ACCELERATED RAY TRACING FOR HEADLAMP SIMULATION

by

Ryota Kimura

A Thesis Submitted to the Faculty of the

COLLEGE OF OPTICAL SCIENCES

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2017

STATEMENT BY AUTHOR

The thesis titled Accelerated Ray Tracing for Headlamp Simulation prepared by Ryota Kimura has been submitted in partial fulfillment of requirements for a master's degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Ryota Kimura

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Russell A. Chipman
Russell A. Chipman
Professor of Optical Sciences

Nov. 22, 2017
Date

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Professor Russell A. Chipman of the College of Optical Sciences at the University of Arizona. He steered me in the right the direction.

I would also like to thank my thesis committee Professor R. John Koshel and Associate Professor Yuzuru Takashima of the College of Optical Sciences at the University of Arizona. Their wide knowledge and advices were very helpful.

I would also like to thank Seiichiro Kitagawa who is President of Nalux. Co., Ltd. He gave me a chance to study in the University of Arizona and supported me financially.

Finally, I thank to my family. They supported me even they had difficult time for living oversea.

Author

Ryota Kimura

Table of Content

LIST OF FIGURES 8

LIST OF TABLE 11

ABSTRACT 12

CHAPTER 1: INTRODUCTION 13

1.1 Ray Tracing 13

1.2 Headlamp for Automotive 13

1.3 Previous Work 15

1.4 Outline of Thesis 15

CHAPTER 2: RAY TRACING 16

2.1 Intersection Point Searching 16

2.1.1 Intersection point between a Ray and a Plane Surface 17

2.1.2 Intersection point between a Ray and a Spherical Surface 18

2.1.3 Intersection point between a Ray and a Complex surface 19

2.1.4 Accelerated Searching Algorithm for Intersection point between a Ray and a Complex Surface 20

2.1.5 Introducing Nagata Patch for Triangular Patches to Interpolate Quadratically 21

2.1.6 Introducing Nagata Patch for Quadrilateral Patch to Interpolate Quadratically 23

2.2 Refraction 24

2.2.1 Fermat's Principle and Snell's Law 24

2.2.2 Snell's Law in Three-Dimensional Space 26

2.2.3 Refractive Index 28

CHAPTER 3: PHOTOMETRY 29

3.1 Photometrical Units 29

3.2 Photometer 30
3.3 Convolution 32
CHAPTER 4: CUDA PROGRAMING 33
4.1 GPU Architecture 33
4.2 GPU Memory Architecture 34
4.3 CUDA 35
4.4 CUDA with Mathematica 36
CHAPTER 5: VALIDATION TESTS 37
5.1 Spherical Lens (Plano-convex lens) 37
5.1.1 Model Analysis with OpticStudio 37
5.1.2 CUDA Ray Tracing for a Spherical Surface and a Plane Surfaces 38
5.1.3 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surfaces (Linear Interpolation) 39
5.1.4 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Triangular Patch Interpolation) 39
5.1.5 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Quadrilateral Patch Interpolation) 40
5.1.6 Comparison of Results 41
5.2 Aspherical Lens (Plano-convex lens) 43
5.2.1 Model Analysis with OpticStudio 43
5.2.2 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Linear Interpolation) 44
5.2.3 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Triangular Patch Interpolation) 45

5.2.4 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Quadrilateral Patch Interpolation) 46
5.2.5 Comparison of Results 47
5.3 Headlamp Lens and Starting ray set 49
5.3.1 Simulation Using SPEOS 49
5.3.2 Simulation Using OpticStudio 50
5.3.3 Simulation Using CUDA Ray Tracing for a Point Cloud Surface (Linear Interpolation) 51
5.3.4 Simulation Using CUDA Ray Tracing for a Point Cloud Surface (Nagata Triangular Patch) 51
5.3.5 Simulation Using CUDA Ray Tracing for a Point Cloud Surface (Nagata Quadrilateral Patch) 52
5.3.6 Comparison of Results between OpticStudio and CUDA Ray Tracing 52
5.3.7 Comparison of Results between SPEOS and CUDA Ray Tracing 54
5.4 Comparison of Calculation Speed 55
CHAPTER 6: CONCLUSION 56
6.1 Summary 56
6.2 Future Work 57
APPENDIX A: FLOW CHART OF CODES 58
Intersection Search with the Brute Force Attack 58
Intersection Search with the K-d Tree (with Linear Interpolation) 59
Intersection Search with the K-d Tree (with Nagata Patch Interpolation) 60
APPENDIX B: INPUT FORMAT FOR COMMANDS WRITTEN IN MATHEMATICA 61

BIBLIOGRAPHY 64

LIST OF FIGURES

		Page
1-1	Projector-type headlamp lens	14
1-2	Light distribution pattern requirements in FMVSS [10]	14
1-3	Actual light distribution model of headlamp	15
2-1	Global coodinates for three-dimensional space	16
2-2	Intersection point between a ray and a plane surface	17
2-3	Intersection point between a ray and a spherical surface	18
2-4	Intersection point between a ray and point cloud, a set of points lying on an optical surface, with an associated grid or surface normals	20
2-5	k-d tree example	21
2-6	Triangular patch [12]	23
2-7	Quadrilateral patch [12]	24
2-8	Light path for Fermat's principle	25
2-9	Snell's law in three-dimensional spaces	28
3-1	Photopic luminous efficiency function K(λ) (CIE1931)	30
3-2	Photometry test setup [3]	32
3-3	Active area of a photometer	32
3-4	2D Binning and convolution	33
4-1	Memory hierarchy of CUDA [2]	35
4-2	Comparison of data copying program in C and Mathematica	36
5-1	Spherical lens test layout	37
5-2	Spot diagram of a spherical lens using OpticStudio	38
5-3	Spot diagram of a spherical lens using CUDA ray tracing (Non-interpolation)	38
5-4	Spot diagram of a spherical lens using CUDA ray tracing (Linear interpolation)	39

5-5	Spot diagram of a spherical lens using CUDA ray tracing (Nagata triangular patch)	40
5-6	Spot diagram of a spherical lens using CUDA ray tracing (Nagata quadrilateral patch)	41
5-7	Aspherical lens test layout	44
5-8	Spot diagram of an aspherical lens using OpticStudio	44
5-9	Spot diagram of an aspherical lens using CUDA ray tracing (Linear interpolation)	45
5-10	Spot diagram of an aspherical lens using CUDA ray tracing (Nagata triangular patch)	46
5-11	Spot diagram of an aspherical lens using CUDA ray tracing (Nagata quadrilateral patch)	47
5-12	Projector type LED headlamp simulation layout	49
5-13	Result of ray tracing with SPEOS	50
5-14	Result of ray tracing with OpticStudio (5,000,000 rays)	50
5-15	Result of ray tracing with CUDA ray tracing (Linear interpolation)	51
5-16	Result of ray tracing with CUDA ray tracing (Nagata triangular interpolation)	52
5-17	Result of ray tracing with CUDA ray tracing (Nagata quadrilateral interpolation)	52
5-18	Comparison of headlamp simulation of OpticStudio and CUDA ray tracing at x=0[deg]	53
5-19	Comparison of headlamp simulation of OpticStudio and CUDA ray tracing at x=2.5[deg]	53
5-20	Comparison of headlamp simulation of SPEOS and CUDA ray tracing at x=0[deg]	54
5-21	Comparison of headlamp simulation of SPEOS and CUDA ray tracing at x=2.5[deg]	54

5-22 Comparison of calculation speed of each interpolation method for 250,000 rays and a single surface and a single point cloud surface 55
5-23 Calculation speed for changing number of input rays (Double precision Nagata triangular patch interpolation) 56

LIST OF TABLE

		Page
3-1	Radiometric units and photometric units	30
5-1	Comparison of calculation accuracy for ray position (Spherical lens)	42
5-2	Comparison of calculation accuracy for ray direction (Spherical lens)	43
5-3	Comparison of calculation accuracy for ray position (Aspherical lens)	48
5-4	Comparison of calculation accuracy for ray direction (Aspherical lens)	48

ABSTRACT

High speed ray tracing for a headlamp lens and advanced algorithms for ray analysis are investigated.

First, the basics of ray tracing, Algorithm to search intersection points between a ray and surfaces and refraction are reviewed, including intersection search for a ray with aspheric surfaces. A spherical surface, a plane surface, and a point cloud are reviewed as objects. Snell's law is introduced from Fermat's principle in 2D. Then, it extended to three dimensional spaces.

Second, photometry is reviewed for the post processing of ray tracing, due to the convolution effect of its area.

To accelerate ray tracing, the Nvidia GPU and CUDA platform of general purpose computing is evaluated in this study. Its architecture and memory architecture is unique. In addition, Mathematica is used in this study for file IO and graphic output with unique CUDA interface.

Then, the each ray tracing method is validated using a spherical lens, aspherical lens, and a headlamp lens. From the comparison, the double precision floating Nagata triangular patch method is best in accuracy. Acceleration of ray tracing using CUDA was successful having 2 times implement in 362 million rays traced, compared to commercially available ray trace packages under the same computing resources.

CHAPTER 1: INTRODUCTION

1.1 Ray Tracing

Ray tracing is the most basic operation of geometrical optics. In the Oxford dictionary, the meaning of ray tracing is described as ”The calculation of the path taken by a ray of light through an optical system such as a lens or a telescope; an instance of this” [8]. In optics, there are several computational light models. One is physical optics which deals with light as a wave. Another is geometrical optics. Third is quantum optics based approach. Geometrical optics deals with light as rays. Rays propagate straight through homogeneous media and are bent by optical elements, lenses and mirrors and such large macroscopic structure, wave optics and quantum nature of light is ignored, thus, ray tracing is the most important and basic calculation for optics.

1.2 Headlamp for Automotive

Nowadays, automobile headlamps employs complex illumination optics. The Ford model T had headlamps even though it was developed over 100 years ago. In recent years, automobiles use projector-type headlamps which can produce flexible yet accurate illumination distribution. Such projector-type lenses for headlamp have tiny features on its exit surface to diverge a part of rays as seen in figure 1-1. Moreover, the headlamp light distribution is regulated by law in each country. Figure 1-2 shows one of the light distribution patterns made from regulation in Federal Motor Vehicle Safety Standard (FMVSS) which is regulated and controlled by National Highway Traffic Safety Administration for automobiles in the United States [10]. Plots of minimum indicate lower limit of intensity in candela. Plots of maximum indicate upper limit of intensity in candela. The top half is about 100× smaller than the bottom half, and these are separated by a transition zone. In addition, the regulation also specifies a slope of the intensity change over the transition region. Therefore, to achieve accurate illumination distribution complying the regulation, a precise optical design is required which is manufactured

to tight tolerances. This design requires many rays tracing and thus consumes time. Reducing calculation time consquently time to deliver product is essential to reduce cost. Therefore, the objective for this study is achieving high-speed ray tracing for headlamp illumination.

Figure 1-1. Projector-type headlamp lens

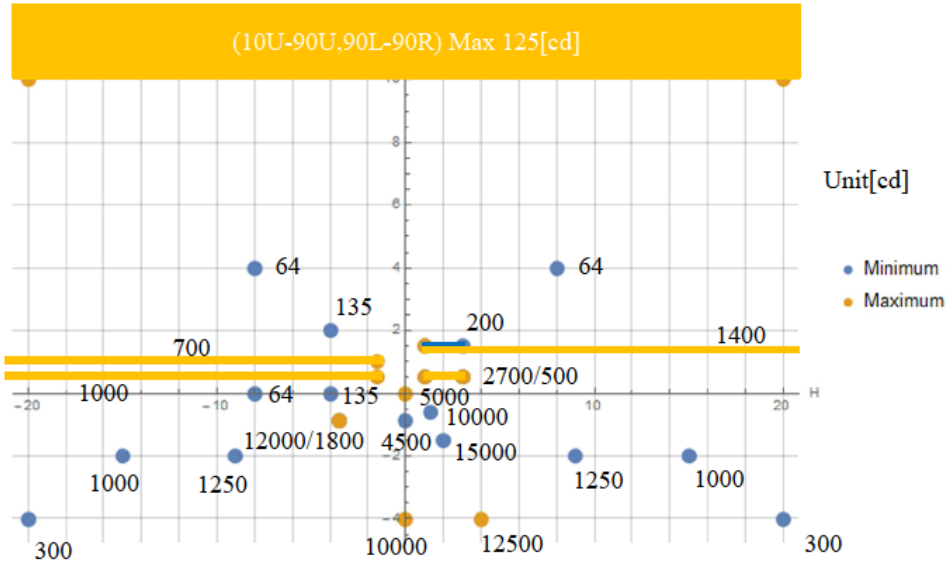


Figure 1-2. Light distribution pattern requirements in FMVSS [10]

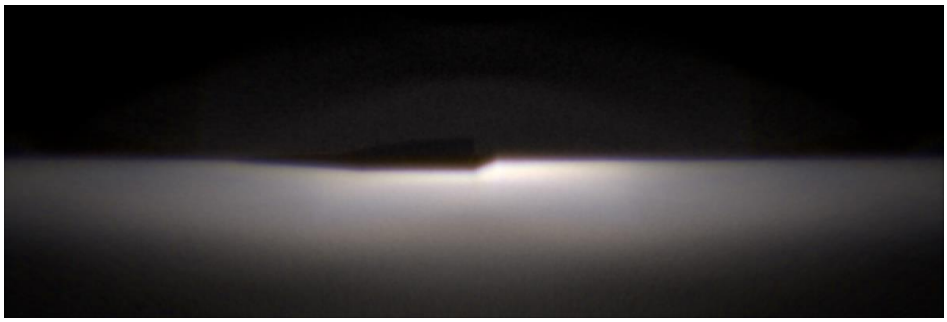


Figure 1-3. Actual light distribution model of headlamp

1.3 Previous Work

There is a lot of previous work for acceleration of ray tracing however many of them are ray tracing for rendering images. There is a gap between ray tracing for rendering images and ray tracing for optical design and simulation because the required accuracy is different. Mauch et al. reported GPU accelerated ray tracing for optical simulation using NVIDIA OptiX which is general ray tracing engine [1]. OptiX is mainly for rendering a photorealistic image. Thus, it uses single precision floating point variables in its calculation. Mauch notes that "This precision is commonly not sufficient for scientific optical simulations, especially if wavefront properties are to be modeled." Therefore, they added double precision floating point variables for ray position and direction in the user defined data structure of OptiX to maintain enough calculation accuracy for optical simulation. Since the GPU has spread to general purpose processing, NVIDIA recognized necessity of incorporating double precision floating point to improve the performance, and accuracy so NVIDIA has supported double precision floating point in their GPU of 2.0.[2] This study uses the full double precision floating point precision CUDA ray tracing is discussed and achieved.

1.4 Outline of Thesis

In this study, we analyze basic ray tracing and the acceleration of calculation speeds using a GPU for optical illumination simulation, to facilitate the tracing of billions of rays using

the automotive headlamp system as the target model. First, we will describe the basics of ray tracing, consisting of intersection point search and refraction calculations. Intersection searching has some variation that is classified by object shape. Plane surfaces are the simplest objects and will be addressed first. Spherical surfaces and complex surfaces will described later. After that, a method for the acceleration of intersection finding is described. Then, refraction calculations are described. Understanding photometry is required for the illumination power calculation. Photometric units and measurement are described. After that, parallel computing using NVIDIA GPU also known as CUDA is described. Then, the result of validation of ray tracing program is described. Finally, summary and future works are described.

CHAPTER 2: RAY TRACING

2.1 Intersection Point Searching

Ray tracing consists of two-parts; searching for the Intersection point between a ray and an object surface, and a refraction calculation at the point. First, intersection searching calculation is described in following. In this study, three-dimensional space coordination is used and described as figure 2-1. In the normal case, rays travels along the +z direction. Vectors are indicated with bold text in sentences and with an overhead arrow in equations.

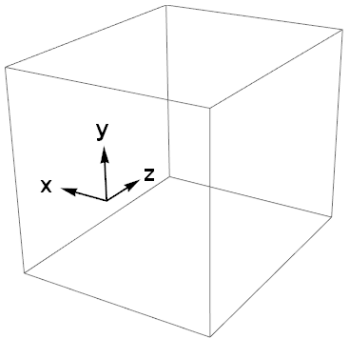


Figure 2-1. Global coodinates for three-dimensional space

2.1.1 Intersection point between a Ray and a Plane Surface

The simplest object for intersection with a ray is a plane surface. The ray in three-dimensional space can be described as,

$$x = x_0 + tk_x, \tag{2.1}$$

$$y = y_0 + tk_y, \tag{2.2}$$

$$z = z_0 + tk_z, \tag{2.3}$$

where $\{x_0, y_0, z_0\}$ is the initial position of the ray. t is distance from $\{x_0, y_0, z_0\}$. $\{k_x, k_y, k_z\}$ is a unit directional vector. The plane surface that contains a point {a, b, c} can be described as,

$$n_x(x - a) + n_y(y - b) + n_z(z - c) = 0, \tag{2.4}$$

where $\{n_x, n_y, n_z\}$ is a normal vector of the surface. Equation (2.1), (2.2), and (2.3) can be substituted into equation (2.4) and solved for t,

$$t = \frac{-n_x x_0 + n_x a - n_y y_0 + n_y b - n_z z_0 + n_z c}{n_x k_x + n_y k_y + n_z k_z}. \tag{2.5}$$

From t, the intersection point {x, y, z} can be derived by substitution of t into equation (2.1), (2.2), and (2.3).

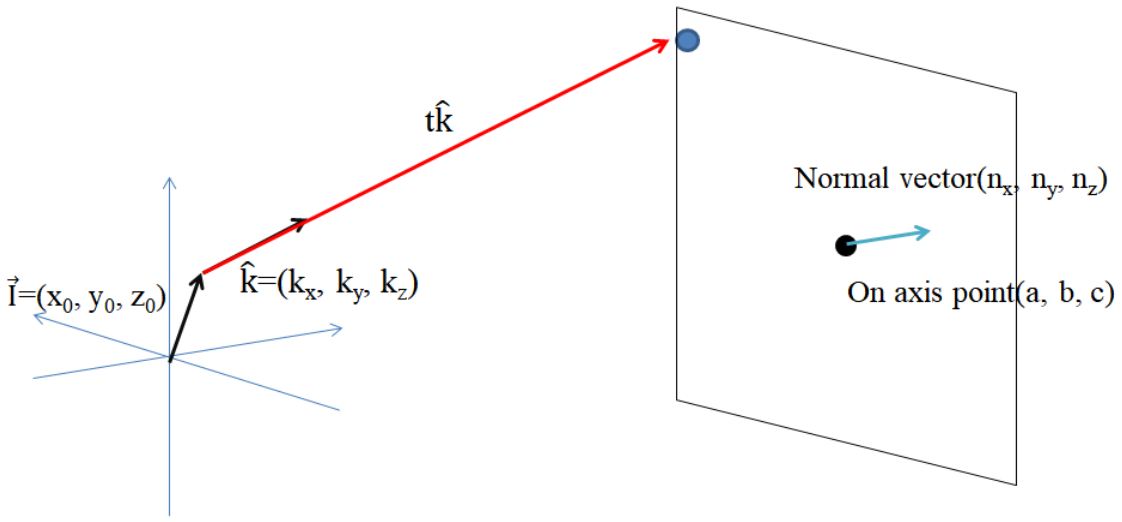


Figure 2-2.Intersection point between a ray and a plane surface

2.1.2 Intersection point between a Ray and a Spherical Surface

The intersection between a ray and a spherical surface can be determined as follows. **p** is vector from the center of the sphere to the surface of the sphere,

$$|\vec{p}| = r, \tag{2.6}$$

where r is the radius of the spherical surface. Furthermore,

$$|\vec{p}| = -\vec{s} + \vec{I} + t\vec{k}, \tag{2.7}$$

where **s** is a vector from the origin of the space to the origin of the sphere and **I** is an initial position of the ray, t is a constant, and **k** is a directional vector of the ray. Substitute equation (2.6) into equation (2.7) and take square of both sides,

$$t^2\left(\left|\vec{k}\right|^2\right) + t\left(-2\vec{s}\cdot\vec{k} + 2\vec{k}\cdot\vec{I}\right) + \left(|\vec{s}|^2 - 2\vec{s}\cdot\vec{I} + \left|\vec{I}\right|^2 - r^2\right) = 0. \tag{2.8}$$

Since equation (2.8) is a quadratic function of t,

$$\mathrm{t} = \frac{-\left(-2\vec{s}\cdot\vec{k} + 2\vec{k}\cdot\vec{I}\right) \pm \sqrt{\left(-2\vec{s}\cdot\vec{k} + 2\vec{k}\cdot\vec{I}\right)^2 - 4\left(\left|\vec{k}\right|^2\right)\left(|\vec{s}|^2 - 2\vec{s}\cdot\vec{I} + \left|\vec{I}\right|^2 - r^2\right)}}{2\left(\left|\vec{k}\right|^2\right)}. \tag{2.9}$$

The equation indicates that t has two answers. The smaller t is for the closer intersection points which are on the front surface of the sphere, and the larger is for farther intersection on the back surface of the sphere. If t is an imaginary number, it indicates that the ray does not intersect with the spherical surface but misses.

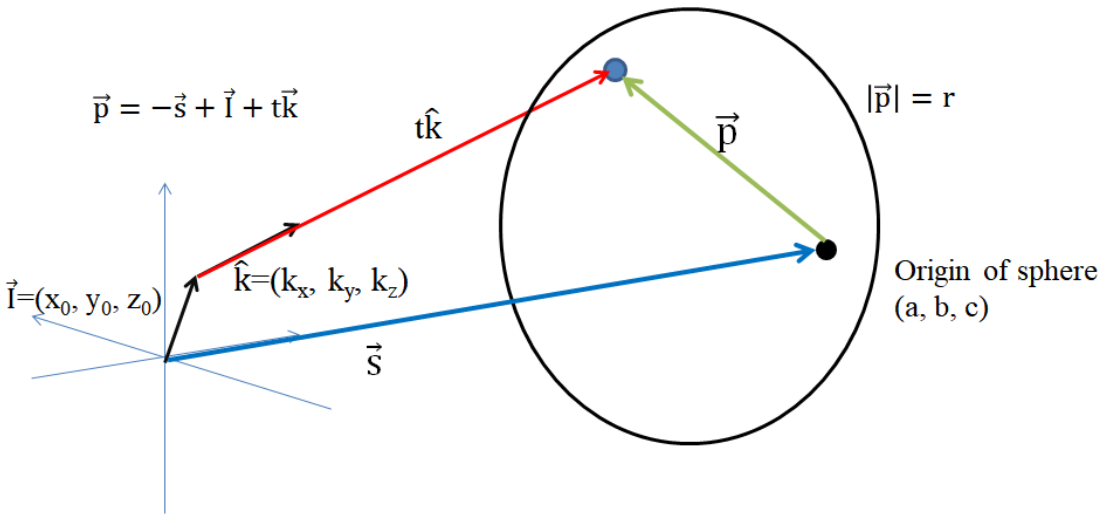


Figure 2-3.Intersection point between a ray and a spherical surface

2.1.3 Intersection point between a Ray and a Complex surface

Actual surface models in ray tracing are often more complex surfaces than plane surfaces and spherical surfaces. In this study, point clouds are used to express complex shaped surfaces because these are often used to communicate complex shapes from CAD programs. Each point has a three-dimensional coordinates and an associated normal vector. The points are addressed as an array and express the complex shaped surfaces. To find the Intersection point between a ray and the surface, **p** which is a vector from the initial point to a point in the cloud is compared with a propagation vector **k**. If the **p** points the same direction with **k,** the ray hit the point. Mathematically, evaluation value f can be calculated by cross product as,

$$f = \frac{\overrightarrow{p_{mn}} \times \hat{k}}{|\overrightarrow{p_{mn}}|} = \sin\theta_{p_{mn}k}\,, \tag{2.10}$$

where $\mathbf{p}_{mn}$ is the vector from the initial point to the point located at m^{th} in horizontal and n^{th} in vertical in the point cloud. θ_{pmnk} is the angle between $\mathbf{p}_{mn}$ and **k**. In the simplest algorithm, the evaluation values for all points in the point cloud are evaluated with a brute force attack. The point which has the smallest evaluation value is the nearest point from the Intersection point between the ray and the surface. When the evaluation value is not zero, interpolation using points in the nearby points are required. A variety of interpolation methods are described in the following section. Since this evaluation method requires a calculation for all points, much unnecessary calculation is performed. If a point cloud consists of 100 by 100 points, the calculation of the nearest point requires 10,000 loops.

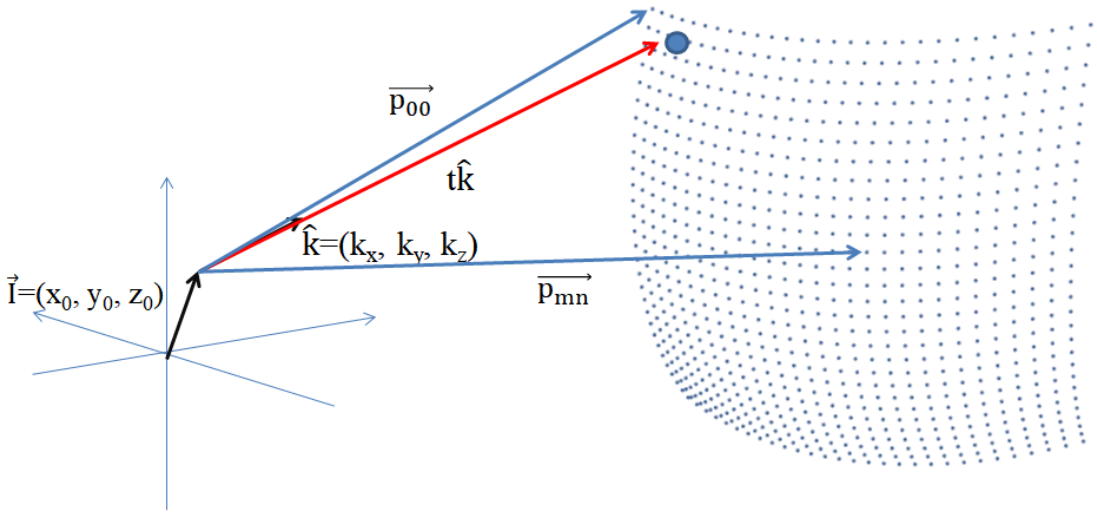


Figure 2-4. Intersection point between a ray and point cloud, a set of points lying on an optical surface, with an associated grid or surface normals

2.1.4 Accelerated Searching Algorithm for Intersection point between a Ray and a Complex Surface

In order to accelerate calculation speed to find the nearest point, many algorithms have been invented. The k-dimensional tree, called k-d tree is one of the major nearest neighbor algorithms. The k-d tree is a space-dividing algorithm. In each step, space is divided at its median value of points. Figure 2-5 shows an example of a k-d tree for one dimension. In this study, we use a two-dimensional evenly spaced grid points which form a point cloud. Thus, space is divided at the mid point in x and y-direction for each step. By applying this algorithm, the nearest candidate points are decreased by 25% in each step. When a point cloud consists of 100 by 100 points, only seven loops are needed to determine the nearest point. Since space is divided by half in each step, the required loop number can be estimated as,

$$2^Q = M, \tag{2.11}$$

where, Q is required loop number and M is the number of the points in a cloud. The equation solves,

$$Q = \frac{\log_{10} M}{\log_{10} 2}, \tag{2.12}$$

Therefore, calculation time is reduced dramatically over the brute force algorithm which is described in the previous section.

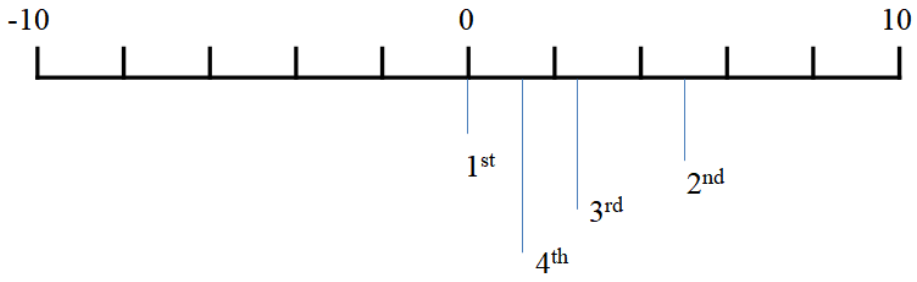


Figure 2-5. k-d tree example

2.1.5 Introducing Nagata Patch for Triangular Patches to Interpolate Quadratically

Nagata introduced a simple quadratic interpolation algorithm from vicinity points and normal vectors which is known as the Nagata patch. [12] In this paper, a key result from that paper is described. Figure 2-6 shows interpolation of a triangular patch using the normal vectors in his paper. Locations of these vicinity points are described with 3D position vector $\mathbf{X}_{00}$, $\mathbf{X}_{10}$, and $\mathbf{X}_{11}$. Normal vectors at each point are described as $\mathbf{n}_{00}$, $\mathbf{n}_{10}$, and $\mathbf{n}_{11}$. η and ξ are parameters in the interpolation equation and indicate the normalized position along the vectors from $\mathbf{X}_{00}$ to $\mathbf{X}_{10}$ and from $\mathbf{X}_{10}$ to $\mathbf{X}_{11}$. The quadratic interpolated point is described as,

$$\vec{X}(\eta,\xi) = \vec{X}_{00}(1-\eta) + \vec{X}_{10}(\eta-\xi) + \vec{X}_{11}\xi - \vec{c}_1(1-\eta)(\eta-\xi) - \vec{c}_2(\eta-\xi)\xi - \vec{c}_3(1-\eta)\xi\ . \tag{2.13}$$

$\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ are curvature parameters and defined as,

$$\vec{c}_1 \equiv \vec{c}(\vec{d}_1, \vec{n}_{00}, \vec{n}_{10})\ , \tag{2.14}$$

$$\vec{c}_2 \equiv \vec{c}(\vec{d}_2, \vec{n}_{10}, \vec{n}_{11})\ , \tag{2.15}$$

$$\vec{c}_3 \equiv \vec{c}(\vec{d}_3, \vec{n}_{00}, \vec{n}_{11}) \quad . \tag{2.16}$$

where, **c** is function and defined as,

$$\vec{c}_m(\vec{d}_m, \vec{n}_a, \vec{n}_b) = \begin{cases} \frac{\Delta d_m}{1-\Delta c}\vec{v} + \frac{d}{\Delta c}\Delta\vec{v} & (c \neq \pm 1) \\ \{0,0,0\} & (c = \pm 1). \end{cases} \tag{2.17}$$

where ν is an average of the normal vectors and Δν is deviation of the normal vectors.

$$\vec{v}_{ab} = (\vec{n}_a + \vec{n}_b)/2 \ , \tag{2.18}$$

$$\Delta\vec{v}_{ab} = (\vec{n}_a - \vec{n}_b)/2 \ . \tag{2.19}$$

d and Δ are defined as,

$$d_m = \vec{d}_m{}^T \vec{v}_{ab} \ , \tag{2.20}$$

$$\Delta d_m = \vec{d}_m{}^T \Delta\vec{v}_{ab} \ . \tag{2.21}$$

Each **d** is defined as,

$$\vec{d}_1 = \vec{X}_{10} - \vec{X}_{00} \ , \tag{2.22}$$

$$\vec{d}_2 = \vec{X}_{11} - \vec{X}_{10} \ , \tag{2.23}$$

$$\vec{d}_3 = \vec{X}_{11} - \vec{X}_{00} \ . \tag{2.24}$$

c and Δc are defined as

$$c_{ab} = 1 - 2\Delta c_{ab} \ , \tag{2.25}$$

$$\Delta c_{ab} = \vec{n}_a{}^T \Delta\vec{v}_{ab} \ . \tag{2.26}$$

In addition, the normal vectors at the interpolated point can be calculated by taking partial derivative of **X** respect to η and ξ as,

$$\vec{X}_\eta = \frac{\partial\vec{X}}{\partial\eta} = \vec{d}_1 + \vec{c}_1\{(\eta - \xi) - (1 - \eta)\} + (\vec{c}_3 - \vec{c}_2)\xi \ , \tag{2.27}$$

$$\vec{X}_\xi = \frac{\partial\vec{X}}{\partial\xi} = \vec{d}_2 + \vec{c}_2\{\xi - (\eta - \xi)\} + (\vec{c}_1 - \vec{c}_3)(1 - \eta) \ . \tag{2.28}$$

The normal vector at η and ξ are calculated as,

$$\vec{n}(\eta, \xi) = \frac{\partial\vec{X}}{\partial\eta} \times \frac{\partial\vec{X}}{\partial\xi} \ , \tag{2.29}$$

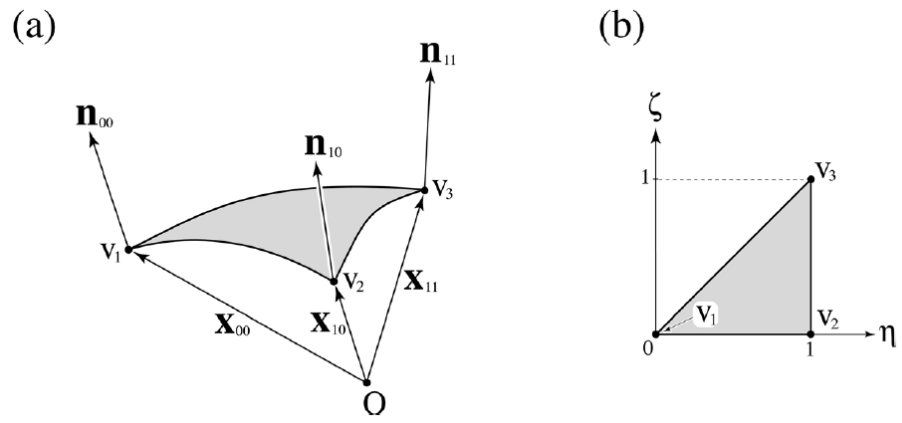


Figure 2-6. Triangular patch [12]

2.1.6 Introducing Nagata Patch for Quadrilateral Patch to Interpolate Quadratically

Nagata also introduced quadratic interpolation for quadrilateral patches. [12] Figure 2-7 shows interpolation of a quadrilateral patch using the normal vectors in his paper. The notation is similar to the previous triangular patch. Quadratic interpolation for the quadrilateral patch is described as,

$$\vec{X}(\eta,\xi) = \vec{X}_{00} + (\vec{d}_1 - \vec{c}_1)\eta + (\vec{d}_4 - \vec{c}_4)\xi + (\vec{d}_2 - \vec{d}_4 + \vec{c}_1 - \vec{c}_2 - \vec{c}_3 + \vec{c}_4)\eta\xi + \vec{c}_1\eta^2 + \vec{c}_4\xi^2 + (\vec{c}_3 - \vec{c}_1)\eta^2\xi + (\vec{c}_2 - \vec{c}_4)\eta\xi^2 \ . \quad (2.30)$$

Each **d** is defined as,

$$\vec{d}_1 = \vec{X}_{10} - \vec{X}_{00} \ , \quad (2.31)$$

$$\vec{d}_2 = \vec{X}_{11} - \vec{X}_{10} \ , \quad (2.32)$$

$$\vec{d}_3 = \vec{X}_{11} - \vec{X}_{01} \ . \quad (2.33)$$

$$\vec{d}_4 = \vec{X}_{01} - \vec{X}_{00} \ . \quad (2.34)$$

The partial derivative of X with respect to η and ξ are described as

$$\vec{X}_\eta = \frac{\partial \vec{x}}{\partial \eta} = (\vec{d}_1 - \vec{c}_1) + (\vec{d}_2 - \vec{d}_4 + \vec{c}_1 - \vec{c}_2 - \vec{c}_3 + \vec{c}_4)\xi + 2\vec{c}_1\eta + \vec{c}_4\xi^2 + 2(\vec{c}_3 - \vec{c}_1)\eta\xi + (\vec{c}_2 - \vec{c}_4)\xi^2 \ , \quad (2.35)$$

$$\vec{X}_\xi = \frac{\partial \vec{X}}{\partial \xi} = (\vec{d}_4 - \vec{c}_4) + (\vec{d}_2 - \vec{d}_4 + \vec{c}_1 - \vec{c}_2 - \vec{c}_3 + \vec{c}_4)\eta + 2\vec{c}_4\xi + (\vec{c}_3 - \vec{c}_1)\eta^2 + 2(\vec{c}_2 - \vec{c}_4)\eta\xi \,. \tag{2.36}$$

The normal vector at η and ξ are calculated as,

$$\vec{n}(\eta, \xi) = \frac{\partial \vec{X}}{\partial \eta} \times \frac{\partial \vec{X}}{\partial \xi} \,. \tag{2.37}$$

In this study, the point clouds consist of linear spaced grid of points. Therefore, the quadrilateral patch is appropriate. Both triangular patch and quadrilateral patch interpolation methods were tried and compared.

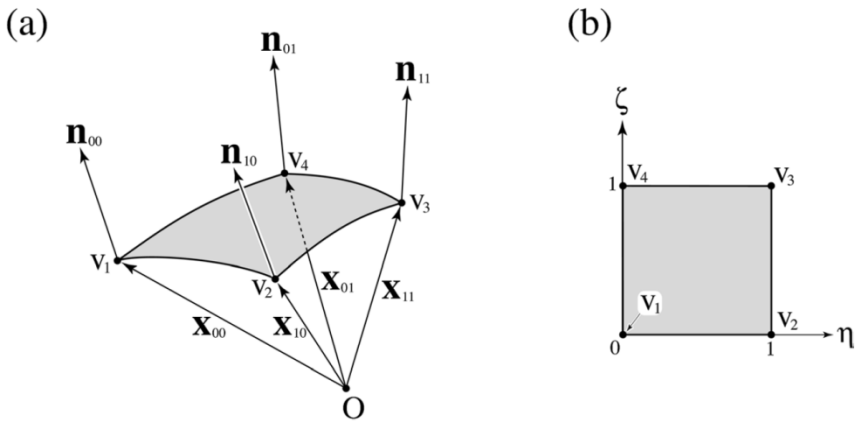


Figure 2-7. Quadrilateral patch [12]

2.2 Refraction

The second step in ray tracing calculation is refraction, the bending of rays at the boundaries of surfaces. Refraction depends on the angle of incidence and the materials before and after the surface. In this section, refraction principle is described.

2.2.1 Fermat's Principle and Snell's Law

Fermat's principle describes why rays travel straight and are bent at a surface boundary. Dereniak & Dereniak (2008) states that "Fermat's principle states that light rays of given frequency traverse the path between tow given points in the least amount of time" [11]. To solve mathematically, a path of the ray from point A to point B is considered as figure 2-8.

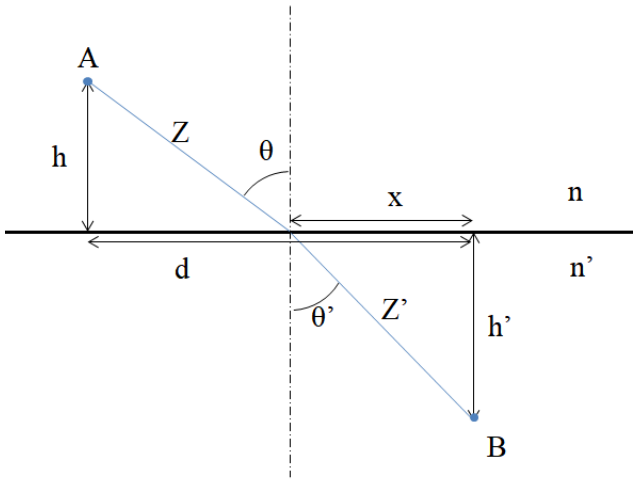


Figure 2-8. Light path for Fermat's principle

Total time from point A to point B is the sum of the distance divided by velocity.

$$\tau_{\text{total}} = \frac{\text{Z}}{v_n} + \frac{\text{Z}}{v_{n\prime}}. \tag{2.38}$$

The velocity of light in medium whose refractive index is n and n' is

$$v_{\text{n}} = \frac{c}{\text{n}} \quad , \quad v_{\text{n}'} = \frac{c}{\text{n}'} \, , \tag{2.39}$$

where, c is the speed of light in vacuum space.

Hence total time is

$$\tau_{\text{total}} = \frac{\text{Zn}}{c} + \frac{\text{Zn}'}{c}. \tag{2.40}$$

Since optical path length (OPL) from A to B is equivalent to velocity times time

$$\text{OPL} = c\tau_{\text{total}} = \text{Zn} + \text{Zn}'. \tag{2.41}$$

From figure 8, using Pythagoras's theorem,

$$\text{Z}^2 = \text{h}^2 + (\text{d} - \text{x})^2. \tag{2.42}$$

$$\text{Z}'^2 = \text{h}'^2 + x^2. \tag{2.43}$$

Substituting (2.42) and (2.43) into (2.41),

$$\mathrm{OPL} = \sqrt{\mathrm{h}^2 + (\mathrm{d} - \mathrm{x})^2}\,\mathrm{n} + \sqrt{\mathrm{h}'^2 + x^2}\,\mathrm{n}'. \tag{2.44}$$

To find the minimum OPL, set the partial derivative with respect to x equal to zero,

$$\frac{\partial \mathrm{OPL}}{\partial \mathrm{x}} = -\frac{\mathrm{n}(\mathrm{d} - \mathrm{x})}{\sqrt{\mathrm{h}^2 + (\mathrm{d} - \mathrm{x})^2}} + \frac{n'x}{\sqrt{\mathrm{h}'^2 + \mathrm{x}^2}} = 0\,. \tag{2.45}$$

Rearranging (2.45),

$$\mathrm{n}\frac{(\mathrm{d} - \mathrm{x})}{\sqrt{\mathrm{h}^2 + (\mathrm{d} - \mathrm{x})^2}} = \mathrm{n}'\frac{x}{\sqrt{\mathrm{h}'^2 + \mathrm{x}^2}}\,, \tag{2.46}$$

$$\mathrm{n}\sin\theta = \mathrm{n}'\sin\theta'\,, \tag{2.47}$$

Equation (2.47) is called Snell's law and describes how a ray is bent at the boundary.

2.2.2 Snell's Law in Three-Dimensional Space

Objects and rays are defined in three-dimensional space for ray tracing, so Snell's law is expanded into three-dimensional space. In figure 2-9, an incident ray unit vector is shown as **i**. A unit refracted exiting ray vector is shown as **r**. **n** shows the unit normal vector at the boundary. **a** and **b** are orthogonal unit vectors with **n** and orthogonal each other. From trigonometry,

$$\hat{\mathrm{r}} = \hat{\mathrm{a}}\sin\theta' - \hat{\mathrm{n}}\cos\theta'. \tag{2.48}$$

Since a, b, and c are orthogonal,

$$\begin{aligned}\hat{\mathrm{a}} = \hat{\mathrm{n}} \times \hat{\mathrm{b}} = \hat{\mathrm{n}} \times \left(\frac{\vec{\mathrm{i}} \times \hat{\mathrm{n}}}{\sin\theta}\right) &= \frac{1}{\sin\theta}\left\{(\hat{\mathrm{n}} \cdot \hat{\mathrm{n}})\hat{\mathrm{i}} - (\hat{\mathrm{n}} \cdot \hat{\mathrm{i}})\hat{\mathrm{n}}\right\} \\ &= \frac{1}{\sin\theta}\left\{\hat{\mathrm{i}} - (\hat{\mathrm{n}} \cdot \hat{\mathrm{i}})\hat{\mathrm{n}}\right\}.\end{aligned} \tag{2.49}$$

Substitute Snell's law (2.47) into (2.49),

$$\hat{\mathrm{a}} = \frac{\mathrm{n}}{\mathrm{n}'\sin\theta'}\left\{\hat{\mathrm{i}} - (\hat{\mathrm{n}} \cdot \hat{\mathrm{i}})\hat{\mathrm{n}}\right\}. \tag{2.50}$$

From trigonometry,

$$\cos\theta = -\hat{\mathrm{i}} \cdot \hat{\mathrm{n}}\ . \tag{2.51}$$

From Pythagoras's identity,

$$\sin^2\theta = 1 - \cos^2\theta = 1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2 \quad . \tag{2.52}$$

Substitute Snell's law (2.47) into (2.52),

$$\left(\frac{n'}{n}\right)^2 \sin^2\theta' = 1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2 \quad . \tag{2.53}$$

$$\sin^2\theta' = \left(\frac{n}{n'}\right)^2 \left\{1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2\right\} . \tag{2.54}$$

From Pythagoras's identity,

$$\cos^2\theta' = 1 - \left(\frac{n}{n'}\right)^2 \left\{1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2\right\}, \tag{2.55}$$

$$\cos\theta' = \sqrt{1 - \left(\frac{n}{n'}\right)^2 \left\{1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2\right\}}. \tag{2.56}$$

Substitute (2.50) and (2.56)into (2.48),

$$\hat{r} = \frac{n}{n'\sin\theta'}\{\hat{\imath} - (\hat{n}\cdot\hat{\imath})\hat{n}\}\sin\theta' - \hat{n}\sqrt{1 - \left(\frac{n}{n'}\right)^2 \left\{1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2\right\}} \tag{2.57}$$

$$= \frac{n}{n'}\{\hat{\imath} - (\hat{n}\cdot\hat{\imath})\hat{n}\} - \hat{n}\sqrt{1 - \left(\frac{n}{n'}\right)^2 \left\{1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2\right\}}$$

$$= \frac{n}{n'}\hat{\imath} - \frac{n}{n'}\left[(\hat{n}\cdot\hat{\imath}) + \sqrt{\left(\frac{n'}{n}\right)^2 - \left\{1 - \left(-\hat{\imath}\cdot\hat{n}\right)^2\right\}}\right]\hat{n}$$

$$= \frac{n}{n'}\hat{\imath} - \frac{n}{n'}\left[(\hat{n}\cdot\hat{\imath}) + \sqrt{\left(\frac{n'}{n}\right)^2 - 1 + (\hat{n}\cdot\hat{\imath})^2}\right]\hat{n}$$

Now, refracted vector **r** is described with refractive indices, the incident vector, and the surface normal vector. When the square root in equation (2.57) is negative, the ray cannot refract at the boundary, so the ray is reflected.

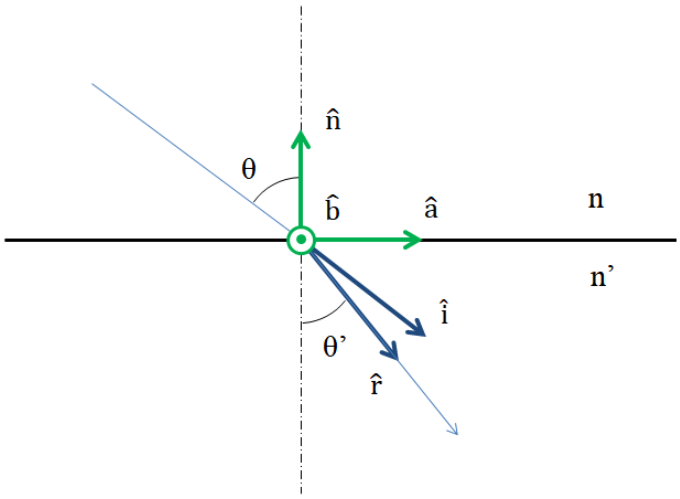


Figure 2-9. Snell's law in three-dimensional spaces

2.2.3 Refractive Index

Refractive index is a parameter of refraction and defines how rays are slow down in medium. The refractive index n is given by

$$\mathrm{n} = \frac{c}{v}, \tag{2.58}$$

where, c is the speed of light in vacuum and v is the speed of light in a medium. Since light is an electromagnetic wave, its propagation speed differs when in a vacuum compared to when it propagates in a medium. Thus, the refractive index depends on the material of the medium and wavelength of light. The refractive index differs for different wavelengths, this effect is called dispersion. There are some calculation models of dispersion such as the Sellmeier equation and Cauchy's equation [4]. The Sellmeier equation is more accurate than Cauchy's equation over wider wavelength ranges. For example, Schott AG, a manufacturer of optical glass materials and Zemax OpticStudio, a major optical design and simulation software package, both use the Sellmeier equation. Therefore, the Sellmeier equation is used for calculation of dispersion in this study as well. The Sellmeier equation is described as,

$$n^2(\lambda) - 1 = \frac{B \cdot \lambda^2}{\lambda^2 - C} \quad . \tag{2.59}$$

Since this refractive index equation is simple, it can be easily calculated inside of a ray tracing program.

CHAPTER 3: PHOTOMETRY

Photometry means measuring light in accordance with the human eye response. For illumination optics, photometric units are usually used because the application typically relates to assisting human observation.

3.1 Photometrical Units

Since the photometry units are based on the human eye response, its units are different from radiometric units. Table 3-1 shows a comparison of the radiometric units and photometric units. Photometric units can be converted from radiometric units using the human eye response function. The human eye response is defined by the International Commission on Illumination (CIE). Figure 3-1 shows the photopic luminous efficiency function in CIE 1931. By multiplying luminous efficiency functions to radiometric measurement, a photometric measurement is calculated. As described in figure 1-1, an illumination distribution is described in luminous intensity whose unit is the candela [cd]. Since the LED output is usually described in radiometric units, it must be converted in the optical simulation.

Table 3-1. Radiometric units and photometric units

	Radiometric unit	Photometric unit
Energy Q	J	T
Flux Φ	J/s, W	lm
Exitance M	W/m^2	$lx = lm/m^2$
Irradiance/ Illuminance E	W/m^2	$lx= lm/m^2$
Radiant intensity/ Luminous intensity I	W/sr	cd = lm/sr

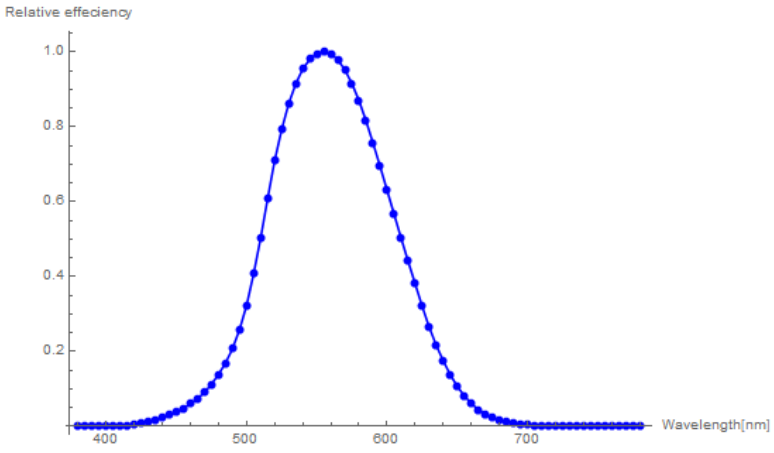


Figure 3-1. Photopic luminous efficiency function K(λ) (CIE1931)

3.2 Photometer

A photometer is used to measure luminous intensity. The National Highway Trafic Safety Administration describes standard measurement setup of headlamp illumination distribution for FMVSS 108 in figure 3-2.[3] The measurement equipment has two goniometer stages that change the azimuth angle and polar angle in order to change the angle of lamp vertically and horizontally. Then, the photometer which is placed at a certain distance, receives light from the lamp and quantizes the flux of the light. Figure 3-3 shows a photometer with its

detection are noted. From the size of the detection area and the distance between the lamp and the detectors, the solid angle Ω is calculated as,

$$\int d\Omega = \iint \sin\theta\cos\theta d\theta d\phi \,, \tag{3.1}$$

where θ is the angle of the cone from the light source to the detector surface. ϕ is the angle of circumference direction of the cone. In this case, ϕ is equal to 2π radians,

$$\Omega = 2\pi \int \sin\theta\cos\theta d\theta = 2\pi \left(-\frac{1}{2}\cos^2\theta \Big|_0^{\theta} \right) = \pi(-\cos^2\theta + 1) \tag{3.2}$$

$$= \pi \sin^2\theta \ \text{[sr]}.$$

Since θ is calculated from r, the detector's radius and d, the distance between the light source and the detector,

$$\Omega = \pi \sin^2(\tan^{-1}\frac{r}{d}) \ \text{[sr]} \,. \tag{3.3}$$

The measured flux at the photometer is divided by the solid angle Ω. Intensity is

$$I = \frac{\Phi}{\Omega} = \frac{\Phi}{\pi \sin^2(\tan^{-1}\frac{r}{d})} \ \text{[W/sr]} \,. \tag{3.4}$$

This is converted to luminous intensity using the luminous efficiency function K figure 3-1,

$$I_l = K\,I = \frac{K\Phi}{\pi \sin^2(\tan^{-1}\frac{r}{d})} \ \text{[lm/sr = cd]} \,. \tag{3.5}$$

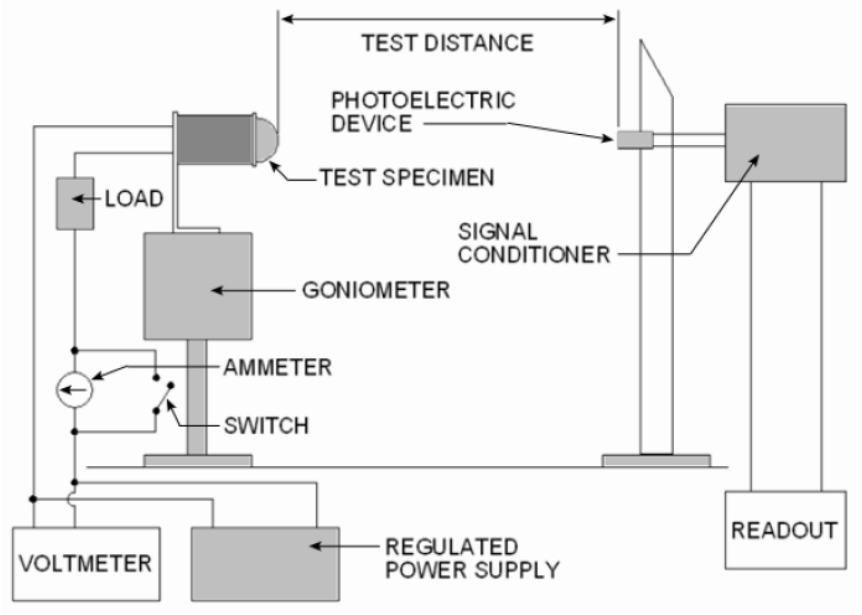


Figure 3-2. Photometry test setup [3]

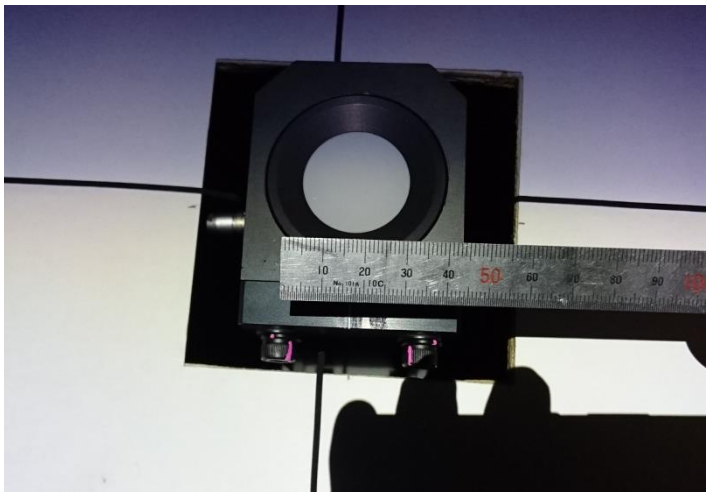


Figure 3-3. Active area of a photometer

3.3 Convolution

Since the photometer has a certain area, πr^2 for a circular area, convolution is required at the detector surface in the simulation. In the optical simulation, an energy value is defined for each ray. The energy of rays which reach the detector surface located at a certain distance from

the lamp are found by dividing the energy of rays hitting the same space with a certain size as a grid. This is called data binning. As a result, the energies in each ray are converted to the energies in each box in the grid. After that, because of the photometer has an area as described, the binning data is processed with 2D convolution to get a simulation result, to simulate the measurement. Figure 3-4 shows the processing model.

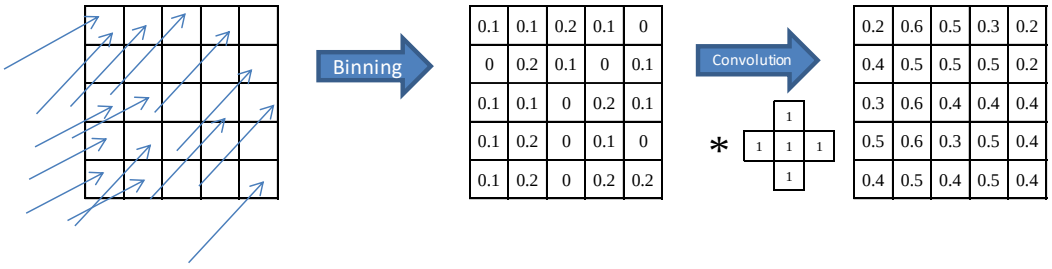


Fig 3-4. 2D Binning and convolution

CHAPTER 4: CUDA PROGRAMING

In this chapter, CUDA, a popular parallel computing method provided by NVIDIA is described.

4.1 GPU Architecture

GPU is an acronym for Graphic Processing Unit. GPUs are circuits developed for image processing in computers. GPUs are more specialized for parallel computing than CPUs, which gives an advantage for image processing, especially when creating images in real-time in computer games. Recently, GPUs are commonly for parallel computing in high-performance computing, such as machine learning, because these applications are a good fit for parallel processing. According to NVIDIA, the GeForce GTX970, one of their mid-range GPUs have 1664 cores, whose base clock is 1050MHz. [13] Even though the speed of each core in a GPU is slower than of a CPU, it has hundreds of times as many cores. As a result, the GPU provides massive processing power compared to a CPU.

4.2 GPU Memory Architecture

GPUs have their own memory that is independent of system memory. Since the standard program uses system memory, data which is needed for processing in a GPU must be copied from system memory to the GPU memory before processing in the GPU. Then, data must be copied from the GPUs memory to the system memory. This account for additional processing times compared to executing program in CPU. Therefore, memory management, especially reducing time for copying data between the GPU memory and system memory is a key consideration when creating a high-speed program. As shown in figure 4-1, a GPU has three types of memories which are independent of the system memory: a global memory, a shared memory, and a local memory. Access permissions for each memory are different. The local memory can only be used inside of a thread. It cannot be accessed from any other threads. The shared memory can be shared with other threads in the same block. The global memory can be shared with other threads in the same GPU. The block and the grid are described in next section.

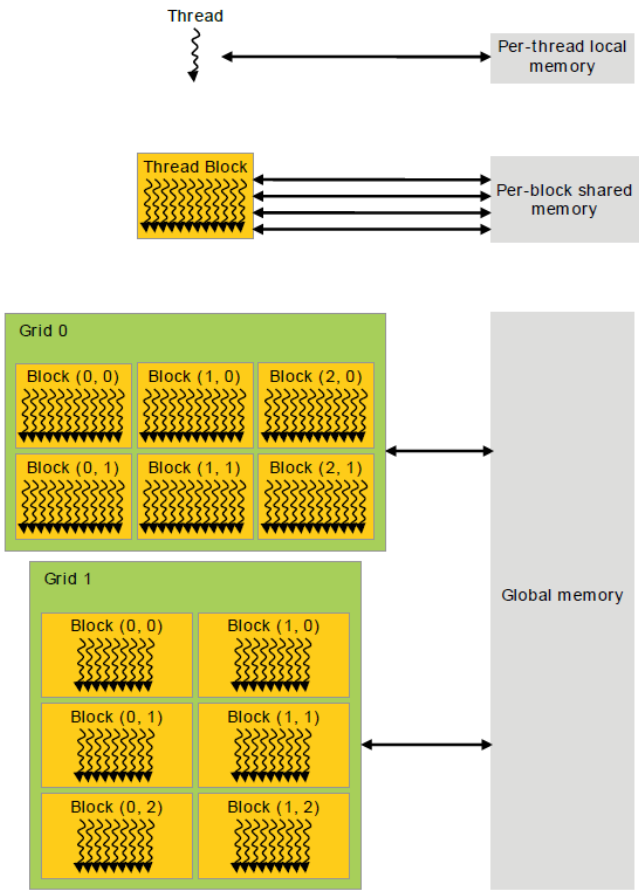


Figure 4-1. Memory hierarchy of CUDA[2]

4.3 CUDA

NVIDIA describes CUDA as “a general purpose parallel computing platform and programming model that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU.”[2] CUDA consists of a driver, a compiler, and APIs. The driver is interface software which connects a device to the system for use in the operating system. The compiler transforms program code into machine code. The CUDA compiler is compatible with the C and C++ and expanded for calculation in GPU. APIs provide program sets which makes it easy to programming effectively. In CUDA, code running on a single core is called a thread. CUDA manages several threads as a group called a block. Several blocks are managed as a grid. The block and the grid can be set in one dimension, two dimensions, or three dimensions when the CUDA program is launched. Its

maximum size depends on the GPU specification. The configuration of block and grid should be considered for effective processing in CUDA. The purpose of CUDA is to provide functions that allow for parallel computing.

4.4 CUDA with Mathematica

In this study, Mathematica is used for programing because it has a variety of input and output interfaces including files, graphics, and CUDA interfaces. Mathematica has a set of memory management functions and interfaces for CUDA called CUDALink. The memory management function makes it easy to copy data from the system memory to the GPU memory. In the C language, data size and data type have to be defined explicitly when the data is copied from a system memory to a GPU memory. However, the memory management function in Mathematica provides data typing automatically. Figure 4-2 shows a comparison of a sample data copying program in C and Mathematica.

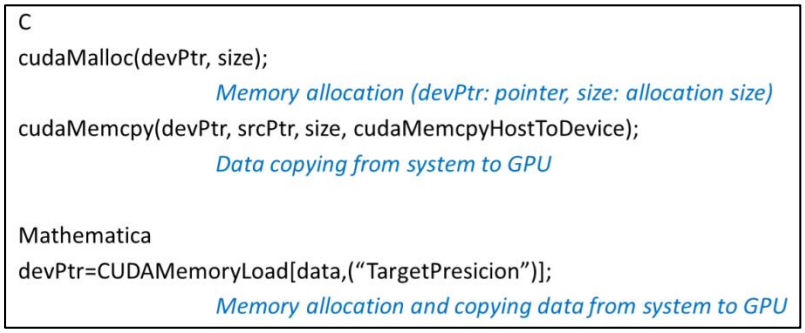


Figure 4-2. Comparison of data copying program in C and Mathematica

In addition, the interface for CUDA provides a way to make a program with CUDA and use it as a function in Mathematica. Because such a CUDA program can be used as a Mathematica function, it is easily associated with other Mathematica functions. Since Mathematica has a large number of built-in functions such as file input and output and 2D or 3D graphics output, CUDA

programing in Mathematica reduces development volume; thus it is easier than CUDA programing in the C language.

CHAPTER 5: VALIDATION TESTS

All tests were done in Microsoft Windows 10 (64 bit) using 16GB of DDR4 RAM, running on Intel Core i5-6500 CPU at 3.2GHz with Wolfram Mathematica 11.0.1.0. A Nvidia GeForce GTX970 whose core clock is 1050MHz and has 1664 CUDA cores with 4GB of GDDR5 RAM is used for CUDA.

5.1 Spherical Lens (Plano-convex lens)

First, ray tracing of a plano-convex spherical lens was done for validation of programs.

5.1.1 Model Analysis with OpticStudio

The plano-convex lens was analyzed by OpticStudio for reference. Figure 5-1 shows the model with a radius of curvature is 25mm and thickness of 3mm. The diameter of the lens is 12mm. The lens is made of PMMA whose refractive index is 1.49 at λ=589.3nm. A focal plane position is optimized for minimization of spot radius located at 48.867mm from the plane surface. Figure 5-2 shows spot diagram of this optical system with a spot size is about 0.18mm.

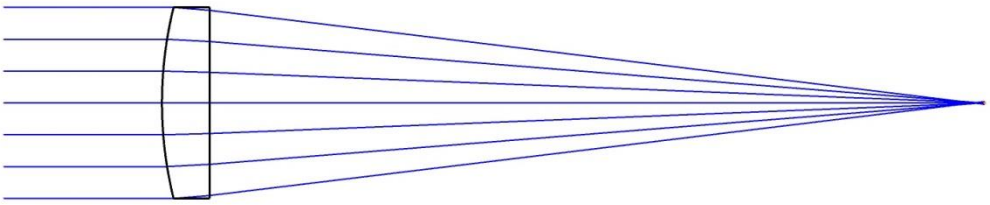


Figure 5-1. Spherical lens test layout

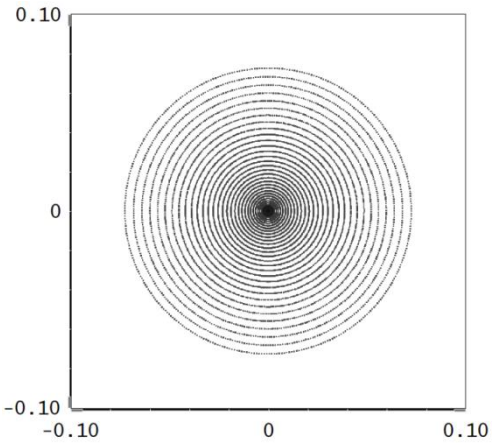
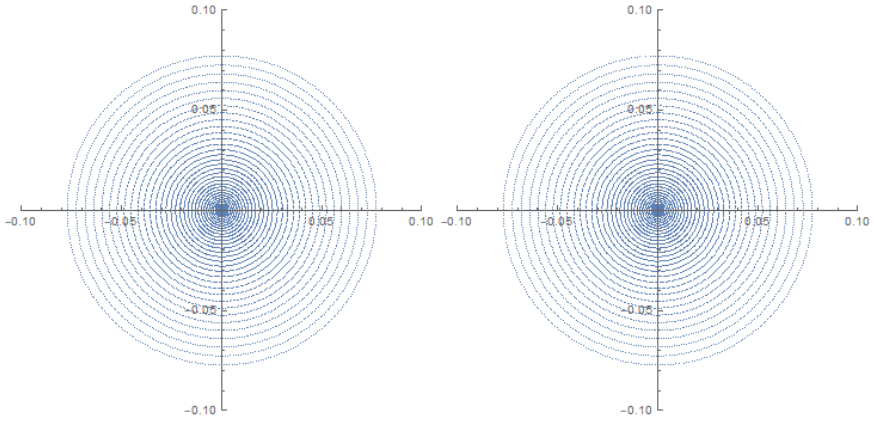


Figure 5-2. Spot diagram of a spherical lens using OpticStudio

5.1.2 CUDA Ray Tracing for a Spherical Surface and a Plane Surfaces

The same plano-convex lens was analyzed by the CUDA ray tracing program. First, intersection search is performed by solving the equation in section 2-1-2 for the Intersection point between a ray and a sphere and in section 2-1-1 for the Intersection point between a ray and a plane. Figure 5-3 shows a spot diagram from the result made with 11,500 rays. The spot size is about 0.18mm and matches with the result from OpticStudio.

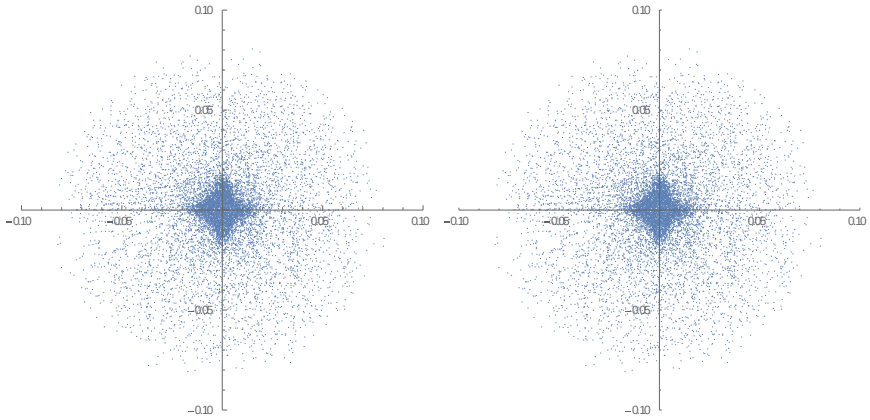


Double precision float Single precision float

Figure 5-3. Spot diagram of a spherical lens using CUDA ray tracing (Non-interpolation)

5.1.3 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surfaces (Linear Interpolation)

The same plano-convex lens was analyzed by CUDA ray tracing again. The spherical surface was represented by a point could of 241 times 241 points for the spherical surface. Plane surfaces were used for the second surface and the focal plane. Figure 5-4 shows spot diagrams of ray tracing result. Even rays in spot diagram are discretized, the spot size is same as reference OpticStudio result.

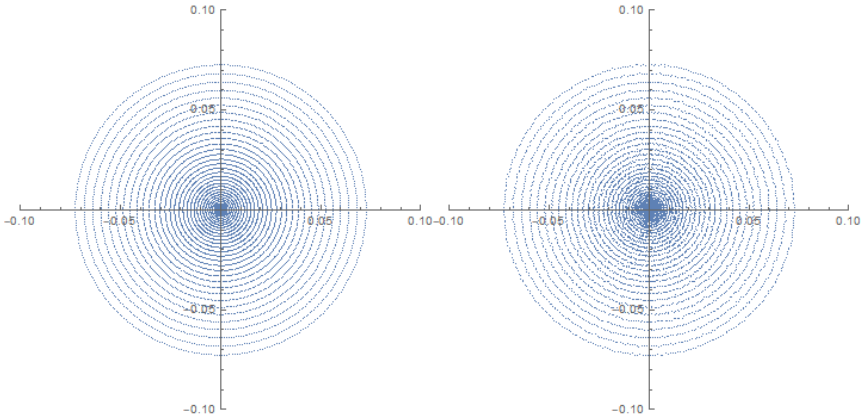


Double precision float Single precision float

Figure 5-4. Spot diagram of a spherical lens using CUDA ray tracing (Linear interpolation)

5.1.4 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Triangular Patch Interpolation)

The same plano-convex lens from the last section was analyzed by CUDA ray tracing with Nagata triangular patches for interpolation of the point cloud. Figure 5-5 shows spot diagrams of the result. It is more similar to the non-interpolation result in figure 5-3. Therefore, there are fewer errors with this interpolation method than in linear interpolation in figure 5-4. In addition, the result of the double precision float is smoother than the result of the single precision float indicating that double precision simulation is more accurate than single precision simulation.

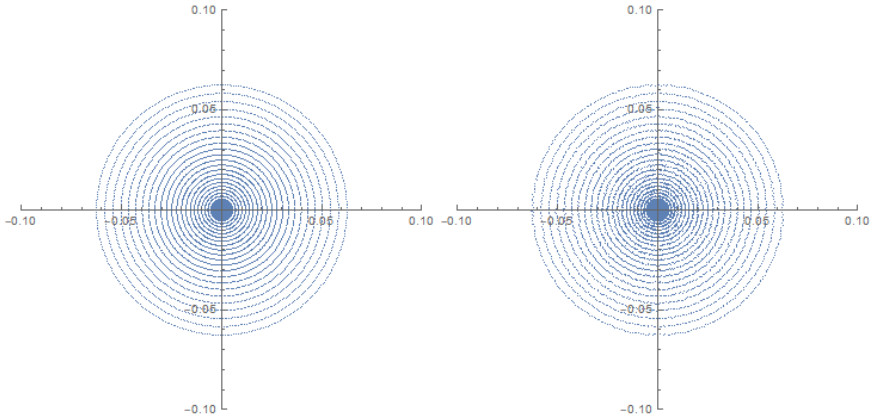


Double precision float Single precision float

Figure 5-5. Spot diagram of a spherical lens using CUDA ray tracing (Nagata triangular patch)

5.1.5 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Quadrilateral Patch Interpolation)

The same plano-convex lens used in the last sections was analyzed by CUDA ray tracing with Nagata quadrilateral patches for interpolation. Figure 5-6 shows a spot diagram of the result but it still does not closely match figure 5.3, the exact result. Rays are more concentrated at the center part, so the spot size is smaller than other interpolation results. Moreover, the result of the double precision float is smoother than the result of the single precision float as it was with the previous tests. This indicates that double precision simulation is more accurate than single precision simulation.



Double precision float Single precision float

Figure 5-6. Spot diagram of a spherical lens using CUDA ray tracing (Nagata quadrilateral patch)

5.1.6 Comparison of Results

To compare the results of each calculation, data for a single ray path data are summarized in table 3-1 and 3-2. Position describes the ray position at each surface in 3D. Direction describes the ray propagation direction as direction cosines for the ray following each surface in 3D. S1 and S2 indicate front and back surface of the lens. Some of the errors can be seen at the focal plane. As expected, the result of double precision interpolation coincides with the result of OpticStudio. Since noninterpolation method is the direct calculation of the Intersection point between a spherical surface and a ray, the error comes from only rounding error. Therefore, double precision noninterpolation calculation should be in the most accurate. This result is from a simulation using spherical surface, so linear interpolation has many errors. Surprisingly, errors in Nagata quadrilateral patch are larger than errors in Nagata triangular patch even though points in the point cloud are allocated as a quadrilateral grid. This comparison shows that double precision noninterpolation method and double precision Nagata triangular patch are a good method for ray tracing for our program.

Table 5-1. Comparison of calculation accuracy for ray position (Spherical lens)

		Initial positon	S1 (Spherical)	S2 (Plane)	Focal plane
OpticStudio	x	5.350840	5.350840	5.178050	-0.059388
	y	-0.726987	-0.726987	-0.703511	0.008069
	z	0.000000	10.590170	13.000000	61.867000
Double Precision Non interpolation	x	5.350840	5.350840	5.178050	-0.059388
	y	-0.726987	-0.726987	-0.703511	0.008069
	z	0.000000	10.590166	13.000000	61.867000
Single Precision Non interpolation	x	5.350840	5.350840	5.178050	-0.059388
	y	-0.726987	-0.726987	-0.703511	0.008069
	z	0.000000	10.590166	13.000000	61.867001
Double Precision Linear interpolation	x	5.350840	5.350840	5.177566	-0.074684
	y	-0.726987	-0.726987	-0.703525	0.007644
	z	0.000000	10.590180	13.000000	61.867000
Single Precision Linear interpolation	x	5.350840	5.350840	5.177566	-0.074684
	y	-0.726987	-0.726987	-0.703525	0.007644
	z	0.000000	10.590179	13.000000	61.867001
Double Precision Nagat triangular patch interpolation	x	5.350840	5.350699	5.177912	-0.059383
	y	-0.726987	-0.726992	-0.703516	0.008068
	z	0.000000	10.590135	13.000000	61.867000
Single Precision Nagat triangular patch interpolation	x	5.350840	5.350701	5.177920	-0.059168
	y	-0.726987	-0.727030	-0.703556	0.007952
	z	0.000000	10.590137	13.000000	61.867001
Double Precision Nagat quadrilateral patch interpolation	x	5.350840	5.350699	5.178212	-0.049942
	y	-0.726987	-0.726992	-0.703557	0.006786
	z	0.000000	10.590135	13.000000	61.867000
Single Precision Nagat quadrilateral patch interpolation	x	5.350840	5.350698	5.178223	-0.049553
	y	-0.726987	-0.726950	-0.703517	0.006754
	z	0.000000	10.590134	13.000000	61.867001

Table 5-2. Comparison of calculation accuracy for ray direction (Spherical lens)

		Initial positon	S1 (Spherical)	S2 (Plane)	Focal plane
OpticStudio	l	0.000000	-0.071515	-0.106556	-0.106556
	m	0.000000	0.009716	0.014477	0.014477
	n	1.000000	0.997392	0.994201	0.994201
Double Precision Non interpolation	l	0.000000	-0.071515	-0.106556	-0.106556
	m	0.000000	0.009716	0.014477	0.014477
	n	1.000000	0.997392	0.994201	0.994201
Single Precision Non interpolation	l	0.000000	-0.071515	-0.106556	-0.106556
	m	0.000000	0.009716	0.014477	0.014477
	n	1.000000	0.997392	0.994201	0.994201
Double Precision Linear interpolation	l	0.000000	-0.071715	-0.106854	-0.106854
	m	0.000000	0.009710	0.014468	0.014468
	n	1.000000	0.997378	0.994169	0.994169
Single Precision Linear interpolation	l	0.000000	-0.071715	-0.106854	-0.106854
	m	0.000000	0.009710	0.014468	0.014468
	n	1.000000	0.997378	0.994170	0.994169
Double Precision Nagat triangular patch interpolation	l	0.000000	-0.071513	-0.106553	-0.106553
	m	0.000000	0.009716	0.014477	0.014477
	n	1.000000	0.997392	0.994202	0.994202
Single Precision Nagat triangular patch interpolation	l	0.000000	-0.071510	-0.106549	-0.106549
	m	0.000000	0.009715	0.014476	0.014476
	n	1.000000	0.997393	0.994202	0.994202
Double Precision Nagat quadrilateral patch interpolation	l	0.000000	-0.071390	-0.106369	-0.106369
	m	0.000000	0.009700	0.014452	0.014452
	n	1.000000	0.997401	0.994222	0.994222
Single Precision Nagat quadrilateral patch interpolation	l	0.000000	-0.071385	-0.106362	-0.106362
	m	0.000000	0.009699	0.014451	0.014451
	n	1.000000	0.997402	0.994222	0.994222

5.2 Aspherical Lens (Plano-convex lens)

5.2.1 Model Analysis with OpticStudio

Second, an aspherical lens was analyzed for validation of the program. Aspherical lenses reduce aberration compare to spherical lenses. Therefore, the spot size becomes smaller. The most common equation for the sag of an aspherical radically symmetric surfaces are described as a conic plus polynomial terms,

$$\mathrm{sag}(r) = \frac{r^2/R}{1+\sqrt{1-(K+1)\left(\frac{r^2}{R^2}\right)}} + \sum_{m=1}^{M} \alpha_{2m} r^{2m}. \tag{3.6}$$

where, K is conic constant. R is the base radius of curvature. Figure 5-6 shows the lens model for a lens with a plane surface and asphere. Its radius of base curvature is 12.5 mm. Its thickness is 3.343mm. The diameter of the lens is 12mm. The lens is made of PMMA whose refractive index is 1.49 at λ=589.3nm. The focal plane position is optimized for minimization of spot radius located at 23.168mm from plane surface. Figure 5-7 shows spot diagram of this optical system. From the spot diagram, spot size is less than 0.001mm.

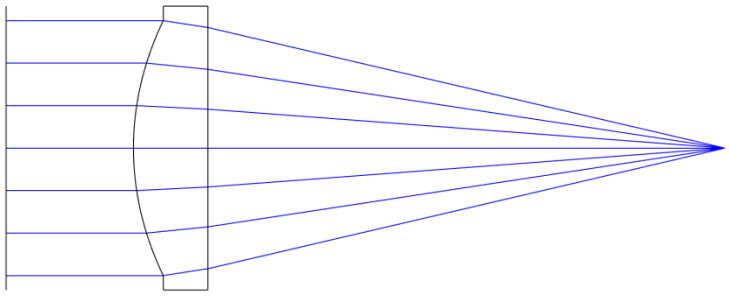


Figure 5-7. Aspherical lens test layout

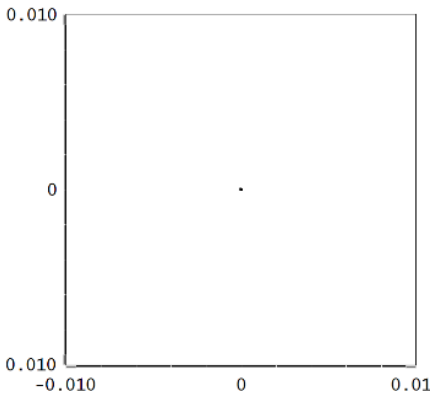
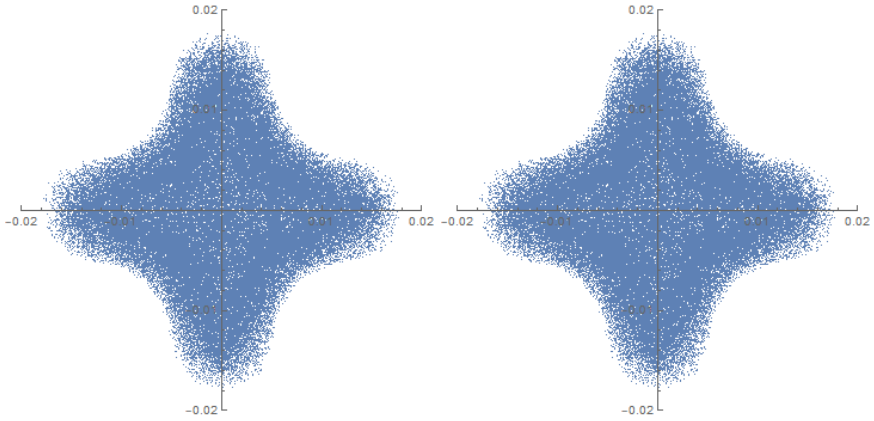


Figure 5-8. Spot diagram of an aspherical lens using OpticStudio

5.2.2 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Linear Interpolation)

The same aspherical lens used in the last sections was analyzed by CUDA ray tracing with the aspheric surface described as a point cloud with 241 times 241 points. Plane surfaces were used for the second surface and the focal plane. Figure 5-8 shows a spot diagram of ray

tracing result with linear interpolation, showing some errors in the result. The spot size almost 0.004mm across, much bigger than the spot size calculated with OpticStudio. In addition, the spot is not radially symmetric, but concentrated near X and Y axes.

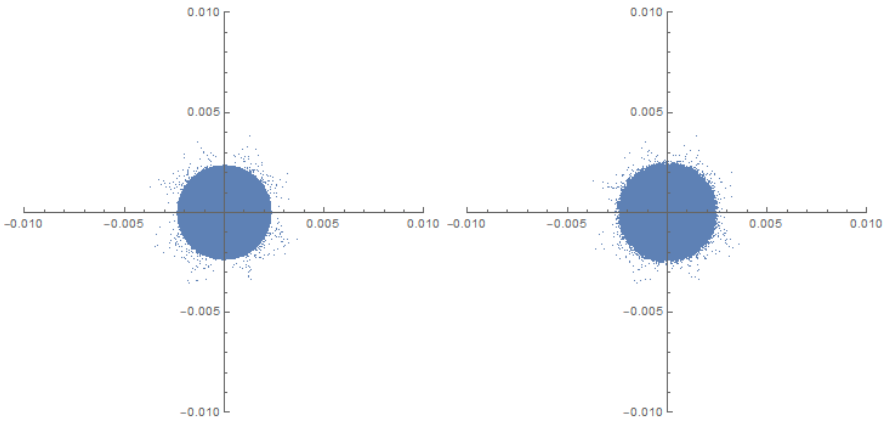


Double precision float Single precision float

Figure 5-9. Spot diagram of an aspherical lens using CUDA ray tracing (Linear interpolation)

5.2.3 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Triangular Patch Interpolation)

The same aspherical lens used in the last sections was analyzed by CUDA ray tracing with Nagata triangular patch interpolation. Surfaces conditions were same as the previous simulation which used linear interpolation. Figure 5-9 shows the results. The spot size is about 0.004 mm which is smaller than the spot of the linear interpolation simulation. Therefore, this simulation method is more accurate than the linear interpolation method.

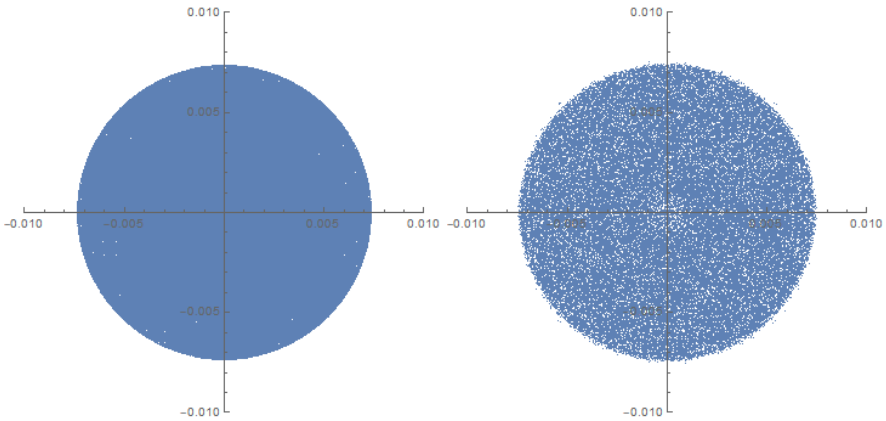


Double precision float Single precision float

Figure 5-10. Spot diagram of an aspherical lens using CUDA ray tracing (Nagata triangular patch)

5.2.4 CUDA Ray Tracing for a Point Cloud Surface and a Plane Surface (Nagata Quadrilateral Patch Interpolation)

Then, the same aspherical lens used in the last sections was analyzed again using CUDA ray tracing with Nagata quadrilateral patch interpolation. Figure 5-11 shows a spot diagram of the result. For both single and double precision float case, the spot size is bigger than the spot size of triangular patch interpolation. In addition, the result of the double precision float is smoother than the result of the single precision float.



Double precision float Single precision float

Figure 5-11. Spot diagram of an aspherical lens using CUDA ray tracing (Nagata quadrilateral patch)

5.2.5 Comparison of Results

Similar to the spherical lens test comparison, a ray path data is summarized in table 5-3 and 5-4 to compare each method.

Table 5-3. Comparison of calculation accuracy for ray position (Aspherical lens)

		Initial positon	S1 (Spherical)	S2 (Plane)	Focal plane
OpticStudio	x	5.350840	5.350840	5.041318	-0.000078
	y	-0.726987	-0.726987	-0.684934	0.000011
	z	0.000000	11.195890	13.342890	36.511000
Double Precision Linear interpolation	x	5.350840	5.350840	5.040385	-0.017106
	y	-0.726987	-0.726987	-0.684947	-0.000107
	z	0.000000	11.195919	13.343000	36.511000
Single Precision Linear interpolation	x	5.350840	5.350840	5.040384	-0.017107
	y	-0.726987	-0.726987	-0.684947	-0.000107
	z	0.000000	11.195918	13.343000	36.511002
Double Precision Nagat triangular patch interpolation	x	5.350840	5.350327	5.040702	-0.002295
	y	-0.726987	-0.727010	-0.684938	0.000310
	z	0.000000	11.195662	13.343000	36.511000
Single Precision Nagat triangular patch interpolation	x	5.350840	5.350326	5.040702	-0.002267
	y	-0.726987	-0.727060	-0.684994	0.000158
	z	0.000000	11.195664	13.343000	36.511002
Double Precision Nagat quadrilateral patch interpolation	x	5.350840	5.350327	5.041234	0.007145
	y	-0.726987	-0.727012	-0.685012	-0.000974
	z	0.000000	11.195662	13.343000	36.511000
Single Precision Nagat quadrilateral patch interpolation	x	5.350840	5.350327	5.041241	0.007262
	y	-0.726987	-0.726961	-0.684967	-0.001025
	z	0.000000	11.195660	13.343000	36.511002

Table 5-4. Comparison of calculation accuracy for ray direction (Aspherical lens)

		Initial positon	S1 (Spherical)	S2 (Plane)	Focal plane
OpticStudio	l	0.000000	-0.142663	-0.212536	-0.212536
	m	0.000000	0.019383	0.028876	0.028876
	n	1.000000	0.989582	0.976726	0.976726
Double Precision Linear interpolation	l	0.000000	-0.143079	-0.213185	-0.213185
	m	0.000000	0.019374	0.028868	0.028868
	n	1.000000	0.989522	0.976585	0.976585
Single Precision Linear interpolation	l	0.000000	-0.143079	-0.213185	-0.213185
	m	0.000000	0.019374	0.028868	0.028868
	n	1.000000	0.989522	0.976585	0.976585
Double Precision Nagat triangular patch interpolation	l	0.000000	-0.142688	-0.212602	-0.212602
	m	0.000000	0.019389	0.028889	0.028889
	n	1.000000	0.989578	0.976712	0.976712
Single Precision Nagat triangular patch interpolation	l	0.000000	-0.142687	-0.212601	-0.212601
	m	0.000000	0.019386	0.028884	0.028884
	n	1.000000	0.989578	0.976712	0.976712
Double Precision Nagat quadrilateral patch interpolation	l	0.000000	-0.142447	-0.212243	-0.212243
	m	0.000000	0.019356	0.028840	0.028840
	n	1.000000	0.989613	0.976791	0.976791
Single Precision Nagat quadrilateral patch interpolation	l	0.000000	-0.142444	-0.212239	-0.212239
	m	0.000000	0.019353	0.028836	0.028836
	n	1.000000	0.989614	0.976792	0.976792

5.3 Headlamp Lens and Starting ray set

Finally, a projector-type headlamp lens was analyzed by the ray tracing algorithm. Figure 5-12 shows a projector headlamp system. It consists of a LED, a reflector, and a projector lens. In this study, ray sets right are after reflector were prepared and used for simulation.

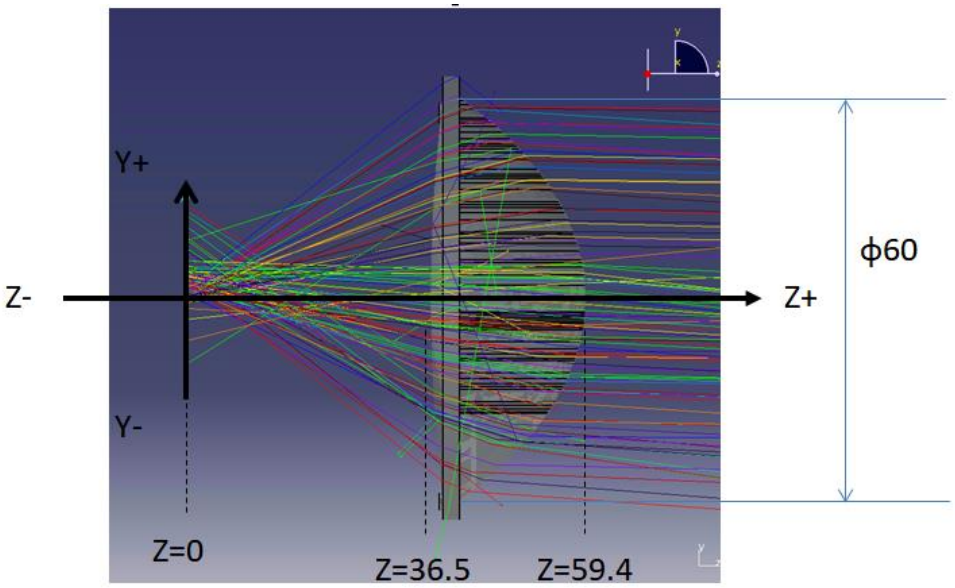


Figure 5-12. Projector type LED headlamp simulation layout

5.3.1 Simulation Using SPEOS

For reference, the lens was analyzed by SPEOS CAA V5 Based from OPTIS (La Farlède, France), an illumination simulation software used for automotive lighting. SPEOS runs on CATIA from Dassault systems (Vélizy-Villacoublay, France), a 3D CAD program can read CATIA model files and import other 3D models such as step file. The simulation was done with a 3D CATIA model of the lens. Figure 5-13 shows a contour plot of simulation result. The transition region is well described.

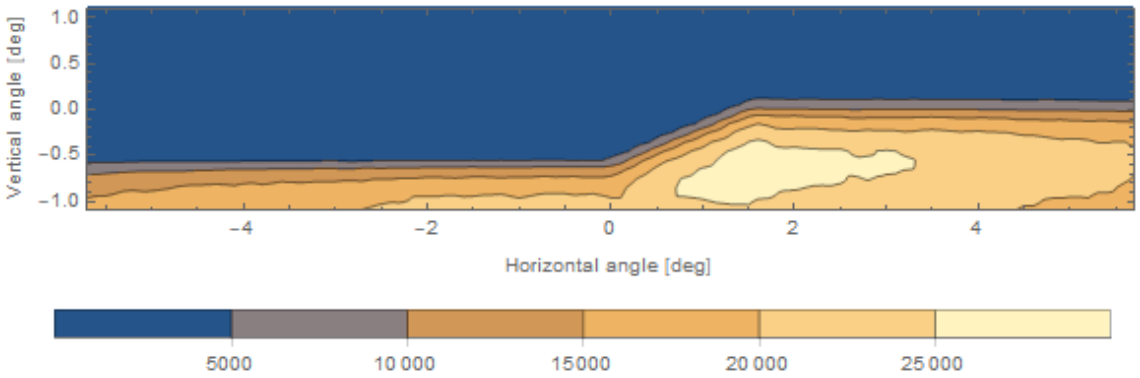


Figure 5-13. Result of ray tracing with SPEOS

5.3.2 Simulation Using OpticStudio

The lens used in the last sections was analyzed again using OpticStudio. Since OpticStudio can import 3D CAD file directly, the lens model expressed as STEP file was used. Since OpticStudio takes much time to tracing a lot of rays, only 5,000,000 parts of rays were analyzed. Figure 5-14 shows a result from an OpticStudio simulation. A characteristic distribution is the cut line which is known as the transition region and is located around horizontal axis line seen in the figure 5-14 simulation result. However, result is noisy because only a part of rays was analyzed.

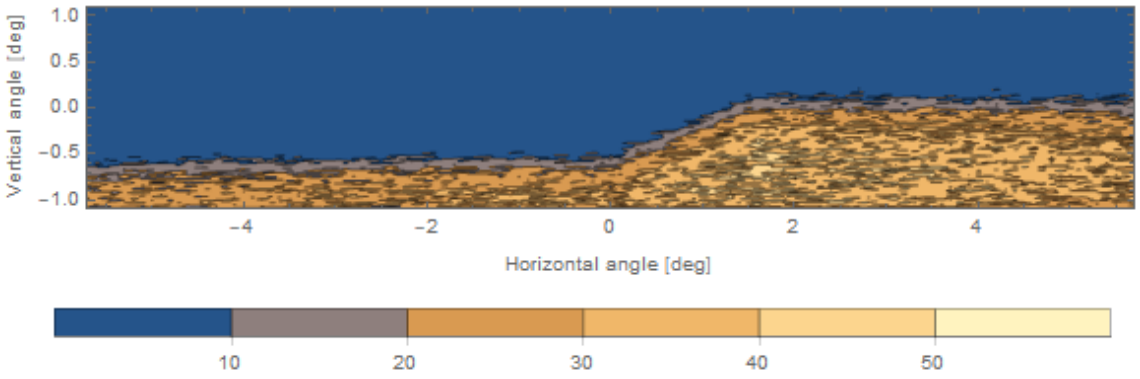


Figure 5-14. Result of ray tracing with OpticStudio (5,000,000 rays)

5.3.3 Simulation Using CUDA Ray Tracing for a Point Cloud Surface (Linear Interpolation)

Next, the lens used in the last sections was analyzed again using CUDA ray tracing with linear interpolation of the point cloud. Since previous spherical and aspherical lens tests showed that the single precision float algorithm is not accurate enough for this application, only double precision float algorithm was used. Figure 5-15 shows the result of CUDA ray tracing with linear interpolation. The cut line is clear in the result.

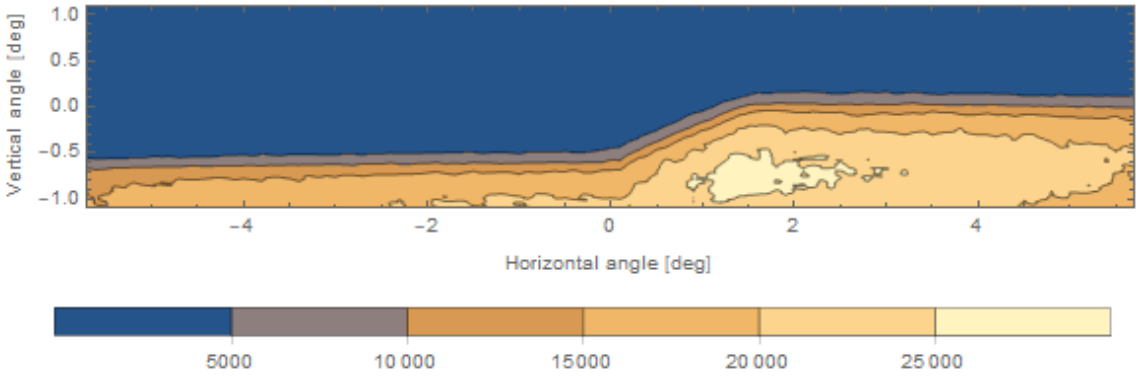


Figure 5-15. Result of ray tracing with CUDA ray tracing (Linear interpolation)

5.3.4 Simulation Using CUDA Ray Tracing for a Point Cloud Surface (Nagata Triangular Patch)

Next, the lens used in the last sections was analyzed again using CUDA ray tracing with Nagata triangular patch interpolation. Figure 5-16 shows result of the simulation. The cut line was characterized similarly with previous simulation. Detail of the comparison will be described in the following section.

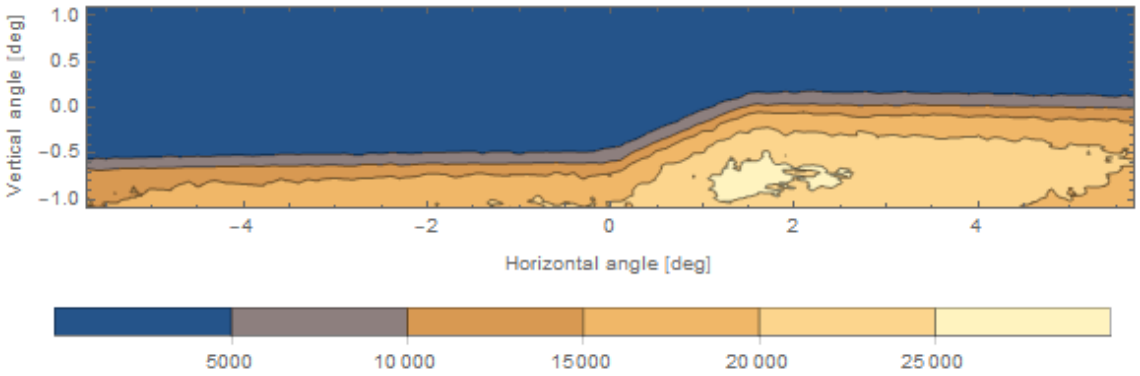


Figure 5-16. Result of ray tracing with CUDA ray tracing (Nagata triangular interpolation)

5.3.5 Simulation Using CUDA Ray Tracing for a Point Cloud Surface (Nagata Quadrilateral Patch)

The lens used in the last sections was analyzed again using CUDA ray tracing with Nagata quadrilateral patch interpolation as well. Figure 5-17 shows result of the simulation. The cut line was characterized similarly with previous simulation. Detail of the comparison is described in following section.

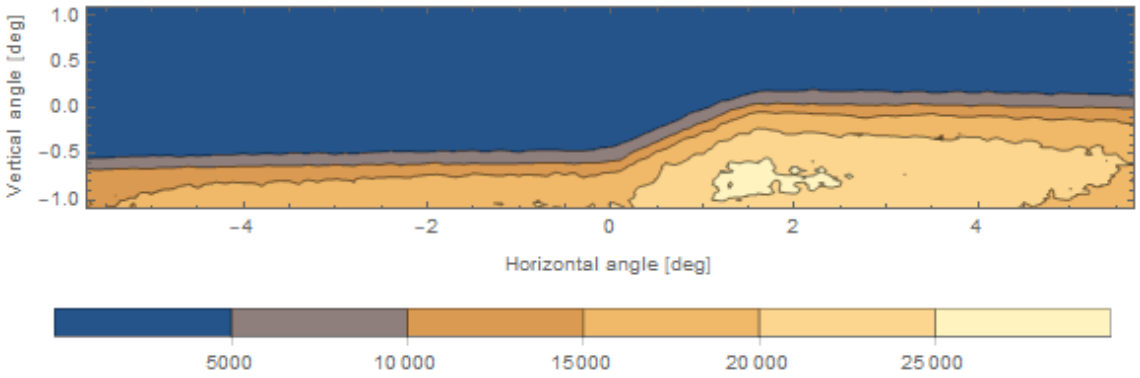


Figure 5-17. Result of ray tracing with CUDA ray tracing (Nagata quadrilateral interpolation)

5.3.6 Comparison of Results between OpticStudio and CUDA Ray Tracing

Since ray tracing with OpticStudio was done with 5,000,000 parts of rays, each CUDA ray tracing with same ray condition were done and compared with them. The results of

comparison are shown in figure 5-18 and 5-19. Figure 5-18 shows intensity distribution at x=0 degree. Figure 5-19 shows intensity distribution at x=2.5 degree. Since the number of traced ray was limited, results were noisy, but those distributions look similar.

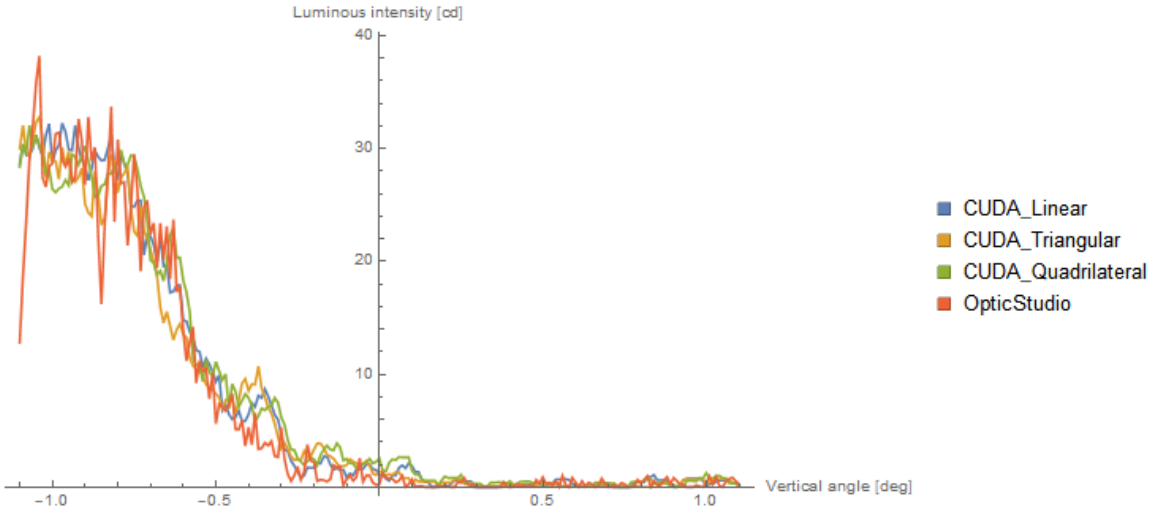


Figure 5-18. Comparison of headlamp simulation of OpticStudio and CUDA ray tracing at x=0[deg]

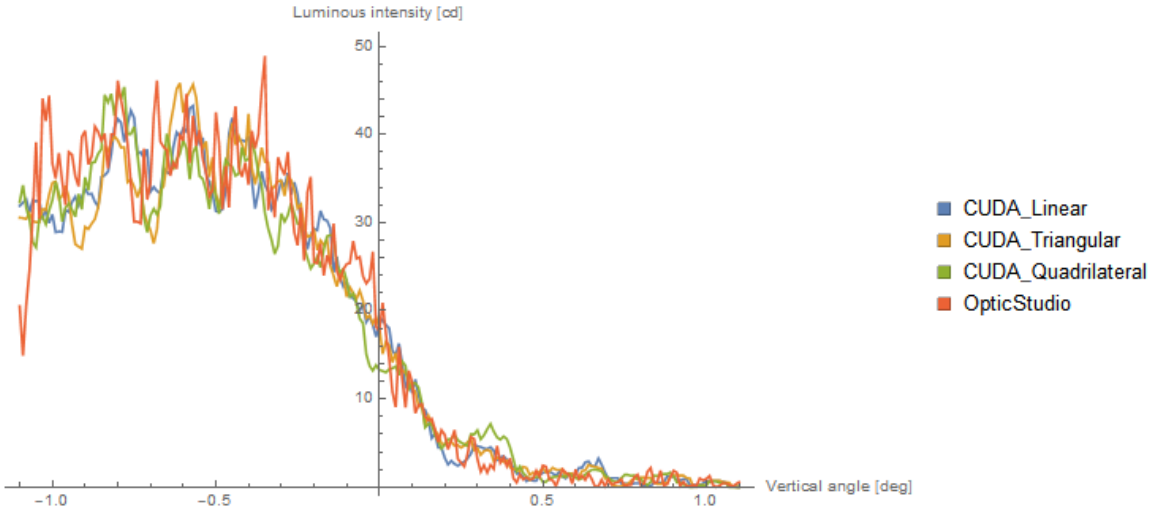


Figure 5-19. Comparison of headlamp simulation of OpticStudio and CUDA ray tracing at x=2.5[deg]

5.3.7 Comparison of Results between SPEOS and CUDA Ray Tracing

The result of the simulation using SPEOS was compared with the results of simulation using CUDA ray tracing using full ray set of 362,000,000 incident rays. The results of comparison are shown in figure 5-20 and 5-21. Figure 5-20 shows intensity distribution at x=0 degree. Figure 5-21 shows intensity distribution at x=2.5 degree. The slope of transition in SPEOS result is larger than it in CUDA ray tracing.

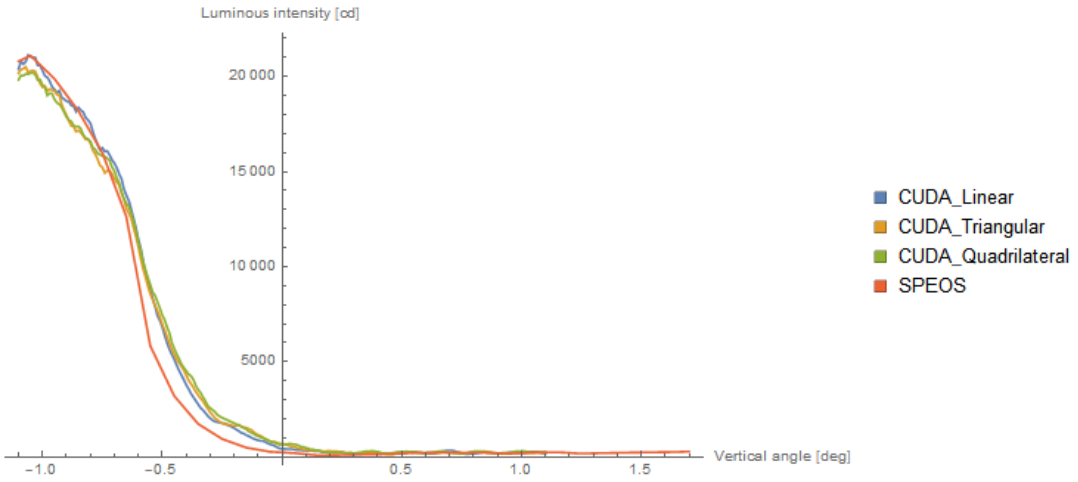


Figure 5-20. Comparison of headlamp simulation of SPEOS and CUDA ray tracing at x=0[deg]

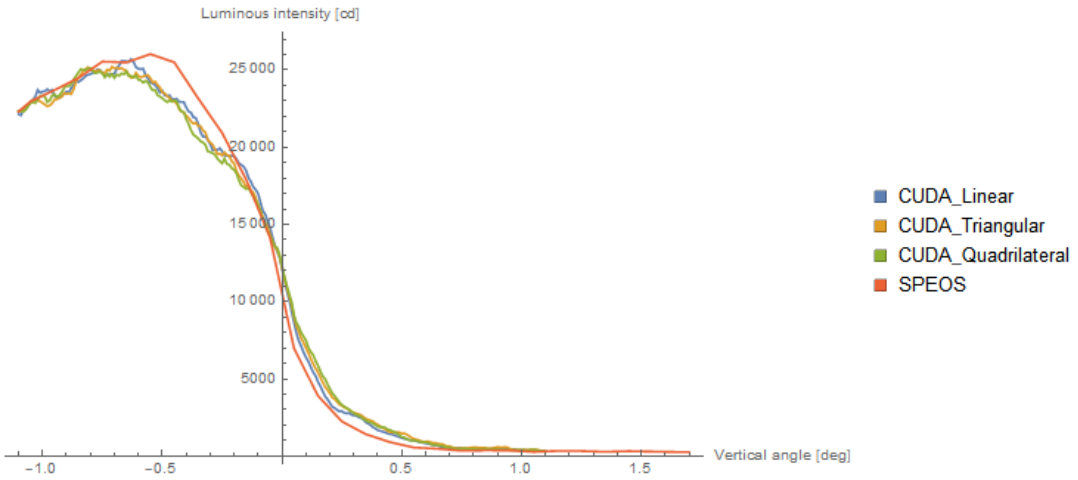


Figure 5-21. Comparison of headlamp simulation of SPEOS and CUDA ray tracing at x=2.5[deg]

5.4 Comparison of Calculation Speed

To confirm the effectiveness of ray tracing using CUDA, the calculation time for a single surface by each method was compared. Figure 5-22 shows calculation time of each CUDA ray tracing methods with 250,000 rays and single surface. In the case of using Non-interpolation and a plane surface, the calculation speed is faster than other methods. However, it takes less than 0.5 second in all cases. Most of time was spent for initializing CUDA.

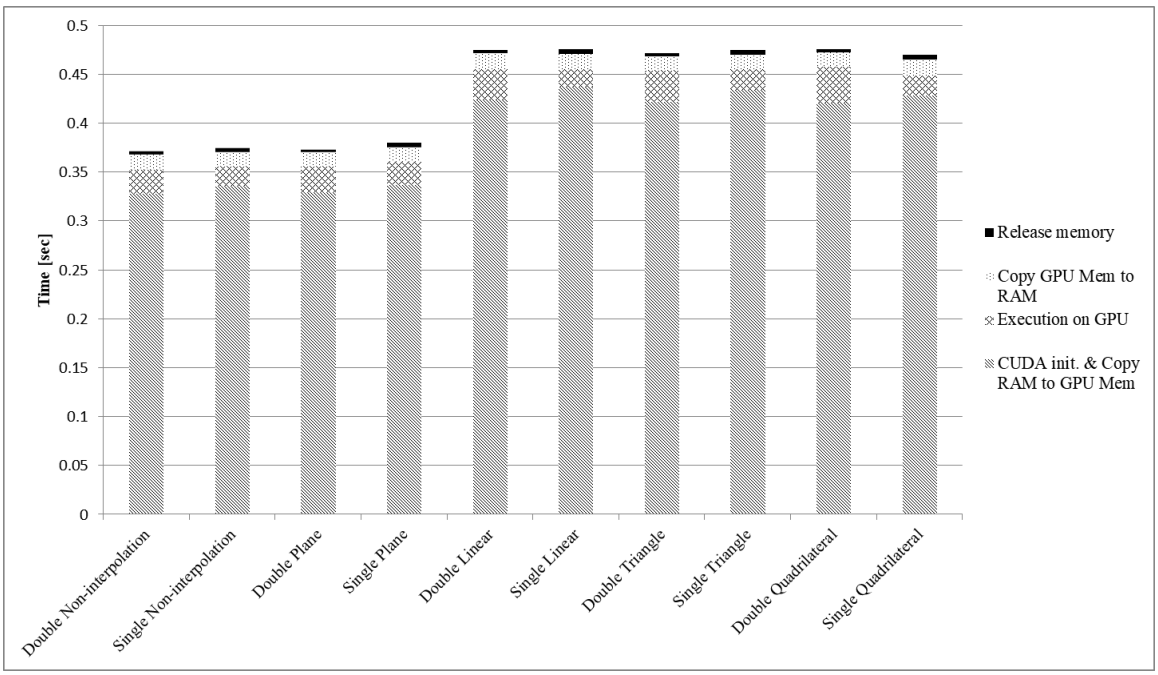


Figure 5-22. Comparison of calculation speed of each interpolation method for 250,000 rays and a single surface and a single point cloud surface

Figure 5-23 shows the calculation time using double precision float Nagata triangular interpolation method. This algorithm provided the best accuracy in our CUDA raytracing method. The result indicates that time consumption minus 0.15 seconds is approximately proportional to the number of rays. In addition, most of the time is spent initializing CUDA driver and copying

data from the system memory to the GPU memory, which includes preparing data and output memory allocation.

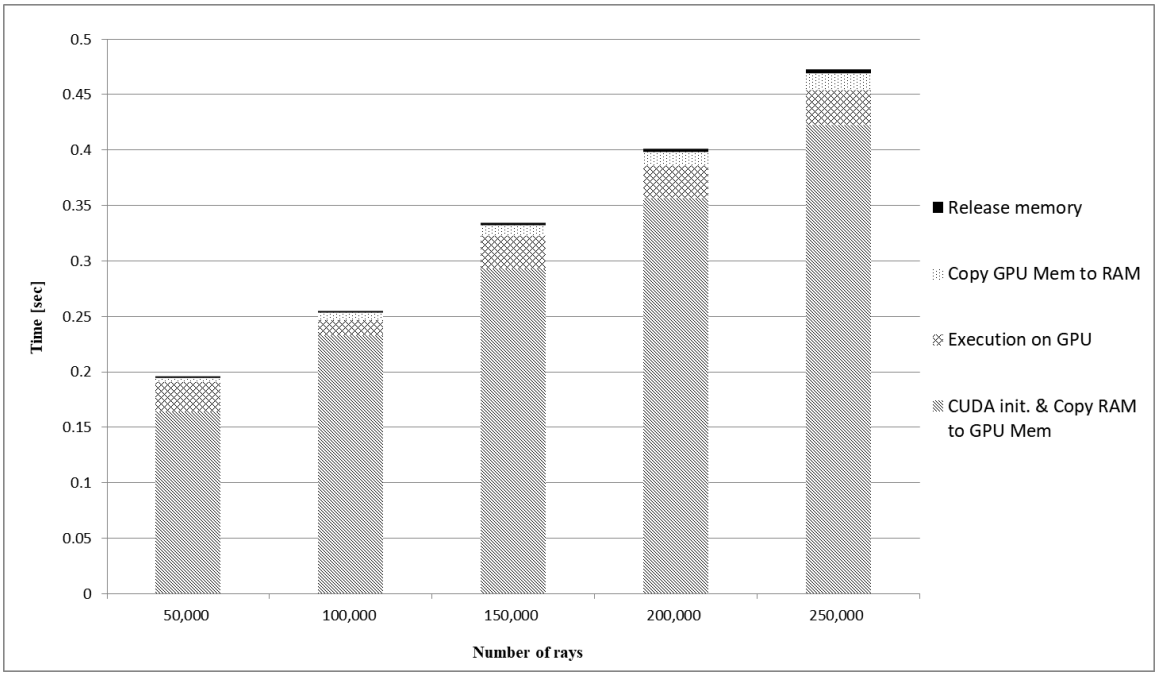


Figure 5-23. Calculation speed for changing number of input rays (Double precision Nagata triangular patch interpolation)

CHAPTER 6: CONCLUSION

6.1 Summary

Several types of intersection searching methods and refraction calculations in 3-dimensional space were reviewed in Chapter 2. The Intersection point between a ray and plane and Intersection point between a ray and a spherical surface can be solved directly using equations. Closed form have been implemented to find the Intersection point between a ray and complex surfaces represented by point clouds. The k-d tree method was described for accelerating the process of finding the nearest points to a ray intersect for intersection searching. When using a point cloud, an interpolation is needed for estimation of the position between

provided points. Linear interpolation, Nagata triangular patch quadric interpolation, and Nagata quadrilateral patch quadric interpolation were analyzed.

Chapter 3 reviewed photometric units compared with radiometric units. When the photometer is used for measurement of illumination, the results include the convolution with the detector aperture, so the basics of convolution were also reviewed.

The key to acceleration calculation speed is parallelizing the ray tracing calculations using CUDA, a parallel computing platform provided by Nvidia and reviewed in chapter 4. CUDA requires memory management separate from system memory. In addition, Mathematica was used to provide a variety of useful interface function was used for this study, which convenient.

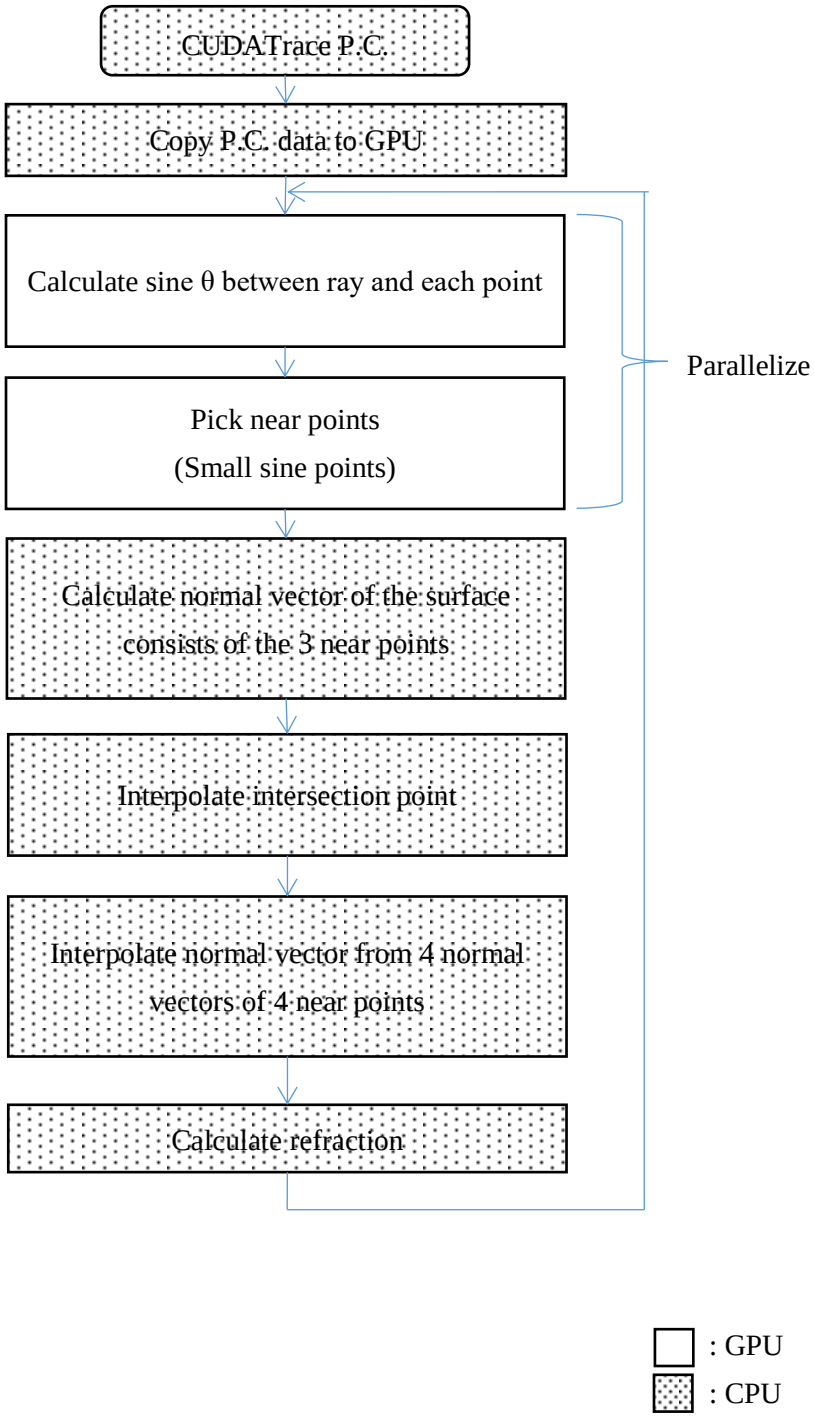
Finally, ray tracing results using each interpolation method were described, and their accuracies were discussed. The first test case was the simplest layout, which was converging a collimated beam using a plano-convex spherical lens. The second test case was used a plano-convex aspherical lens instead of the spherical lens. Both results indicate that the double float precision Nagata triangular patch interpolation in the most accurate. Finally, the headlamp lens simulation which was a main objective of this study was done.

6.2 Future Work

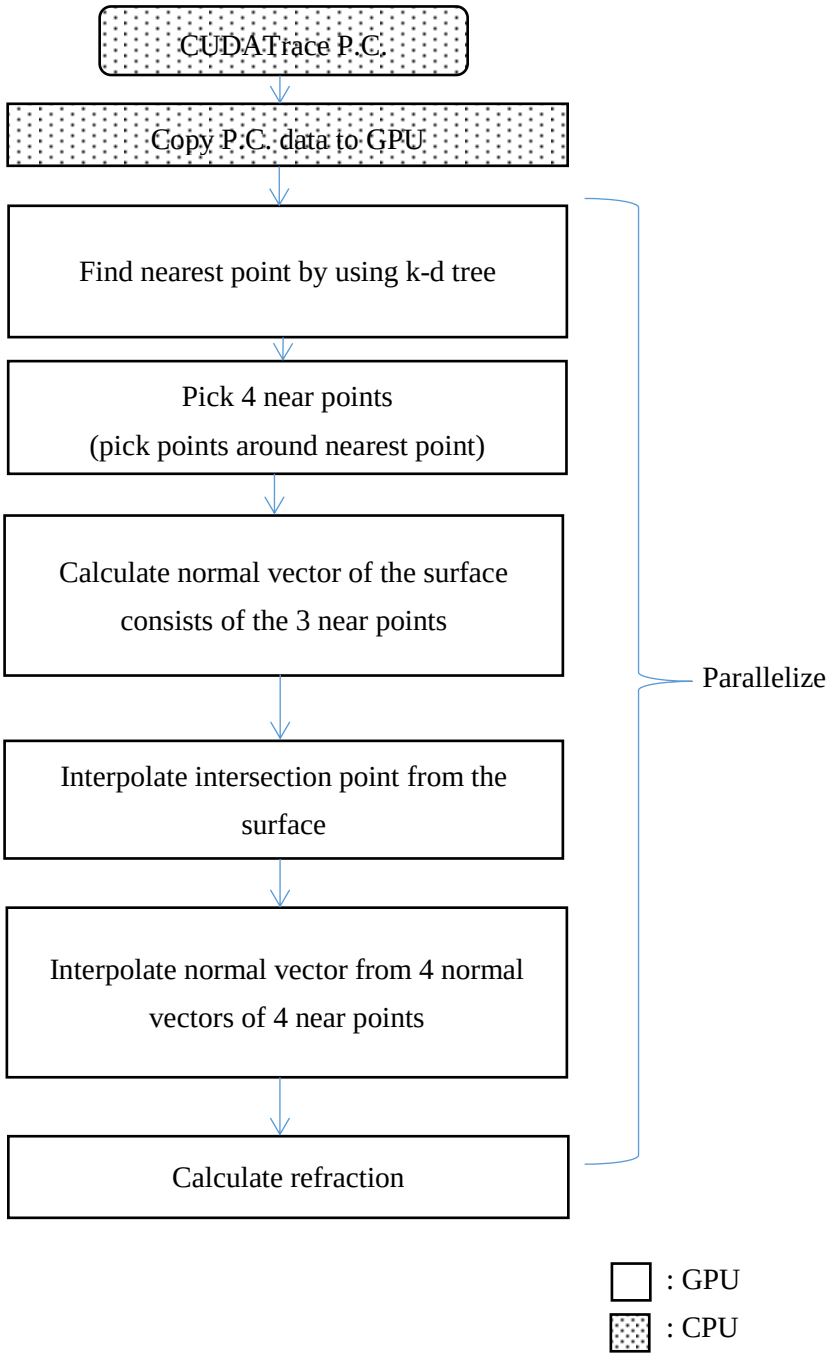
An accelerated ray tracing method using CUDA for models with spherical surfaces, plane surfaces, and point clouds with linear and quadratic interpolation have been developed in this study. High-speed ray tracing was achieved. However, some errors were documented from interpolation. One of solution that reduces errors from interpolation is using exact surface equations in the models so that interpolation is not required. Specifically, if the CAD model can be used for the simulation directly, interpolations might not be necessary. For example, STEP files which are regulated by International Organization for Standardization are used in many CAD programs. The STEP file format and interface APIs are publically disclosed. Therefore, to reducing errors in CUDA ray tracing, it will be best to use STEP files.

APPENDIX A: FLOW CHART OF CODES

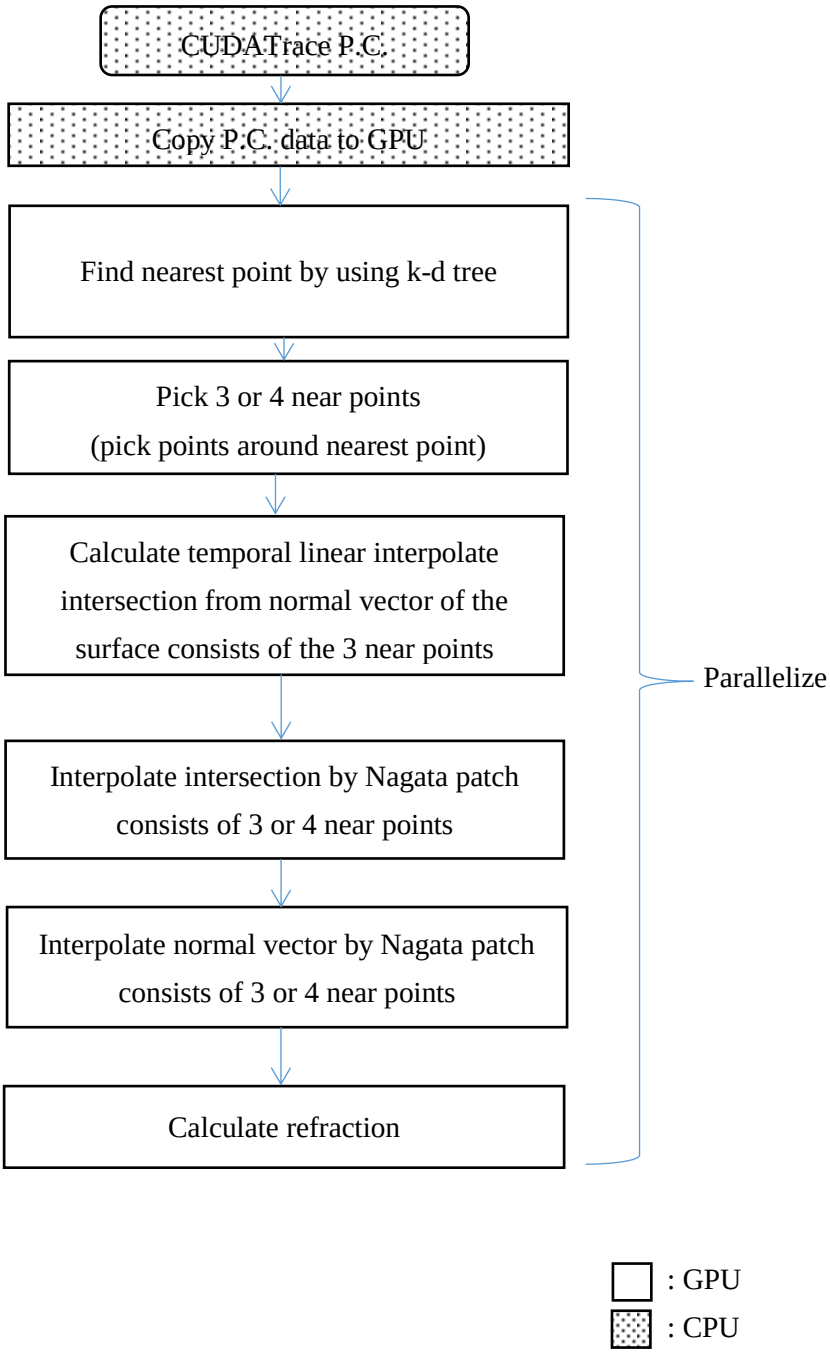
Intersection Search with the Brute Force Attack



Intersection Search with the K-d Tree (with Linear Interpolation)



Intersection Search with the K-d Tree (with Nagata Patch Interpolation)



APPENDIX B: INPUT FORMAT FOR COMMANDS WRITTEN IN MATHEMATICA

1. Ray tracing with sphere surface

CUDATraceSphered[rayin, radius, vertex, aptype, apsize, mat1, mat2] (Double precision)

CUDATraceSpheref[rayin, radius, vertex, aptype, apsize, mat1, mat2] (Single precision)

rayin: input ray array. Format is {{x , y , z , kx, ky, kz, λ, P},{x , y , z , kx, ky, kz, λ, P},...}

x,y,z indicate initial position of ray in mm. kx, ky, kz indicate ray direction. (direction cosine.)

λ is wavelength in nm. P is flux of ray in watt.

radius: radius of curvature in mm.

vertex: vertex position in mm. $\{x_v, y_v, z_v\}$

aptype: Aperture type. 0 is rectangular aperture. 1 is circular aperture.

apsize: Aperture size. {xmin, xmax, ymin, ymax} if aptype=0. {r, 0, 0, 0} if aptype=1.

mat1: Material before surface. 0 is Air. 1 is PMMA.

mat2: Material after surface. 0 is Air. 1 is PMMA.

2. Ray tracing with plane surface

CUDATracePlaned[rayin, normalvector, origin, mat1, mat2] (Double precision)

CUDATracePlanef[rayin, normalvector, origin, mat1, mat2] (Single precision)

normalvector: Surface normal vector {nx, ny, nz} (direction cosine)

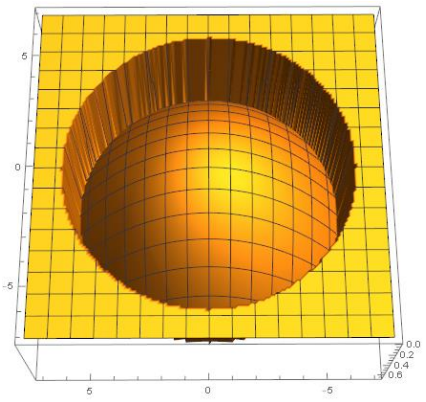
origin: A point on the surface. {x, y, z}

3. Ray tracing with point cloud (Brute Force algorithm, linear interpolation)

CUDATracePCd[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Double precision)

CUDATracePCf[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Single precision)

pc: Point Cloud. {{x0, y0, z0},{x1, y1, z1},...}. Point should be start from left upper (maximum x and maximum y) and equal space grid. z=0 indicate out of aperture. To define aperture, z=0 are required at most outer perimeter of point cloud. Dimension of list should be {hmax*wmax,3}. After {xmax, ymin, z}, {xmax-1 incriment, ymax, z} is followed.



pcnv: Normal vectors of points. It correspond to point cloud.

hmax: Number of points in y direction

wmax: Number of points in x direction

4. Ray tracing with point cloud (k-d tree algorithm, linear interpolation)

CUDATracePCd2[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Double precision)

CUDATracePCf2[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Single precision)

5. Ray tracing with point cloud (k-d tree algorithm, Nagata triangular patch interpolation)

CUDATracePCd3[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Double precision)

CUDATracePCf3[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Single precision)

6. Ray tracing with point cloud (k-d tree algorithm, Nagata quadrilateral patch interpolation)

CUDATracePCd4[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Double precision)

CUDATracePCf4[rayin, pc, pcnv, mat1, mat2, hmax, wmax] (Single precision)

7. Binning and Convolution

CUDABinConv[rayin3, xmin, xmax, xincr, ymin, ymax, yincr, radius, distance]

rayin3: 2D rayset. {x,y,P}

xmin,xmax,xincr: x of output grid size in degree. (Minimum, Maximum, Increment)

ymin,ymax,yincr: y of output grid size in degree. (Minimum, Maximum, Increment)

radius: detector radius

distance: distance between the lens and the detector

BIBLIOGRAPHY

[1]Mauch, F., Gronle, M., Lyda, W., & Osten, W. (2013). Open-source graphics processing unit–accelerated ray tracer for optical simulation. Optical Engineering, 52(5), 53004.

[2] NVIDIA Corporation. (2017). *CUDA C Programming Guide* (v8.0). http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

[3] U.S. Department of Transportation National Highway Trafic Safety Administration. (2007, December). *Laboratory Test Procedure For FMVSS 108 Lamps, Reflective Devices, and Associated Equipment* (Report No. TP-108-13 DRAFT) Retrieved from https://www.nhtsa.gov/DOT/TP-108-13.pdf

[4]Schott AG. (2014, February). Technical Information, TIE-29: Refractive Index and Dispersion. Retrieved September 7, 2017, from http://www.schott.com/d/advanced_optics/02ffdb0d-00a6-408f-84a5-19de56652849/1.0/tie_29_refractive_index_and_dispersion_eng.pdf

[5]Hughes, J. F., 1955. (2014). *Computer graphics: Principles and practice* (3rd ed.). Upper Saddle River, N.J: Addison-Wesley.

[6]Koshel, R. J. (2013;2012;). *Illumination engineering: Design with nonimaging optics* (1. Aufl.;1; ed.). Hoboken, New Jersey: Wiley-IEEE Press.

[7]Schwiegerling, J. (2014). *Optical specification, fabrication, and testing*. Bellingham, WA: SPIE Press.

[8]*The oxford english dictionary* (1989). . England;United Kingdom;:

[9]CUDALink guide. (n.d.). Retrieved January 10, 2017, from http://reference.wolfram.com/language/CUDALink/guide/CUDALink.html

[10]49: Transportation, §571.108 Standard No. 108; Lamps, reflective devices, and associated equipment. (2017, August 25). U.S. Government Publish Office.

[11]Dereniak, E. L., & Dereniak, T. D. (2008). Geometrical and trigonometric optics. Cambridge, UK;New York;: Cambridge University Press.

[12]Nagata T. Simple local interpolation of surfaces using normal vectors. Computer Aided Geometric Design. 2005;22(4):327-47.

[13] NVIDIA Corporation. (n.d.). Specifications. Retrieved September 18, 2017, from https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-970/specifications

[14]Lengyel, E., & Books24x7, I. (2012;2011;). *Mathematics for 3D game programming and computer graphics, third edition* (3rd;3; ed.). US: Cengage Learning.