2005.