Simulating the plenoptic camera.

Arash. Razavi
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
Simulating The Plenoptic Camera

by

Arash Razavi

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada
2003

Copyright © 2003 by Arash Razavi
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-80511-5
ABSTRACT

This thesis presents the simulation of a CCD based camera called the Plenoptic Camera. Plenoptic Camera is a novel single lens camera capable of capturing 3D information of the scene and is used to find the depth of the objects. The main concept behind the design of this camera lies in the fact that light which passes through the lens of a camera is quite rich in information which regular cameras are not designed to take advantage of. The plenoptic camera keeps track of the 3D information of the light rays by distributing these rays over a two dimensional array of converging lenses which we call a lenticular array, causing the rays to be separated based on the angle of incident. The plenoptic camera was designed in 1992 and is capable of taking several snapshots of the scene from a continuum of viewpoints in a single shot, resulting in multiple images which we call plenoptic images.

The goal of this thesis is to fully simulate this camera and generate the plenoptic images. The simulation is carried out using ray tracing, which is a technique for image synthesis. The ray tracing problem is divided into two separate parts. The first part is the simulation of the camera lens system, and the second part is the ray tracing in the scene which is calculating the distribution of light within the environment.

Since no simulation has ever been done on this research domain camera, the simulation which is proposed in this thesis can be used to carry out tests and customize the camera for a specific application before the actual model is built.
To my family...
ACKNOWLEDGEMENTS

The author would like to express his sincere thanks and appreciation to Dr. Ahmadi for his assistance and guidance.

The author would also like to thank Dr. Boufama for his suggestions and ideas, and Dr. Miller for taking time to review the work and participate in the examination committee.

The author would like to acknowledge the financial support provided by Micronet Network Of Centers Of Excellence\(^1\) and DALSA Corp.\(^2\) which formed the basis for developing this research project.

\(^1\)www.micronetrd.ca
\(^2\)www.dalsa.com
# TABLE OF CONTENTS

ABSTRACT ........................................ iii
DEDICATION ........................................ iv
ACKNOWLEDGEMENTS ............................... v
LIST OF FIGURES .................................. ix

I  INTRODUCTION .................................. 1
   1.1 Statement of the problem ...................... 1
   1.2 Challenges .................................. 3
   1.3 Outline of the thesis ......................... 4

II  PLENOPTIC CAMERA .............................. 5
   2.1 Overview .................................. 5
   2.2 Description ................................ 7
   2.3 Image Formation ............................. 9
   2.4 Depth Measurement .......................... 10

III MODELLING THE PLENOPTIC CAMERA LENS SYSTEM 14
   3.1 Overview .................................. 14
   3.2 Surface Physics .............................. 14
      3.2.1 Specular Surfaces ....................... 15
         3.2.1.1 Specular Reflection .................. 15
         3.2.1.2 Specular Transmission ............... 16
      3.2.2 Diffuse Surfaces ....................... 18
         3.2.2.1 Diffuse Reflection .................. 18
         3.2.2.2 Diffuse Transmission ............... 19
   3.3 Lenses and Cameras ......................... 20
      3.3.1 Lens .................................. 20
      3.3.2 Thin Lens ............................. 21
      3.3.3 Thick Lens ............................ 22
3.3.4 Lens System ........................................ 23
3.3.5 Pinhole Camera .................................... 25
3.4 Plenoptic Camera Lens System Simulation ............. 28
  3.4.1 Implementation .................................. 28
  3.4.2 Results .......................................... 29

IV RAY TRACING ............................................ 34
  4.1 Overview ............................................ 34
  4.2 Global Illumination .................................. 34
  4.3 Surface Reflection Models ............................ 35
  4.4 Rendering Equations ................................ 37
  4.5 Monte Carlo Integration .............................. 38
  4.6 Sampling Techniques ................................ 41
    4.6.1 Aliasing ....................................... 41
    4.6.2 Uniform Supersampling .......................... 42
    4.6.3 Adaptive Sampling ............................... 42
    4.6.4 Stochastic Sampling .............................. 43
  4.7 Texture Mapping .................................... 45
    4.7.1 Inverse Mapping ................................ 45
      4.7.1.1 Sphere Inverse Mapping ..................... 45
      4.7.1.2 Disk Inverse Mapping ....................... 46

V PATH TRACING ............................................ 48
  5.1 Overview ............................................ 48
  5.2 Solution to Monte Carlo Integration .................. 49
  5.3 Pure Path Tracing ................................... 50
  5.4 Adding Direct Lighting .............................. 52
    5.4.1 Direct lighting calculations .................... 52
    5.4.2 Sampling light sources ......................... 54
      5.4.2.1 Spherical light sources ...................... 54
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Plenoptic Camera Simulation Block Diagram</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Plenoptic Camera Simulation</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Stereoscopic Vision</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>The Plenoptic Camera Structure</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Image Formation</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>Plenoptic Images</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Effect of Depth</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>Geometry of a Specular Reflection</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>Geometry of a Specular Transmission</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>Diffuse Reflection</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>Diffuse Transmission</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>Lens Geometry</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Thin Lens</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>Thick Lens</td>
<td>23</td>
</tr>
<tr>
<td>15</td>
<td>Lens system</td>
<td>24</td>
</tr>
<tr>
<td>16</td>
<td>Modified Pinhole Camera Model</td>
<td>26</td>
</tr>
<tr>
<td>17</td>
<td>Backward Ray Tracing</td>
<td>27</td>
</tr>
<tr>
<td>18</td>
<td>Sampling the Lenticular Array</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>Generating the Initial Rays</td>
<td>29</td>
</tr>
<tr>
<td>20</td>
<td>Plenoptic Camera Lens System Simulation Flowchart</td>
<td>31</td>
</tr>
<tr>
<td>21</td>
<td>A Lens System</td>
<td>32</td>
</tr>
<tr>
<td>22</td>
<td>Tabular Description of the Lens System</td>
<td>32</td>
</tr>
<tr>
<td>23</td>
<td>Plenoptic Image</td>
<td>33</td>
</tr>
<tr>
<td>24</td>
<td>Extracted Plenoptic Images</td>
<td>33</td>
</tr>
<tr>
<td>25</td>
<td>Illumination Hemisphere</td>
<td>36</td>
</tr>
<tr>
<td>26</td>
<td>Importance Sampling</td>
<td>40</td>
</tr>
<tr>
<td>27</td>
<td>Adaptive Sampling</td>
<td>43</td>
</tr>
<tr>
<td>Page</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>28</td>
<td>Stochastic Sampling</td>
<td>44</td>
</tr>
<tr>
<td>29</td>
<td>Sphere Inverse Mapping</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>Disk Inverse Mapping</td>
<td>47</td>
</tr>
<tr>
<td>31</td>
<td>A Path of Depth 3</td>
<td>50</td>
</tr>
<tr>
<td>32</td>
<td>Direct Lighting</td>
<td>53</td>
</tr>
<tr>
<td>33</td>
<td>A Surface Shadows Itself</td>
<td>56</td>
</tr>
<tr>
<td>34</td>
<td>Results with different number of samples per pixel</td>
<td>64</td>
</tr>
<tr>
<td>35</td>
<td>Results with different number of light source samples</td>
<td>65</td>
</tr>
<tr>
<td>36</td>
<td>Results with texture mapping</td>
<td>66</td>
</tr>
<tr>
<td>37</td>
<td>Room with shelves and two spheres</td>
<td>67</td>
</tr>
<tr>
<td>38</td>
<td>Room with a table and two chairs and a frame</td>
<td>67</td>
</tr>
<tr>
<td>39</td>
<td>Plenoptic Image with Path Tracer</td>
<td>70</td>
</tr>
<tr>
<td>40</td>
<td>Extracted Plenoptic Images with Path Tracer</td>
<td>70</td>
</tr>
<tr>
<td>41</td>
<td>Displacement Analysis</td>
<td>71</td>
</tr>
<tr>
<td>42</td>
<td>The Two Coordinate Systems (Object and Camera)</td>
<td>75</td>
</tr>
<tr>
<td>43</td>
<td>Shapes Class Structure</td>
<td>78</td>
</tr>
<tr>
<td>44</td>
<td>Lens Class Structure</td>
<td>79</td>
</tr>
<tr>
<td>45</td>
<td>Camera Class Structure</td>
<td>79</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

1.1 Statement of the problem

Depth recovery is a central concern in computer vision. In this thesis we introduce the simulation of a novel single lens camera which is called the Plenoptic Camera [1], [2]. This camera which was designed in 1992, is capable of capturing the 3D information of the scene and is used to find the depth of the objects in the scene. Due to complexity of the plenoptic camera lens system structure, the simulation cannot be carried out using mathematical modelling and approximations. Therefore ray tracing, which is a technique for image synthesis, was used to simulate this camera.

Plenoptic camera is capable of taking several snapshots of the scene from a continuum of viewpoints in a single shot, resulting in multiple images which we call plenoptic images. The goal of the simulation is to generate the plenoptic images. As can be seen in figure (1), the outputs of the simulation are the plenoptic images and the inputs are:

- **Scene Description**: Geometry of the objects of the scene, object materials and properties, and location and properties of light sources.

![Figure 1: Plenoptic Camera Simulation Block Diagram](image-url)
- **Camera Description**: Geometry and description of the camera lens system.

We divided the simulation problem into two separate parts. As can be seen in figure (2), the first part is the simulation of the camera lens system, and the second part is the ray tracing in the scene, which is calculating the distribution of light within the environment. For the second part we implemented a ray tracing algorithm which is called the **Path tracer**.

![Diagram of Plenoptic Camera Simulation](image)

**Figure 2**: Plenoptic Camera Simulation

The reasons for choosing this algorithm are:

- **Realistic results**. As will be discussed later, path tracer solves the global illumination problem completely, and supports both the **diffuse** and **specular** reflection of the surfaces. Because most of the possible applications of this camera would be industrial, choosing a ray tracing algorithm which supports both the specular materials (metal) and the diffuse materials (plastic) is essential.

- **Speed**. Path tracer algorithm is fast, compared to other ray tracing algorithms, and because simulation of the camera lens system is also time consuming, path tracer is a good choice.
• **Implementation.** Compared to other ray tracing algorithms, path tracer is easy to implement.

### 1.2 Challenges

The main challenges of the project were:

• *Simulating the lens system.* As will be discussed later, plenoptic camera has a main lens and an array of lenses in the lenticular array. Simulating this lens system accurately was an important part of the thesis.

• *Implementation of a path tracer.* We had to implement a complete ray tracer to get realistic results. Almost all of the available ray tracing packages, like Rayshade [3] or RenderPark [4], use a pinhole camera for their camera system which was not suitable for our camera. Furthermore, the sources of better ray tracing software are not available. Even if we had used an available ray tracing package, merging the ray tracing part to the camera lens system simulation could have been difficult. Therefore a complete and realistic ray tracing algorithm had to be implemented.

• *Sampling.* Owing to the discrete nature of the pixels, sampling is one of the most important parts of a ray tracing algorithm. An image can be thought of as a set of discrete samples of a signal representing intensity variations over space, which can be called *spatial signal.* Since the sampling is discrete, loss of information of the continuous spatial signal results in an artifact which is called *aliasing.* The pixel resolution dictates an upper bound for spatial frequencies that can be displayed. This upper bound is the *Nyquist limit.* When a uniform sampling grid is used, frequencies higher than the Nyquist limit appear as aliasing artifacts. If the sample grid is nonuniform, the high frequencies may appear as noise, which is less objectionable that aliasing.
1.3 Outline of the thesis

This thesis is organized as follows.

Chapter 2 introduces the Plenoptic camera. Starting with the description of the camera and going through the image formation and depth measurement.

Chapter 3 presents the simulation of the plenoptic camera lens system, which involves Ray tracing inside the camera. The simulation needs some knowledge of optics and cameras which is briefly discussed in this chapter.

Chapter 4 presents the basic concepts behind a ray tracing algorithm. Concepts like Global illumination, Rendering equations and Monte Carlo integration. An important section of this chapter describes the Aliasing problem and the sampling techniques.

Chapter 5 details the implementation of the Path tracer, which is the ray tracing algorithm used to simulate the distribution of light in the environment.

Chapter 6 presents the final results of the plenoptic camera simulation using path tracing, which merges the results of chapter 3 and 5.

Chapter 7 summarizes the contribution of the thesis and points out directions for future work.
CHAPTER II

PLENOPTIC CAMERA

2.1 Overview

In this chapter we describe a novel camera that captures 3D information of the scene. The camera which is called the Plenoptic Camera [1],[2] achieves depth recovery through a process known as the single lens stereo [5].

A range image which represents the distances between a viewing point and various surfaces in a scene, can be derived by active or passive techniques. In active ranging some form of energy is directed toward an object and then measured on its return, like sonar, laser or structured light. In passive ranging the light from a normally illuminated scene is captured and analyzed by one or more cameras.

Some methods of passive ranging and depth estimation are depth from focus [6] and depth from shading [7], but the most popular method of passive ranging is Binocular stereo vision or Stereoscopic Vision [8], which extracts depth information by the use of two cameras. Two images from two different viewpoints are taken by the cameras and after matching the corresponding image points in the two views, depth is estimated from the disparity of corresponding points using epipolar geometry [9]. Epipolar geometry describes the relationship between two viewpoints and simplifies the problem of point matching from the domain of all the pixels in the image to the pixels on a specific line in the image. As can be seen in figure (3), a point X in 3D-space is imaged in two views, at X₁ in the first, and X₂ in the second. As shown in this figure, the image points X₁ and X₂, space point X and camera centers are coplanar, which we call the Epipolar plane. The rays back-projected from X₁ and X₂ intersect at X which are also in the epipolar plane. Suppose X₁ is known and we want
to solve the correspondence problem by finding $X_2$. The epipolar plane is determined by the baseline $C_1C_2$ and the ray defined by $C_1X_1$. Now we know that $X_2$ is in a line $L$ which we call the epipolar line. This line is the intersection of the epipolar plane and the second image plane. Therefore the search for the point corresponding to $X$ does not need to cover the entire image plane and can be restricted to the line $L$.

![Figure 3: Stereoscopic Vision](image)

Despite the popularity of this technique, stereoscopic vision has some disadvantages.

- Two separate cameras must be used, which increases the bulk and expense of the system.

- These two cameras must be accurately positioned and calibrated. Camera calibration is the process of determining a camera’s geometric and optical constants and it’s pose in a 3D world coordinate system.

- Stereoscopic vision only extracts information about one axis like horizontal or vertical parallax.
• Correspondence problem, which is the problem of determining the matches between corresponding image points, is known to be demanding from a computational point of view.

• Depth recovery relies highly on the accuracy of points correspondence, which is highly sensitive to noise.

The last two problems can be somehow solved by the use of a Trinocular system [10],[11] which uses three cameras, or a Motion parallax system [12] which moves the camera along a track and takes a dense sequence of images from a series of closely spaced viewpoints. For the Trinocular system, the problem which still remains is the size and expense of the system and the calibration problem. Also in the Motion parallax system, the necessity of movement of the camera along a trajectory over an extended period of time is the main disadvantage.

Single lens stereo with a plenoptic camera provide us with a dense sequence of images, along both axes, without the necessity of a complex moving system.

2.2 Description

The Plenoptic Camera was designed in 1992 by Edward H. Adelson [1] to achieve single lens stereo, by taking several snapshots of the scene from a continuum of viewpoints in a single shot. Plenoptic is derived from the word roots for complete and view. Since this camera records information about how the scene appears from all the possible viewpoints within the lens aperture, it is called the plenoptic camera.

The advantages of depth recovery using the plenoptic camera to the systems discussed in the previous section are:

• Only one single lens camera is used in this system which has no moving parts.

• Results in multiple images of the scene with a single snapshot.
• Deriving depth information is simpler, because the correspondence problem is minimized.

• This camera extracts information about both horizontal and vertical parallax, which improves the reliability of the depth estimates.

The main concept behind the design of this camera lies in the fact that light which passes through the lens of a camera is quite rich in information which regular cameras were not designed to take advantage from. In an ordinary camera most of the information is lost at the final moment of image formation when all of the light rays are projected onto a single planar surface and each sensor element registers the average of all the light rays striking it from different angles.

The plenoptic camera keeps track of the 3D information of the light rays by distributing these rays over a two dimensional array of converging lenses which we call a *lenticular array*, causing the rays to be separated based on the angle of incident. The lenticules can be arranged in a rectangular or hexagonal lattice. The inventors [1] suggested a rectangular arrangement with spherical lenses which we use in this simulation.

The Plenoptic camera structure can be seen in figure (4),

![Figure 4: The Plenoptic Camera Structure](image-url)
In the plenoptic camera the main lens bends light rays towards a lenticular array in the image plane which is attached to an array of Charged Coupled Devices (CCD) in the backplane of the camera. A CCD element is a photo-detector semiconductor which generates an output voltage proportional to the intensity of light incident upon it. Behind each lenticule there is a CCD macro-pixel which is subdivided into a group of sub-pixels. Every lenticule captures a chunk of light of the main lens aperture and redirects it to the group of macro-pixels it is attached to. A lenticular array is usually made in such a thickness that its rear surface coincides with the focal plane of the lenses. As will be discussed in the next section, the number of lenticules defines the spatial resolution of the sub-images and the number of macro-pixels per lenticule defines the number of extracted sub-images (number of viewpoints). As can be seen this camera does not have any aperture or moving parts which is a great advantage.

2.3 Image Formation

The advantage of the Plenoptic camera is the capability of taking multiple images with a single snapshot. These images are formed as follows.

Suppose we have a 3 by 3 rectangular array of lenticules and each macro-pixel behind a lenticule is divided to 16 sub-pixels, as can be seen in figure(5). Sub-pixels $A_1, B_1, ..., P_1$ are the pixels behind the first lenticule which is positioned at the top left corner of the array, and $A_9, B_9, ..., P_9$ belong to the lenticule at the bottom right corner of the array. We form image $A$ by assembling all the subparts $A_1, A_2, ..., A_9$ together. Therefore the application of this procedure results in 16 different images, shown in figure (6), which could be obtained by an ordinary camera if the viewing position had been shifted slightly up, down, left or right. We call these images the Plenoptic Images. There was 16 sub-pixels behind each lenticule which results in 16 different images, and as can be seen the resolution of each image is 3 by 3 which is equal to the number of lenticules.
Number of lenticules determines the \textit{Spatial Resolution} of the received images, and the number of sub-pixels per macro-pixel determines the number of different views or the \textit{Motion Resolution} or \textit{Parallax Resolution} of the images.

\section*{2.4 Depth Measurement}

Now we describe how this camera captures the depth information of the scene. Each of the macro-pixels receives an image which contains light passed by the entire main lens. However, the light received by each macro-pixel is received from a different angular position. The role of the lenticular array is to separate these light rays in order to extract the 3D information which they carry.

Assume we are tracing the light rays from a point which is in the focal plane of the main lens of the plenoptic camera, as shown in figure (7). These rays come in focus on the surface of the lenticules, and because of this focusing the light is received by only one of the lenticules and there is no overlap of light between lenticules. As can be seen the light is evenly distributed among all three sub-pixels of the center lenticule and intensity of light received by each of the three sub-pixels across one dimension is the same. Now if we display the sub-images of this point, which is in focus, in sequence as an animation, the point will remain stationary. Because as can be seen in figure (7), \( B_x \) will become \( B_y \) and then \( B_z \), and all the three intensities are the equal.

Now if we trace the rays from a point which is beyond the focal plane, they come to a focus before reaching the lenticular array. Thus the light diverges past the focal point and spills over into the lenticules adjacent the center lenticule, and therefore light is received by more than one lenticule. Now if we display the sub-images of point \( A \) in sequence, \( A_x \) will become \( A_y \) and then \( A_z \). As can be seen the intensity of light decreases from left image to the right image. For point \( B \), intensity first increases then decreases, and for point \( C \), the intensity increases. Because these three points
Figure 5: Image Formation
$A$, $B$ and $C$ are adjacent in the images, if we look at the animation we will notice that somehow this point, which is beyond the focal plane, moves from left to right when we change the angle of view form left to right.

For a point which is closer than the focal plane, the rays do not converge before reaching the lenticules, and light is therefore received by more than one lenticule. In this case if we look at the animation we will notice that somehow this point moves from right to left when we change the angle of view form left to right.

Therefore depth of the object point can be determined by how much parallax exists relative to the focal plane of the main lens. The more an object appears to rearranging, changing position from one image to the next, the farther it is from the focal plane. Objects farther from the main lens than the focal plane appear to change position toward the right when viewing the images in the order of left image through right image (X through Z), And objects closer to the main lens than the focal plane appear to therefore, change position toward the left if the same viewing order is used.
Figure 7: Effect of Depth
CHAPTER III

MODELLING THE PLENOPTIC CAMERA
LENS SYSTEM

3.1 Overview

In this chapter we start the simulation of the plenoptic camera by simulating the plenoptic camera lens system. This process requires some knowledge of optics, like interaction of light and a surface, and image formation by a lens. Therefore we first briefly discuss some basic concepts in optics and in the last section we implement the simulation of the plenoptic camera lens system and present some results.

3.2 Surface Physics

There are two popular models for the nature of light; the wave model, and the particle model. The techniques of computer graphics and ray tracing are based mostly on the particle model, which essentially says that a light ray is the straight path of a particle of light which is called a photon.

In order to generate realistic images in computer graphics we need to understand how light behaves at the surfaces of objects. The way light interacts with a surface is very complex, as many techniques have been developed over the years to model it [13], [14]. We divide the interaction of light and a surface into four classes. specular reflection, diffuse reflection, specular transmission, and diffuse transmission. When a given photon hits a surface, it will undergo changes in direction and color as a result of these four effects. The amount of influence of each effect on the photon is mostly dependent on the surface material. But this material may behave differently for light
arriving at different frequencies and angles of incident.

3.2.1 Specular Surfaces

3.2.1.1 Specular Reflection

Most of the time when we deal with a very hard and shiny surface we model the reflection from this surface as a perfect specular reflection, which is not subject to any absorption or re-radiation. As an example, most of what we see in a mirror is the perfect specular reflection of the incoming light, or the highlights on a shiny surface are also an example of perfect specular reflection.

Figure (8) shows a photon arriving at a hard, shiny surface and bouncing off. \( N \) is the surface normal, which is the vector that indicates a direction perpendicular to the surface and points away from the object’s inside. \( \theta_i \) which is called the angle of incidence is the angle between the surface normal and the direction of the incoming ray \( I \). And \( \theta_r \), which is called the angle of reflection is the angle between the surface normal and the direction of the reflected ray \( R \). 

\( N \), \( I \) and \( R \) are all normalized vectors, it means \( \| N \| = \| I \| = \| R \| = 1 \).

![Figure 8: Geometry of a Specular Reflection](image)

Given \( N \) and \( I \) we wish to find \( R \). There are two ways to solve this problem, algebraic and geometrical. We present an algebraic solution.

Two physical laws help us find \( R \). The first is that \( I \), \( N \) and \( R \) are coplanar, so we
can write $R$ as a linear combination of $I$ and $N$.

$$R = \alpha I + \beta N$$  \hspace{1cm} (1)

The second principle is that in a perfect specular reflection the angle of incidence is equal to the angle of reflection.

$$\theta_i = \theta_r$$  \hspace{1cm} (2)

Therefore we will have an equality between the cosines of two angles and we can get the cosine by using internal product of two vectors.

$$\cos \theta_i = \cos \theta_r$$

$$-I \cdot N = N \cdot R$$

$$= N \cdot (\alpha I + \beta N)$$

$$= \alpha (N \cdot I) + \beta$$  \hspace{1cm} (3)

Therefore if we set $\alpha = 1$, and solve for $\beta$:

$$\beta = -2(N \cdot I)$$  \hspace{1cm} (4)

Substituting equation(4) into equation(1) gives:

$$R = I - 2(N \cdot I)N$$  \hspace{1cm} (5)

3.2.1.2 Specular Transmission

In a transparent object, like an optical lens, light can arrive from behind the object’s surface and pass through. Such light is called transmitted light.

To properly handle transmitted light, we need to handle the bending of the light as it crosses the interface between two media. This bending is called transmission or refraction. Each medium has an Index of refraction $\eta$, which actually describes the speed of light in that medium compared to the speed of light in a vacuum.

Figure (9) shows an incoming light ray $I$ striking a surface with normal $N$. The incident light makes an angle $\theta_i$ with the surface normal, which is again called the
angle of incidence. The transmitted light $T$ makes an angle of $\theta_t$ with the reflected normal in the second medium which we call the angle of refraction. $I$, $N$ and $T$ are again coplanar.

![Figure 9: Geometry of a Specular Transmission](image)

The equation relating the angles of incidence and refraction is called the Snell's Law [15]:

$$\frac{\sin(\theta_i)}{\sin(\theta_t)} = \eta_{12} = \frac{\eta_1}{\eta_2}$$  \hspace{1cm} (6)

where $\eta_1$ is the index of refraction of medium 1 with respect to vacuum. $\eta_2$ is the index of refraction of medium 2 with respect to vacuum. And $\eta_{12}$ is the index of refraction of medium 1 with respect to medium 2.

Like reflection, given $N$ and $I$ we wish to find $T$. Since $I$, $N$ and $T$ are coplanar, we can write $T$ as a linear combination of $I$ and $N$.

$$T = \alpha I + \beta N$$  \hspace{1cm} (7)

From the Snell's Law (6) we get:

$$(1 - \cos^2 \theta_i)\eta_{12}^2 = (1 - \cos^2 \theta_t)$$  \hspace{1cm} (8)
Which we can rewrite:

\[ (1 - \cos^2 \theta_i) \eta_{12}^2 - 1 = \cos^2 \theta_i \]

\[ = [-N \cdot T]^2 \]

\[ = [-N \cdot (\alpha I + \beta N)]^2 \]

\[ = [\alpha(-N \cdot I) - \beta]^2 \]

\[ = [\alpha(\cos \theta_i) - \beta]^2 \]  \hspace{1cm} (9)

We get the second condition on \( \alpha \) and \( \beta \) from the fact that \( T \) is normalized.

\[ 1 = T \cdot T \]

\[ = (\alpha I + \beta N) \cdot (\alpha I + \beta N) \]

\[ = \alpha^2 + 2\alpha\beta(I \cdot N) + \beta^2 \]

\[ = \alpha^2 - 2\alpha\beta(\cos \theta_i) + \beta^2 \]  \hspace{1cm} (10)

Now we can solve for \( \alpha \) and \( \beta \) from equations (9) and (10) and the final algebraic solution for \( T \) will be:

\[ T = \eta_{12} I + (\eta_{12} \cos \theta_i - \sqrt{1 + \eta_{12}^2 (\cos^2 \theta_i - 1)})N \]  \hspace{1cm} (11)

3.2.2 Diffuse Surfaces

3.2.2.1 Diffuse Reflection

Unfortunately the above derivation for specular reflection is only valid for hard, shiny surfaces. A rougher surface behaves in quite a different manner and the characteristics of the reflected light from such a surface do not have a simple geometry like the specular reflection.

Specularly reflected light is not subject to any absorption or re-radiation since it bounces off of the surface of the object. But diffusely reflected light actually interacts with the surface. When a photon is absorbed by an atom of the surface, the photon may be turned into heat or it may eventually be re-radiated. If the photon is re-radiated, there is nothing that determines in which direction the photon ought to
proceed. Although any given photon will go in only one direction, many photons over the course of time will tend to go in all possible directions. So the distribution of the reflected light from a diffuse surface is equal for all reflection angles as can be seen in figure (10).

![Diagram of Diffuse Reflection](image)

**Figure 10: Diffuse Reflection**

### 3.2.2.2 *Diffuse Transmission*

Like specular reflection, specular transmission holds only for a very small number of materials, like fine crystals. Usually when light travels from one medium to another, the particles of the second medium interfere with the travelling photons. Consider a material like plastic, it allows light to pass and colors it along the way. But it is not possible to clearly see anything on the other side of the plastic.

A perfectly diffuse transparent medium scatters light evenly in all directions as it passes through, just as a perfectly diffuse reflective surface scatters light in all directions as it is reflected. Therefore the intensity of diffusely transmitted light would be the same in all directions, as can be seen in figure (11).
3.3 Lenses and Cameras

3.3.1 Lens

A light ray passing through a lens undergoes two changes of trajectory, because of the refraction phenomenon that occurs due to the change of medium. As discussed before, we model these two refractions as perfect specular refractions which is governed by equation (11).

The lens geometry is illustrated in figure (12). Where $d$ is the lens thickness, $I$ is the incidence ray on the first surface of the lens, $N$ are the normal vectors, $R$ is the transmitted ray inside the lens and $T$ is the transmitted ray from the second surface of the lens.

Figure 11: Diffuse Transmission

Figure 12: Lens Geometry
One of the important properties associated to a lens is the *Focal length*, which is the distance from the lens to the focal plane when the lens is focused on infinity. The longer the focal length, the greater the magnification of the image.

### 3.3.2 Thin Lens

A *Thin lens* may be defined as one whose thickness is small in comparison with the distances generally associated with its optical properties. Such distances are, for example, the *primary and secondary focal lengths* and object and image distances. Focal points and image formation of a thin lens is shown in figure (13).

![Primary and secondary focal points of a thin lens](image1)

![Thin lens image formation model](image2)

**Figure 13:** Thin Lens

Every thin lens has two focal points, one on each side of the lens and equidistant from the center. *Paraxial rays*, which are parallel to the optical axis, approaching
from object space converge at the secondary focal point $F'$ after refraction through the lens. Similarly, paraxial rays from image space converge at the primary focal point $F$. The distance between the center of a lens and either of its focal points is its focal length.

The position of the image of an object is given by

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'}$$

(12)

where $s$ is the object distance, $s'$ is the image distance, and $f$ is the focal length, all measured from the center of the lens. Points $p$ and $p'$ in figure (13) are called the conjugate points.

3.3.3 Thick Lens

When the thickness of a lens cannot be considered as small compared to its focal length, the thin lens formulae are no longer applicable, and the lens must be treated as a Thick lens. The behavior of a thick lens is modelled by its focal points and principal planes, which is shown in figure (14).

Paraxial rays approaching from the object space converge at the secondary focal point $F'$ after refraction through the lens. Similarly, paraxial rays from image space converge at the primary focal point $F$. In either case, the intersection of incident and emergent rays defines a principal plane. There are thus two principal planes, primary and secondary, at which refraction is assumed to occur. So the same equation which was used for the thin lens can be applied here if the distances $s$ and $s'$ are measured from the principal points $H$ and $H'$ respectively.

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'}$$

(13)

The distance from $H$ to $H'$ is the effective thickness of the lens.
3.3.4 Lens System

Real cameras use a lens system [16] instead of a single thin or thick lens. In this case the thickness of lenses usually cannot be considered negligible, thus usually this lens system is approximated with a thick lens model.

A camera lens system usually consists of individual spherical glass lenses placed on a common optical axis, as can be seen in figure (15). The specification of a lens system is traditionally presented in a tabular format.

In order to find a thick lens approximation for a lens system, we trace a paraxial ray through the lens system, using equation (11). As can be seen in figure (15), the
The intersection of incident and emergent rays defines the secondary principal plane, we repeat the same process to get the primary principal plane. The intersection of the emergent rays and the optical axis of the lens system are the focal points. So all the parameters of an approximated thick lens model can be derived by this procedure.

After replacing the lens system with a thick lens approximation, ray tracing can be done by an algorithm introduced by Kolb [17],[18].

The lens is represented by a pair of principal planes. To trace a ray from \( x' \), a point on the lens is chosen. Then the intersection point with the principal plane \( H' \)
is translated parallel to the optical axis. The ray is then directed towards the image, or the conjugate point $x$. Ray tracing then proceeds in the usual manner.

### 3.3.5 Pinhole Camera

Pinhole camera model is probably the simplest camera model, but despite its simplicity, it is quite practical.

A flat piece of photographic film is placed at the back of a light-proof box. A pin is used to pierce a single hole in the front of the box, which is then covered with a piece of tape. When you wish to take a picture, you remove the tape for a while. Light will enter the pinhole and strike the film, causing a chemical change. The pinhole is a necessary part of the camera. If we remove the box and the pinhole and simply expose the entire sheet of film to the scene, light from all directions would strike all points on the film, saturating the entire surface and result in an overexposed (white) image. The pinhole eliminates this problem by allowing only a very small number of light rays to pass from the scene to the film. In particular, each point on the film can receive light only along the line joining that piece of film and the pinhole. As the pinhole gets bigger, each bit of the film receives more light rays from the scene, and the image gets brighter and of course more blurry.

Although more complicated camera models have been used in computer graphics, the pinhole camera model is still popular because of its simplicity and wide range of application.

For convenience in programming and modelling, we usually move the plane of the film out in front of the pinhole, and rename the pinhole as the eye, as shown in figure (16). Although we have moved things around, each component of the pinhole camera model is accounted for in this new model which we call the *Modified pinhole camera model* [19]. This makes the computer simulation a lot easier. And as can be seen the requirement that all light rays pass through the pinhole is translated into the
requirement that all light rays pass through the eyepoint.

\[\text{Figure 16: Modified Pinhole Camera Model}\]

When we generate an image we are basically determining what color to place in each pixel (Picture-Element) of the image. A small distribution of light rays can arrive from the scene, pass through the pinhole, and strike the film at a particular pixel. After the exposure has completed and the pinhole is covered, that pixel has absorbed many different rays of light. If we wish to describe the entire pixel with a single color, a good approximation might be to simply average together all the colors of all the light rays that struck it.

In order to form these light rays, we need to start from the light sources, follow these rays through the scene and finally through the pinhole of our camera. We call this approach Ray Tracing, because we follow (or trace) the path of a photon (or a light ray) as it bounces around the scene. Usually the light gets a little dimmer on each bounce, and after a couple of bounces the light is so dim it will not have any effect in the final color of the pixel.

More specifically, what we discussed was Forward Ray Tracing, that is we followed photons from their origin at the light source and into the scene, tracing their path in
a forward direction, just as the photons themselves would have travelled it.

Although forward ray tracing seems to be a good idea, it has a problem. The problem is the amount of time it would take to produce the image. Because many of the thousands of photons which originate from the light source do not reach the pinhole and therefore the image plane at all. Most of these rays hit the objects and fly in another direction or go out of the scene, and we end up with a very small number of rays with a very low intensity inside the camera. This way it might take years to make one dim picture!

Therefore almost all of the ray tracing packages do Backward Ray Tracing [20]. In this approach we are following rays not forward, from the light source to objects to the eye, but backward, from the eye to objects to the light source. So we restrict our attention to rays that we know will be useful to our image.

For example backward ray tracing of a light ray is shown in figure (17). We start by forming an eye ray, which is a ray beginning from the eye and passing through a particular pixel. This rays bounces 3 times in the scene till it reaches a light source.

![Diagram of Backward Ray Tracing](image)

**Figure 17:** Backward Ray Tracing
3.4 **Plenoptic Camera Lens System Simulation**

3.4.1 Implementation

In order to simulate the plenoptic camera lens system, we start by generating the initial rays and tracing them through the lens system. As will be fully discussed in the next chapter, we use stochastic sampling to get the sample points in the image plane. Each pixel is subdivided to sub-pixels and random sample points are taken from each sub-pixel. The plane on the edge of the lenticules is also sampled the same way, as can be seen in figure (18).

![Image](image.png)

**Figure 18:** Sampling the Lenticular Array

Then initial rays are formed by linking the pixel sample points to the sample points on the plane at the edge of the lenticules. If we link the first image pixel to the first sample on the second plane, this would lead to correlation and artifacts. So Shirley [21] suggested linking these two sets of samples together randomly as can be seen in figure (19). $(U, V)$ are the sets of image sample points, and $(A, B)$ are the sets of sample points on the second plane.

Now these rays are traced through the camera main lens and finally through the scene. For now we assume there are no light sources in the scene and we only find the intersection of the rays with the closest object in the scene and return the color of that point. In the next two chapters we implement a ray tracing algorithm which completely calculates the distribution of light within the environment.

The simulation process is illustrated in figure (20).
Figure 19: Generating the Initial Rays

3.4.2 Results

In this section we present the results of the plenoptic camera lens system simulation. The scene is a room with two objects (sphere) in it, with no light source.

For the main lens, a real lens system was used which is shown in figure (21). The lens which is selected from LensView [22], a database of the optical designs found in the United States and Japanese patent literature, is designed by Gilbert Dey for Kodak company, with the patent number 1160148. A tabular description of the lens is shown in figure (22). Each row in the table describes a surface of a lens element. Surfaces are listed in order from the front (nearest object space) to rear (nearest image space), with measurements given in millimeters. The first column gives the signed radius of curvature of a spherical element. A positive radius of curvature indicates a surface that is convex when viewed from the front of the lens, while a negative radius is concave. The next entry is thickness, which measures the distance from this surface to the next surface along the central axis. The last column is the index of refraction of the medium. This lens was replaced with its thick lens approximation in the simulation process.

For the lenticular array, an array of 50 by 50 lenticules was used. With the
thickness of the lenticules equal to 2mm, and radius of curvature equal to 10mm. The CCD sensor behind the array has a resolution of 250 by 250.

The whole plenoptic image before extracting the sub-images (illustrated in figure (5)), which we call the raw plenoptic image, is shown in figure (23). With 100 samples per pixel, rendering time was about 20 minutes.

The extracted plenoptic sub-images (illustrated in figure (6)) can be seen in figure (24). If we take the sub-images in a row or column and display them in sequence as an animation, the displacement can be seen clearly.
Figure 20: Plenoptic Camera Lens System Simulation Flowchart
Figure 21: A Lens System

<table>
<thead>
<tr>
<th>Radius</th>
<th>Thickness</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.802309</td>
<td>2.120206</td>
<td>1.61</td>
</tr>
<tr>
<td>-66.116441</td>
<td>1.710166</td>
<td>AIR</td>
</tr>
<tr>
<td>-36.143512</td>
<td>1.060103</td>
<td>1.58</td>
</tr>
<tr>
<td>36.143512</td>
<td>1.765171</td>
<td>AIR</td>
</tr>
<tr>
<td>INFINITY</td>
<td>1.765171</td>
<td>AIR</td>
</tr>
<tr>
<td>-36.143512</td>
<td>1.060103</td>
<td>1.58</td>
</tr>
<tr>
<td>36.143512</td>
<td>1.710166</td>
<td>AIR</td>
</tr>
<tr>
<td>66.116441</td>
<td>2.120206</td>
<td>1.61</td>
</tr>
<tr>
<td>-23.802309</td>
<td>93.623771</td>
<td>AIR</td>
</tr>
</tbody>
</table>

Figure 22: Tabular Description of the Lens System
Figure 23: Plenoptic Image

Figure 24: Extracted Plenoptic Images
CHAPTER IV

RAY TRACING

4.1 Overview

Ray tracing is a technique for Image Synthesis: Creating a 2-D picture of a 3-D world. Image synthesis techniques rely on simulating the distribution of light around an environment in order to calculate how much light reaches a particular pixel of the image.

In this chapter we review the basis of a ray tracing algorithm and in the subsequent chapter we detail the implementation of a Path Tracer which uses the idea of Markov Chains [23] to solve the Rendering Equations [24]. A Markov Chain is a sequence of random samples where each sample is probabilistically determined by the previous one in the sequence. Rendering Equation is a formula describing the Global illumination.

4.2 Global Illumination

The most important part of a ray tracing algorithm is solving the Global illumination problem, that is, correctly calculating the distribution of light within an environment, taking into account all forms of scattering, absorption and inter-reflection which commonly occur. In Global illumination we deal with the interaction of light which reaches a surface directly from a light source (Local illumination) as well as the interaction of light which reaches a surface as a result of scattering or transmission from or through other surfaces.

The distribution of light is governed by the position, strength and nature of the light sources in the scene, by the geometry of the objects in the scene along with their associated properties, and by how light interacts with the objects. This interactions
can be in the form of reflection, transmission and absorption, but usually reflection is the dominant mode of interaction. To determine the intensity of light which reaches a particular pixel, we must first determine which point on which object is visible through that pixel and then determine how much light is reflected off this point in the direction of the eye through that pixel. In order to determine how much light is reflected from that point we must know the amount of the light which hits that point and then provide some means of calculating how much of this light is reflected in the appropriate direction. In order to calculate this we first need to find a model for the reflection characteristics of a surface.

4.3 Surface Reflection Models

Bidirectional reflectance distribution function (BRDF) is a function which describes the spectral and spatial reflection characteristics of a surface. It is the ratio of reflected radiance to incident radiance at a particular wavelength.

Radiance $L$ can be defined as the power arriving at or leaving from a surface in a certain direction, per unit solid angle around that direction, per unit projected area perpendicular to the direction of travel as can be seen in figure (25).

The directional property of the energy emission is described in a so-called Illumination hemisphere which contains those solid angles to where the surface point $p$ can emit energy. A solid angle $d_w$ is a cone or a pyramid, with its size determined by its subtended area of a unit sphere centered around the apex. The solid angle, in which a differential surface $d_A$ can be seen from point $p$, is the projected area per the square of the distance of the surface. If the angle between the surface normal of $d_A$ and the directional vector from $d_A$ to $p$ is $\theta$, and the distance from $d_A$ to $p$ is $r$, then this solid angle $d_w$ is:

$$d_w = \frac{d_A \cdot \cos \theta}{r^2}$$ (14)
The *Radiance* $L$, is the differential light flux $d\Phi$ leaving a surface element $dA$ in a differential solid angle $d\omega$ per the projected area of the surface element and the size of the solid angle. Light power or *Flux* $\Phi$ is the energy radiated through a boundary per unit time over a given range of spectrum (say $[\lambda, \lambda + d\lambda]$)

$$L = \frac{d\Phi(d\omega)}{d\omega \cdot dA \cdot \cos \theta}$$  \hspace{1cm} (15)

Now we can define a BRDF $f_r(x, \theta_i, \theta_o)$ as a function that is associated with a surface and when given an incoming radiance returns an outgoing radiance, where $x$ is a point on the surface, $\theta_i$ is the incoming direction, and $\theta_o$ is the outgoing direction. For example the *Phong local illumination model* [14] which is being used by many graphic packages is a simplified version of a BRDF. Later we use a modified version of this model for our BRDF.
4.4 **Rendering Equations**

In 1986 Kajiya [24] proposed a formula in the form of an infinitely recursive integral equation to describe the global illumination.

\[
L(x, \Theta_x) = L_e(x, \Theta_x) + \int_{\Omega_x} L(y, \Theta_y) f_r(x, \Theta_y, \Theta_x) |\Theta_y . N_x| d\omega_y
\]  

(16)

where \(L(x, \Theta_x)\) is the radiance from point \(x\) in some direction \(\Theta_x\), so if \(x\) is a point in the scene that we are looking at, then this term is the radiance coming from this point in the direction of the eye and therefore is the color of the pixel we are interested in. \(N_x\) is the surface normal at point \(x\) and \(L_e(x, \Theta_x)\) is the emitted surface radiance in direction \(\Theta_x\) at point \(x\). This will be zero if the surface is not an emitter itself. The integral expresses the radiance contribution of the secondary emitters. The domain of the integral \(\Omega_x\) is the hemisphere of directions above the surface at point \(x\). This represents all the possible incoming directions and for each one we need to calculate what the incoming radiance is, sum them together and use the BRDF of the surface \(f_r\) to calculate the outgoing radiance in the direction \(\Theta_x\). But in order to calculate the incoming radiance from point \(y\) along direction \(\Theta_y\) we need to solve an integral equation of the same form as above and hence the whole process is *recursive*.

In order to solve the global illumination problem, we need to find a solution to this infinitely recursive integral. We first review some of the solutions and then in the next section we talk about the *Monte Carlo Integration* [25] which is used in the *Path Tracing* algorithm. Monte Carlo methods provide means of solving the rendering equation completely and hence provide a basis for implementing physically accurate lighting simulation and rendering systems.

Traditional image synthesis techniques try to simplify the rendering equation. In *Whitted Ray Tracing* [26] the BRDF is replaced with the simple phong model and the hemisphere of directions is replaced by a finite set of directions which are the directions towards the visible light sources and the perfect specular reflection direction.
Infinite recursion is avoided by placing a limit on how far the process can recurse. *Radiosity* [27] treats the environment as a discrete set of patches rather than a continuous set of points. So the integral is replaced by a summation. This technique assumes that all the surfaces are perfectly diffuse Lambertian reflectors, as a result a complete BRDF model is not used in this method. Therefore Radiosity ignores specular reflection completely and concentrates on modelling the diffuse inter-reflections, while whitted method models the specular reflection and inter-reflection successfully but is limited to only direct lighting for the diffuse component of a surface.

### 4.5 Monte Carlo Integration

*Monte Carlo methods* [28] are techniques for computing answers to problems using the properties of *random* numbers. It was first used to compute the value of π in 1873. After World War II it was used to simulate the flight paths of neutrons in design of thermonuclear weapons (H-Bomb), and later during 50’s and 60’s were applied to a wide variety of physical and mathematical problems. They can be used to carry out simulations of problems which themselves have a probabilistic nature or they can be applied to problems for which solutions are impossible or difficult to obtain using analytical and numerical techniques.

One of the applications of Monte Carlo methods is as a numerical means of evaluating integrals. For example suppose we have some function that we wish to integrate over the range [0,1].

\[
I = \int_{0}^{1} f(x) dx
\]  

(17)

We can estimate the value of this integral by taking a uniform random number \( r \) between [0,1] and evaluating \( f(r) \). This is called a primary estimator for the integral and when evaluated for a specific value of \( r \) it forms an estimate for the value of the
integral. The basis of Monte Carlo methods is to make this estimate more accurate by taking numerous primary estimators and combining them into a secondary estimator by averaging them. So the secondary estimator is the average of \( N \) primary estimators:

\[
I_{sec} = \frac{1}{N} \sum_{i=1}^{N} f(r_i)
\]  

(18)

The accuracy of the result depends on how many samples are taken. When the number of samples approaches infinity the answer converges to the correct result. If we take \( N \) primary estimators, the variance of the secondary estimator is reduced by a factor of \( \frac{1}{N} \). So in order to reduce the variance of the secondary estimator, we need to reduce the variance of the primary estimators, which is largely dependent on the distribution of the samples. We can do this by importance sampling [25], which is taking more samples from regions of the function which have higher values and hence affect the value of the integral more than those with low values. Hence we carry out importance sampling by distributing the samples according to a Probability Density Function (PDF) that has a roughly similar shape to the integrand.

*Probability Distribution Function* for a random variable \( y \) is a function \( P(y) \) which gives the probability that the value of the random variable is less than or equal to \( y \). If \( P(y) \) is continuous over the domain of the random variable, then the *Probability Density Function* (PDF) is the derivative of the distribution. Therefore integrating the PDF between a range \([a, b]\) in the domain of the random variable gives us the probability that the random variable will fall within this range:

\[
P(a < y \leq b) = \int_{a}^{b} p(x)dx
\]  

(19)

In order to distribute the samples according to our PDF, we use the inverse of the probability distribution function \( P(x) \), associated with our PDF, \( p(x) \). we first select a random sample \( r \) then our new sample will be:

\[
x = P^{-1}(r)
\]  

(20)
This is represented in figure (26).

![Figure 26: Importance Sampling](image)

The shape of our PDF is similar to the function $f(x)$. By taking the integral of $p(x)$ we find our $P(x)$ function and inverse of $P(x)$ gives us the new samples. As can be seen $r$ samples are random, but the new samples $x$ are grouped in the area where our function has higher values.

In order to get the correct convergence the primary estimators will be:

\[ I_{imp} = \frac{f(r)}{p(r)} \]  \hspace{1cm} (21)

and the second estimator:

\[ I_{impsec} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(r)}{p(r)} \]  \hspace{1cm} (22)

The success of this approach is highly dependent on the choice of the appropriate PDF which should be as close as possible in shape to the function being integrated. The other requirements are: It should be positive in the domain $[0, 1]$. Integral of
the PDF in the domain $[0, 1]$ must be equal to 1, and inverse of the $P(x)$ function associated to the PDF should exist.

As can be seen Monte Carlo methods provide a feasible means of solving integral equations. The important advantage of these methods is that the convergence is not affected by the dimensionality of the integral. The integrals we will be dealing with exhibit high dimensionality. The radiance itself is a five-dimension function since it is defined at a 3-D point in space and it also has a direction. We will also be sampling over pixels which further increases the dimensionality of the problem by two. In the next section we review the sampling techniques which are used to sample the pixels.

4.6 Sampling Techniques

One of the most important objectives in ray tracing is to ensure that the final color of a pixel correctly represents the objects that contributed to it. In order to achieve this, we should use the appropriate sampling techniques.

4.6.1 Aliasing

When a signal is undersampled, high-frequency components of the original signal can appear as lower frequency components in the sampled version. These high-frequencies assume the alias (or false identity) of the low frequencies, because after sampling these very different phenomena cannot be distinguished.

In ray tracing due to the uniform nature of the pixel grid, we often face the Spatial Aliasing problem which shows itself in two forms. First there will be jaggies around object edges. Second small objects can fall between rays so that the rays miss these small objects entirely. No matter how closely the rays are packed, they can always miss a small object or a large object farther away. The solution is instead of casting only one ray per pixel we cast multiple rays for each pixel and then we find the color at each individual pixel by averaging the colors of all the rays within that pixel. This is called Anti-aliasing. In the following sections we review some of these techniques.
which reduce the aliasing problem.

4.6.2 Uniform Supersampling

The simplest way in which a pixel can be sampled is by uniformly distributing the ray positions across that pixel. For example instead of casting one ray through the center of the pixel we can cast four rays through the corners. Unfortunately because of the uniform sampling we still have the jaggies along object edges or intersections, specially those which are regularly shaped. Irregular shapes are by their nature sampled correctly by this method, but since most scenes contain a large number of objects with straight edges, a different method must be considered.

4.6.3 Adaptive Sampling

Another way is to cast more rays for areas which are rich in information or have a large color variance. Some areas of an image may be quite uniform in their color, and so shooting a large number of rays is not necessary to get the desired picture quality. It would be a great idea to spend more time on the interesting portions of an image, such as edges of the objects and shadow regions. Adaptive sampling works by first casting a certain number of rays per pixel, examining the variance across that pixel, and using it to decide which portion of the image requires extra sampling.

For example we start casting five rays per pixel, one through each corner and one through the center. If each of the five rays is about the same color, we assume that they probably hit the same object, and we will just use their average color for this pixel. If the rays have sufficiently different colors, then we will subdivide the pixel into smaller regions. Then we’ll treat each smaller region just as we did the whole pixel. Because this technique subdivides where the colors change, it adapts to the image in a pixel, and is thus called Adaptive sampling [19]. An example of the process is shown in figure (27).

This technique often works fairly well, but the problem that persists is the issue
of small objects, which can still slip through the initial five rays.

Another problem is that the nature of Path Tracing, which is the ray tracing algorithm we used. It is difficult to compare two paths in order to see if the variance between them warrants more sampling, because in path tracing they may represent different types of paths. For example one path may have followed diffuse reflection at the first intersection, while the adjacent path may have followed specular reflection. This is not so much of a problem for surfaces which are well lit, because the variance will not be large, however when looking at a surface which is lit by indirect light alone, it may become apparent. Therefore we choose the next sampling technique which does not have these problems and solves aliasing problem better.

4.6.4 Stochastic Sampling

As we saw, adaptive sampling still ends up sending out rays on a regular grid, even though this grid is somewhat more finely subdivided in some places than in others. Thus we can still get popping edges, jaggies and all the other aliasing problem that regular grids give us, although they will usually be reduced. Another solution is to get rid of the fixed grid, but continue using a fixed number of samples per pixel, which are scattered evenly across the pixel. If each pixel gets covered with these rays in a
different pattern, then we have successfully eliminated any regular grid. Now that we have removed the regular sampling grid, regular aliasing artifacts is also eliminated. Because we are randomly (or stochastically) distributing the rays across the pixel, this technique is called *Stochastic Sampling* [29],[30].

If we take the whole domain of the pixel and randomly choose a number of rays through it, we will end up with a concentration of rays in certain parts of the pixel domain which result in badly sampling of another area. So the best way is to split the domain into a number of sub-pixels, and to choose randomly inside these sub-pixels, as can be seen in figure (28).

![Image of One Pixel with 16 Samples and A 3X3 Pixels Image]

**Figure 28:** Stochastic Sampling

We first select the number of sub-pixels based on the required resolution, then each sample point is placed in the center of a sub-pixel, and then noise is added to the $x$ and $y$ locations independently so that each sample point occurs at some random location within its sub-pixel. This technique is also called *Stratified Sampling* or *Jittering* [21]. Finally because splitting of the domain is uniform, the final pixel color may again be found by taking the average of the sub-pixels.

But although stochastic sampling solves many of the problems of regular ray tracing,
we have picked up something new: *Noise!*. This noise spreads out over the whole picture, but it turns out that the human visual system is much more forgiving of this form of random noise than the regular aliasing problems like the jaggies, so in this way stochastic sampling is a good solution to aliasing problems.

### 4.7 Texture Mapping

*Texture Mapping* [31] is a shading technique for image synthesis in which a texture image is mapped onto a surface in a three dimensional scene, much as wallpaper is applied to a wall.

#### 4.7.1 Inverse Mapping

In image texture mapping, we take an image in a form such as BMP or GIF and map the pixels of this image to the 3D points on the surface of the object, which is called *Inverse Mapping*. So each 3D point \((x, y, z)\) is mapped to a pixel \((u, v)\) of the texture image.

##### 4.7.1.1 Sphere Inverse Mapping

The parametric equation of a sphere of radius \(R\) and with center \((x_c, y_c, z_c)\), in polar coordinates is:

\[
\begin{align*}
x &= x_c + R \cos \phi \sin \theta \\
y &= y_c + R \sin \phi \sin \theta \\
z &= z_c + R \cos \theta
\end{align*}
\]  

(23)

The \((\theta, \phi)\) can be found using:

\[
\begin{align*}
\theta &= \arccos \left( \frac{z - z_c}{R} \right) \\
\phi &= \arctan \left( \frac{y - y_c}{x - x_c} \right)
\end{align*}
\]  

(24)
Because $0 \leq \theta \leq \pi$ and $-\pi \leq \phi \leq \pi$, we convert to $(u,v)$ as follows:

After first adding $2\pi$ to $\phi$ if it is negative:

$$u = \frac{\phi}{2\pi}$$
$$v = \frac{\pi - \theta}{\pi}$$

(25)

Where $0 \leq (u,v) \leq 1$. Now to get the pixel location in the image texture, we multiply $(u,v)$ by the height and width of the texture image.

$$i = \lfloor u \cdot width \rfloor$$
$$j = \lfloor v \cdot height \rfloor$$

(26)

Sphere inverse mapping is shown in figure (29).

![Figure 29: Sphere Inverse Mapping](image)

### 4.7.1.2 Disk Inverse Mapping

The inverse mapping of a disk is mostly just a problem of converting from cartesian to polar coordinates. Consider a circle laying on the $XY$ plane with its center at the origin and a radius $R$.

$$x_e^2 + y_e^2 = R^2$$

(27)
Also given is an intersection point \((x, y, z)\) which lies on the \(XY\) plane. \((u, v)\) can be derived by converting to the polar coordinates:

\[
\begin{align*}
    v &= \sqrt{\frac{x^2 + y^2}{R^2}} \\
    u' &= \frac{\arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right)}{2\pi}
\end{align*}
\]

(28)

If \(y < 0\) then \(u = 1 - u'\), else set \(u = u'\).

Disc inverse mapping is shown in figure (30).

\[\text{Figure 30: Disk Inverse Mapping}\]
CHAPTER V

PATH TRACING

5.1 Overview

In this chapter we detail the implementation of Path Tracing [23]. First we start with the solution to Monte Carlo integration in the Path Tracing algorithm and then we present the implementation of the Pure Path Tracer. Later we improve the results by adding the direct lighting from the light sources and also adding the specular component to the objects of the scene.

As we discussed earlier Kajiya [24] proposed a solution to the Rendering Equation which involved setting up a Markov chain through the scene. This implies following a ray through the scene, but unlike ray tracing at each intersection only one ray is followed, creating a Path. The type of the ray (specular or diffuse) and its direction is determined by characteristics of the material and some random factor.

Path Tracing solves the rendering equation completely. It is a generalization of the Whitted [26] ray tracing algorithm. In Whitted ray tracing we follow rays backward according to the specular reflection angle. This means that we follow Specular light transport chains. In path tracing we follow the same algorithm, but at each intersection point we choose random directions to follow. By doing this we construct random walks through the scene and compute the light transport along these walks.
5.2 Solution to Monte Carlo Integration

McCabe [23] solved the rendering equations by using Markov Chains. The rendering equation is a Fredholm Integral of the Second Kind.

\[ f(x) = g(x) + \int_0^1 K(x, y) f(y) dy \quad (29) \]

where \( g(x) \) is a known function and \( f(x) \) is unknown. \( K(x, y) \) is the Kernel operator. \( f(y) \) will be an equation of the same form, as shown below.

\[ f(y) = g(y) + \int_0^1 K(y, y') f(y') dy' \quad (30) \]

So the process is recursive. At each point we take a sample point \( \epsilon_i \) over the interval [0,1] according to some Probability Density Function PDF \( p_i(x) \). so an estimator can be derived as follows:

\[ I = \sum_{i=0}^{\infty} \left( \prod_{j=1}^{i} \frac{K(\epsilon_{j-1}, \epsilon_j)}{p_j(\epsilon_j)} \right) g(\epsilon_i) \quad (31) \]

where \( \epsilon_0 = x \). The series of \( \epsilon_1, \epsilon_2, \ldots \) is called a Markov Chain because the PDF for each point depends only on the previous point. This is also known as a Random Walk [32]. In practice we cannot extend a random walk to infinity, therefore we need some methods of terminating the walk. This cannot be done by simply terminating at some fixed number of recursions because this introduces bias. The usual way to deal with this situation is to employ some form of Russian Roulette. This means that the paths are terminated probabilistically and results in an unbiased estimator.

Considering the rendering equation from the previous chapter:

\[ L(x, \Theta_x) = L_e(x, \Theta_x) + \int_{\Omega_x} L(y, \Theta_y) f_r(x, \Theta_y, \Theta_x) |\Theta_y.N_x| d\omega_y \quad (32) \]

Where \( \Omega_x \) is the hemisphere of incoming directions around x. \( \Theta \) represents the direction and \( f_r \) is the Bidirectional Reflectance Distribution Function (BRDF), this equation has the same form of equation (29) so can be solved using (31), when we present the path tracing algorithm.
5.3 Pure Path Tracing

In this section we implement the Pure Path Tracing, by assuming that all the surfaces are Lambertian (perfectly diffuse) and we don’t consider the effect of direct lighting. As mentioned before, the difference between path tracing and ray tracing is that random directions are chosen rather than a fixed direction (specular reflection angle). By choosing random directions we are sampling the hemisphere above the point according to some PDF. By doing this we build up a random walk through the scene and these walks will only contribute to the pixel in question if they eventually hit a light source.

for example a path of depth 3 is shown in figure (31).

\[ I(x_0, \Theta_{x_0}) = \left( \prod_{i=1}^{n} \frac{f_r(x_i, \Theta_{x_i}, \Theta_{x_{i-1}}) \cos(\theta_{x_i})}{pdf(\Theta_{x_i})} \right) L_s \]  

\( L_s \) is the radiance of the light source which was struck after \( n \) bounces and the \( pdf(\Theta_{x_i}) \) is the PDF which was used to sample the hemisphere above the point \( x_i \).
we also replaced \(|\Theta_y, N_x|\) with \(\cos(\theta_x)\). Now we should define BRDF and PDF.

For BRDF we use *Phong* model for pure lambertian surfaces. Such a surface reflects light equally in all directions. Later we will use *Modified Phong Model* when we add specularity properties to the surfaces. So for now BRDF can be expressed as:

\[
f_r(x, \Theta_i, \Theta_o) = \frac{k_d}{\pi} \tag{34}
\]

Where \(k_d\) is the *diffuse reflection coefficient*. By using only one \(k_d\), we get monochromatic images. Since we need color images with RGB color model we should have three diffuse reflection coefficients, \(k_{dr}, k_{dg}\) and \(k_{db}\), and at each point we carry out three separate independent calculations.

The PDF we use to sample the hemisphere is a simple cosine distribution.

\[
pdf(\Theta) = \frac{\cos \theta}{\pi} \tag{35}
\]

\(\theta\) is the angle between the direction \(\Theta\) and the surface normal. This means that, if we consider the cylindrical coordinate system with the object normal as the \(z\) axis, \(\theta\) will be the polar angle. So with this PDF the probability of sampling a direction when \(\theta\) is near \(\frac{\pi}{2}\) is very small. Lafortune [33] calculated the direction based on this PDF.

In polar coordinates system, directions are two-dimensional vectors. Therefore PDFs can be sampled by selecting 2 uniform stochastic variables \(\xi_1\) and \(\xi_2\) over the interval \([0,1]\). In term of polar angles \((\theta, \phi)\), where \(\theta\) is the polar angle with the surface normal and \(\phi\) is the azimuthal angle:

\[
(\theta, \phi) = (\arccos \sqrt{\xi_1}, 2\pi \xi_2) \tag{36}
\]

In terms of \((x,y,z)\):

\[
(x, y, z) = (\sqrt{1 - \xi_1} \cos(2\pi \xi_2), \sqrt{1 - \xi_1} \sin(2\pi \xi_2), \sqrt{\xi_1}) \tag{37}
\]

But this direction is defined in the object coordinate system, we need to change it to the camera coordinate system. The process is detailed in the Appendix(A).
Now if we replace the terms from equation (34) and equation (35) into equation (33) we get:

\[ I(x, \Theta_x) = \left( \prod_{i=1}^{n} k d_i \right) L_s \]  
(38)

This gives the main calculation for the Pure Path Tracing for one color. Three independent calculations need to be done at each point for RGB.

We implemented the algorithm and obtained some results. The images we obtained from pure path tracing were a bit dark. This is because in this algorithm only paths which eventually hit a light source will contribute to the color of the pixel and the probability of hitting a light source with a path is very low. Most paths which are extended through the pixel do not return anything and hence they are black. If we increase the resolution (number of samples per pixel) or increase the maximum path depth we will get better results but the factor which really improves the results is adding the direct lighting effect to the scene.

### 5.4 Adding Direct Lighting

In order to add the effect of direct lighting along the path, McCabe [23] divided the hemisphere of directions above \( x \) into two parts. One part is composed of all those directions which lead to light sources and the other is composed of those directions which do not. So we have two integrals which can be evaluated independently and therefore we can use different PDFs in each case.

#### 5.4.1 Direct lighting calculations

We therefore need to calculate the reflected radiance from a point \( x \) in direction \( \Theta_x \) as a result of incoming radiance coming directly from light sources. To calculate this, we rewrite the rendering equation and we get:

\[ L(x, \Theta_x) = \int_{\Omega} g(x, y) L_e(y, \Theta_y) f_r(x, \Theta_y, \Theta_x) \Theta_y, N_x d\omega_y \]  
(39)
where \( g(x, y) \) is a visibility function which gives 0 if the path from \( x \) to \( y \) is occluded and 1 otherwise. This term is evaluated by casting a ray, which is often called a shadow feeler or shadow ray from \( x \) to \( y \). Now we need to take some sample points from the surface of the light source and we change the domain of the integration from all the directions in the hemisphere to the surface points of the light source:

\[
L_D(x, \Theta_x) = \int_y g(x, y) L_c(y, \Theta_y) f_r(x, \Theta_y, \Theta_x)|\Theta_y, N_x| \frac{|\Theta_y - N_y|}{\|x - y\|^2} dA(y)
\]

(40)

This represents the change of the integration domain from \( \Omega \) to \( y \). \( N_x \) is the surface normal and \( N_y \) is the normal to the light source surface. As can be seen in figure (32), if \( \theta_1 \) is the angle between \( N_x \) and the line from \( x \) to \( y \), \( \theta_2 \) the angle between \( N_y \) and the line from \( y \) to \( x \), \( d \) the distance from \( x \) to \( y \), and \( I_s \) the intensity of the light source in any direction, we get:

\[
L_D(x, \Theta_x) = \int_y g(x, y) f_r(x, \Theta_y, \Theta_x) I_s \cos \frac{\theta_2}{d^2} dA(y)
\]

(41)

Figure 32: Direct Lighting

Since it is an integral equation over the surface area of the source we can use monte carlo integration to solve this by picking random sample point \( y \) on the surface
according to some PDF and averaging the results. So by choosing the phong model for BRDF and N samples on the light source surface, we get:

\[
L_D(x, \Theta_x) = \frac{I_s k_d}{\pi N} \sum_{i=1}^{N} g(x, y_i) \cos \theta_1 \frac{\cos \theta_2}{pdf(y)d^2}
\]  

(42)

5.4.2 Sampling light sources

Sampling a particular type of light source involves choosing a PDF for the surface of the light source and picking sample points according to this PDF. The light sources we have are Spherical light sources and Disk light sources.

5.4.2.1 Spherical light sources

The simplest form of PDF for a spherical light source is a uniform distribution on the surface of the sphere.

\[
pdf(y) = \frac{1}{4\pi r^2}
\]  

(43)

Putting this into equation(42) gives:

\[
L_D(x, \Theta_x) = \frac{4r^2 I_s k_d}{N} \sum_{i=1}^{N} g(x, y_i) \cos \theta_1 \frac{\cos \theta_2}{d^2}
\]  

(44)

In order to get the sample points, Shirley [34] suggested generating two random numbers \(\varepsilon_1\) and \(\varepsilon_2\) to get the sample point in the polar coordinates system:

\[
(\theta, \phi) = (\arccos(1 - 2\varepsilon_1), 2\pi \varepsilon_2)
\]  

(45)

Converting to Cartesian coordinates:

\[
y = \begin{pmatrix}
Cx + 2r \cos(2\pi \varepsilon_2) \sqrt{\varepsilon_1(1 - \varepsilon_1)} \\
Cy + 2r \sin(2\pi \varepsilon_2) \sqrt{\varepsilon_1(1 - \varepsilon_1)} \\
Cz + r(1 - 2\varepsilon_1)
\end{pmatrix}
\]  

(46)

Where \((Cx, Cy, Cz)\) is the center of the light source and \(r\) is the radius. We should only choose the sample points which are on the part of the sphere facing the object sample point and reject the others which are on the opposite side of the sphere. And
increasing the number of sample points on the light source $N$ will result in softer shadows.

5.4.2.2 Disk light sources

For a Disk light source, a uniform distribution has the following PDF.

$$\text{pdf}(y) = \frac{1}{\pi r^2}$$ (47)

Putting this into equation (42) gives:

$$L_D(x, \Theta_x) = \frac{r^2 I_s k_d}{N} \sum_{i=1}^{N} g(x, y_i) \cos \theta_i \frac{\cos \theta_2}{d^2}$$ (48)

To choose a random sample from a disk, we suppose that its center is at $(Cx, Cy, Cz)$, the radius is $r$, and one of the coordinate system axis is perpendicular to the disk plane. For example if $Z$ is the disk normal, a sample point on the disk surface in cartesian coordinates is:

$$y = \begin{pmatrix}
Cx + r \sqrt{\varepsilon_1} \cos(2\pi \varepsilon_2) \\
Cy + r \sqrt{\varepsilon_1} \sin(2\pi \varepsilon_2) \\
Cz
\end{pmatrix}$$ (49)

where $\varepsilon_1$ and $\varepsilon_2$ are two random numbers over the interval [0,1].

5.4.3 Precision Problem

During implementation of the Direct Lighting, we encountered a precision problem which arises because of the floating-point calculations. The problem is that sometimes when we cast a shadow ray from a particular surface, the ray hits the same surface and the surface somehow shadows itself.

As you see in figure (33) due to precision problems the calculated intersection of the ray and the surface is beneath the surface, so when a shadow ray starts from this point it hits the same surface and therefore the surface is in shadow. This precision problem will result in appearance of blotches and spots on the surface.
A solution is to move the intersection point outside the surface as needed. This can be done by moving each new ray's origin along the surface normal until it is found to be on the proper side of the surface.

### 5.5 Adding Specularity

In this section we add the specularity characteristics of the materials to the path tracer which affects both the indirect and direct lighting. It also affects the direction of the outgoing ray. In order to achieve this we use the Modified Phong Model [33].

Lafortune [33] introduced the modified Phong reflectance model as:

\[
    f_r(x, \Theta_i, \Theta_o) = \frac{k_d}{\pi} + k_s \frac{n + 2}{2\pi} \cos^n \alpha
\]

where \( \alpha \) is the angle between the perfect specular reflection direction and the outgoing direction, \( n \) is the specular exponent, \( k_d \) is the diffuse reflection coefficient, and \( k_s \) is the specular reflection coefficient.

The physical properties which a physically plausible reflectance model must have are: first it should be reciprocal, or the incoming and the outgoing light directions are interchangeable. second it should ensure conservation of energy, or the fraction
of incoming light from any direction reflected over the hemisphere is always less than 1. We can see that the Modified Phong Model satisfies these two conditions as long as $k_d + k_s \leq 1$, where $k_d$ and $k_s$ are the diffuse and specular reflection coefficients.

The fist step is to sample the outgoing direction according to this BRDF. The solution is to split the BRDF into a diffuse part and a specular part and to sample each of them separately. This means that when we want to choose a ray direction we either randomly generate a diffuse ray (means sampling the diffuse part of the BRDF) or a specular ray (sampling the specular part). If we generate a diffuse ray we multiply the result by the diffuse part of the BRDF and divide it by the PDF of generating the diffuse ray and if we generate a specular ray we multiply the result by the specular part of the BRDF and divide it by the PDF of generating the specular ray.

We described how to cast a diffuse ray in section 5.3. To cast a specular ray we use the specular part of the BRDF and we use the following PDF to sample the hemisphere above $x$:

$$pdf(\theta) = \frac{n + 1}{2\pi} \cos^n \alpha$$  \hspace{1cm} (51)

In order to sample this PDF we choose 2 uniform stochastic variables $\xi_1$ and $\xi_2$ over the interval [0,1]. In term of $(\alpha, \phi)$, where $\alpha$ is the angle between the perfect specular reflection direction and the outgoing direction and $\phi$ is the azimuthal angle:

$$(\alpha, \phi) = (\arccos \xi_1^{\frac{1}{n+1}}, 2\pi \xi_2)$$  \hspace{1cm} (52)

In terms of $(x, y, z)$:

$$(x, y, z) = (\sqrt{1 - \xi_1^{\frac{2}{n+1}}} \cos(2\pi \xi_2), \sqrt{1 - \xi_1^{\frac{2}{n+1}}} \sin(2\pi \xi_2), \xi_1^{\frac{1}{n+1}})$$  \hspace{1cm} (53)

and it is required to change this direction from the object coordinate system to the camera coordinate system.

If we look at the random distribution of the directions we see that with a very high
value of \( n \) (the specular component) the distribution of directions will be tightly grouped around the perfect specular reflection direction. In other words \( n \) describes the sharpness of the specular reflection.

Now that we can calculate the direction of the diffuse or specular ray, the question is how we choose which type of the ray to follow?

### 5.5.1 Determining the type of ray to follow

The obvious approach is to cast more diffuse rays from surfaces which are primarily diffuse and more specular rays from surfaces which are primarily specular. We associate with each surface a number which represents the probability of casting a diffuse ray and a number which represents the probability of casting a specular ray. We then generate a random number \( \varepsilon \) and based on this number we choose the type of the ray to follow. There are two ways to do this:

#### 5.5.1.1 Regular Method

Because we are using the RGB system, we take the average of the three diffuse reflection coefficients and we call it \( \lambda_d \). We do the same thing with the specular reflection coefficients and we call it \( \lambda_s \). Then we calculate two parameters \( \rho_d \) and \( \rho_s \) in such a way that \( 0 \leq \rho_d, \rho_s \leq 1 \) and \( \rho_d + \rho_s = 1 \)

\[
\rho_d = \frac{\lambda_d}{\lambda_d + \lambda_s}, \quad \rho_s = \frac{\lambda_s}{\lambda_d + \lambda_s}
\]

(54)

where \( \lambda_d = \frac{k_{dr} + k_{dg} + k_{db}}{3} \) and \( \lambda_s = \frac{k_{sr} + k_{sg} + k_{sb}}{3} \).

Now we generate a random number \( \varepsilon \) over the interval \([0,1]\).

- if \( 0 \leq \varepsilon \leq \rho_d \) : we cast a diffuse ray.
- if \( \rho_d \leq \varepsilon \leq 1 \) : we cast a specular ray.

#### 5.5.1.2 Russian Roulette Method

If we use the above solution, we simply terminate the walk at some fixed path depth, which introduces bias. So in order to produce an unbiased solution we should use
the Russian Roulette method. However the removal of the resulting small amount of bias comes at the expense of the addition of a small amount of noise. In this method instead of using ρ we use λ for the boundaries:

if $0 \leq ε \leq λ_d$ : we cast a diffuse ray.

if $λ_d \leq ε \leq λ_d + λ_s$ : we cast a specular ray.

if $λ_d + λ_s \leq ε \leq 1$ : we terminate the path.

### 5.5.2 Specularity and Direct Lighting

In section 5.4.1 we calculated the direct lighting and we assumed that we are using the simple phong model. Now if we use the *Modified Phong Model* from equation (41) we get:

$$L_D(x, Θ_x) = \frac{I_s}{\pi N} (k_d + k_s \frac{n + 2}{2} \cos^n \alpha) \sum_{i=1}^{N} g(x, y_i) \cos \theta_1 \frac{\cos \theta_2}{pdf(y) d^2}$$  \hspace{1cm} (55)

This is the equation for direct lighting with *Modified Phong Model* and $N$ samples on the light source.

### 5.5.3 Specularity and Indirect Lighting

As mentioned above, at each intersection we decide which type of ray to cast, and we multiply the result by that part of the BRDF (Specular or Diffuse) and divide it by the appropriate PDF. So we introduce a *Transport Operator* $T_x$ which represents, for any point $x$ on the path, the term by which we multiply the indirect lighting:

$$T_{xi} = \frac{f_r(x_i, Θ_i, Θ_{i-1}) \cos(θ_{zi})}{pdf(Θ_{zi})}$$  \hspace{1cm} (56)

So if we cast a diffuse ray:

$$T_{xi} = k_d$$  \hspace{1cm} (57)

and if we cast a specular ray:

$$T_{xi} = k_s \frac{n + 2}{n + 1} \cos(θ_{zi})$$  \hspace{1cm} (58)

59
In the above equations, \( \cos(\theta_{zi}) \) is the angle between the direction \( \Theta \) and the surface normal, \( n \) is the specular exponent, \( k_d \) is the diffuse reflection coefficient, \( k_s \) is the specular reflection coefficient, and \( f_r \) is the BRDF.

Since we are using the RGB color model, at each point we have 3 independent equations for red, green and blue.

### 5.6 Implementation and Results

#### 5.6.1 Implementation

The following formula describes the intensity calculation for the complete path tracer with direct lighting and surfaces with both specular and diffuse reflection properties:

\[
I(x_0, \Theta_{x_0}) = \sum_{i=1}^{M} L_D(x_i, \Theta_{x_i}) \left( \prod_{j=0}^{i-1} T_{xj} \right)
\]  

(59)

where \( x_0 \) is the first point on the path (the point on the pixel) and \( \Theta_{x_0} \) is the direction towards this point to the eye. \( L_D(x_i, \Theta_{x_i}) \) is the direct lighting at point \( x_i \) on the path, \( T_{xj} \) is the transport operator \( (T_{x_0} = 1) \) and \( M \) is the path depth.

For example if the ray hits 2 objects and then hits nothing (background, black) the radiance will be:

\[
I(x_0, \Theta_{x_0}) = L_D(x_1, \Theta_{x_1}) + T_{x_1} \cdot L_D(x_2, \Theta_{x_2})
\]  

(60)

If the ray hits 2 objects and then hits a light source with intensity \( I_s \) the radiance will be:

\[
I(x_0, \Theta_{x_0}) = L_D(x_1, \Theta_{x_1}) + T_{x_1} \cdot L_D(x_2, \Theta_{x_2}) + T_{x_1} \cdot T_{x_2} \cdot I_s
\]  

(61)

And if the ray hits 3 objects and reaches the max path depth the radiance will be:

\[
I(x_0, \Theta_{x_0}) = L_D(x_1, \Theta_{x_1}) + T_{x_1} \cdot L_D(x_2, \Theta_{x_2}) + T_{x_1} \cdot T_{x_2} \cdot L_D(x_3, \Theta_{x_3})
\]  

(62)

Therefore at each intersection we calculate the direct lighting and we add the effect of indirect lighting only if we hit a light source. An algorithm for the path tracer is
presented in the appendix (B), and the structure of the programming is illustrated in the appendix (C).

As discussed earlier, we use RGB model for colors, and modified phong model for the BRDF. Therefore we have 3 diffuse reflection coefficients \((k_{dr}, k_{dg}, k_{db})\), and 3 specular reflection coefficients \((k_{sr}, k_{sg}, k_{sb})\) for each surface, in such a way that:

\[
\begin{align*}
    k_{dr} + k_{sr} & \leq 1 \\
    k_{dg} + k_{sg} & \leq 1 \\
    k_{db} + k_{sb} & \leq 1
\end{align*}
\]

When we talk about the color of a surface, what it actually means is that which wavelength of the spectrum the surface reflects, so the color of a surface and the reflection coefficients are closely related. Therefore in order to describe the reflection properties of a surface, all we need is the color of the surface, and a parameter which describes the percentage of the specular reflection to total reflection, which we call the Specular ratio.

For example the reflection coefficients of a surface with the color \((Red, Green, Blue)\), and with the specular ratio of \(Sratio\) can be calculated as follows:

\[
\begin{align*}
    k_{sr} & = \frac{Sratio}{100} \left( \frac{Red}{255} \right) \\
    k_{sg} & = \frac{Sratio}{100} \left( \frac{Green}{255} \right) \\
    k_{sb} & = \frac{Sratio}{100} \left( \frac{Blue}{255} \right) \\
    k_{dr} & = (1 - \frac{Sratio}{100}) \left( \frac{Red}{255} \right) \\
    k_{dg} & = (1 - \frac{Sratio}{100}) \left( \frac{Green}{255} \right) \\
    k_{db} & = (1 - \frac{Sratio}{100}) \left( \frac{Blue}{255} \right)
\end{align*}
\]

The reason for dividing the color by 255 is that in RGB model we use one byte (one character) for each of the three components.
5.6.2 Results

Figure (34) shows a test scene, which is a room with an object and two spherical light sources in it. The pinhole camera is located at the center of the coordinate system, $Z$ is the axis which the camera is looking at, and $Y$ is the up direction. The object inside the room is a sphere which is closer to the bigger spherical light source at the left of the room. Each of the five walls have different color and different specular and diffuse reflection properties. The images in figure (34) were generated with different number of sample points per pixel. For all the images in this set, the number of sample points on the light sources is 5, and the resolution of the images is 320 by 200.

A detailed description of the room with the object locations and properties:

**Right Wall** - Location: (20,0,0), Normal: (-1,0,0), Color: (143,250,120), Specular ratio: (30), Specular exponent: (30).

**Left Wall** - Location: (-20,0,0), Normal: (1,0,0), Color: (143,250,120), Specular ratio: (30), Specular exponent: (30).

**Opposite Wall** - Location: (0,0,120), Normal: (0,0,-1), Color: (255,128,2), Specular ratio: (50), Specular exponent: (10).

**Floor** - Location: (0,-12,0), Normal: (0,1,0), Color: (200,200,2), Specular ratio: (70), Specular exponent: (50).

**Ceiling** - Location: (0,12,0), Normal: (0,-1,0), Color: (2,102,200), Specular ratio: (50), Specular exponent: (30).

**Sphere** - Center: (-5,-5,100), radius: (3), Color: (255,2,2), Specular ratio: (90), Specular exponent: (100).

**Left spherical light source** - Center: (-10,4,100), Radius: (4), Color: (255,255,255).

**Right spherical light source** - Center: (10,4,100), Radius: (3), Color: (255,255,255).

In the next set of images in figure (35), we change the number of samples on the light sources. The scene is the same room, but with only one light source, and
The number of samples per pixel for all the images is 16. As can be seen with only one sample point on the light sources, the image is a little bit dark and it has sharp shadows. Increasing the number of samples results in softer shadows.

The next results in figure (36) shows the texture mapping on the surface of the sphere. The top image is without texture, and in the next two images, textures on the right are mapped on the surface of the sphere. For these images we used 25 samples per pixel, and 5 samples per light source.

Figure (37) and (38) shows a room with more objects. The first one has shelves and two spheres on the floor. The second one has a table, two chairs and a frame on the wall. For these images we used 25 samples per pixel, and 5 samples per light source. The rendering time was 40 minutes for the first one and 80 minutes for the second one.
Figure 34: Results with different number of samples per pixel
(a) 1 sample, 13 minutes

(b) 5 samples, 28 minutes

(c) 10 samples, 46 minutes

Figure 35: Results with different number of light source samples
Figure 36: Results with texture mapping
**Figure 37:** Room with shelves and two spheres

**Figure 38:** Room with a table and two chairs and a frame
CHAPTER VI

PATH TRACING WITH PLENOPTIC CAMERA

6.1 Results

In this chapter we present the final results which are simulating the plenoptic camera using a path tracer.

For the lenticular array, an array of 100 by 100 lenticules was used, with the thickness of the lenticules equal to 2mm, radius of curvature equal to 10mm, and index of refraction equal to 1.61. For the main lens, the same lens system in figure (21) was used. The CCD sensor behind the array has a resolution of 500 by 500. This means the number of sub-pixels behind each lenticule is 5 by 5.

For these images we have used 25 samples per pixel, and 5 samples per light source.

The scene is a room with 16 objects. There are 4 boxes, shelves and a sphere with a map of the world texture on it. There is one light source in the room which is not visible in the picture.

A raw plenoptic image can be seen in figure (39) and the extracted plenoptic images in figure (40).

Figure (41) shows the displacement analysis of the sub-images. Points A and B are selected for the analysis, which are the upper left corners of the shelves and one of the boxes respectively. To observe the displacement the left and the right sub-images from the third row are selected. Main lens is focused on the opposite wall (5 meters away), so the displacement of points A (4 meters) and B (2 meters), which are closer than the focal plane, should be from right to left when we see the sub-images in sequence from left to right. More displacement should be observed at point B, since it is farther away from the focal plane.
Location of point $A$ in the left sub-image: (28,27)
Location of point $A$ in the right sub-image: (25,27)
Location of point $B$ in the left sub-image: (70,70)
Location of point $B$ in the right sub-image: (68,70)

As can be seen the displacement in point $A$ is more than point $B$. Since the resolution of the sub-images is low (100 by 100 pixels), the displacement is only two to three pixels. As discussed earlier the resolution of the plenoptic sub-images is equal to number of lenticules. Microstructure optics is capable of making a lenticule as small as 0.12mm. Therefore in a 35mm film we can have more than 300 by 300 lenticules, which results in the plenoptic sub-images with the resolution of 300 by 300.
Figure 39: Plenoptic Image with Path Tracer

Figure 40: Extracted Plenoptic Images with Path Tracer
Figure 41: Displacement Analysis
CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Summary and Conclusions

The structure of light that fills the space around an object contains a great deal of information about the 3D shape of the object. Ordinary camera systems capture only a tiny portion of this information, but plenoptic camera keeps track of the 3D information of the light rays by distributing these rays over a two dimensional array of lenses.

The system has a number of advantages: it requires only a single stationary camera; it uses both horizontal and vertical disparity; there is no need to establish and maintain calibration between multiple cameras, and since the output of the plenoptic camera are multiple images, the image processing algorithms can be simple and fast.

The system’s main limitation is the resolution of the plenoptic sub-images, which is equal to the number of lenticules. Today’s microstructure optics is capable of fabricating 300 by 300 lenticules in a 35mm film. Another limitation is the nonuniform distribution of the light energy on the lenticules, which causes the appearance of the black regions at the edge of the sub-images. This problem can be solved by placing a field lens between the main lens and the lenticular array. The field lens bends back the diverging rays to converge on the lenticular array.

In this thesis we have introduced the simulation of the plenoptic camera which was carried out using ray tracing. We divided the ray tracing problem into two separate parts. The simulation of the camera lens system was presented in chapter III and calculating the distribution of light within the environment was presented in chapter V. By merging these two parts we got the final simulation results in chapter VI. The
other chapters provided the necessary backgrounds for the simulation procedure.

The simulation program is written in an object oriented programming environment with Microsoft Visual C++ 6.0. A complete ray tracing package capable of simulating the plenoptic camera, as well as other cameras and lens systems, is being prepared for DALSA Corp. Every parameter of the camera will be accessible through a user interface, which makes the simulation package a very powerful tool for customizing the camera for a specific application. Since no simulation has ever been done on this camera, the simulation which is proposed in this thesis can be used to carry out tests and customize the camera for a specific application before the actual model is built.

In the next sections we review some of the simulation package features and point out directions for future work.

7.2 Software Features

- Four types of primitive objects (sphere, plane, disk, box).

- Spherical and disk light sources.

- Capability of simulating lens systems.

- Capability of simulating cameras.

- Image texture mapping.

- Texture database.

- Realistic modelling of materials by using both the specular and diffuse properties.

- Anti-aliasing through stochastic sampling.

- Easy addition of new geometry, textures, light sources and other features.

Currently under development:
• Adding more objects to the object class.

• Adding more light sources to the light source class.

• Using acceleration techniques, like bounding volumes, to speed up the ray tracing process.

7.3 Future Works

• Depth recovery from the plenoptic images

• Customizing the prototype based on a specific application
APPENDIX A

CHANGING THE COORDINATE SYSTEM

In this appendix we want to transfer a vector from one coordinate system to another one. The vector is the direction of the ray which we calculated from the Probability Density Function (PDF). This vector is defined in the coordinate system attached to the object point. we need to transfer this vector to the camera coordinate system.

Let’s call the basis vectors of the object (or the New) coordinate system $U_1$, $U_2$ and $U_3$. The basis vectors of the camera (or the Old) coordinate system are $V_1$, $V_2$ and $V_3$. Because we are only interested in the new direction of the vector, let us assume these two coordinate systems have the same origin, see figure (42). When we obtain the new direction we just transfer the vector to the object point.

![Diagram of two coordinate systems](image)

**Figure 42:** The Two Coordinate Systems (Object and Camera)

If we write our vector in terms of the basis vectors in two systems we get:

$$ W = \alpha_1 V_1 + \alpha_2 V_2 + \alpha_3 V_3 $$

$$ W = \beta_1 U_1 + \beta_2 U_2 + \beta_3 U_3 $$
\[ b^T U = W = a^T V \] (63)

\[ a^T = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{pmatrix} \]

\[ b^T = \begin{pmatrix} \beta_1 & \beta_2 & \beta_3 \end{pmatrix} \]

\( \beta_1, \beta_2 \) and \( \beta_3 \) are known and \( \alpha_1, \alpha_2 \) and \( \alpha_3 \) are unknown.

we can represent the basis vectors of the New coordinate system in terms of the Old coordinate system:

\[
\begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}
\]

\[ U = BV \] (64)

from (63) and (64) we get:

\[ a^T = b^T B \]

Therefore by knowing the direction of the vector in the object coordinates system and the transfer matrix between the two coordinate systems we calculate the direction of the vector in the camera coordinate system and we translate this vector to the object point.
APPENDIX B

PATH TRACING ALGORITHM

Trace(Ray)

Get Closest Intersection

If (Intersection not valid) { Return Direct Color }

Calculate Surface Properties

If (Intersection is a Light Source) { Return (Direct Color + Indirect Color)}

If (Path Depth Reached) { Return Direct Color }

If (Rand < \(\rho_d\))

Get Diffuse Direction

Calculate Transport Operator for Diffuse Ray

Else

Get Specular Direction

Calculate Transport Operator for Specular Ray

Calculate Direct Lighting

Direct Color \(\ast\) Direct Color \(\ast\) Path Transport Operator

Path Transport Operator \(\ast\) Transport Operator

Trace(NewRay)

End
APPENDIX C

PROGRAMMING STRUCTURE

The simulation program is written in an object oriented programming environment with *Microsoft Visual C++ 6.0*.

Figure (43) shows the *Shapes* class structure. Objects in the ovals are *Classes* and the others are *Variables*.

![Shapes Class Structure Diagram](image)

**Figure 43**: Shapes Class Structure

Figure (44) and (45) show the *Lens* and *Camera* class structures.
Figure 44: Lens Class Structure

Figure 45: Camera Class Structure
REFERENCES


(http://www-bcs.mit.edu/people/jyawang/demos/plenoptic/plenoptic.html)

(http://www-graphics.stanford.edu/cek/rayshade/rayshade.html)

(http://www.cs.kuleuven.ac.be/cwis/research/graphics/RENDERPARK/)


(http://www.zemax.com/lensview/)


(www.biz.uiowa.edu/faculty/jgeweke/papers/paper59/paper59a.pdf)


VITA AUCTORIS

• Name: Arash Razavi

• Place of Birth: Tehran, Iran

• Year of Birth: 1975

• Education:

  – School of Exceptional Talents, Tehran, Iran.
    1989-1993

  – University of Tehran, Tehran, Iran.
    1993-1999 B.Sc.

  – University of Windsor, Ontario, Canada.