UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO DE MELO COSTA

Spectral photon mapping

Recife

2025

BRUNO DE MELO COSTA

Spectral photon mapping

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Ciência da Computação
da Universidade Federal de Pernambuco,
como requisito parcial para obtenção do título
de bacharel em Ciência da Computação.

Orientador: Silvio de Barros Melo

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Costa, Bruno de Melo.
Spectral photon mapping / Bruno de Melo Costa. - Recife, 2025.
29p.

Orientador(a): Silvio de Barros Melo
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2025.

1. Spectral Rendering. 2. Wavelength-Dependent Refraction. 3. Chromatic Aberration. 4. Photon Mapping. I. Melo, Silvio de Barros. (Orientação). II. Título.

000 CDD (22.ed.)

BRUNO DE MELO COSTA

Spectral photon mapping

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: 01/04/2025

Spectral Photon Mapping

Bruno de Melo Costa

April 10, 2025

Abstract

This paper introduces a photon mapping framework that accommodates wavelength-dependent refraction through a model based on the Sellmeier equation. In contrast to the standard RGB-based techniques, the new method traces photons through several spectral channels to better represent dispersion and color shift within transparent media called 'spectral bins'. During the first stage, the photons are released from light sources and scattered arbitrarily throughout the scene, each path being reflected, refracted, or absorbed according to wavelength-dependent characteristics. Afterward, the photons are separated into two K-D trees that differentiate between refractive and regular interactions. In a second pass, a reverse ray-tracing technique approximates final pixel intensities by fetching local photon density estimates for diffuse lighting and recursively systematically calculating specular reflections and refractions.

The model supports a realistic material dispersion of materials such as glass and water by calculating refractive indices at specific wavelengths. One can apply the Sellmeier equation, enabling significant color shifting and multicolored caustics within transparent objects. Russian roulette sampling also eliminates the issue of very long paths while maintaining statistical accuracy, and parallelization alleviates computational overhead. The experimental findings validate that the spectral method effectively reproduces intricate optical phenomena, such as multicolored caustics, color fringing, and secondary color bleeding, thus providing a step up from conventional three-channel models in physical realism. The findings demonstrate the applicability and versatility of spectral photon mapping to high-quality global illumination and optical simulation.

Keywords — Spectral Rendering, Wavelength-Dependent Refraction, Chromatic Aberration, Photon Mapping

1 Introduction

Simulation of chromatic aberrations—exemplified by the colorful dispersion in prisms or on soap bubbles' surfaces [WS00, Hec20]—has long been an open problem in physically-based rendering. In most current rendering pipelines, a single isotropic index of refraction is assigned to the entire spectral range [Jen01], thus neglecting the wavelength-dependent nature of light in authentic materials. This simplification removes prism-like color separation and subtle iridescent gradations, leaving out important optical effects from the images. Consequently, rainbow edge effects, complex caustics, and multicolored reflections are poorly represented or left out.

To amend these limitations, we created an approach that utilizes a wavelength-dependent refraction model, utilizing a Sellmeier-like equation [Sel72] to compute the index of refraction for each discrete wavelength 'bin.' This technique allows for more physical realism in the simulation by ensuring that each light spectrum segment refracts at slightly disparate angles. As an illustration, the simulation of a prism within our system will automatically produce the traditional dispersion and rainbow splitting of dispersive prisms. By recording per-wavelength intensities, we can capture subtle color bleeding and more subtle types of chromatic aberration [MS07].

Although path tracing [Laf95] is a popular global illumination method, we utilize photon mapping [Jen01] specifically to leverage its two-stage nature, which more efficiently handles dense diffuse interactions [Jen01, HOJ08]. During the forward pass, photons are emitted from light sources and propagated through the scene, constructing a photon map with diffuse and refractive interactions. During the backward pass, camera rays utilize this photon map to calculate indirect lighting contributions and perform wavelength-dependent refraction to calculate final pixel intensities. This separation of photon emission and pixel calculation is helpful for diffuse bounce tracing, where many photons can efficiently approximate global illumination.

This study aims to assess the potential of a spectral photon mapping framework to reproduce dispersion and color-bleeding effects that are not feasible with single-index methods. We restrict our study to comparatively straightforward geometric configurations to enable us to examine these effects exhaustively with minimal scene complexity. Initial tests reveal that multi-wavelength channels produce high-quality images, particularly for transparent objects and soap-bubble-like materials. However, multi-threaded implementation [PBMH02] continues to experience significant performance overhead with larger numbers of bins, suggesting that additional optimization or adaptive wavelength sampling algorithms will be necessary for larger or more complicated scenes.

In the next sections, we specify our spectral model using the Sellmeier equation, describe the photon-mapping method, and present our results [Jen01]. We also examine the realism vs. rendering speed trade-off and illustrate the viability of real-world spectral methods. These methods, while computationally expensive, result in much greater fidelity in the simulation of optical dispersion and diffusion than traditional single-index or three-channel RGB approaches [Jen01].

Apart from their theoretical significance, spectral photon mapping methods can find application in practical fields like optical design, virtual prototyping of lenses and materials, scientific visualization, and cinematic rendering for the purposes of physically accurate visual effects. Simulation of chromatic aberration in custom optical lenses, realistic reproduction of caustics in jewelry rendering, and visualization of spectral properties of translucent biological tissues are areas where traditional RGB-based approaches are lacking. The ability to simulate wavelength-dependent interactions enables advanced material simulation, augmented reality realism, and even optics education. These uses are fertile ground for future research, particularly in the application of spectral rendering in applied and commercial settings.

2 Background

2.1 Photon Mapping Fundamentals

Photon mapping is a two-phase global lighting approach described by Henrik Wann Jensen [Jen01, JC95]. Its strength lies in its ability to accurately reproduce complicated light interactions, such as caustics and indirect diffused illumination, by separating the rendering into the following.

1. **Forward Photon Tracing**: Light sources release photons, which scatter through reflection, refraction, and absorption mechanisms. When they strike a surface, they are grouped into a data structure, usually a K-D tree [Bro15, Ben75]. This collection of photon positions and trajectories is commonly known as the 'photon map' [Jen01].

2. **Backward Ray Tracing**: Rays are emitted from the camera. Upon intersecting an object, the system records localized photon information to approximate diffuse lighting at that location. Recursive ray bouncing addresses special effects, including reflection and refraction, as in traditional ray tracing methods [Vea98].

This technique effectively produces diffuse interreflections and demonstrates proficiency in producing caustics, such as the colored patterns cast by glass objects onto surrounding surfaces. Photon mapping forms the foundation of the spectral process in our method: we insert spectrally differentiated photons into the scene, follow their interactions through dispersion, and deposit the results in specially designed K-D trees for subsequent gathering [Jen01], as illustrated in Figure 1.

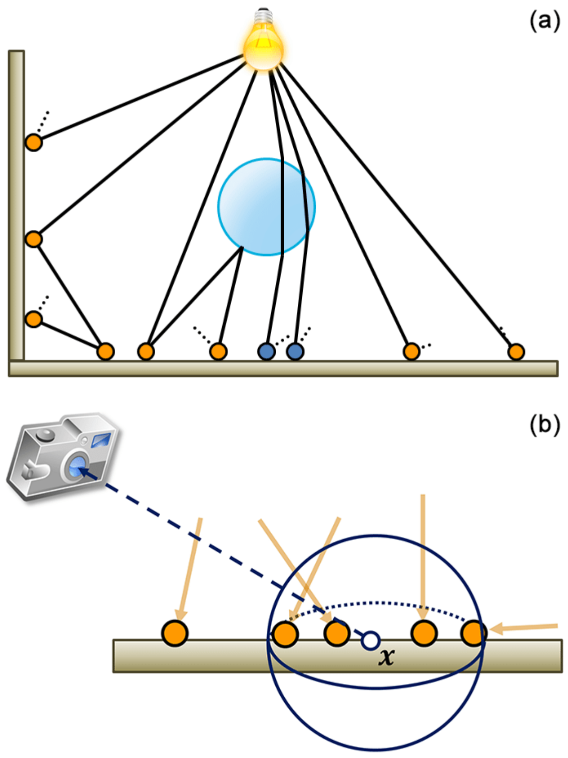


Figure 1: Photon mapping pipeline: (a) forward photon tracing; (b) radiance estimation via local photon gathering. [RHBD17]

2.2 Wavelength-Dependent Refraction: Sellmeier Equation

The primary method of doing this, particularly for effects such as prismatic color separation, is to have the refractive index n of the material wavelength dependent. A popular model of this effect is the Sellmeier equation, an empirical formula widely used [Hec20, AG23, Web02], traditionally expressed in the form:

$$n(\lambda) = \sqrt{1 + \sum_{k=1}^{N} \frac{\lambda^2 B_k}{\lambda^2 - C_k}} \tag{1}$$

In equation 1 λ is the wavelength of light (typically in micrometers), and B_k and C_k are experimentally determined coefficients for each material. They can be found in data from the literature such as Schott glass catalogs or other spectroscopic measurements for standard optical glasses and liquids [AG23]. Figure 2 shows how the Sellmeier equation fits real-world data for BK7 optical glass.

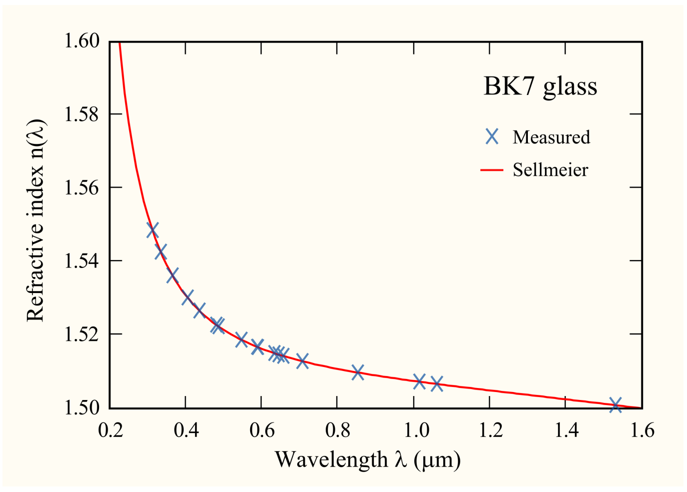


Figure 2: Measured refractive indices for BK7 glass (blue crosses) and Sellmeier model fit (red curve). The equation accurately captures wavelength-dependent dispersion effects. [Mel07]

Under this definition, each 'bin' for a given wavelength in our spectral rendering pipeline is prescribed some specific refractive index. Whenever a light ray or photon of wavelength λ intersects a boundary between two media with refractive indices $n_1(\lambda)$ and $n_2(\lambda)$, the ray is bent by Snell's law:

$$n_1(\lambda)\sin(\theta_i) = n_2(\lambda)\sin(\theta_r) \tag{2}$$

In equation 2, θ_i is the angle of incidence and θ_r is the angle of refraction. As $n(\lambda)$ is different for different wavelengths λ, each light spectral component is deflected differently, resulting in chromatic dispersion.

Classical rendering systems often assume a constant refractive index across all wavelengths, causing all light components to refract identically. In contrast, using the wavelength-dependent function $n_2(\lambda)$ from the Sellmeier equation allows a realistic simulation of optical dispersion. Although per-bin refraction increases computational cost, it significantly improves physical accuracy, especially for transparent materials like glass, crystals, and soap films that exhibit strong color variation under varying lighting.

2.3 K-D trees for Photon Storage

A central challenge in photon mapping is the ability to rapidly store and query potentially millions of photons. A K-D tree [Bro15, Ben75, Kel97] is well suited to this task, providing efficient spatial partitioning through recursive, axis-aligned splits. Figure 3 illustrates this process, where the space is divided at each level by alternating between vertical and horizontal splits.

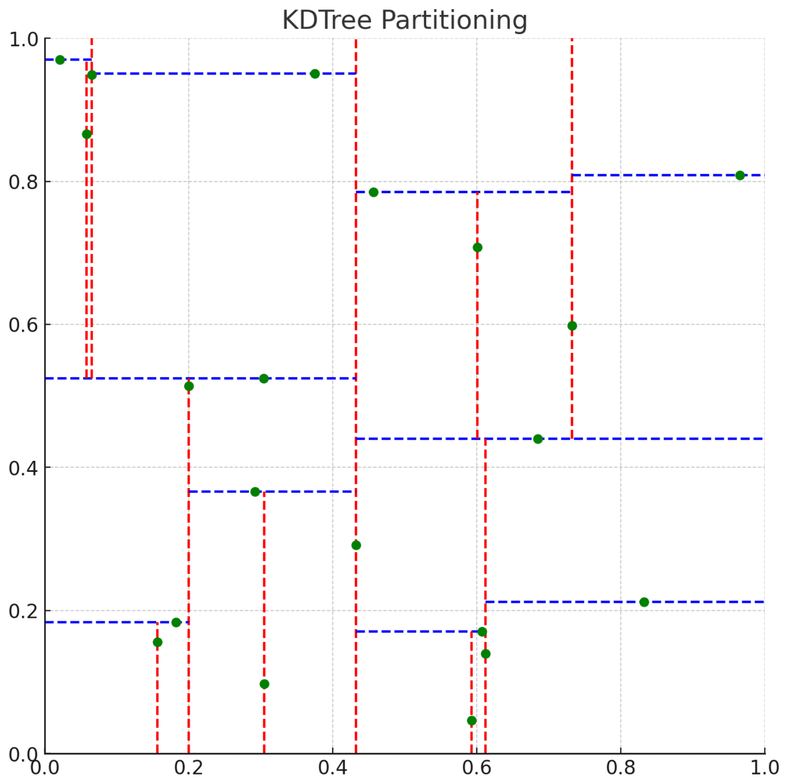


Figure 3: Visualization of K-D Tree partitioning. Green dots represent stored photons, while red and blue dashed lines indicate vertical and horizontal splits, respectively. This hierarchical subdivision enables efficient photon lookups by progressively narrowing the search space.

- **Construction**: A typical K-D tree build proceeds by finding the median photon coordinate along the current splitting axis (x, y, or z). The photons are then divided into two halves: those whose coordinates are below (or equal to) the median and those above it. This process repeats recursively, cycling through the splitting axes in $x \to y \to z \to x \to \cdots$ order. The result is a balanced binary tree that allows for queries in the average logarithmic time [Bro15]. As depicted in Figure 3, this recursive partitioning ensures that the photons are organized efficiently for fast retrieval.

- **Nearest-Neighbor and Range Searches**: During the final gathering phase (the backward ray-tracing pass), the renderer must locate photons within a certain radius of a surface intersection point. The K-D tree supports this by traversing only those nodes (subtrees) likely to fall within the query region. When it is determined that a subtree cannot intersect with the query sphere, that branch is pruned, reducing redundant checks [Kel97]. This behavior is implicit in Figure 3, where only a subset of the stored photons will be relevant to a given query.

- **Integration with Spectral Photon Mapping**: Since every photon in this system carries wavelength-specific data, each node in the K-D tree effectively tracks the position, direction, object reference, and spectral intensity of a photon. At lookup time, we gather neighboring photons for the relevant wavelength channels, which is especially important for computing wavelength-dependent caustics or diffused color bleeding [Ben75]. The partitioning in Figure 3 reflects how photons are stored efficiently, allowing queries to extract relevant wavelength-dependent contributions with minimal computational overhead.

The K-D tree details in our code demonstrate this approach, including a radius-based intensity summation to compute diffuse contributions at a given surface point. By leveraging balanced partitioning, we mitigate the computational overhead that arises when dealing with large numbers of photons across many wavelength bins.

2.4 Spectral Rendering and Color Handling

Our system uses a spectral rendering approach rather than a typical three-channel (RGB) method to capture the difference in refractive indices and other optical properties across different regions of the visible spectrum [WS00]:

- **Spectral Bins**
  We divide the visible range from about 380 to 750 nm into various bins. Thus, each photon or ray carries a certain "wavelength channel", which determines how it refracts (Section 2.2) and interacts with the objects of the scene.

- **Spectral to RGB Conversion**
  Although internal calculations are made using these spectral bins, the final result is typically presented in RGB. Consequently, after all spectral intensities have been accumulated, spectral information is transformed into an RGB color space by a matrix transformation derived from known color matching functions [Hec20]. This transformation is accomplished by an iterative matrix inversion and fitting procedure, which guarantees that the color of each pixel is very closely in line with human vision [MS07].

- **Benefits and Trade-Offs**
  - **Realistic Dispersion**: Our framework simulates realistic dispersion by simulating varying angles of refraction across wavelength bins, resulting in subtle color fringing and caustics.
  - **Increased Computation**: Using several channels increases the computation per ray. Even with multithreading optimizations, scenes with dense geometry or high bin counts consume more processing time and memory than a standard RGB or single-index method.

This spectrum method, along with photon mapping and K-D tree structures, enables us to generate physically plausible images with dispersion, secondary color bleeding, and soap bubble-like phenomena that would be difficult to recreate in simpler pipelines [MS07].

2.5 Reconstructing Spectra from RGB Using a Least-Squares Approach

A spectral distribution to RGB conversion involves mapping several wavelength channels onto only three color components. Such a many-to-one mapping implies no unique solution to the inverse transformation. An infinite number of spectral power distributions can produce the identical RGB triplet [Smi99]. We resolved this uncertainty employing a least-squares solution using the Moore-Penrose pseudoinverse to calculate an approximate spectrum that best matches a desired RGB value [Pen55, GVL96].

In the following subsections, we detail the use of this least-squares solution, discuss iterative negative clamping to guarantee that the intensities are physically valid (nonnegative), and explain how this method fits nicely into our spectral photon mapping pipeline framework.

2.5.1 The Underlying Math

Let M be a $3 \times N$ matrix that transforms a N dimensional spectral vector s (ie intensities at discrete wavelengths) into a 3-element RGB color vector c. Formally,

$$c = M \times s \tag{3}$$

Because $N > 3$ is in a typical spectral pipeline, this operation is not strictly invertible; many different spectra can yield the same RGB values. To obtain a best-fit solution in the least-squares sense, we apply the Moore-Penrose pseudoinverse M^+, which yields:

$$\begin{cases} s^* = M^+ \times c \\ M^+ = M^T \times (M \times M^T)^{-1} \end{cases} \tag{4}$$

This s^* is the approximate spectrum that, when multiplied by M, recovers an RGB value closest (in the least squares sense) to c.

2.5.2 Iterative Nonnegative Constraint

A potential drawback is that the initial solution s^* may contain negative coefficients for some wavelengths, which is physically meaningless. In practice, we can clamp the negative values to zero, but that step alone risks offsetting the color accuracy for each iteration. A more refined approach is as follows:

1. **Compute** $s^* = M^+ \times c$
2. **Clamp** negative coefficients in s^* to zero.
3. **Recompute** $\Delta c = c - M \times s^*$ to see how far the reconstructed color is from the target.
4. **Adjust** s^* by adding $M^+ \times \Delta c$ until the reconstructed color is sufficiently close to c.

This iterative procedure helps correct for the color shifts introduced by clamping. Each iteration nudges the coefficients toward a non-negative solution and remains near the desired RGB.
Although it does not produce a unique physical spectrum, this Moore-Penrose-based method provides a convenient spectral reconstruction that can be incorporated into dispersive rendering pipelines. By performing multiple iterations and zero-clamping negative coefficients, one can reconcile color accuracy with physically plausible (non-negative) spectral distributions, broadening the usability of spectral methods when starting from standard RGB assets [Lin07,LH95].

3 Implementation

This section describes the development of a spectral-photon mapping framework founded on state-of-the-art ideas [Jos12], adequately simulating the photon tracing algorithm incorporating refraction depending on wavelength using the Sellmeier equation. Although the architecture is based on existing work, every element—photon emission, wavelength-dependent refraction, and conversion from spectral to RGB—was built from the ground up in C++ to allow for top numerical accuracy and parallel processing control. The framework operates through a four-phase pipeline comprising scene setup, photon emission, K-D tree construction, and ray tracking. In addition, the processes of multithreading and spectral conversion are elaborated upon.

3.1 Overview of the Rendering Pipeline

3.1.1 Scene Setup

We instantiate geometry (spheres, planes, etc.) along with its optical properties, define lights with intensities and colors, and specify how many photons each light emits.

3.1.2 Photon Emission

A forward pass shoots photons from the lights into the scene. Each photon carries spectral data in bins. When a photon hits a surface, it may reflect, refract, or be absorbed, according to the probabilities of Russian roulette and the Sellmeier-based index of refraction of the material. All surviving photons and their associated data are stored [Vea98,JC95].

3.1.3 K-D tree Construction

Once all photons have been traced, we sort them “offline” to build a balanced K-D tree. This data structure is then ready for efficient spatial queries during rendering [Bro15,Ben75].

3.1.4 Ray Tracing

Rays are cast from the camera into the scene. For diffuse contributions, each intersection queries local photons in the K-D tree. The code calculates refractive bounces on a per-wavelength basis for transparent objects, leveraging the Sellmeier equation [Hec20].

3.2 Scene Setup

In the scene, we instantiate:

- Objects
  - Geometry (e.g., Triangles, Spheres, ...)
  - Optical Properties: diffusion, reflection, refraction
  - Color: the color of the object
  - Sellmeier function
- Lights
  - Type (e.g., Spherical, Cylindrical, etc.).
  - Intensity: how intense the light is
  - Color: the color of the light
  - Photons quantity

By default, 10^6 photons and 32 spectral bins are used. This choice balances rendering speed with visual fidelity, but can be adjusted for experimentation.

3.3 Photon Emission (Forward Pass)

Each light spawns its designated number of photons. A random direction is selected for each photon (spherical emission), and it carries a channel intensity vector stored in 32-bit float number to manage memory use [Jen01]. When a photon intersects an object, the code enters a loop where it repeatedly determines the fate of the photon through Russian roulette, continuing until the photon is absorbed or a predefined maximum number of interactions is reached. At each interaction, the photon may:

- **Diffuses**: randomly scattered, coloring the surface;
- **Reflects**: mirror-like reflection;
- **Refracts**: wavelength-dependent direction from the Sellmeier equation, generating multiple photons for different wavelength bins;
- **Is absorbed**: photon terminates.

In the case of refraction, the photon is split into multiple secondary photons, one for each wavelength bin that still carries intensity. Each of these secondary photons is traced independently, following the same loop-based decision process until absorption or the maximum depth is reached. This approach ensures that wavelength-dependent effects, such as dispersion, are properly simulated.

At each step, the photon's final position, direction, wavelength bin intensities, and object encountered are recorded. Every photon is added to a global buffer, and once all photons have been traced, we proceed to construct the K-D tree in Section 3.4.

Figure 4 illustrates this process, showing photons emitted from a light source and interacting with various surfaces. Some photons reflect off objects, while others refract, splitting into multiple wavelength components, or are absorbed. The dashed lines indicate potential scattering paths that contribute to indirect illumination, emphasizing the importance of tracking multiple bounces for accurate global illumination computation.

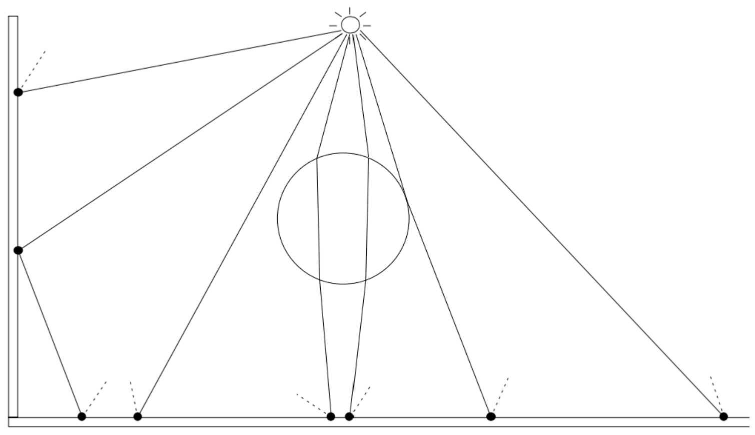


Figure 4: Photon emission in the forward pass. Photons are emitted from the light source and travel through the scene, interacting with surfaces via reflection, refraction, or absorption. Some photons scatter multiple times before reaching the final surface, while others are absorbed or leave the scene. Dashed lines indicate potential photon paths that contribute to indirect illumination [Jen04].

3.4 K-D tree

The K-D tree is a fundamental data structure that is used to efficiently store and retrieve photons. Once the emission phase is complete, all both photon lists are organized into a balanced K-D tree for fast nearest-neighbor searches. This section outlines both the construction of the tree and its role in radiance estimation.

3.4.1 Building the K-D Tree

Once the emission phase is over, both photon lists are sorted into a balanced K-D tree.

- **Median Selection**: Depending on the depth of the tree, we select a pivot (the median photon) along x, y, or z.
- **Partition**: Photons with coordinates below the median go to the left subtree; the rest go to the right subtree.
- **Recursion**: Repeat, cycling the axes in $x \to y \to z \to x \to \cdots$ order.

This offline build, illustrated in Listing 1, ensures quick range or nearest-neighbor queries when we gather diffuse illumination later in the rendering pass. The data structure is read-only once built, so concurrency issues are avoided at this stage [Bro15, Ben75].

Listing 1: Sort function for kd-tree.

```
template <int dimension>
void sort(size_t b, size_t e) {
    if (e - b <= 1)
        return;

    constexpr auto lessthan = [](const Photon &p1, const Photon &p2) {
        return p1.point[dimension] < p2.point[dimension];
    };

    const auto &v_b = begin();
    const size_t middle = (b + e) / 2;
    std::nth_element(v_b + b, v_b + middle, v_b + e, lessthan);

    sort<(dimension + 1) % 3>(b, middle);
    sort<(dimension + 1) % 3>(middle + 1, e);
}
```

3.4.2 Radiance Estimation Using Photon Queries

After constructing the K-D tree, we use it to estimate the radiance at a given surface point. The main principle is to add together the contributions of neighboring photons, weighted by their distance and incident direction:

- **Photon Lookup**: We used the K-D tree's nearest-neighbor search for a given intersection point to locate photons within a set radius.
- **Kernel Density Estimation**: Each photon's contribution is weighted using a kernel function that smoothes intensities based on the photon distance.
- **Directional Filtering**: Because photons contain directional information, their contribution is regulated by the surface normal and incident angle to prevent artifacts.

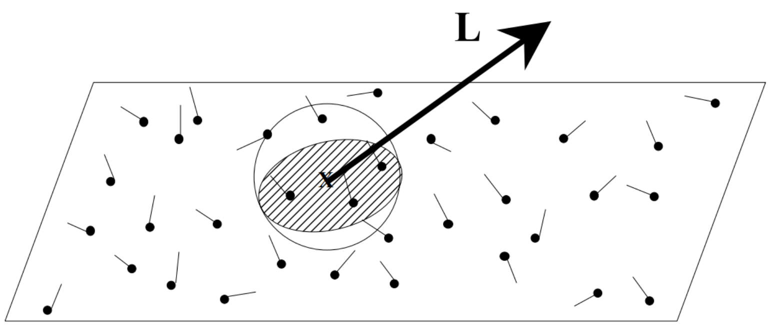


Figure 5: Radiance estimation with photon mapping. Photons (black dots) are stored with directionality, and a spherical query region gathers nearby photons. Their contributions, weighted by distance and incident angle, determine the outgoing radiance L, capturing indirect lighting and caustics. [Jen04].

Figure 5 illustrates this process. Photons are stored in the plane, while a spherical region collects nearby photons with directional influence. The radiance at a surface point is computed by integrating the weighted contributions of photons within the query region.

3.5 Ray Tracing (Backward Pass)

The backward pass computes the final color of each pixel by following rays from the camera into the scene and modeling light's interaction with objects. Unlike the forward pass, which propagates photons from light sources, this stage employs a standard ray-tracing technique, gathering both direct and indirect illumination to precisely reconstruct the radiance arriving at the camera [Jen01]. The method includes the following basic steps:

1. **Primary Rays**: The camera emits a central ray for each pixel traced into the image to determine the first visible surface. This process is performed in parallel to accelerate rendering.

2. **Intersection Computation**: The scene is examined for any intersections with the primary ray. Once an intersection is discovered, the surface's normal at the impact point is obtained, and the qualities of the material determine whether reflection, refraction, or diffusion are used.

3. **Diffuse Look-ups and Indirect Illumination**: If the intersected surface contains a diffuse component, the K-D tree is queried within a small radius of the intersection site to retrieve stored photons. The radiance estimation technique described in Section 3.4 calculates the local photon density, and the adjacent photon intensities are added to simulate indirect illumination. This explains global illumination effects such as color bleeding and soft shadows.

4. **Reflection and Refraction Handling**: If the material has reflecting or refractive qualities, we calculate deterministic reflection and refraction vectors:

   - **Specular Reflection**: The reflection vector is calculated using the surface normal, and a new ray is created to mimic the mirror-like bounce. The method then repeats the previous stages to compute more interactions.
   - **Wavelength-dependent Refraction**: Refraction is calculated per wavelength bin using the Sellmeier equation for transparent objects, resulting in a physically precise spectral dispersion. Each wavelength bin with retained intensity generates a distinct ray, which is traced independently. The intensity contribution for each wavelength is calculated separately, allowing effects like chromatic aberration to appear naturally.

This cyclical process continues until the ray hits a light source, quits the scene, or its contribution is negligible. The final color of the pixels is determined by combining the contributions of the diffuse, reflecting, and refractive components.

The ray-tracing method involves the camera emitting primary rays interacting with the scene's surfaces. When a surface is hit, a shadow ray decides whether the location is illuminated or shadowed. Reflective or refractive surfaces produce secondary rays, which simulate realistic reflections and refractions. This is shown in Figure 6.

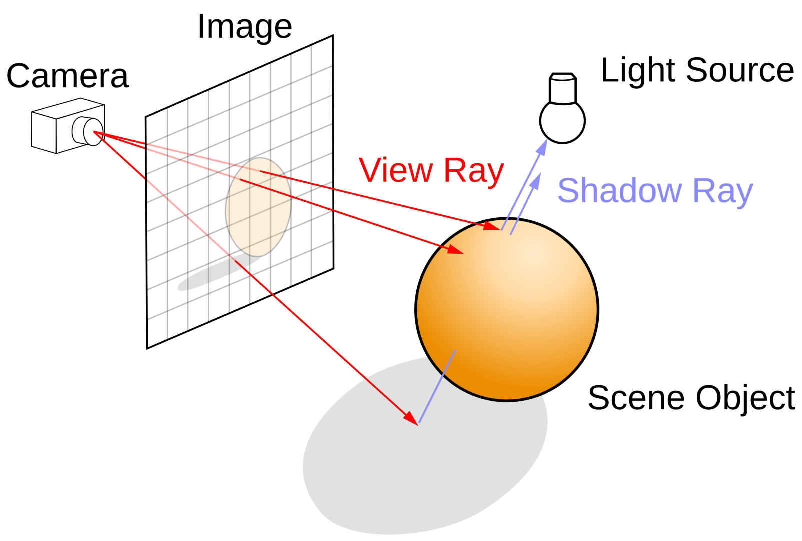


Figure 6: Figure depicts the ray-tracing technique [Hen08]. The camera emits primary rays (red), intersecting objects in the scene. A shadow ray (blue) is cast to see if the spot is in the shade or directly lighted. If the item is reflective or refractive, more rays are generated to simulate light transit, resulting in realistic reflections and refractions.

3.6 Wavelength-to-RGB Conversion

When ray tracing is complete, each pixel contains intensities for various discrete wavelength bins. To display these results in conventional RGB format, we follow the following steps:

1. **Spectral Summation**: Each wavelength bin is multiplied by the relevant color matching function and summed, producing intermediate red, green, and blue components [MS07].

2. **Normalization and Clamping**: We verify that the raw values of red, green, and blue are within the displaying range $[0, 1]$. Any component outside the range is limited to this range to avoid negative or extreme intensities that cannot be represented in standard image formats.

3. **Gamma Correction**: Because most displays expect non-linear color input, we employ a gamma function on the clamped data. In practice, c is a color channel in $[0, 1]$, and we set $c_\gamma = c^{1/2.2}$ to ensure that the final intensity of the image coincides with a human perceptual response [GJH$^+$09].

By adding clamping to legal display ranges and gamma correction, the system generates more perceptually accurate images that follow conventional output conventions while retaining the intricacies acquired by the underlying spectral computations, as shown in Listing 4.

3.7 Inverse Color Transformation: RGB to Spectral

Certain materials or textures enter the pipeline as RGB values only. We integrate an iterative Moore–Penrose pseudoinverse procedure (see Section 2.5) to reconstruct approximate spectra [LH95, GVL96].

1. Compute an initial least-squares solution $s^{(0)} = M^{+} \times c$.
2. Fix the negative values to zero.
3. Recompute the difference Δc between the target RGB and the reconstructed color, adjusting s until the error is sufficiently small.

This process runs for each RGB-based material, enabling physically plausible multi-channel data even if the user or scene specification only provides traditional three-component colors, as shown in Listing 2 and fully in Listing 3.

Listing 2: Color approximation loop.

```
// Pseudocode for the iterative negative clamp approach
Vector s = Mplus * c; // initial guess
clampNegative(s);

for(int iter = 0; iter < maxIters; ++iter) {
    Vector deltaC = c - (M * s);
    if( length(deltaC) < colorErrorThreshold ) break;
    s = s + (Mplus * deltaC);
    clampNegative(s);
}
```

3.8 Parallelization

To manage the computational overhead from emitting millions of photons and handling multiple spectral bins, we employ multithreading in two main steps:

1. **Photon Emission**: Each thread retrieves a range of photon indices from an atomic counter, emits photons for those indices, and stores them in a thread-local buffer. After all threads finish, their local buffers merge into a single global list. This deferred approach avoids synchronization bottlenecks during emission and defers K-D tree construction until the complete photon set is ready.
2. **Ray Tracing**: We divide the image by rows, assigning each thread a distinct row segment. Every thread reads from shared (read-only) scene data and the K-D tree, but writes pixel results independently, ensuring minimal need for synchronization.

By parallelizing these two major stages, we substantially reduce overall rendering time while preserving the algorithm's physically based spectral nature [PBMH02].

4 Results

This section describes the findings of three carefully planned experiments to assess our spectral photon-mapping framework under various optical events. The first experiment investigates a "Blood Moon" analog, using wavelength-dependent refraction in a transparent sphere to simulate the selective scattering seen during lunar eclipses. The second experiment investigates dispersion and diffusion by placing a highly translucent sphere in a controlled environment, isolating the interaction between color-separating refraction and intense internal scattering. Finally, the final experiment demonstrates the algorithm's adaptability to various light sources, diffusive objects, and prismatic geometries in enclosed situations.

Each experiment uses simple geometry (spheres, planes, or prisms) within box-like surroundings, which are illuminated by either a single overhead source or additional strategically placed emitters. We compare standard (single-index) refraction with our Sellmeier-based multichannel technique, showing variations in chromatic aberration, caustics, and overall indirect illumination quality. Through these many configurations, we aim to highlight the method's main strengths and potential limitations, particularly its ability to capture delicate wavelength-dependent effects and multi-bounce global illumination in a single rendering framework.

All experiments were carried out on a *12th Gen Intel(R) Core(TM) i9-12900H* CPU using 10 threads and 32GB of RAM, running Windows 11. GPU acceleration was not used. The final render resolution was 750×750 pixels, with 5×10^6 photons emitted per scene and 32 spectral bins.

4.1 Experiment 1: Blood Moon Effect

This experiment uses transparent and opaque spherical objects to examine photon mapping and diffusion in a controlled setting. The main focus is understanding how light interacts with these materials, particularly with respect to refraction, diffusion, and spectral dispersion.

A fascinating natural analogy to this phenomenon is the Blood Moon, which is observed during a total lunar eclipse. Because Earth's atmosphere scatters sunlight, the shorter wavelengths (blue and green) are scattered more than the longer wavelengths (red and orange), creating a deep reddish hue on the Moon. In this experiment, Sphere A is an optical medium that scatters and refracts the light before its interaction with Sphere C. Spectral detail differences seen in images are the result of wavelength-dependent refraction in the same manner as the Earth's atmosphere selectively allows longer wavelengths to pass through during a lunar eclipse.

4.1.1 Scene Setup

The experiment is conducted in a simulated 3D environment with the following components:

- **Sphere A**: Positioned at $(0, 0, 2)$ with a radius of 0.5. This sphere is transparent and exhibits high diffusion properties. The refraction of light through Sphere A follows the Sellmeier equation (mentioned in Section 2.2) with a single term, where $B_1 = 0.12961$ and $C_1 = 125311$.
- **Sphere B**: Also at $(0, 0, 2)$, but with a radius of 0.455 and 0.44. It is solid and white, serving as an obstacle to transmitted light.

- **Sphere C**: Located at $(-0.75, 0, 2)$ with a radius of 0.125. It is solid white and interacts with refracted light.
- **Plane**: Placed at point $(-0.875, 0, 2)$ with a normal vector of $(1, 0, 0)$. It is a solid black surface that is used to show stray refraction rays.
- **Light Source**: A collimated beam of light enters the scene from infinity, illuminating Sphere A uniformly.
- **Camera**: Positioned at $(0, 0, -0.5)$, pointing towards $(-0.2, 0, 2)$ with an upward vector of $(0, 1, 0)$.

4.1.2 Simulation Results

To evaluate the effects of photon mapping and wavelength-dependent refraction, we analyzed the interaction of light under different configurations of Sphere C. The results are divided into two cases, each providing insight into how object size influences light transmission, spectral dispersion, and photon scattering. Each case is accompanied by the following.

- **A Full Lighting Simulation**, which shows the rendered scene with photon-mapped lighting.
- **A Starfield Projection**, which visualizes photon scattering and refraction patterns.
- **A Starfield Projection with Uniform Refractive Index**, which isolates the effects of chromatic dispersion.

4.1.2.1 Case 1: Sphere B with Radius 0.455

- **Full Lighting Simulation** (Figure 8a): The bigger A Sphere exhibits intense refraction and diffraction of light, resulting in a strong spectral dispersion. The left half is predominantly white due to direct lighting, whereas subtle color fringing is present on the right-hand side due to wavelength dependence of the refraction.

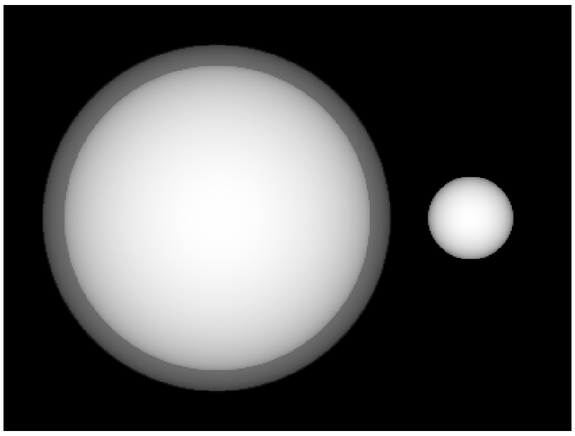
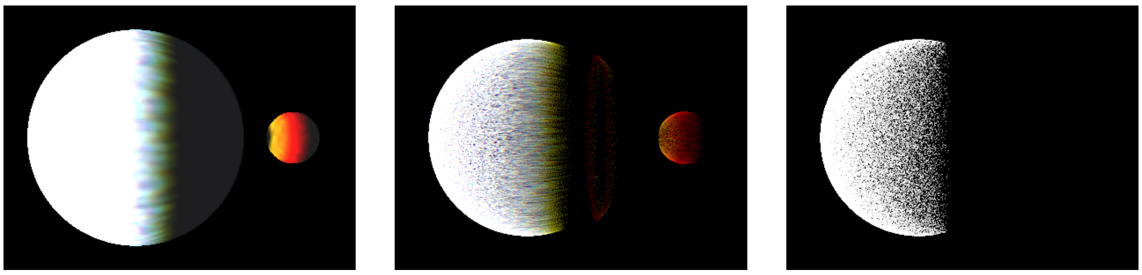


Figure 7: (Scene Structure - No Lighting): A reference image illustrating the spatial arrangement of objects without applied lighting effects.

- **Starfield Projection** (Figure 8b: This depiction shows the scattering of photons upon their interaction with Sphere A and Sphere B. A dense annular structure forms around Sphere C, representing a zone of concentration where refracted light is accumulated.

- **Starfield Projection with Uniform Refractive Index** (Figure 8c): The chromatic dispersion is effectively removed when the refractive index is kept the same for all wavelengths. The projected result has an even light distribution without the typical color separation of the default setting.


(a) Rendered scene with full lighting simulation showing chromatic dispersion effects around Sphere A.

(b) Photon scatter visualization indicating focal regions where refracted light converges.

(c) Starfield projection without chromatic dispersion, showing a uniform refractive light pattern.

Figure 8: A comparative visualization of all images illustrating the effects of photon mapping, refraction, and dispersion.

4.1.2.2 Case 2: Sphere B with Radius 0.44

- **Full Lighting Simulation** (Figure 9a): Including a slightly reduced Sphere B permits more light to be transmitted through the instrument, thereby enhancing spectral dispersion. The red and yellow bands on the right-hand side show improved visibility, signifying a higher chromatic disjunction.

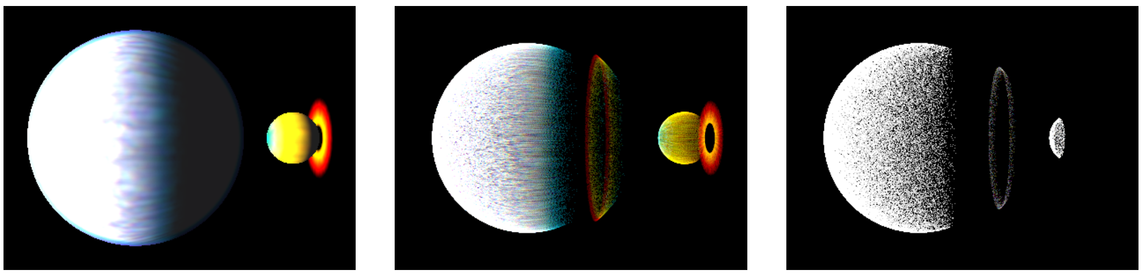
- **Starfield Projection** (Figure 9b: This setup observes more photon scattering. Compared to the earlier case, photons are more broadly distributed, while intensity fluctuations exhibit an intense focal spot.

- **Starfield Projection with Uniform Refractive Index** (Figure 9c): As in Case 1, eliminating chromatic dispersion leads to a more uniform distribution of photons. The lack of wavelength-dependent effects underscores the dominance of refraction in determining the resultant light distribution.

4.1.3 Discussion and Key Observations

These results are consistent with theoretical expectations based on the Sellmeier equation, indicating that the wavelength-dependent refraction is an important aspect of the unconventional lighting simulations of the photon map.

1. **Influence of Sphere B's Size**: The decreased radius of Sphere B increases light transmission and chromatic effects because less light is obstructed. In Case 2, due to lesser obstruction, the red spectrum experienced a shift, leading to Sphere C being illuminated in yellow and green.


(a) Rendered scene with full lighting simulation showing chromatic dispersion effects around Sphere A.

(b) Photon scatter visualization indicating focal regions where refracted light converges.

(c) Starfield projection without chromatic dispersion, showing a uniform refractive light pattern.

Figure 9: A comparative visualization of all images illustrating the effects of photon mapping, refraction, and dispersion for Sphere B with 0.44 radius.

2. **Chromatic Dispersion and Wavelength Separation**: The distinction between standard and uniform refractive index projections confirms that dispersion is responsible for the essential roles played in the scene's color variations. The simulation lacks any realistic spectral enhancement typical of light interactions unless the benefits of wavelength-based refraction are considered.

3. **Photon Scattering Patterns**: Starfield projections reveal how light bends and diffuses, collecting at different focal points depending on object positioning and material properties. The other light pattern noticed on Sphere C during Case 2 further illustrates how important proper spectral distribution is for a realistic scene, for without it, light would naturally act to scatter and refract according to wavelength.

4. **Light Transmission Variations**: A larger Sphere B blocks more light, decreasing the intensity of the transmitted color, whereas a minor Sphere B allows the light to pass and thus increases the vividness of the color and chromatic separation.

5. **Effect of a Uniform Refractive Index**: The chromatic dispersion disappears, and the uniformity of light propagated with that makes the colors not shift. This confirms that spectrum dispersion allows for realistic rendering in optical simulations.

4.1.4 Performance Consideration

In the spectral rendering case, the whole process took about 42 seconds, with 4 seconds of photon emission. For comparison, the RGB version with the constant index of refraction took 35 seconds and 3 seconds for photon emission. That is about a 20% overhead of the total render time using the spectral method. The extra cost mainly comes from the refraction logic for every wavelength and color mixture in multiple channels. However, the added realism—especially in portraying color spreading and wavelength-dependent effects—justifies the performance price, especially since the increase in computing time on standard home computers is extremely slight.

4.1.5 Conclusion

The work presented here successfully shows the effects of photon mapping, refraction, and chromatic dispersion in a controlled setting. By varying the size of Sphere B, we saw significant divergences in light transmission per spectral distribution and photon scattering. The results show the necessity for wavelength-dependent refraction in realistic physical rendering and optical simulations. The main conclusions that can be drawn from the present study are as follows:

- **Impact of Object Size on Light Transmission**: In Case 1, when a larger sphere presents an obstacle to transmission, light transmission is severely lower than that of a narrower sphere. This leads to less spectral separation than in Case 2. A smaller Sphere B in Case 2 allows the transmission of more light, thus increasing the chromatic dispersion and producing more spectra detail.
- **Role of Chromatic Dispersion**: The differences between the standard and uniform refractive index tests highlight the importance of chromatic dispersion in real-world light interactions. Otherwise, the simulation would have no spectral separation, and, therefore, less physically believable results.
- **Photon Scattering and Spectral Distribution**: The starfield projection tests powerfully demonstrate how realistic rendering is affected by photon-to-photon interactions. For Sphere C in Case 2, realistic results can be produced if the spectral distributions are correctly and accurately modeled.

This experiment compares the Blood Moon phenomenon and elaborates on how refraction and diffusion affect color perception in an astronomical and artificial setting. The discovery also stresses that realistic photon mapping strictly depends on accurate spectral modeling in rendering, optics, and atmospheric physics.

4.2 Experiment 2: Analysis of Photon Dispersion and Diffusion

Light has complex interactions with transparent materials, causing dispersion and diffusion effects. As light passes through a medium having a gradient index of refraction, different wavelengths are refracted to varying degrees, causing chromatic dispersion. Furthermore, incoming photons undergo multiple internal scatterings in materials with high diffusion coefficients before escaping, altering the spatial and spectral properties of the transmitted light. Understanding these effects is critical for precise optical modeling in computer graphics, optical engineering, and materials science applications.

This experiment investigates the interplay between dispersion and diffusion in a characteristic setting. Placing a highly diffuse yet extremely transparent ball in an area with a single light source allows us to see how light flows, scatters, and interacts with its environment. This setup allows us to understand how light redistribution and color dispersion affect the appearance of materials in rendering and optical simulations.

4.2.1 Scene Setup

This experiment investigates the interaction of light with a very diffuse transparent sphere, with a special emphasis on the processes of dispersion and diffusion and their effects on color distribution in a controlled context. The apparatus contains the following:

- **Sphere A**: Positioned at $(0, 0, 1.5)$ with a radius of one. This sphere is translucent and has high diffusion qualities. The refraction of light through Sphere A follows the Sellmeier equation (cited in Section 2.2) with two terms: $B_1 = 0.42163$, $C_1 = 144192$, $B_2 = -0.17956$, and $C_2 = 144311$.
- **Sphere B**: Positioned at $(0, 0, 5)$, with a radius of 0.075. In the scene, it is the only source of light, and it is a solid, white sphere.
- **Camera**: Positioned at $(0, 0, -1)$ and pointing at $(0, 0, 0)$. It captures photon interactions with Sphere A and the dispersion effects that arise.

The experiment is designed to reveal how photons emitted by Sphere B interact with Sphere A, refracting, diffusing, dispersing, and producing intricate lighting effects.

4.2.2 Simulation Results

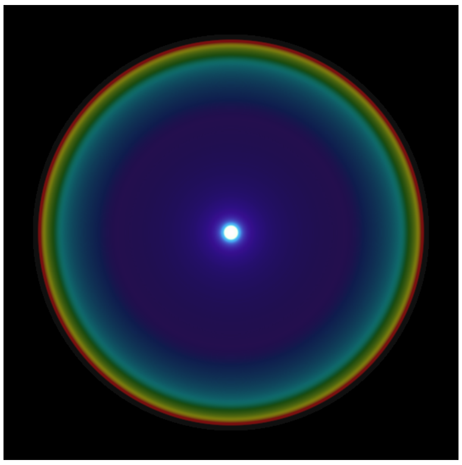


Figure 10: This graphic illustrates the photon diffusion and dispersion that Sphere A, which is highly transparent and diffusive, interacts with light from Sphere B in a convoluted fashion, resulting in noticeable chromatic dispersion and diffused transmission.

Figure 10 depicts the key physical phenomena investigated in this study.

- Chromatic dispersion is seen in the color separation around Sphere A.
- Diffuse scattering produces a spread-out, non-directional illumination pattern rather than a narrowly focused transmission.
- The dark background emphasizes the contrast between direct and diffuse illumination while separating the effects of transparency and internal scattering.

4.2.3 Discussion and Key Observations

The findings of this experiment provide a thorough examination of how light behaves when interacting with a transparent, very diffuse medium. By studying dispersion and diffusion effects in a controlled environment, we can better understand how wavelength-dependent refraction and internal scattering affect perceived color and illumination. The following subsections cover the observed phenomena and their consequences for light-transport models.

4.2.3.1 Chromatic Dispersion and Wavelength Separation

Sphere A's transparency creates a chromatic dispersion when photons flow through its medium. The material's refractive index varies with wavelength, causing bending angles to fluctuate between colors. Shorter wavelengths, such as blue light, have more refraction than longer wavelengths, such as red light. As a result, photons emitted from Sphere A exhibit a distinct spectral spread, resulting in visual color separation.

4.2.3.2 Diffuse Scattering and Light Redistribution

Sphere A has a higher diffusion coefficient than refractive media, resulting in significant internal scattering of incident light before it leaves the medium. This causes the following observable effects:

- The transmitted light has no refraction route and appears as a collection of scattered components.
- The diffusion process modifies the sharp definition of refracted light, resulting in a more subtle transition of hues.
- Compared to a strictly refractive object, the illumination pattern around Sphere A seems fuzzy and desaturated.

4.2.3.3 Contrast Between Direct and Diffused Illumination

Because Sphere B is the only light source in the image, its illumination must be sent directly to the camera or altered by Sphere A. This study compares direct and diffused light transfer.

- Because there are no intervening scattering effects, the area around Sphere B remains very bright, giving it a pure white appearance.
- The intensity of light and its spectral variations, as it travels through Sphere A, are determined by its penetration depth and exit angle.
- Some photons encounter many internal reflections within Sphere A, causing a mild secondary illumination effect at the margins and a discernible halo-like glow.

4.2.4 Performance Consideration

This experiment, which benefited from indirect illumination and the lack of photon emission, was rendered in about 5 seconds. There was only one emissive object in the scene and its light traveled through a highly translucent sphere before arriving at the camera. The calculation was restricted to direct ray tracing with wavelength-dependent refraction and diffusion since no global photon mapping was needed. This demonstrates how, in the absence of global illumination, the spectral rendering pipeline can efficiently and computationally cheaply simulate intricate optical behaviors, including chromatic dispersion and internal scattering.

4.2.5 Conclusion

This experiment investigates how photon dispersion and diffusion influence light transit in a transparent medium. The findings emphasize important principles in light behavior, such as:

- Wavelength-dependent chromatic dispersion arises when colors refract at different angles.
- Diffuse scattering combines and softens spectral components.
- Modulated illumination properties are those in which light leaving the medium deviates significantly from direct transmission.

These findings have important implications for realistic light modeling in computer graphics and optical simulations, particularly in accurately depicting glass, liquids, and translucent materials.

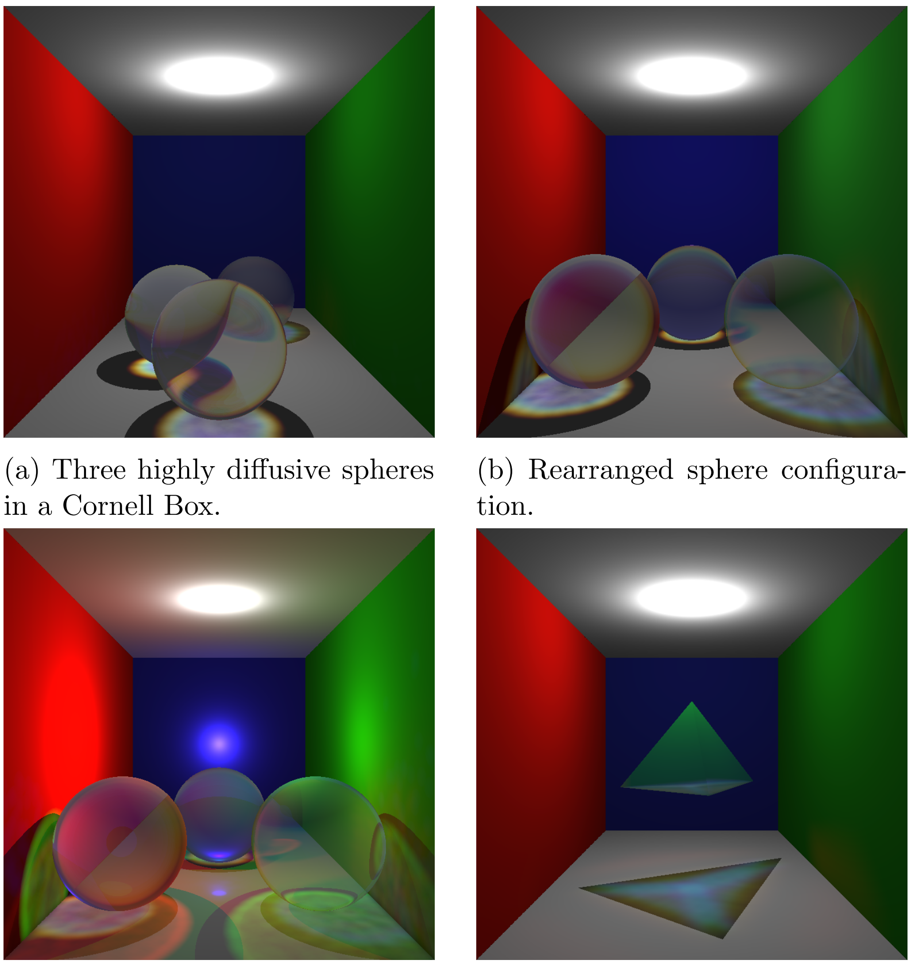
4.3 Experiment 3: Extended Photon-Mapping Demonstrations with Diffuse Interreflections

In this last series of experiments, we extend the preceding testing to demonstrate the adaptability of our spectrum photon-mapping approach to growing complex lighting and geometry arrangements. Figure 11 shows four images that exhibit different scenario layouts, material attributes, and lighting source placements.

4.3.1 Discussion

1. **Diffuse Interaction among Spheres**: Figure 11a shows three spheres with significant diffusion coefficients. The spheres exhibit unusual global lighting effects, with photons bouncing numerous times before reaching the camera. Extensive interreflections within the restricted region produce soft gradients and subdued shadows.

2. **Rearranging the Diffusive Objects**: Figure 11b shows how the reorganization of the spheres affects indirect lighting. Notably, the shifting geometry modifies the cumulative photon pathways, causing differences in the distribution of diffused light across surfaces and enhancing the spatial complexity of the resulting caustics.



(a) Three highly diffusive spheres in a Cornell Box.

(b) Rearranged sphere configuration.

(c) Added light sources on each wall.

(d) A tetrahedron to exhibit prismatic effects.

Figure 11: Additional results obtained using spectral photon mapping and diffuse interreflections.

3. **Multi-Wall Illumination**: Figure 11c illustrates the insertion of light sources into each wall. This configuration dramatically increases the amount of indirect photon bounces, emphasizing volumetric and surface scattering. The overlapping spheres also produce more elaborate shadow and penumbra patterns, demonstrating the photon-map-based approach's ability to manage increased illumination complexity.

4. **Prismatic Tetrahedron**: Finally, Figure 11d shows a tetrahedron with partially refractive faces, highlighting prismatic splitting. Photons flowing through the tetrahedron show nuanced color separation and delicate caustics, demonstrating the algorithm's ability to handle diffuse and refractive features in the same picture.

4.3.2 Key Observations:

1. **Diffuse Interreflections**: Each setting shows how photon mapping can record many low-intensity bounces that progressively build up to create realistic lighting effects.

2. **Sensitivity to Geometry**: Changing the layout or adding new forms can

significantly affect indirect lighting, shadows, and caustics.

3. **Complexity vs. Accuracy**: Adding additional emitters, as in Figure 11c, greatly deepens the complexity of the scene, which increases the computational load required to maintain noise-free outcomes.

4. **Combining Diffuse and Refractive Elements**: The prismatic effects of the tetrahedron demonstrate the generality of the algorithm, accurately portraying diffusion-dominated and wavelength-dependent refractions inside a single unified framework.

Overall, these demonstrations confirm the robustness and adaptability of our spectral photon mapping method in scenarios that require precise modeling of light scattering, reflection, and refraction.

5 Conclusion

The main aim of this study was to present a spectral photon mapping method designed to address chromatic aberration and light scattering within transparent materials. We could simulate realistic refractive caustics and dispersion effects by computing wavelength-dependent refractive indices using the Sellmeier equation and utilizing a photon-mapping technique with several spectral bins. This mimicked effects such as rainbow fringing and more gradual color changes than standard single-index techniques.

Nevertheless, the higher fidelity comes at the expense of a high memory cost. As each photon carries more spectral information, storing and manipulating millions of photons takes more capacity than regular, less detailed lighting techniques. This restriction may significantly translate to higher hardware demands and longer process times as the bin count or photon count increases.

Despite these constraints, the system demonstrates the potential of fully spectral techniques to replicate complex optical effects that are not otherwise possible. By combining Sellmeier dispersion concepts, K-D-tree-enabled photon gathering, and multithreaded architecture, this research moves towards a rendering method for transparent materials more inherently based on physical principles. Future optimizations—such as adaptive binning and progressive photon mapping—can be expected to increase efficiency further and expand the range of spectral photon mapping in both academic research and production visual effects pipelines.

6 Potential Extensions

Although the proposed spectral photon mapping framework has successfully replicated chromatic aberration and realistic diffusion, there are several ways to improve both visual fidelity and performance.

1. **Adaptive Spectral Binning**
   Rather than partitioning the visible spectrum into intervals of equal spacing, a technique that allocates more intervals to areas where the dispersion changes significantly could minimize computational demands without compromising detail in important wavelength ranges [HJW$^+$08].

2. **GPU Acceleration**
   Porting these fundamental operations—photon emission, K-D tree construction, and ray-object intersection—to the GPU would even more aggressively pursue parallelism [PBMH02, WWB+14]. Implementing it would significantly decrease rendering time, particularly in scenes requiring large quantities of photons or high bin resolution.

3. **Progressive Photon Mapping**
   The present method uses one photon emission phase and a last ray-tracing phase. An iterative method, by which partial-photon maps are progressively updated, can decrease noise and achieve even more satisfactory results by taking multiple short passes [HOJ08]. This technique can be stopped or restarted, which makes it capable of handling large volumes of complicated scenes.

4. **Adaptive Photon Distribution**
   Rather than emitting photons uniformly, the renderer can bias photon emission to regions that produce challenging effects (e.g., narrow caustics or multilayer refractions). Allowing more photons in those regions would improve overall image smoothness and reduce artifacts, especially where the spectral dispersion is most pronounced. Adherence to these standards enables the system to progress toward improved performance, greater complexity of material models, and a more faithful simulation of the behavior of light throughout the visible spectrum [HOJ08].

By following these directions, the system can evolve toward higher performance, more sophisticated material models, and more realistic depictions of light behavior across the visible spectrum.

References

[AG23] Schott AG. Optical glass datasheets. Online, 2023. `https://www.schott.com/en-gb/products/optical-glass-p1000267/downloads`.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[Bro15] Russell A. Brown. Building a balanced k-d tree in $o(kn \log n)$ time. *Journal of Computer Graphics Techniques (JCGT)*, 4(1):50–68, March 2015.

[GJH+09] Xu Guan, Su Jian, Pan Hongda, Zhang Zhiguo, and Gong Haibin. An image enhancement method based on gamma correction. In *2009 Second International Symposium on Computational Intelligence and Design*, volume 1, pages 60–63, 2009.

[GVL96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[Hec20] Eugene Hecht. *Optics (5th Edition) by Eugene Hecht Book PDF*, chapter 10. Zenodo, Apr 2020.

[Hen08] Henrik. Ray trace diagram. Wikimedia Commons, 2008. CC BY-SA 4.0, via Wikimedia Commons.

[HJW$^+$08] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, New York, NY, USA, 2008. Association for Computing Machinery.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27(5), December 2008.

[JC95] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping.* A. K. Peters, Ltd., USA, 2001.

[Jen04] Henrik Wann Jensen. A practical guide to global illumination using ray tracing and photon mapping. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, page 20–es, New York, NY, USA, 2004. Association for Computing Machinery.

[Jos12] N.M. Josuttis. *The C++ Standard Library: A Tutorial and Reference.* Pearson Education, 2012.

[Kel97] Alexander Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 49–56, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Laf95] Eric P. Lafortune. Mathematical models and monte carlo algorithms for physically based rendering, 1995.

[LH95] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1995.

[Lin07] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[Mel07] Bob Mellish. Sellmeier equation. Wikimedia Commons, 2007. CC BY-SA 4.0, via Wikimedia Commons.

[MS07] Dragos Miha and Eugen Strajescu. From wavelength to r g b filter. *University" Politehnica" of Bucharest Scientific Bulletin, Series D: Mechanical Engineering*, 69(2):77–84, 2007.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Trans. Graph.*, 21(3):703–712, July 2002.

[Pen55] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.

[RHBD17] Max-Gerd Retzlaff, Johannes Hanika, Jürgen Beyerer, and Carsten Dachsbacher. Physically based computer graphics for realistic image formation to simulate optical measurement systems. *Journal of Sensors and Sensor Systems*, 6:171–184, 05 2017.

[Sel72] W. Sellmeier. Ueber die durch die aetherschwingungen erregten mitschwingungen der körpertheilchen und deren rückwirkung auf die ersteren, besonders zur erklärung der dispersion und ihrer anomalien. *Annalen der Physik*, 223(11):386–403, 1872.

[Smi99] Brian Smits. An rgb-to-spectrum conversion for reflectances. *Journal of Graphics Tools*, 4(4):11–22, 1999.

[Vea98] Eric Veach. *Robust monte carlo methods for light transport simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. AAI9837162.

[Web02] M.J. Weber. *Handbook of Optical Materials*. Laser & Optical Science & Technology. Taylor & Francis, 2002.

[WS00] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley Series in Pure and Applied Optics. Wiley, 2000.

[WWB+14] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. Embree: a kernel framework for efficient cpu ray tracing. *ACM Trans. Graph.*, 33(4), July 2014.

Appendix

Listing 3: RGB to Spectral conversion using wavelength-based mapping.

```
#include "matrix.h"
#include "vec.h"

// Static members of the Color class
Matrix<float> Color::C; // Transformation matrix
float Color::fix[CN] = {}; // Fix array for color normalization

// Function to set up the color transformation matrix and normalization
    factors
void setupColor() {
    Matrix<float> M(3, CN); // Create a 3xCN matrix

    // Populate the matrix with RGB values for each wavelength
    for (size_t i = 0; i < CN; ++i) {
        Color3 c(Color::get_wavelength(i)); // Get RGB values for the
            wavelength
        M[0][i] = c.R; // Red channel
        M[1][i] = c.G; // Green channel
        M[2][i] = c.B; // Blue channel
    }
    // Compute the transformation matrix
    Matrix<float> M_t = M.transpose(); // Transpose of M
    Color::C = M_t * (M * M_t).inverse(); // C = M_t * (M * M_t)^-1

    // Compute normalization factors for white color
    Color white(Color3(1, 1, 1)); // White color (R=1, G=1, B=1)

    for (size_t i = 0; i < CN; ++i)
        Color::fix[i] = 1 / white[i]; // Normalize each channel
}

// Constructor for the Color class that converts RGB to internal
    representation
Color::Color(const Color3 &rgb) {
    // Lambda function to calculate the RGB values from the internal
        representation
    constexpr auto calculate = [&](Matrix<float> m) {
        Matrix<float> Color(3, 1); // 3x1 matrix to store RGB values
        for (size_t i = 0; i < CN; ++i) {
            if (m[i][0] < 0) continue; // Skip negative values
            Color3 c(get_wavelength(i)); // Get RGB values for the
                wavelength

            Color[0][0] += m[i][0] * c.R; // Accumulate Red channel
            Color[1][0] += m[i][0] * c.G; // Accumulate Green channel
            Color[2][0] += m[i][0] * c.B; // Accumulate Blue channel
        }

        return Color; // Return the calculated RGB matrix
    };

    // Goal matrix representing the target RGB values
    const Matrix<float> goal({{rgb.R}, {rgb.G}, {rgb.B}});
    Matrix<float> M = C * goal, newgoal; // Initial internal
        representation and error matrix

    size_t cnt = 0; // Iteration counter
    do {
        newgoal = goal - calculate(M); // Compute the error
        M += C * newgoal; // Update the internal representation

        // Ensure non-negative values in the internal representation
        for (size_t i = 0; i < CN; ++ i)
            if (M[i][0] < 0)
                M[i][0] = 0;
    } while (++cnt < 1e4 && newgoal.sumValuesAbsolute() >= 5e-3); //
        Convergence criteria

    // Store the final internal representation in the channels array
    for (size_t i = 0; i < CN; ++i)
        channels[i] = M[i][0];
}
```

Listing 4: Spectral to RGB conversion using wavelength-based mapping.

```
// Constructor to create a Color3 object based on a wavelength (wl) in
    nanometers.
// The wavelength is mapped to RGB values based on the visible spectrum.
Color3 (float wl) {
    if (wl >= 380 && wl < 410) {
        R = 0.6 - 0.41 * (410 - wl) / 30;
        G = 0.0;
        B = 0.39 + 0.6 * (410 - wl) / 30;
    } else if (wl >= 410 && wl < 440) {
        R = 0.19 - 0.19 * (440 - wl) / 30;
        G = 0.0;
        B = 1.0;
    } else if (wl >= 440 && wl < 490) {
        R = 0.0;
        G = (wl - 440) / 50;
        B = 1.0;
    } else if (wl >= 490 && wl < 510) {
        R = 0.0;
        G = 1.0;
        B = (510 - wl) / 20;
    } else if (wl >= 510 && wl < 580) {
        R = (wl - 510) / 70;
        G = 1.0;
        B = 0.0;
    } else if (wl >= 580 && wl < 640) {
        R = 1.0;
        G = (640 - wl) / 60;
        B = 0.0;
    } else if (wl >= 640 && wl < 700) {
        R = 1.0;
        G = 0.0;
        B = 0.0;
    } else if (wl >= 700 && wl <= 780) {
        R = 0.35 + 0.65 * (780 - wl) / 80;
        G = 0.0;
        B = 0.0;
    }
}
// Constructor to create a Color3 object from a Color object.
// The Color object contains multiple channels, each corresponding to a
    specific wavelength.
explicit Color3(const Color &c) {
    R = 0, G = 0, B = 0;
    for (size_t i = 0; i < CN; ++i) {
        if (c[i] <= 0) continue;
        const float wl = Color::get_wavelength(i);
        *this += Color3(wl) * c[i];
    }
}
```