UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"MEJORA EN IMÁGENES GENERADAS POR GRAFICACIÓN POR COMPUTADORA MANIPULANDO LOS MODELOS DE ILUMINACIÓN"

TESIS
QUE PARA OPTAR POR EL GRADO DE
DOCTORA EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
CINTHYA LIZETH CEJA MENDOZA

DR. EDGAR GARDUÑO ÁNGELES
IIMAS, UNAM

CIUDAD UNIVERSITARIA, CD. MX, MAYO 2023

Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

"Mejora en Imágenes Generadas por Graficación por Computadora Manipulando los Modelos de Iluminación"

T E S I S

que para obtener el grado de:

Doctora en Ciencias

(Computación)

presenta:

Cinthya Lizeth Ceja Mendoza

director de tesis:

Dr. Edgar Garduño Ángeles

Ciudad de México, México mayo, 2023

A toda mi familia.

Agradecimientos

Siempre me es difícil escribir esta sección porque siento que no logro transmitir mi agradecimiento como quisiera. Agradezco a mis sinodales y comité tutor, Pablo Barberis, Alfonso Gastelum, Jorge Pérez, Bruno M. Carvalho y Luis de la Cruz, así como a mi tutor, Edgar Garduño, por su tiempo para revisar este trabajo y sus comentarios que sin duda fueron de gran ayuda para mejorarlo.

También agradezco al Posgrado en Ciencia e Ingeniería de la Computación de la UNAM por haberme aceptado para realizar este proyecto, a Lulú, Cecilia y Amalia por siempre estar al pendiente de los alumnos y hacer que el posgrado funcione con su trabajo. Así mismo, doy gracias al Departamento de Ciencias de la Computación del IIMAS por darme un espacio de trabajo.

Quiero agradecer a mis papás por haberme apoyado todo este tiempo y darme las bases para convertirme en la persona que soy ahora, jamás podré terminar de agradecerles a ustedes y a mis tíos todo lo que han hecho por mí. A mis hermanas, Ilse y Denise, por siempre estar conmigo aunque estemos lejos. Y a Héctor, porque definitivamente gracias a ti he llegado tan lejos, por aguantarme en mis malos ratos y por darme ánimos cuando más lo necesito. A todos mis amigos, en especial a Rosario, Caleb, Gibrán, Wendy, Miguel y Montse por acompañarme y hacer más amenas mis comidas.

Finalmente, agradezco al CONACyT por haberme otorgado la beca 331081/231743 para realizar mis estudios de doctorado.

Cinthya Ceja

Resumen

Los gráficos por computadora (graficación por computadora o computación gráfica) son un campo de las ciencias de la computación que estudia los métodos para la creación y manipulación de imágenes a través de la computadora. Sin embargo, el término gráficos por computadora comúnmente se refiere a un ambiente en tres dimensiones, un proceso que genera una imagen por medio de $\psi : \mathbb{R}^3 \to \mathbb{N}^2$, donde ψ es la aproximación de un sistema óptico, generalmente modelado mediante la simulación de una cámara fotográfica *estenopeica*. El proceso involucra la obtención de representaciones tridimensionales de los objetos (*modelos geométricos*); objetos que serán colocados en una *escena* virtual para posteriormente ser proyectados para generar la imagen final que puede almacenarse o desplegarse en algún dispositivo (*renderizado*).

La forma más común de representar a un objeto en graficación por computadora es a través de su superficie (una malla poligonal). Para obtener la superficie hay varios métodos y uno de ellos es rastreando, o extrayendo, la superficie de un objeto voxelizado (un objeto discretizado con una colección de pequeños vóxeles cúbicos contiguos). En la literatura hay varias propuestas para la extracción de superficies de objetos voxelizados, una de ellas simplemente extrae la colección de las caras rectangulares de los vóxeles en la frontera del objeto, otras hacen directamente una aproximación suave de dicha colección. La colección de caras rectangulares tiene la ventaja de ser topológicamente correcta, con respecto al objeto voxelizado, pero comúnmente resulta en renderizados muy burdos (voxelizados). Para evitar este efecto, muchos trabajos se han enfocado en la refinación del mallado del objeto para suavizar las representaciones sobre la pantalla; sin embargo, pueden lograrse cambios en el renderizado

de la superficie de un objeto voxelizado sin necesidad de modificar su superficie ni el objeto voxelizado subyacente.

Al igual que en el caso de la fotografía, el manejo de la interacción de la luz con los objetos en dicha escena es un proceso que juega un papel fundamental para dar realismo o simplemente mejorar la visualización del objeto en una imagen generada por computadora. Esto ocurre porque ψ representa un sistema óptico en dónde la iluminación es responsable de la ilusión de tridimensionalidad en el proceso de formación de la imagen.

Para la generación de imágenes agradables a la vista o de apariencia natural es importante modelar correctamente: los objetos, la luz (*modelos de iluminación*), y el material de los objetos; además de hacer un proceso de renderizado adecuado. Cambios en el modelo de iluminación y sus componentes involucrados pueden producir cambios drásticos en la imagen resultante, por ejemplo, suavizar un mallado burdo. Así mismo, modelos de iluminación más sofisticados pueden generar imágenes más cercanas a las obtenidas por una cámara fotográfica. Sin embargo, buscar realismo conlleva frecuentemente un costo computacional que impide realizar una simulación exacta de dicha interacción, por lo tanto, se han creado técnicas de graficación por computadora que permitan lograr imágenes similares a la realidad sin ser necesariamente simulaciones físicamente correctas.

Los métodos para la creación de imágenes por computadora se pueden clasificar en dos grandes grupos según la forma en que se modela la interacción entre luz y objetos: los que utilizan modelos de iluminación local, que dan prioridad a tener resultados rápidos, y los que utilizan modelos de iluminación global, que dan prioridad a tener imágenes foto-realistas.

Sin embargo, dentro de los métodos que producen imágenes foto-realistas ha habido pocos esfuerzos por incluir el comportamiento dual de la luz a pesar de que el comportamiento de onda es el responsable de efectos como la difracción, el *speckle* y la polarización.

Este trabajo presenta dos métodos que aprovechan o modifican el modelo de iluminación para mejorar los renderizados finales. El primer método genera mejores

renderizados de mallas de polígonos rectangulares provenientes de objetos voxelizados (con propiedades matemáticas deseables pero renderizados muy cúbicos) creadas por un algoritmo de detección de límites. Logramos esta mejora al asignar normales más apropiadas a los vértices de la malla usando una combinación lineal de funciones suaves conocidas como *blobs* y aprovechando la forma en que los métodos de graficación por computadora estándar renderizan las superficies. Se presentan dos aproximaciones de asignación de normales, uno general (GMAN) y otro (SMAN) que utiliza conocimiento *a priori* del método usado para reconstruir el modelo geométrico, las imágenes resultantes de ambos métodos muestran una suavización del modelo en el renderizado y se comparan ambos métodos con los resultados de una asignación de normales usando un filtro de Sobel y con una biblioteca especializada, donde los resultados de este trabajo son en muchos casos similares o ligeramente superiores.

De igual manera, se presenta un modelo para generar imágenes foto-realistas que toma en cuenta el comportamiento de onda de la luz para generar efectos de difracción y polarización. El método se apoya de la teoría de Fraunhoffer y aprovecha la posibilidad de calcular el patrón de difracción resultante de atravesar una abertura calculando la transformada de Fourier de dicha abertura. El espectro de Fourier resultante se usa para generar una imagen compuesta a color (que representa la distribución de la luz difractada) que se coloca sobre el objeto usando una modificación al proceso del mapeo de texturas para tomar en cuenta la posición de la luz en la colocación del patrón. El método se implementó en un programa de renderizado basado en rayos al que además se le agregó el manejo de vectores de Stokes y matrices de Mueller para el manejo de los efectos de polarización. Los resultados muestran cambios de tonalidad esperados debido al comportamiento de onda de la luz.

Abstract

Computer graphics is a field of computer science studying the methods for the creation and manipulation of images with a computer. However, the term computer graphics usually refers to rendering three-dimensional scenes, the process yielding an image through $\psi : \mathbb{R}^3 \to \mathbb{N}^2$, where ψ is an optical system approximation that is generally modeled through the simulation of a *pinhole* camera. This process involves the adequate representation of three-dimensional objects (*geometric models*); these objects are positioned in a virtual *scene* that will be projected onto a plane to generate a final image that can be saved or displayed on some device (*render process*).

The prevalent way to represent an object in computer graphics is through its surface (i.e., a polygon mesh). There are several methods to obtain such a surface, one consisting of tracking, or extracting, the surface from a voxelized object (an object discretized by a collection of small neighboring cubic voxels). The literature presents various proposals for extracting surfaces from voxelized objects, one of them simply extracts the collection of all rectangular faces on the object's voxelized boundary, other methods perform a smooth approximation to said collection. The rectangular faces collection has the advantage of being topologically correct, with reference to the voxelized object, but usually results on blocky renderings.

To avoid this effect, several works have focused on refining the object's "quadrilateral" mesh to smooth the representation over the screen, however, changes in the rendering of the surface of an object's surface can be achieved without modifying its surface nor the subjacent voxelized object.

As in the case of photography, the way light-objects interaction is modeled in a scene has a fundamental role on giving realism or simply improve the visualization of

the object in computer graphics. This occurs because ψ represents an optical system where illumination is responsible for the perception of tridimensionality in the process of image formation.

To generate images pleasing to the eye or that feel natural it is important to correctly model: objects, light (*illumination models*), and surface materials; besides carrying out an appropriate rendering process. Changes in the illumination model and associated components can result in significantly different images, for example, smoothing a blocky mesh. In addition, sophisticated illumination models can yield images closer in appearance to the ones produced by a photographic camera. However, aiming for realism often entails a computational cost that does not allow performing exact simulations of such interaction, and thus, computer graphics techniques have been developed to render reality-like images that are not physically correct simulations.

Methods for computer-generated images can be classified into two groups, depending on the way they model light-object interaction: local and global illumination methods. The former prioritizes interactivity and fast rendering whereas the latter prioritizes rendering photo-realistic images regardless of computing time. However, there has been few efforts to include duality-of-light behavior in the illumination models of photo-realistic-rendering methods, in spite of wave behavior being responsible of effects such as diffraction, speckle, and polarization.

This work presents two methods that take advantage of, or modify, the illumination model to improve the final renderings. The first method yields visually-pleasing renderings of rectangular meshes from voxelized objects (with desirable mathematical properties but very blocky renderings) created by a boundary detection algorithm. This improvement is achieved by assigning appropriate normals to the mesh vertices using a combination of lineal functions known as *blobs* and taking advantage of the way standard computer graphics methods render surfaces. Two approaches to assigning normals are presented, one general (GMAN) and another (SMAN) using *a priori* knowledge on how the underlying geometric model was created; the resulting images from these methods show smooth renderings. The results of both methods are

compared to renderings produced with a Sobel filter and with a specialized library to assign such normals, the results from our proposed methods are similar or better in several cases.

The other method consists of a model to generate photo-realistic images and incorporates wave behavior of light to generate diffraction and polarization effects. This method relies on Fraunhoffer theory and takes advantage of computing the diffraction pattern produced by the light after going through a diffracting aperture. The resulting Fourier spectrum is used to generate a compound a color image (representing the distribution of the diffracted light) that is placed on the object using a modified texture mapping process that takes light position into account. The method was implemented in a modified ray-based rendering software to manage Stokes vectors and Mueller matrices for polarization calculation. The results of our proposed method show the change of colors expected due to wave light behavior.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los gráficos por computadora (también conocido como graficación por computadora o computación gráfica) son un campo de las ciencias de la computación que se dedica al estudio de métodos para la creación y manipulación de contenido visual a través de la computadora. Abarcan una gran diversidad de representaciones, incluyendo gráficos en 2D y 3D, sin embargo, el término se emplea más comúnmente para referirse a los gráficos en tres dimensiones, convención que usaremos en este trabajo. A continuación definiremos conceptos y mencionaremos algunas aplicaciones básicas de graficación por computadora para posteriormente concentrarnos en la presentación de la contribución de este trabajo: mejorar las imágenes resultantes apoyándonos en modificaciones al funcionamiento del modelo de iluminación, en primer lugar para suavizar las visualizaciones de mallas rectangulares, que representan la superficie de objetos voxelizados binarios, sin cambiar el modelo geométrico, y en segundo, para incluir efectos de onda en la generación de imágenes foto-realistas.

1.1. Graficación por computadora

Los gráficos por computadora tienen un gran número de aplicaciones en diversos campos del conocimiento, que abarcan desde visualización científica de datos, con la creación de representaciones y análisis de fenómenos o modelos, hasta la creación de imágenes realistas y surrealistas usadas extensivamente en el mundo del arte, el entretenimiento (tales como animación, vídeo juegos y películas) y la publicidad.

Sin embargo, la finalidad básica de cualquier aplicación de gráficos por computadora es hacer una representación en dos dimensiones (imagen) de un conjunto de objetos en tres dimensiones, es decir un mapeo $\psi : \mathbb{R}^3 \to G$, donde ψ es la aproximación a un sistema óptico y $G \subset \mathbb{Z}^2$. El proceso para generar una imagen por computadora se puede considerar análogo a tomar una fotografía y consiste esencialmente en utilizar cálculos matemáticos para proyectar la representación en 3D de los objetos sobre un plano (que, típicamente, es la pantalla de una computadora).

El sistema óptico más usado para modelar el proceso de proyección en gráficos por computadora es el de la cámara estenopeica (*pinhole camera*), una de las formas más sencillas de tomar fotografías en el mundo real. Estas cámaras consisten de una caja bien sellada con un pequeño agujero (teóricamente, una entrada puntual) en uno de sus lados y papel fotográfico fijado en el lado contrario [30]. Cuando la caja es colocada frente a una escena (es decir, aquello que se encuentre frente al agujero de la cámara) con el agujero descubierto, la luz entra a través de dicho agujero y llega al papel fotográfico; después de dejar la cámara cierto tiempo se habrá formado una imagen en el papel.

En el caso de graficación por computadora, para llevar a cabo el proceso de proyección es necesario, entre otras cosas, contar con una representación computacional de los objetos. La representación por superficie es la más popular en gráficos por computadora y consiste en representar a los objetos únicamente por medio de su superficie la cuál es comúnmente un conjunto de polígonos simples que forman una malla (malla poligonal); claramente, dicha malla es una colección de vértices y aristas (gráfica). El proceso mediante el cual se generan es conocido como *modelado geométrico*. Las representaciones de los diferentes objetos que formaran una imagen son distribuidos y acomodados en una porción del espacio tri-dimensional (escena). Sobre la escena se utiliza el modelo matemático de una cámara desde la cual se formará la imagen. Una vez determinada la escena y los parámetros de la cámara, se realiza la simulación del comportamiento de la luz (definido con *modelos de iluminación y sombreado*) interactuando con los objetos de la escena. Para definir el proceso de interacción entre luz y materiales es necesario definir las propiedades de éstos (*modelado de materiales*).

Una vez que se tienen definidos la escena, la cámara, los materiales y el modelo de iluminación se realiza el proceso de generación de la imagen que conlleva una proyección de la escena a un plano y al final de este proceso obtendremos el equivalente a una fotografía de la escena que debe discretizarse para poder ser almacenada en una imagen digital o desplegarse en una pantalla. En algunos casos es necesario modificar algunos elementos del proceso de formación de la imagen (por ejemplo, la forma, la posición de un objeto o la posición de la cámara) y que esos cambios ocurran en tiempo real. La necesidad de tener o no interacción en tiempo real determina la complejidad de varios de los elementos del proceso de formación de la imagen. Esto ocurre porque para lograr un alto grado de realismo visual (por ejemplo, la inclusión de efectos como texturas, sombras, reflexiones, irregularidades, colores, entre otros) los costos computacionales pueden ser altos; por otra parte, los modelos suelen ser significativamente más simples cuando se necesita interactividad y esto se logra reduciendo el número o calidad de efectos lo que significa una reducción de cálculos necesarios para generar las imágenes y, por lo tanto, el tiempo necesario para obtenerlas.

A pesar de las variaciones que se pueden encontrar en el proceso de generación de una imagen según su aplicación, el proceso de generación de representaciones gráficas es básicamente el mismo y se presenta a continuación.

1.1.1. Proceso de renderizado

El proceso por el que pasa una descripción de una escena tridimensional para producir una imagen bidimensional final es conocido como *rendering pipeline* o proceso de *renderizado* (anglicismo usado debido a que en Español no existe una palabra que implique el proceso completo de formación de la imagen). Durante este proceso se hacen varias operaciones que afectarán la apariencia de la imagen final y puede considerarse análogo a tomar una fotografía usando el modelo de la cámara estenopeica. Aunque existen distintas variaciones e implementaciones de este proceso; la Figura 1.1 muestra un diagrama general de los pasos básicos del proceso de renderizado.

- *Transformaciones de modelo*: son transformaciones, típicamente geométricas,

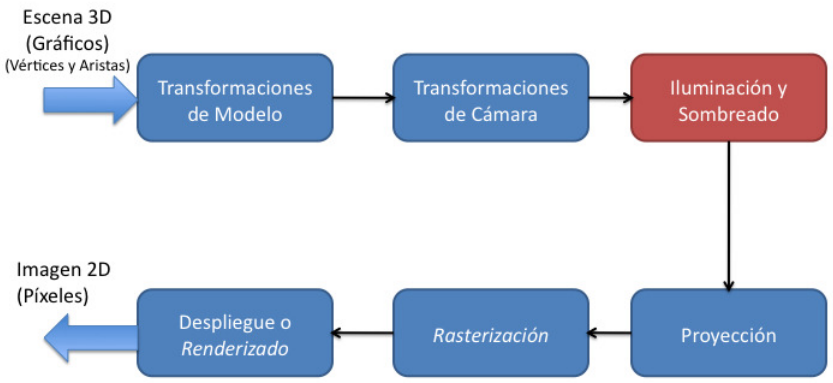


Figura 1.1: Diagrama que muestra el proceso de renderizado básico, donde se hace la transformación de una escena en tres dimensiones a una imagen en dos dimensiones. El rectángulo rojo señala la tarea de iluminación y sombreado, que es la tarea en la que se enfoca este trabajo.

realizadas sobre los objetos (los cuales son modelos geométricos, generalmente descritos por gráficos) para colocarlos en el sistema coordenado del espacio 3D de la escena virtual.

- *Transformaciones de cámara*: son transformaciones realizadas sobre la escena de acuerdo a la orientación y posición de una cámara virtual para mapear las coordenadas 3D de la escena al sistema de coordenadas 3D de la cámara.

- *Iluminación y sombreado*: son un conjunto de procesos que consisten en modelar la interacción de la luz con los materiales de los objetos y a través de ellos dar una apariencia lo más realista posible a la imagen final; en estos procesos las normales a las superficies tienen un papel fundamental.

- *Proyección*: este proceso consiste en mapear la escena, cuyas coordenadas están en el marco de la cámara, a un plano. La proyección puede ser ortogonal o en perspectiva. Cuando se hace una proyección ortogonal, el tamaño de los

objetos no se altera en función de su distancia a la cámara; cuando se realiza en perspectiva, los objetos cambian su tamaño de acuerdo con su distancia a la cámara.

    - *Clipping*: es un subproceso encargado de eliminar aquellas partes de la escena que no son visibles desde el cubo, o pirámide truncada, de proyección (la región del espacio 3D que será visible desde el plano).

- *Rasterización* (*Rastering*): es el conjunto de procesos que producen los valores finales de cada píxel al discretizar los valores de la escena que ha sido proyectada sobre el plano, obtenidos en la operación de proyección. Los valores discretizados se almacenan en una región de memoria conocida como *frame buffer*. Esta operación incluye procesos tales como:

    - *Determinación de superficie visible*. Este proceso consiste en la eliminación de las regiones de los objetos que se encuentran detrás de otros objetos en relación con el observador (*buffer* de profundidad o *z-buffer*).
    - *Mapeo de Texturas*. Proceso que usa imágenes 2D, discretas o continuas, para asignar diversos valores o propiedades a las superficies de los objetos o a la escena con la finalidad de otorgarles alguna propiedad de realismo; estas propiedades incluyen color, detalles, o sombras.

- *Despliegue de la imagen*: los valores de los píxeles finales en el *frame buffer* son mostrados en la pantalla, o cualquier otro dispositivo de despliegue; está etapa también es conocida como *renderizado* (*rendering*).

De este conjunto de procesos podemos afirmar que el modelo matemático de graficación por computadora es $\psi : \mathbb{R}^3 \to \mathbb{P}$, donde ψ representa al modelo óptico que proyecta una escena $\subset \mathbb{R}^3$ (vértices en un espacio tridimensional) a un espacio $\mathbb{P} \subset \mathbb{N}^2$ (píxeles en una imagen).

Varias de las etapas del proceso de renderizado son amplias áreas de investigación por sí mismas, por ejemplo, el modelado geométrico [104, 36] o de materiales [24], o

de las estructuras de datos apropiadas. En el área de modelos de iluminación existen básicamente dos tendencias en el manejo de la interacción de la luz con la escena, uno genera imágenes rápidamente pero los efectos ópticos que puede representar son limitados y el otro reproduce más y mejores efectos ópticos pero requiere de más tiempo y recursos; sin embargo, el *pipeline* general no cambia de forma importante (normalmente, el cambio se da en el orden en que se llevan a cabo las etapas) independientemente del modelo de iluminación que se esté utilizando.

Los términos de realismo y foto-realismo se usan, tanto en arte como en graficación por computadora, para referirse a una imagen que captura de forma relativamente precisa los efectos de la luz al interactuar con los objetos físicos y es casi indistinguible de aquellas que capturaría una cámara, o un dispositivo óptico, para una escena similar. Aunque para ciertas aplicaciones de graficación por computadora no es necesario el foto-realismo, recientemente ha crecido el interés en generar escenas realistas debido, en parte, al aumento del poder de cómputo generado por el desarrollo de *hardware* como las tarjetas gráficas.

El modelo de iluminación es crucial en el proceso de renderizado y en la generación de imágenes más agradables a la vista y en los casos en que se requieran imágenes realistas, la contribución de este trabajo se encuentra en esa clase de modelos; por ello, en las siguientes secciones elaboraremos acerca de ellos.

1.1.2. Modelado Geométrico

Las computadoras han facilitado el estudio de objetos tridimensionales reales, sin embargo, estos necesitan discretizarse para ser adecuados para el procesamiento por computadora. Una posible forma de discretizar un objeto es muestrear de alguna forma su superficie (por ejemplo, con un sistema RGB-D, escáner que regresa la distancia además del color del punto medido). Otra forma de obtener la representación discreta de un objeto 3D real consiste en reemplazarlo por un conjunto discreto de sus puntos; en tal representación, el vecindario Voronoi de cada uno de esos puntos forma un elemento de volumen (*vóxel,* se pueden pensar en un vóxel como la generalización

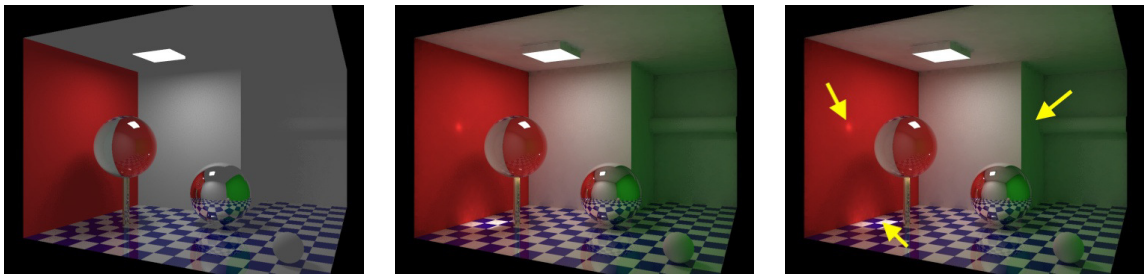
a tres dimensiones de un píxel), el cuál es un poliedro cuyas caras dependen de la posición del conjunto discreto de puntos; en este trabajo sólo consideraremos vóxeles cúbicos (es decir, cubos que representan un valor o un promedio de valores contenido en su volumen) y por ello, nos referiremos a objetos 3D voxelizados a aquellos objetos discretizados con vóxeles cúbicos. Una consecuencia directa de la discretización de los objetos es que el estudio de sus propiedades geométricas ha tomado gran importancia, en particular en áreas como los gráficos por computadoras, el procesamiento digital de imágenes, la visión computacional y el reconocimiento de patrones [67].

Las propiedades de un objeto 3D voxelizado de interés incluyen su volumen, su topología [68] y su superficie. Una superficie puede usarse, entre otras tareas importantes, para visualizar un objeto 3D. Ha habido varios trabajos para extraer o aproximar las superficies de objetos 3D voxelizados a través de un proceso usualmente llamado reconstrucción, el cuál produce una malla poligonal.

1.1.3. Modelos de iluminación

Una parte importante del proceso de generación de una imagen por computadora es el modelado de la interacción de la luz con los materiales asignados a los objetos en una escena, ya que esto influye en la preservación de la apariencia tridimensional de la escena al momento de realizar la proyección de la escena. El color que se percibe de un objeto real no depende sólo del objeto mismo y sus propiedades, si no también se ve influido por las propiedades de la fuente de luz, el color de los objetos que rodean al objeto y el sistema óptico que lo observa [29], ya sea el sistema de visión humano o la tecnología que capturó la escena. Por ejemplo, si en una imagen generada por computadora sólo se "colorean" las superficies de los objetos éstas aparecerán como siluetas planas, lo que claramente no es un comportamiento esperado; por lo tanto, el siguiente paso para lograr realismo es sombrear las superficies visibles con base en la distancia a las fuentes de luz, de esta forma se puede dar la ilusión de tridimensionalidad.

Los procesos encargados de generar esta apariencia involucran el modelado de la



(a) (b) (c)

Figura 1.2: Imagen que muestra la diferencia entre los modelos de iluminación (a) local y (b) global, en esta última pueden notarse efectos que no están presentes en la imagen superior, tales como caústicas y transferencias de color; en (c) se señalan estos efectos con flechas amarillas (Imágenes de uso libre obtenidas de [114, 115]).

interacción de las fuentes de luz con los objetos en la escena, dichos procesos son conocidos como *modelos de iluminación*. En general, se puede hacer una clasificación de dos grupos de métodos para modelar el comportamiento de la luz, los que realizan cálculos rápidos, conocidos como modelos de *iluminación local*, y los que tardan más tiempo pero generan imágenes más realistas, conocidos como modelos de *iluminación global* [29]. La principal diferencia entre ambos tipos de métodos es la forma en que manejan la luz que llega indirectamente a los objetos. En principio, es relativamente sencillo encontrar la cantidad de luz que llega directamente de la fuente de iluminación al objeto cuando no hay oclusión, entre más cerca a la fuente de luz se encuentre el objeto éste debe verse más brillante. Sin embargo, si en el proceso se toma en cuenta la luz que llega indirectamente al objeto debido a las reflexiones de los rayos de luz en otros objetos de la escena los modelos se vuelven más complicados. Para este caso, los modelos de iluminación locales se restringen a calcular únicamente la iluminación directa y consideran que la iluminación debida a otras reflexiones puede manejarse como una constante, mientras que los modelos de iluminación globales siguen algunas de estas reflexiones para calcular su contribución en la escena. Una comparación entre ambos modelos se puede observar en la Figura 1.2, donde es notable que el modelo de iluminación global produce fenómenos que no son visibles en el modelo de iluminación local (por ejemplo, concentración de luz y transferencia de colores).

Ambos modelos serán explicados brevemente a continuación, pero revisaremos los

modelos globales y la importancia de la iluminación en la visualización de imágenes con más detalle en el Capítulo 5.

1.1.3.1. Modelos de Iluminación local

Los investigadores de gráficos por computadora comúnmente aproximan las reglas fundamentales de la óptica y la radiación térmica, ya sea para simplificar los cálculos o porque no se conocen modelos más precisos [29]. En consecuencia, muchos de los modelos de iluminación utilizados en aplicaciones prácticas y/o interactivas, tales como videojuegos, consisten en trucos o simplificaciones extremas, pero que dan resultados visuales aceptables en la práctica.

Los modelos que sólo toman en cuenta la luz que llega directamente desde las fuentes de luz al punto en la escena que está siendo sombreado son conocidos como *modelos de iluminación local*. Uno de los métodos más representativos de dicho enfoque es el modelo de iluminación de Phong [99], que además es de los modelos más empleados debido a su simpleza y resultados suficientemente buenos para la percepción visual humana. Este modelo supone que la intensidad de la luz I en un punto $\boldsymbol{x}$ de la superficie y su reflejo pueden calcularse como la combinación lineal de tres componentes de intensidades de luz independientes: la componente ambiental (la luz que ha rebotado de todas las superficies en la escena, típicamente constante por lo que no captura mucha información), la componente difusa (la luz que llega al objeto cuyo valor depende de la dirección y distancia a la fuente de luz) y la componente especular (la luz reflejada por las superficies brillantes). La ecuación del modelo es

$$I = I_a(\boldsymbol{x})\kappa_a + I_e(\boldsymbol{x})\kappa_e \left\langle \mathbf{r} \cdot \mathbf{e} \right\rangle^{\gamma} + I_d(\boldsymbol{x})\kappa_d \left\langle \vec{\mathbf{n}} \cdot \boldsymbol{\ell} \right\rangle , \tag{1.1}$$

donde $I_a(\boldsymbol{x})$, $I_e(\boldsymbol{x})$ e $I_d(\boldsymbol{x})$ son las intensidades de la luz ambiental, especular y difusa, respectivamente, κ_a, κ_e y $\kappa_d \in \mathbb{R}_+$, son coeficientes para sopesar la intensidad de la reflexión de cada componente de luz, $\mathbf{r}$ es el vector de dirección de la reflexión, $\mathbf{e}$ el vector de dirección del observador o cámara, $\vec{\mathbf{n}}$ es el vector unitario normal a la superficie en el punto $\boldsymbol{x}$, $\boldsymbol{\ell}$ es el vector de dirección de la luz (todos los vectores están

definidos en $\mathbb{R}^3$) y γ es el exponente de reflexión especular. En esta representación, los vectores normalizados son representados con una flecha encima, por ejemplo el vector dirección $\overrightarrow{\boldsymbol{o}}$. Adicionalmente, la operación $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ entre dos vectores $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ es el producto interno (producto punto).

Parte del modelado de los objetos de una escena implica asignarles un material y cada tipo de material tiene cierta sensibilidad a cada componente de la luz. Por lo tanto, el color final de un píxel, que se encuentra dentro de la proyección de un objeto en la escena, se calcula en función del material y las propiedades de las luces que interactúan con su superficie. Comúnmente, la *Función de Distribución de Reflectancia Bidireccional* (o *BRDF* por sus siglas en Inglés) [91] se emplea como formalismo para describir la reflexión de la luz sobre una superficie en dirección al observador a lo largo de una orientación de incidencia de la luz. En la literatura existen una gran cantidad de métodos para crear BRDFs de acuerdo a los distintos materiales que se quieran modelar [98]. Por ejemplo, el modelo descrito por la ecuación (1.1) describe muy bien las propiedades de los materiales plásticos. Los modelos de iluminación local no generan de forma "natural" efectos más complejos como sombras o traslucidez, por lo que es común hacer uso de algún truco de graficación para incorporarlos al proceso de renderizado.

1.1.3.2. Modelos de Iluminación Global

La creación de imágenes realistas es una meta en campos como la simulación, el diseño, el entretenimiento, la publicidad, la investigación y la educación. Crear imágenes realistas generadas por computadora puede ser una forma más asequible y efectiva de ver resultados preliminares, principalmente en diseño de prototipos o arquitectura.

En muchas ocasiones, crear imágenes complejas tiene un costo menor a la utilización de los objetos físicos reales; además de que también permite generar escenarios imposibles en la realidad. Por otra parte, las imágenes realistas se están convirtiendo en una herramienta esencial para la investigación y la educación ya que pueden utilizarse para analizar, explicar o simular diversos fenómenos.

Un *modelo de iluminación global* calcula el color de un punto sobre un objeto en la escena usando tanto la luz emitida directamente por las fuentes de luz, así como la luz que alcanza dicho punto después de su reflexión y transmisión a través de varias superficies. Es decir, también toma en cuenta la luz indirectamente reflejada y transmitida, a diferencia de los métodos locales que han modelado esa luz con un término de iluminación ambiental que se mantiene como constante para todos los puntos en todos los objetos. El valor de esta constante no depende de la posición del objeto ni del observador, ni toma en cuenta la presencia, o ausencia, de objetos cercanos que pudieran bloquear, o permitir el paso de, la luz ambiental. Sin embargo, mucha de la luz en los ambientes reales no viene de fuentes de luz directas si no de los rebotes de la luz presente en, y fuera de, la escena. Al considerar que la componente ambiental no es constante se pueden obtener efectos como las cáusticas (la envolvente de rayos de luz reflejados o refractados por una superficie u objeto curvo, o bien, la proyección de esa envolvente de rayos en otra superficie), zonas de penumbra y transferencia de color, fenómenos que no se pueden simular con modelos locales, como puede verse en la Figura 1.2.

Dentro de los enfoques que se pueden seguir para llevar a cabo un modelado global de iluminación se destacan los enfoques que utilizan *ray tracing*, concepto introducido en [2] y que consiste en lanzar rayos desde el plano de proyección o pantalla y encontrar las intersecciones más cercanas al observador de los rayos con los objetos en la escena, acumulando la luz reflejada por el rebote de los rayos para determinar el color en dichas intersecciones; y los que utilizan el método de radiosidad (*radiosity*) [38, 40], el cual se encuentra basado en modelos de ingeniería térmica para la emisión y reflexión de radiación.

El algoritmo de *ray tracing* intercala la determinación de la superficie visible con los cálculos de sombras, reflexión y refracción, por lo que es considerado un método dependiente del punto de vista. Los algoritmos dependientes de la vista son adecuados para manejar fenómenos especulares que son altamente dependientes de la posición del observador, pero puede que realicen trabajo extra cuando modelan fenómenos difusos que cambian poco a lo largo de grandes áreas de la imagen o

entre imágenes hechas desde distintos puntos de vista. En contraste, el método de radiosidad separa completamente la determinación del sombreado de la determinación de superficie visible. Los algoritmos independientes del punto de vista, como *radiosity*, modelan los fenómenos difusos de forma eficiente pero requieren una gran cantidad de almacenamiento para capturar suficiente información sobre los fenómenos especulares [57]. Sin embargo, para obtener un renderizado "completo" se suelen modelar todas las interacciones del ambiente con fuentes de luz, primero en una etapa independiente de la vista, para posteriormente calcular una o más imágenes para los puntos de vista deseados usando algoritmos convencionales de superficie visible y sombreado [57, 58].

A medida que se ha buscado incluir efectos más complejos en los modelos de iluminación se han introducido funciones similares al concepto de BRDF que capturan con más precisión la interacción entre la luz y los materiales. Ejemplos de ello son las BSDFs [98](Función de Distribución de Dispersión Bidireccional o *Bidirectional Scattering Distribution Function*) que tratan de simular la forma en que la luz es dispersada desde una superficie dependiendo de las propiedades de un material. De igual forma, la *Bidirectional Scattering Surface Reflectance Distribution Function*, BSSRDF [98], permite renderizar superficies traslúcidas con dispersión de luz en subsuperficies (capas).

Aunque ciertamente los modelos globales son más completos, una de sus premisas dominantes es modelar la luz como partículas, dejando de lado el comportamiento de onda de la luz que, entre otras cosas, produce efectos ópticos que se obvian o se obtienen de manera "artificial", es ahí donde se encuentra la contribución de este trabajo y por lo tanto nos enfocaremos a la descripción de dichos efectos y los trabajos que se han realizado al respecto.

1.1.3.3. Importancia de las normales en la iluminación

De la ecuación (1.1) se puede notar que la normal a la superficie es fundamental para los cálculos de iluminación, para el proceso de visualización de la superficie y para crear la sensación de tridimensionalidad. En el proceso de creación del modelo geométrico para representar un objeto se acostumbra el cálculo de las normales a

la superficie. Típicamente, se genera una malla poligonal donde cada polígono tiene asociada al menos una normal. De esta situación podemos deducir que entre más polígonos conformen nuestro modelo geométrico tendremos más normales y mejorará la visualización del objeto discretizado. Por esta razón, ha habido muchos esfuerzos en desarrollar métodos de adquisición y refinación del mallado [13], aunque alternativamente se pueden usar modelos matemáticos que permiten una evaluación más continua de las normales.

Por lo tanto, un buen renderizado puede lograrse a través de una combinación de refinado y suavizado de la malla poligonal extraída, por medio de una asignación de normales suaves a la misma superficie o con una combinación de ambas (porque producir una malla fina sería computacionalmente costoso). Las normales a la superficie ayudan a asignar colores a los polígonos visibles de la malla de forma suave. Esto se lleva a cabo combinando un modelo de iluminación e interpolación del color [39] o de la normal [99], que serán descritos de forma un poco más detallada a continuación.

Sombreado de Gouraud El sombreado de Gouraud es un método de interpolación usado para generar sombreado continuo de superficies representadas por mallas poligonales. Se implementa calculando la iluminación correspondiente (usando cualquier modelo de iluminación, como el de Phong) en los vértices de cada polígono y los colores resultantes se interpolan para cada píxel cubierto por el polígono.

Para cada vértice se debe especificar una estimación de la normal a la superficie sobre cada uno de los vértices en el modelo poligonal 3D, o bien, encontrar una normal promediando las normales a cada uno de los polígonos que coinciden en cada vértice. Usando estas normales estimadas se pueden hacer los cálculos de iluminación que producen un valor de intensidad de color para cada vértice. Posteriormente, para cada píxel que se encuentra en la proyección de la malla poligonal, el color correspondiente puede interpolarse de los valores de color calculados en los vértices, esta interpolación puede ser lineal o hiperbólica.

El problema del sombreado de Gouraud sucede cuando los brillos especulares se encuentran en el medio de un polígono grande, ya que el modelo interpola basado en

el color de los vértices, el brillo estará ausente del interior del polígono.

Sombreado de Phong No debe confundirse con el modelo de iluminación de Phong, aunque es común que se usen en conjunto. Es una técnica de interpolación para sombreado de superficies, interpola las normales de la superficie a lo largo del polígono y calcula los colores correspondientes al píxel basados en las normales interpoladas y el modelo de reflexión.

Mejora el sombreado de Gouraud al generar una mejor aproximación del sombreado de una superficie suave. El sombreado de Phong asume un vector normal a la superficie que varía suavemente, las normales son interpoladas linealmente usando las normales en los vértices y normalizadas en cada píxel para ser usadas en el modelo de iluminación para obtener el valor final del color en cada píxel.

Como las normales son interpoladas a lo largo de la superficie del polígono se evitan los problemas del sombreado de Gouraud, sin embargo, es un método más costoso ya que la iluminación se calcula en cada píxel que se encuentra en la proyección del polígono en vez de en cada vértice.

1.1.4. Efectos de onda

A pesar de que los modelos de iluminación global generan imágenes significativamente más cercanas a los que percibiría un humano (o sea, lo que comúnmente se conoce como *imágenes realistas*) que los modelos de iluminación local, existen varios efectos ópticos que no pueden reproducir debido a que estos modelos no toman en cuenta la naturaleza dual de la luz: los métodos de iluminación global comunes simulan el comportamiento de la luz únicamente como partícula, una aproximación conocida como modelo de óptica geométrica.

Aunque se considera, en general, que incorporar el comportamiento de onda es muy costoso y no contribuye con efectos prominentes a la apariencia de una escena promedio [124, 126], la mayor pérdida por la adopción de modelos geométricos es que los efectos de difracción e interferencia (por ejemplo, efecto tornasol en burbujas de jabón o discos compactos) no pueden reproducirse fácilmente [98], ni tampoco aque-

llos patrones de oscurecimiento o decoloración en ciertos materiales (como metales y cristales) dependientes de la polarización. Sin embargo, la evolución del *hardware* y *software* hace que actualmente sea más factible la implementación de modelos de iluminación más exactos y complicados.

Una de las contribuciones del proyecto que se presenta en este documento intenta incorporar efectos más cercanos al modelo físico real en un modelo de iluminación global al incluir efectos debidos al comportamiento de onda de la luz. El trabajo se centra en el modelado del comportamiento de onda que permite la aparición de patrones de difracción (donde la interferencia constructiva y destructiva genera patrones con distinta intensidad o color cuando una onda de luz encuentra un obstáculo o una abertura) y *speckle* (patrón de interferencia en la dispersión de luz láser debido a la rugosidad de una superficie iluminada, con áreas en dónde la luz se ve más intensa por interferencia constructiva y áreas oscuras por interferencia destructiva) en la difusión del rayo láser y el modelado de luz polarizada.

1.1.5. Justificación y objetivos

Un problema en el área de graficación por computadora es generar imágenes que sean visualmente atractivas, que se generen de forma rápida y eficiente pero que adicionalmente representen de manera cercana a la realidad los objetos y sus interacciones con la luz sin llegar a ser simulaciones físicas o representaciones exactas, no porque no se quiera tener dichas simulaciones y representaciones exactas, si no porque las restricciones de *hardware* y *software* no lo permiten. Por lo tanto, es importante tener aproximaciones que generen resultados aceptables y que, de ser posible, tengan una base física. Como lo hemos visto en este capítulo, los modelos de iluminación afectan fuertemente las imágenes resultantes del proceso de renderizado. El objetivo general de este trabajo es mejorar la apreciación visual de dichas imágenes a través de dos planteamientos que, aunque diferentes, se basan en el modelo de iluminación actual en gráficos por computadora, uno de ellos se aprovecha de sus características para suavizar el renderizado de modelos geométricos voxelizados sin modificar su

mallado y el otro modifica el modelo de iluminación para actualizar algunas de sus limitaciones incorporando efectos debidos al comportamiento dual de la luz. En ambos casos se busca aprovechar el funcionamiento de los modelos de iluminación y los procesos de renderizado estándar sin realizar modificaciones muy fuertes para mejorar las imágenes, ya sea suavizando el resultado final o añadiendo efectos de onda.

En particular, para el primer planteamiento, tomamos en cuenta que un "buen" renderizado de una malla poligonal burda, en particular aquellas extraídas de objetos 3D voxelizados, puede lograrse por medio de la combinación de refinar y suavizar la malla, a través de la asignación de normales suaves a la misma malla, o bien, una combinación de ambos procesos (porque producir una malla fina es computacionalmente costoso). En este trabajo seguiremos el segundo enfoque, estimar las normales directamente de las superficies sin hacer modificaciones a los objetos 3D voxelizados, pero tomando en cuenta información de cómo se discretizó el objeto o, alternativamente, su geometría. A pesar de producir renderizados burdos, las superficies obtenidas por algoritmos de seguimiento de superficies tienen una gran ventaja: preservan la topología de los objetos 3D voxelizados. Tales representaciones también tienen propiedades geométricas deseables (por ejemplo, no producen artefactos complejos, tales como agujeros) y el proceso para generar una malla rectangular (aquella extraída de un objeto 3D voxelizado y por ello formada por un conjunto de rectángulos) es rápido y eficiente. Otra ventaja de usar una malla rectangular producida por un algoritmo de seguimiento de superficies es que no introduce un sesgo adicional a la superficie, ya que la misma es representada por las caras rectangulares de los vóxeles cúbicos producidos por el proceso de adquisición de datos. En este trabajo proponemos un método que asigna normales suaves a una superficie discreta evaluando, en la posición de sus vértices, el gradiente de una combinación lineal de funciones base suaves que son simétricas esféricamente, tienen soporte compacto y son diferenciables de forma múltiple; nos referiremos a estas funciones como *blobs* (nombre dado por la semejanza entre la mancha generada por la punta de una pluma fuente sobre papel y la visualización de estas funciones base). De esta manera creamos una aproximación a un objeto 3D discreto que es continua y que al visualizar su superficie rectangular

puede lograr renderizados suaves sin necesidad de generar una malla compleja o sofisticada; esto se logra por medio de la asignación de normales suaves a la superficie del objeto. Para asignar normales a la superficie discreta, el método presentado en este trabajo usa dos aproximaciones. La primera asume la existencia de información sobre la representación del objeto 3D real, antes de discretizarse, a través de la combinación lineal de los *blobs*. Es importante notar que existen métodos que producen este tipo de aproximaciones continuas de objetos 3D por medio de una combinación lineal de funciones base; en particular, los métodos de reconstrucción de imágenes a través de proyecciones obtenidas de un objeto 3D físico o virtual. Como resultado, una aproximación continua obtenida por tales métodos puede usarse *a priori* para asignar normales suaves a la superficie discreta extraída del objeto 3D voxelizado. Por el contrario, el segundo no asume la existencia de información subyacente sobre el objeto 3D real, en su lugar, crea una aproximación continua del objeto 3D voxelizado usando una combinación lineal de *blobs* que sólo utiliza la información acerca del objeto 3D voxelizado. Usando principios de procesamiento de señales seleccionamos los parámetros de los *blobs* que producen normales que resultan en las representaciones buscadas del objeto suave. Los resultados obtenidos muestran que existe un suavizado en el renderizado final, aunque por circunstancias ajenas al modelo algunas visualizaciones presentan artefactos.

Para el segundo planteamiento, buscamos mejorar la generación de imágenes foto-realistas, imágenes que son importantes por sus aplicaciones en las áreas del entretenimiento, el análisis y la simulación. Adicionalmente, las imágenes foto-realistas pueden ser utilizadas como una métrica para la evaluación de la calidad de otros métodos de renderizado. Casi ninguno de los trabajos revisados para este proyecto ha presentado resultados de un modelo de onda que tome en cuenta los efectos de difracción, interferencia y polarización en un entorno unificado ([45] lo definió teóricamente y [103] lo define sólo para gotas de agua). Por lo tanto, el presente proyecto propone un modelo de luz estocástica que incluya todos los efectos mencionados y muestre un comportamiento más cercano al modelo físico real. En particular, nos interesa modelar el comportamiento de onda que permite la aparición de patrones de difracción y *speckle*

en la difusión de un rayo láser y el modelado de luz polarizada. La meta fundamental del trabajo es generar imágenes que, además de ser visualmente atractivas, reflejen de forma más fiel la realidad; por lo tanto, es importante que la aproximación se encuentre basada en un modelo físico (aunque no sea completamente exacto) y produzca efectos interesantes. A pesar de que el campo de graficación por computadora se sirve de modelos del comportamiento real de luz y materiales, en general no pretende hacer una simulación exacta de los fenómenos involucrados, ya que esto implica un gran número de cálculos y memoria que harían que los métodos fueran prohibitivos. Sin embargo, tener modelos sustentados en la física ayuda a generar imágenes realistas, de ahí la importancia de incorporarlos a los modelos de graficación. Para ello hacemos uso de la teoría en óptica de Fourier para calcular el patrón de difracción resultante de un objeto difractante y aprovechamos los métodos de mapeo de texturas para colocarlo en dirección de la reflexión especular sobre el objeto, además de hacer un manejo de la polarización a través de un mecanismo conocido como vectores de Stokes y matrices de Mueller. Los resultados muestran como las imágenes presentan los efectos de cambios de color esperados debido a la difracción aunque el método presentado tiene varias limitaciones, principalmente que la formación del patrón de difracción no depende de la distancia del objeto a la fuente de luz.

El resto del documento está dividido en dos partes, la primera describe la solución al problema de suavizar modelos geométricos voxelizados sin modificar la malla subyacente y se encuentra organizada de la siguiente manera, el Capítulo 2 presenta los detalles de la obtención de superficies a partir de funciones discretizadas, el Capítulo 3 ahondará en los métodos propuestos de aproximación de normales para superficies voxelizadas y el Capítulo 4 muestra resultados y comparaciones visuales de dichos métodos. Mientras que la segunda parte describe el método para incorporar efectos de onda en un renderizador basado en física y se encuentra organizada de la siguiente manera, el Capítulo 5 ahondará en los modelos de iluminación global, los trabajos relacionados, así como los formalismos para la descripción de la luz; el Capítulo 6 presentará la propuesta del modelo unificado para incorporar efectos de onda de difracción y polarización; el Capítulo 7 presenta los experimentos y resultados visuales;

y finalmente, las conclusiones y la discusión pueden encontrarse en el Capítulo 8.

Capítulo 2

Renderizado de Superficies: Antecedentes

Existen varios métodos que permiten obtener modelos geométricos para realizar un renderizado [3, 5, 13], en este capítulo nos enfocaremos en aquellos modelos que se obtienen a partir de la extracción de una superficie de un objeto 3D voxelizado. La superficie discreta de dicho objeto puede obtenerse fácilmente usando un algoritmo de seguimiento de superficie. Esos algoritmos producen un grafo no dirigido (también llamado malla), que representa un conjunto de caras rectangulares adyacentes. Tal malla puede utilizarse para representar el objeto 3D voxelizado en una pantalla de computadora usando técnicas de graficación por computadora estándar para el renderizado de superficies. Desafortunadamente, los renderizados de las mallas rectangulares producen imágenes burdas o "voxelizadas", algo que puede mejorarse ya sea suavizando la malla o asignándole normales suaves.

En este capítulo se ahondará sobre los trabajos en la literatura relacionados con suavizar el mallado y asignar normales más suaves a la superficie, en la obtención de las superficies voxelizadas, así como en la razón de la selección de las funciones base para la asignación de normales de nuestro método publicado en [14][1].

[1]Copyright Topology Proceedings
Reproducción de información y resultados con permiso de la revista.

2.1. Trabajos relacionados

Los trabajos más recientes involucrando asignación de normales se enfocan en extraer las superficies y estimar las normales a partir de nubes de puntos (un estudio cubriendo métodos hasta el 2014 puede encontrarse en [5]) o en suavizar la geometría del modelo al subdividir la malla de la superficie ([13] presenta un estudio de estos métodos hasta el 2012), sin embargo, concentramos esta revisión de trabajos relacionados en aquellos con aproximaciones que se basan únicamente en modificar las normales sin modificar la malla de la superficie. Un estudio completo de métodos anteriores a 1992 para asignar normales a superficies discretas y sus respectivos rendimientos puede encontrarse en [136], a continuación nos concentramos en trabajos más recientes.

Uno de los métodos más antiguos para asignar las normales apropiadas a una superficie discreta fue propuesto por Chen et al. [16] quiénes sugirieron que, para una superficie discreta hecha de caras rectangulares y extraídas de un objeto 3D voxelizado conformado de un conjunto finito de vóxeles cúbicos, se puede colocar una normal por cada cara usando el producto vector (producto cruz) cuya orientación puede calcularse usando la configuración local de sus caras adyacentes. Otros trabajos relacionados han propuesto el cálculo de normales basado en el promedio ponderado de una vecindad; algunas aproximaciones usan un tamaño fijo para dicha vecindad mientras otros proponen un tamaño variable. Un ejemplo de la primera aproximación es [96], dónde se propone una generalización del método presentado en [16]. Por otro lado, el método en [117] adapta el tamaño de la vecindad dependiendo del plano tangente así como el ángulo entre normales; un método similar se presentó en [59], en dónde las normales deseadas son estimadas por la suma ponderada de las normales asignadas a las caras vecinas y el peso asignado a la cara es calculado de acuerdo al grado de similitud a las caras vecinas. Otro método que promedia vecinos para estimar las normales en una superficie discreta fue propuesto en [31] en dónde para una cara dada los centros de las caras dentro de su vecindario son promediados y posteriormente los vectores en el espacio tangente a la superficie se obtienen de sus

derivadas parciales. Los vectores normales finales se obtienen con el producto cruz de los vectores en el espacio tangente. Que tanto se promedian los centros depende de una máscara de promediado predefinida con soporte local y un número de convoluciones iterativas, de esta forma es posible controlar el tamaño de la vecindad que afecta el promediado.

Por otra parte, los autores de [78] proponen calcular las normales a una superficie discreta al aproximar los momentos 3D de valores del objeto 3D voxelizado, estos momentos son calculados sobre una región esférica con un radio predeterminado. El método relacionado propuesto en [70] sugiere un estimador de curvatura para producir renderizados suaves, este método también sigue el límite del objeto binario usando una extensión del método propuesto por Artzy et al. [3]. Después, el vector normal asociado a la superficie discreta se obtiene moviendo un *kernel* esférico de tamaño predefinido a lo largo del borde y calculando integrales volumétricas invariantes en su intersección. Para obtener la curvatura y sus normales, el método calcula los eigenvectores asociados a la matriz de covariancia en dichas intersecciones y extrae algunas cantidades diferenciales, tales como la primera y segunda curvaturas, vectores normales o vectores tangentes.

Otra aproximación a la estimación de normales usa el producto cruz entre dos vectores tangenciales. Cada cara en la superficie discreta es cruzada exactamente por dos planos perpendiculares a los ejes canónicos y los contornos de la superficie están contenidos en dichos planos. Los vectores tangenciales para cada cara en la superficie discreta son obtenidos de los contornos 2D cruzándola [74, 116]. Es posible considerar las contribuciones de los vecinos realizando una convolución con un *kernel* Gaussiano para producir normales que varían suavemente [74]. De manera similar, Braquelaire et al. [10, 66] definieron y usaron redes Euclideanas (es decir, una extensión de caminos Euclidianos en dónde cada punto de la superficie puede moverse dentro del vóxel limitante que lo contiene) y los puntos de la superficie discreta son movidos de acuerdo a una proyección en los planos tangentes discretos.

Otros métodos proponen el uso de una función que actúe como filtro de interpolación para suavizar la reconstrucción en combinación con un filtro derivativo para

la estimación de normales. Möller et al. [83] presentaron un método para analizar, clasificar y evaluar esos filtros usando una expansión de series de Taylor de la suma de convolución; este método fue usado para encontrar un nuevo filtro derivativo. En otra aproximación de este tipo, Sramek and Kaufman [110] propusieron el uso de una función de densidad que depende de un mapa de distancias no lineal (un campo de distancias truncado) para representar sólidos y superficies, entonces usaron filtros derivativos para calcular las normales. Este método se extendió en [93] para representar detalles nítidos, detectando y removiendo artefactos al corregir el campo de distancia en una etapa de postprocesamiento.

Por otro lado, Hong et al. [52] usaron una función polinomial de segundo grado para representar una distribución de densidad en un vecindario local. Este método obtiene los coeficientes de la función polinomial minimizando el error de la aproximación y calcula el vector gradiente en un punto arbitrario usando la derivada analítica. Alternativamente, los autores de [28] proponen un algoritmo basado en obtener el gradiente del mapa de distancias de fondo discreto de la imagen 3D original.

Otra aproximación propuesta por los autores de [71] usa una estructura de datos tipo *octree* disperso para almacenar datos de vóxeles, incluyendo información del contorno para incrementar la resolución geométrica y permite una codificación más compacta de superficies suaves. Posteriormente, una técnica de filtrado en postprocesamiento es usada para suavizar el efecto de bloques causado por el muestreo discreto de los atributos de sombreado.

Otra aproximación, propuesta en [101], trata de resolver inconsistencias entre las normales de cara y las normales usadas para asignar colores en imágenes renderizadas (es decir, sombreado). El método propuesto permite reflexiones físicamente plausibles y usa un parámetro escalar que caracteriza la desviación entre las normales por vértice y las normales por cara para ajustar linealmente las normales interpoladas para crear la apariencia de una superficie suave.

2.2. Superficies Discretas

Los dispositivos de imagenología producen un conjunto de valores acomodados en el espacio 3D; esto es particularmente cierto para dispositivos biomédicos. Los valores físicos caracterizan la modalidad de imagenología; por ejemplo, en la Tomografía Computarizada de rayos x, el escáner mide los coeficientes de atenuación de rayos x dentro del cuerpo que se está estudiando y en la Tomografía de Emisión de Positrones, el escáner mide la concentración de un radionucleido emisor de positrones. El acomodo más común para el conjunto de valores físicos medidos es la llamada rejilla cúbica simple, aunque existen propuestas para otro tipo de configuraciones. La rejilla está definida por

$$G_{\Delta_\gamma} = \left\{ \Delta_\gamma \boldsymbol{x} \,|\, \boldsymbol{x} \in \mathbb{Z}^3 \text{ and } \Delta_\gamma \in \mathbb{R}_+ \right\} \tag{2.1}$$

donde Δ_γ es conocida como la distancia de muestreo y los vecindarios de Voronoi asociados con esta rejilla son los vóxeles cúbicos cuya longitud de arista es Δ_γ. Es importante notar que los dispositivos de imagenología o discretización solo producen valores físicos para un subconjunto de G_{Δ_γ}.

Por lo tanto, un conjunto de datos producidos por un dispositivo de imagenología puede ser caracterizado por el conjunto G_{Δ_γ} y la función $f \in \mathbb{R}$ tal que para una ubicación $\boldsymbol{x} \in G_{\Delta_\gamma}$, $f(\boldsymbol{x})$ produce el valor físico medido en tal ubicación; este valor es asignado al vecindario de Voronoi asociado a la posición $\boldsymbol{x}$ (es decir, al vóxel centrado en $\boldsymbol{x}$ se le asigna el valor físico de $f(\boldsymbol{x})$). Típicamente, hay varios objetos representados en un par dado $\left(G_{\Delta_\gamma},\, f\right)$, por ejemplo, varios órganos en un estudio biomédico. Para identificar cuales vóxeles en un par $\left(G_{\Delta_\gamma},\, f\right)$ representan un objeto dado, un método de segmentación (la aplicación de un método que determina cuales puntos de la función de densidad discreta pertenecen al objeto y cuales puntos no, un proceso también conocido como binarización) es aplicado para obtener el par $\left(G_{\Delta_\gamma}, \{0,1\}\right)$ en dónde el valor uno se asigna a los vóxeles que representan el objeto de interés y cero de otra manera. Como resultado, el método de segmentación particiona G_{Δ_γ} en subconjuntos $\mathcal{I}$, el conjunto de vóxeles con valor uno, y $\mathcal{E}$, el conjunto de vóxeles con

valor cero. Decimos que un *surfel* (elemento de superficie, una cara cuadrada en una rejilla cúbica simple) es la intersección entre un vóxel $\boldsymbol{c}$ en $\mathcal{I}$ y un vóxel $\boldsymbol{d}$ en $\mathcal{E}$ tal que $(\boldsymbol{c}, \boldsymbol{d}) \in \omega$, donde $(\boldsymbol{c}, \boldsymbol{d}) \in \omega \Leftrightarrow ||\boldsymbol{c} - \boldsymbol{d}|| = 1$. El límite entre los conjuntos $\mathcal{I}$ y $\mathcal{E}$ es tal que

$$\partial(\mathcal{I}, \mathcal{E}) = \{(\boldsymbol{c}, \boldsymbol{d}) \mid \boldsymbol{c} \in \mathcal{I}, \boldsymbol{d} \in \mathcal{E}, (\boldsymbol{c}, \boldsymbol{d}) \in \omega\}. \tag{2.2}$$

El límite se conforma de todos los *surfels* en $\left(G_{\Delta_\gamma}, \{0, 1\}\right)$ y nos referiremos a él como la superficie discreta S. Un algoritmo eficiente para obtener la superficie discreta S fue propuesto por Artzy et al. en [3] y nosotros lo usaremos para obtener las superficies de este trabajo. Vale la pena mencionar que en la práctica es posible representar una superficie discreta S al almacenar ya sea el centro de cada uno de sus *surfels* o un conjunto de todos los vértices de cada uno de los *surfels*. El último es el camino más común en graficación por computadora, ya que permite representar cualquier superficie poligonal. Por lo tanto, en este trabajo usamos dicha representación y denotamos el conjunto de todos los vértices de una superficie discreta S como el conjunto V.

2.3. Aproximación de normales

Como se mencionó anteriormente, podemos representar un objeto tridimensional real por medio de un conjunto discreto de puntos, normalmente generando vóxeles a través de un método de reconstrucción que posteriormente se usan para producir una malla poligonal.

En el proceso de renderizado de dicha malla, es una práctica común calcular las normales a la superficie; tales vectores se usan, entre otras cosas, en el proceso de visualización de la superficie. Las normales a la superficie ayudan a asignar colores a los polígonos visibles de la malla de forma suave. Esto se lleva a cabo combinando un modelo de iluminación e interpolación del color [39] o de la normal [99]. Por lo tanto, un buen renderizado puede lograrse a través de la combinación de refinamiento y suavizado de la malla poligonal extraída y de la asignación de normales suaves a la misma superficie o de una combinación de ambas (porque producir una malla fina

sería computacionalmente costoso). En este trabajo seguiremos el segundo enfoque, estimando las normales directamente de las superficies sin modificación de los objetos 3D discretos que comprenden un conjunto finito de vóxeles cúbicos.

Podemos obtener la superficie discreta de un objeto 3D voxelizado usando un algoritmo de seguimiento de superficie. La malla (un grafo no dirigido) resultante de este algoritmo representa un conjunto de caras rectangulares adyacentes. Se puede considerar que tal superficie corresponde al objeto 3D voxelizado y puede utilizarse para proyectarse sobre una pantalla de computadora usando técnicas de graficación por computadora estándar para el renderizado de superficies. Desafortunadamente, los renderizados de las mallas rectangulares producen imágenes muy voxelizadas (burdas), algo que puede mejorarse ya sea suavizando la malla o asignándole normales suaves. A pesar de producir renderizados burdos, las superficies obtenidas por algoritmos de seguimiento de superficies tienen una gran ventaja: preservan la topología de los objetos 3D discretos. Tales representaciones también tienen propiedades geométricas deseables (por ejemplo, no producen artefactos complejos, tales como agujeros) y el proceso de generación de una malla rectangular es rápido y eficiente. Otra ventaja de usar una malla rectangular producida por un algoritmo de seguimiento de superficies es que no introduce un sesgo adicional a la superficie, ya que la misma es representada por las caras rectangulares de los vóxeles producidos por el proceso de adquisición de datos. En este trabajo proponemos un método que asigna normales suaves a una superficie discreta evaluando, en la ubicación de sus vértices, el gradiente de una combinación lineal de funciones base suaves, *blobs*, que son simétricas esféricamente, tienen soporte compacto y son diferenciables de forma múltiple. De esta manera creamos una aproximación continua a un objeto 3D voxelizado con la que se puede lograr un renderizado suave de su superficie discreta sin generar una malla compleja o sofisticada, a través de la asignación de normales suaves a la superficie del objeto. Para asignar normales a la superficie discreta, el método propuesto no asume la existencia de información subyacente sobre el objeto 3D real, en su lugar, crea una aproximación continua del objeto 3D discreto usando una combinación lineal de *blobs*.

2.3.1. Aproximación por una combinación lineal de funciones base

En esta aproximación, ponemos normales suaves en las posiciones en V por medio del uso de una función continua $\upsilon : \mathbb{R}^3 \to \mathbb{R}$ que puede representarse por la siguiente combinación lineal

$$\upsilon(\boldsymbol{x}) = \sum_{j=1}^{J} c_j b_j(\boldsymbol{x}), \tag{2.3}$$

donde $\{b_j\}$ es un conjunto conocido de funciones base, cada una ponderada por el coeficiente correspondiente c_j. El centro de cada función individual b_j de (2.3) se localiza en un punto $\boldsymbol{p}_j$ y nos referiremos a ese conjunto $\{\boldsymbol{p}_j\}$ de puntos como *rejilla*. En la práctica, las funciones base b_j tienen que caer a un valor insignificante más allá de un radio moderado y varias funciones se han sugerido para ello y los vóxeles cúbicos son las funciones más comunes. Sin embargo, las transiciones bruscas de los vóxeles no son apropiadas para representar objetos naturales y, consecuentemente, otras funciones suaves han sido propuestas, tal como las Gaussianas [7], *wavelets* multiescala [87, 88], cuadráticas por partes [92], *splines* [121] y polinomios [133, 134]. Al usar una combinación lineal de funciones base, suponiendo que la aproximación en (2.3) es buena, sabemos que υ debe ser una función continuamente diferenciable y que el gradiente de υ, en cualquier punto, debe estar dado por

$$\nabla\upsilon(\boldsymbol{x}) = \sum_{j=1}^{J} c_j \nabla b_j(\boldsymbol{x}). \tag{2.4}$$

De hecho, aproximar funciones por medio de (2.4) es una práctica común en varios campos; por ejemplo, en graficación por computadora (normalmente conocido como modelo *orgánico* o modelo *blobby*) y en reconstrucción de imágenes a partir de proyecciones; sin embargo, (2.3) es usada de manera diferente en ambos campos. Como mencionamos anteriormente, la elección del conjunto de funciones base $\{b_j\}$ influencia grandemente el resultado de la aproximación por (2.3). En nuestro trabajo usamos funciones con soporte compacto, simetría esférica y transiciones suaves de uno a cero

[75, 76]; funciones comúnmente conocidas como *blobs*. Estos han sido usados como funciones ventana en procesamiento de señales [61] y como funciones base en reconstrucción de imágenes a partir de proyecciones [12, 76, 79, 80, 81]. La forma general de un blob [75] está dada por

$$b(m,\alpha,a;r)=\begin{cases}\frac{I_m(\alpha\, w)}{I_m(\alpha)}w^m, & \text{si} \quad 0\leq r\leq a,\\ 0, & \text{de otra manera,}\end{cases} \tag{2.5}$$

donde I_m denota una función Bessel modificada del primer tipo de orden m (un entero no negativo), $\alpha\in\mathbb{R}_+$, a es el radio del *blob*, r es la distancia radial desde el centro del *blob* y $w=\sqrt{1-\left(\frac{r}{a}\right)^2}$. Los parámetros a y α controlan la suavidad y la forma de *blob* y, por lo tanto, la selección apropiada de los mismos es muy importante. El parámetro m controla la continuidad del *blob* en a: para $m>0$ el *blob* es una función continua con $m-1$ derivadas continuas [76]. Es una práctica común usar $m=2$ porque con ese valor los *blobs* ya son funciones suaves con primeras derivadas continuas y valores más grandes de m no proveen resultados substancialmente diferentes en la aproximación por (2.3). Posteriormente, mostraremos como seleccionar los valores para α y a, una vez que se fija el valor de m, usando principios de procesamiento de señales, así como optimización, lo que resulta en "buenas" aproximaciones de v usando (2.3). Un factor importante para calcular las normales de la función v, usando la aproximación (2.3), es que la función b de (2.5) tenga simetría rotacional, por lo tanto la dirección del gradiente de la función base b_i, centrado en $\{\boldsymbol{p}_i\}$, en la posición $\boldsymbol{x}$ es la misma que la dirección de $(\boldsymbol{x}-\boldsymbol{p}_i)$, y su magnitud está dada por $|\nabla b_i|(\boldsymbol{x})=\frac{\partial b_i}{\partial r}(r)$, donde r es la distancia entre $\boldsymbol{x}$ y $\boldsymbol{p}_i$. La fórmula analítica para calcular $\frac{\partial b_i}{\partial r}(r)$ es (véase [75])

$$\frac{\partial}{\partial r}b(m,\alpha,a;r)=-\frac{r\, z^{m-1}I_{m-1}(z)}{a^2\,\alpha^{m-2}\,I_m(\alpha)},\ 0\leq r\leq a, \tag{2.6}$$

donde $z=\sqrt{\alpha\left(1-\left(\frac{r}{a}\right)^2\right)}$ y las variables m, a y α, así como la función I_m tienen el mismo significado que en (2.5).

Como mencionamos previamente, el método presentado en el siguiente capítulo

asigna normales suaves a una superficie discreta evaluando en sus vértices el gradiente de una combinación lineal de funciones base suaves conocidas como *blobs*. Esto nos permite generar renderizados suaves de una superficie discreta ya que se crea una aproximación continua del objeto 3D voxelizado sin modificar la malla. Este método está basado en los principios presentados en [33], donde se presenta un método para seleccionar parámetros de *blob* adecuados tanto para reconstrucciones a partir de proyecciones como para visualizaciones; sin embargo, la visualización propuesta en ese trabajo es de una representación continua de una combinación lineal de *blobs* usando *raycasting*. Aquí, proponemos el uso de *blobs* para mejorar la visualización de un objeto 3D discreto ubicado en una rejilla cúbica (o sea, voxelizado).

Capítulo 3

Modelo e Implementación para la Aproximación de Normales

En este capítulo describiremos el método general para crear una aproximación continua de un objeto 3D voxelizado, también presentaremos un método que asigna las normales sobre los vértices de la malla bajo la suposición de la existencia de una representación subyacente de una aproximación continua al objeto 3D a partir de una combinación lineal de *blobs*.

3.1. Normales a partir de una rejilla cúbica simple

A continuación presentaremos los dos métodos que proponemos para la asignación de normales sobre las posiciones del conjunto V usando la combinación lineal de *blobs* de (2.3). El primer método sólo usa la superficie discreta S y los conjuntos $\mathcal{I}$ y $\mathcal{E}$ subyacentes. Por otra parte, el segundo método usa la superficie discreta S e información *a priori* sobre la aproximación con (2.3) al objeto 3D real que posteriormente se discretiza para producir el objeto 3D voxelizado.

3.1.1. Método general para asignar normales suaves (GMAN)

Con el fin de asignar normales suaves a las posiciones de V, este método crea una aproximación a un objeto discreto 3D usando la combinación lineal de *blobs* de (2.3) y hace que el conjunto $\{\boldsymbol{p}_j\}$ para ubicar los centros de los *blobs* de (2.3) coincida

con el conjunto de posiciones de G_{Δ_γ}. Por otro lado, en este enfoque, los valores de coeficientes del conjunto $\{c_j\}$ son asignados de la siguiente manera: si la posición $\boldsymbol{p}_j$ asociada con c_j pertenece al conjunto $\mathcal{I}$, asignamos el valor de uno a c_j y cero de otra manera. Finalmente, las normales son asignadas a las posiciones en V al calcular el gradiente de la aproximación continua v en dichas posiciones con (2.6). Como estamos usando la combinación lineal de blobs (2.3) con $\mathcal{I}$ como su conjunto $\left\{\boldsymbol{p}_j\right\}$ solo necesitamos seleccionar los valores apropiados para Δ_γ, α y a. Para este propósito seguimos un enfoque similar al propuesto en [80], que presenta un método que selecciona los parámetros de los *blobs* en una combinación lineal suponiendo que aproximarán una función de valor constante y por ello se asigna $c_j = 1$, para $1 \leq j \leq J$. Para el caso en que $\left\{\boldsymbol{p}_j\right\}$ es un subconjunto G_{Δ_γ}, el lado derecho de (2.3) es una convolución del *blob* b en (2.5) con una versión truncada del tren de pulsos $\text{Ш}_{G_{\Delta_\gamma}}$. La transformada de Fourier de esa convolución es aproximadamente

$$\mathcal{F}\left\{b * \text{Ш}_{G_{\Delta_\gamma}}\right\} = \hat{b} \times \mathcal{F}\left\{\text{Ш}_{G_{\Delta_\gamma}}\right\}, \tag{3.1}$$

donde $\hat{b}$ es la transformada de Fourier del *blob* definido en (2.5) cuya fórmula analítica ha sido provista por Lewitt [76] y para $m = 2$, se define de la siguiente manera:

$$\hat{b}\left(2, \alpha, a; R\right) = \frac{(2\pi)^{\frac{3}{2}} a^3 \alpha^2}{I_2\left(\alpha\right)} \begin{cases} \dfrac{I_{\frac{7}{2}}\left(\sqrt{\alpha^2 - (2\pi a R)^2}\right)}{\left(\sqrt{\alpha^2 - (2\pi a R)^2}\right)^{\frac{7}{2}}}, & \text{si } 2\pi a R \leq \alpha, \\ \dfrac{\mathcal{J}_{\frac{7}{2}}\left(\sqrt{(2\pi a R)^2 - \alpha^2}\right)}{\left(\sqrt{(2\pi a R)^2 - \alpha^2}\right)^{\frac{7}{2}}}, & \text{si } 2\pi a R \geq \alpha, \end{cases} \tag{3.2}$$

donde $\mathcal{J}_{\frac{7}{2}}$ es la función Bessel del primer tipo de orden $\frac{7}{2}$. La transformada de Fourier del tren de pulsos $\text{Ш}_{G_{\Delta_\gamma}}$ está dada por

$$\mathcal{F}\left\{\text{Ш}_{G_{\Delta_\gamma}}\right\} = \frac{1}{\Delta_\gamma^3} \text{Ш}_{G_{\frac{1}{\Delta_\gamma}}}, \tag{3.3}$$

que también es un tren de pulsos.

Ya que $\mathcal{F}\left\{\text{Ш}_{G_{\Delta_\gamma}}\right\}$ es también un tren de pulsos, para que $\hat{b} \times \mathcal{F}\left\{\text{Ш}_{G_{\Delta_\gamma}}\right\}$ aproxi-

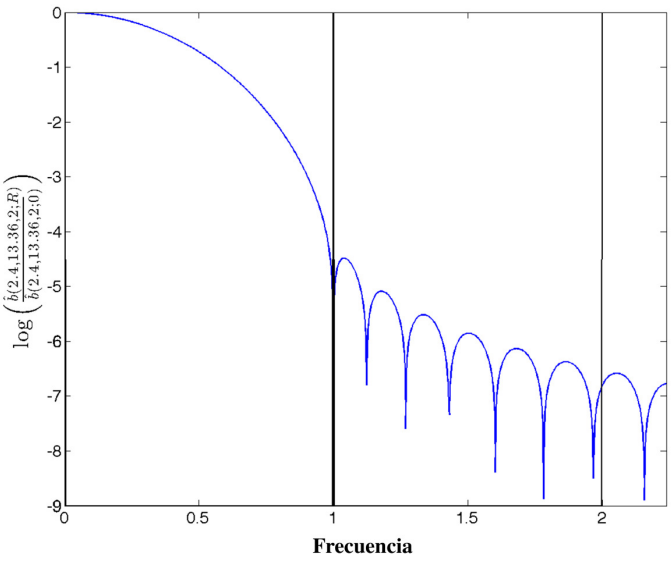


Figura 3.1: La transformada de Fourier de un *blob*. Se graficó $\log\left(\frac{\hat{b}(2,13.36,2.40;R)}{\hat{b}(2,13.36,2.40;0)}\right)$ como función de la frecuencia espacial R.

me mejor la transformada de Fourier de una función de valor constante (es decir, un impulso en el origen) es útil seleccionar b de tal manera que $\hat{b}$ tenga valor de cero en las posiciones de los pulsos en $\mathcal{F}\left\{\text{Ш}_{G_{\Delta_\gamma}}\right\}$ que tiene la menor distancia positiva al origen (es decir, $\frac{1}{\Delta_\gamma}$) porque los valores de $\hat{b}$ en frecuencias más altas decaen más rápido; véase la Figura 3.1. La función $\hat{b}$ es cero sólo cuando el término $\mathcal{J}_{\frac{7}{2}}\left(\sqrt{(2\pi aR)^2-\alpha^2}\right)$ de (3.2) es igual a cero y el menor valor positivo x para el cuál $\mathcal{J}_{\frac{7}{2}}(x)=0$ es $x=6.987932$. De esta discusión y de la fórmula analítica para $\hat{b}$ se deduce que para el primer cero de la función Bessel $\mathcal{J}_{\frac{7}{2}}$ de la Figura 3.1 nosotros tenemos la siguiente relación

$$\alpha=\sqrt{4\pi^2\left(\frac{a}{\Delta_\gamma}\right)^2-6.987932^2}. \tag{3.4}$$

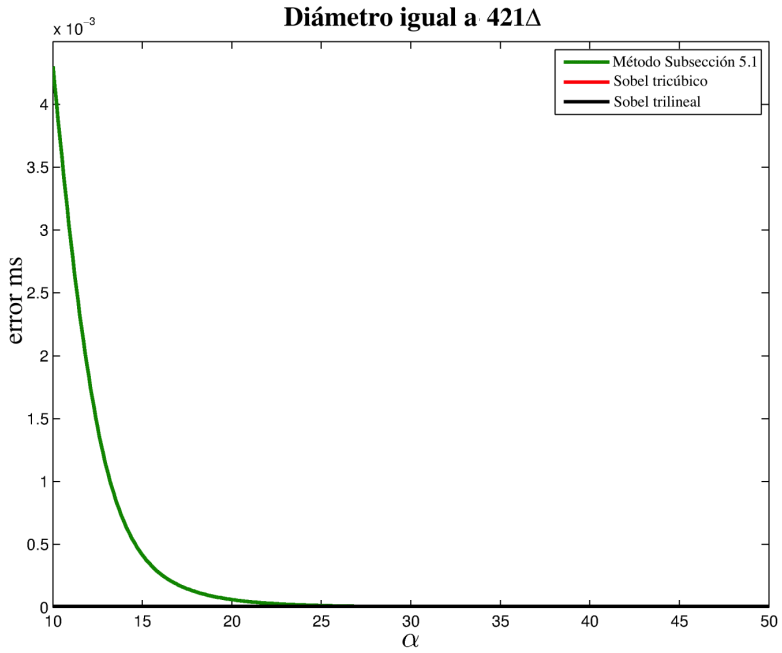
Sin embargo, este criterio no es suficiente para la determinación única de los *blobs* a ser usados porque hay un número infinito de posibles valores del radio $\frac{a}{\Delta_\gamma}$ y α que satisfacen (3.4). Por lo tanto, es posible (y deseable) agregar un criterio de selección secundario que restringirá los valores a aquellos que producen buenas aproximaciones. Nosotros seleccionamos tal criterio usando el error cuadrático medio (*ms*, por sus siglas en Inglés) entre las normales de una esfera analítica y las normales estimadas por la

combinación lineal de blobs (2.3) usada para aproximar una bola con el mismo radio de la esfera analítica. Los detalles son los siguientes, una bola continua centrada en el origen es discretizada en los puntos de G_{Δ_γ} de tal manera que a los vóxeles cuyos centros caen dentro de la bola se les asigna el valor de uno y cero de otra manera. La superficie discreta S es obtenida a partir de la bola discretizada produciendo el conjunto V. El error entre una normal $\tilde{\mathbf{n}}_{\mathbf{c}}$ asignada a un vértice $\boldsymbol{c} \in V$ y una normal de la esfera analítica $\tilde{\mathbf{a}_{\mathbf{c}}}$ en el mismo vértice (es decir, el vector unitario en la dirección desde el origen hacia $\boldsymbol{c}$) es calculado usando el producto interno entre estos dos vectores. Ya que distintos valores de α producirán un conjunto diferente de normales asignadas, para un valor dado de α el error ms para el conjunto V es calculado usando el siguiente valor medio ya que sabemos que para normales visibles el producto interno entre la normal analítica y la normal aproximada varía en el rango [0, 1] acercándose a 1 conforme disminuye la diferencia entre ellas

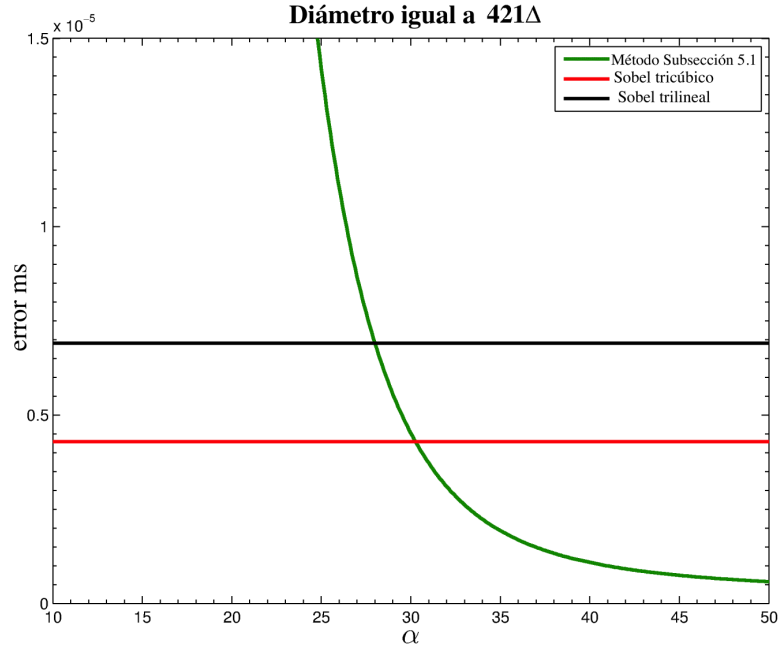
$$\text{mse}_\alpha = \frac{1}{|V|} \sum_{v \in V} \left(1 - \langle \tilde{\mathbf{a}}_{\mathbf{v}}, \tilde{\mathbf{n}}^\alpha_{\mathbf{v}} \rangle\right)^2, \tag{3.5}$$

donde $|V|$ es la cardinalidad del conjunto V. Consecuentemente, al variar los valores para α y $\frac{a}{\Delta_\gamma}$ que satisfacen (3.4) es posible producir una serie de errores ms diferentes (véase la línea verde en la Figura 3.2).

Basado en la relación de (3.4) se puede ver que, para un valor de Δ_γ fijo, entre mayor es el valor de a, mayor es el valor de α y, consecuentemente, menor es el valor menor de ms (véase la Figura 3.2). Sin embargo, valores grandes de a corresponden a *blobs* con el soporte compacto más amplio, resultando en un costo computacional mayor (es decir, para un punto dado en la superficie habrá más funciones base contribuyendo al cálculo del gradiente en dicha ubicación). Nosotros quisiéramos escoger valores para a y α para producir una "buena" aproximación a la esfera analítica (es decir, tener un error ms pequeño) por la expansión de series de *blobs* así como para ahorrar costos computacionales (por ejemplo, valores de a lo más pequeños posible). Una forma de lograr lo anterior es usar el error ms obtenido por otro método como referencia, tal como el siguiente enfoque basado en Sobel. Primero, obtenemos el gra-



(a)



(b)

Figura 3.2: Error cuadrático medio entre la normales de la superficie analítica de una esfera con radio igual a uno y las normales asignadas a una superficie discretizada obtenidas por el método presentado en la Subsección 3.1.1 usando varios valores para α. La bola fue discretizada a un objeto discreto 3D de dimensiones $421 \times 421 \times 421$ vóxeles. Para seleccionar el valor final de α (30.2523), usamos el valor de error ms obtenido por un método basado en Sobel e interpolación tricúbica.

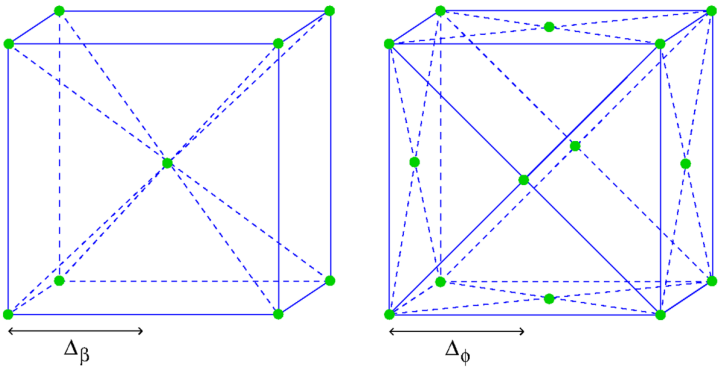
diente en el centro de cada vóxel en G_{Δ_γ} cuyo subconjunto $\mathcal{I}$ aproxima mejor la esfera que encierra la bola. El gradiente en cada posición $\boldsymbol{c} \in G_{\Delta_\gamma}$ es obtenido al calcular las tres derivadas parciales del par $\left(G_{\Delta_\gamma}, \{0,1\}\right)$ por componente; lo hacemos aplicando un filtro de Sobel. Para todos los experimentos usamos un filtro de Sobel compuesto de tres *kernels* $3 \times 3 \times 3$ vóxeles aproximando las derivadas parciales de $\left(G_{\Delta_\gamma}, \{0,1\}\right)$ en la posición $\boldsymbol{c}$. Después de obtener el gradiente para cada posición en G_{Δ_γ} lo usamos para calcular el gradiente en las posiciones del conjunto V. Para una posición $\boldsymbol{c} \in V$, el gradiente se obtiene al interpolar los gradientes ya asignados a los puntos vecinos de $\boldsymbol{c}$ en G_{Δ_γ}; para nuestra aproximación basada en Sobel usamos tanto interpolación trilineal como interpolación tricúbica. En esta aproximación, las normales asignadas a las posiciones en V dependen solo de los filtros Sobel y el método de interpolación. Debido a que decidimos los valores de estos filtros, el error *ms* para un conjunto V es fijo para un método de interpolación dado. Para los experimentos decidimos que los valores de α y Δ_γ fuesen igual a uno, debido a que estos valores representan la separación entre vóxeles y ese valor facilita la generalización y los cálculos, y una discretización de la bola tal que un número de puntos pertenecientes a $\mathbb{Z}^3$ dentro de la bola sea tal que hay 421 puntos a lo largo del diámetro de la esfera. Bajo estas condiciones, el error *ms* para la aproximación basada en Sobel usando interpolación trilineal es igual a 4.29269×10^{-6}. Debido a que la aproximación usando interpolación tricúbica tiene un error más pequeño esta fue la aproximación que usamos como referencia; el mismo *ms* se obtiene usando la aproximación basada en *blobs* si el valor de α es igual a 30.2523, véase la Figura 3.2. Con el fin de satisfacer (3.4), el radio a tiene que ser igual a 4.9416. Sin embargo, es posible obtener un valor de error *ms* más bajo al incrementar el valor de α lo que lleva a un radio más grande, obtenido al satisfacer (3.4) con un valor computacional mayor.

3.1.2. Normales a partir de un método de expansión de series (SMAN)

En muchos campos de la ciencia, las funciones que representan la distribución de algún parámetro físico que varía espacialmente pueden obtenerse a partir de sus proyecciones; estas mediciones están relacionadas a las interacciones locales de algún tipo de energía, viajando de la fuente al detector, con el objeto 3D bajo estudio y son consideradas como integrales de línea del parámetro que varía espacialmente. Algunos métodos usados para reconstruir funciones a partir de sus proyecciones usan como modelo la combinación lineal de (2.3), en particular la llamada "expansión de series" que incluye métodos como la Técnica de Reconstrucción Algebraica (ART, por sus siglas en Inglés), la Técnica Reconstructiva Iterativa Simultánea (SIRT), o la Técnica de Reconstrucción Algebraica Simultánea (SART) [47]. Todos estos métodos estiman el conjunto de coeficientes $\{c_i\}$, para una rejilla $\{\boldsymbol{p}_j\}$ y funciones base b dadas; algunas implementaciones de estos métodos usan el *blob* de (2.5) para las funciones base b_i de (2.3). Mientras que en el método descrito en la subsección previa suponemos que no existe información sobre la función que produce el par $\left(G_{\Delta_\gamma},\ f\right)$, en este método suponemos que sabemos que el par $\left(G_{\Delta_\gamma},\ f\right)$ es el resultado de muestrear la función continua υ, aproximada por la combinación lineal de (2.3). Para propósitos de reconstrucción de imágenes a partir de proyecciones, los puntos del conjunto $\{\boldsymbol{p}_j\}$ están acomodados en un subconjunto de la rejilla *cúbica centrada en cuerpo* (*body-centered cubic grid* o *bcc*). Esto siguiendo la sugerencia dada por Matej y Lewitt [80], quiénes demostraron que en este campo siempre que se desea utilizar una combinación lineal de *blobs* para obtener una función continúa υ, estas rejillas proveen un conjunto de ubicaciones deseables para los centros de lo *blobs*. Tales rejillas están definidas por

$$B_{\Delta_\beta} = \{\Delta_\beta k \,|\, \boldsymbol{k} \in \mathbb{Z}^3 \text{ and } k_1 \equiv k_2 \equiv k_3 \ (mod\ 2)\}, \tag{3.6}$$

donde Δ_β es un número real positivo que determina la separación entre puntos discretos; véase la Figura 3.3(a). Por lo tanto, para $\{\boldsymbol{p}_j\}$ usaremos el conjunto obtenido por la intersección de alguna región convexa finita en el espacio con un B_{Δ_β} de (3.6) para



(a) (b)

Figura 3.3: Puntos en las rejillas (a) cúbica centrada en el cuerpo, *bcc*, y (b) cúbica centrada en la cara, *fcc*, en una porción del espacio $2 \times 2 \times 2$ vóxeles (asumiendo $\Delta_\beta = \Delta_\phi$). El resto de los puntos pueden obtenerse llenando el espacio con la repetición más natural de la porción $2 \times 2 \times 2$ indicada.

obtener la función v de (2.3). Por consiguiente, para este método supondremos que contamos con una superficie S así como con los conjuntos $\{c_j\}$ y $\{\boldsymbol{p}_j\}$ acomodados sobre la rejilla de (3.6).

Debido a que los centros de los *blobs* están acomodados en un subconjunto de una rejilla *bcc*, en este enfoque es necesario obtener la transformada de Fourier del tren de pulsos acomodado en tal rejilla. Debido a que la transformada de Fourier del tren de pulsos $\text{Ш}_{B_{\Delta_\beta}}$ produce otro tren de pulsos organizado en la rejilla *cúbica centrada en la cara* (*face-centered cubic grid* o *fcc*, véase la Figura 3.3(b)), la relación resultante entre α y $\frac{a}{\Delta_\beta}$ es

$$\alpha = \sqrt{2\pi^2 \left(\frac{a}{\Delta_\beta}\right)^2 - 6.987932^2}. \tag{3.7}$$

Esta relación permite la implementación de un método para seleccionar valores para a, α y Δ_β con un costo computacional aceptable al calcular la raíz cuadrada del error cuadrático medio *rms* (sacando la raíz cuadrada del error *ms*) debido a que es fácil de calcular y es más sensible a valores atípicos entre una constante apropiada y el lado derecho de (2.3) usando varias opciones de α y $\frac{a}{\Delta_\beta}$. Sin embargo, de manera similar a lo que sucede en el método de la sección previa, hay muchos posibles valores de a, α y Δ_β. Por lo tanto, los autores de [33] también proponen un criterio adicional

para la selección de los parámetros que consiste en seleccionar aquellos valores de a, α y Δ_β que producen una función v cuya superficie es convexa cuando solo dos *blobs*, cuyos centros están en las dos posiciones vecinas más cercadas de una rejilla *bcc*, tienen coeficientes con el mismo valor y éste es diferente de cero. Basados en este procedimiento, los autores de [33] encontraron que a igual a 2.40 y α igual a 13.3633 para Δ_β igual a $\frac{1}{\sqrt{2}}$ resulta en funciones continuas v cuyas superficies también son suaves. Para encontrar dichos valores se restringe la solución usando solo dos *blobs* unidos, acomodados de forma que cumplan con los parámetros establecidos para las normales aproximadas y que su unión sea un objeto convexo con Δ_γ como el valor estándar. Consecuentemente, al obtener la información previa $\{c_j\}$ y $\{\boldsymbol{p}_j\} \subset B_{\Delta_\beta}$ (ambos producidos por algún método tal como un algoritmo de reconstrucción a partir de proyecciones) es posible generar v y asignarle normales suaves a los vértices en V al calcular el gradiente de v en dichas posiciones. La implementación del método antes mencionado calcula las normales en los vértices del límite de la superficie al iterar sobre todos los *blobs* y para cada uno de ellos verifica los vértices que influencia, dicho procedimiento toma lugar cuando la superficie limítrofe ya está disponible y por lo tanto muchos *blobs* pueden descartarse. Además, al probar si un vértice está bajo la influencia de un *blob*, cuando tal vértice está dentro del radio de un *blob*, entonces la parte correspondiente de (2.4) es calculada; por lo tanto, después de iterar a través del conjunto de *blobs*, todos los vértices tendrán una normal calculada con (2.4).

3.1.3. De GMAN a SMAN

En el caso de que uno quisiera usar las normales similares a aquellas producidas por un método de expansión de series pero sólo tiene acceso a la imagen 3D voxelizada generada por dicho método, puede usarse el procedimiento que describiremos ahora. Nosotros podemos generar los coeficientes para el volumen voxelizado reconstruido al ejecutar ART [11, 12], un algoritmo de reconstrucción iterativo cuya actualización está dada por:

$$
\begin{aligned}
c^{(0)} & & & \text{es arbitrario,} \\
c_j^{(n+1)} = & \; c_j^{(n)} + \lambda^{(n)} \frac{y_i - \sum_{k=1}^{N} a_{i,k} c_k^{(n)}}{\sum_{k=1}^{N} a_{i,k}^2} a_{i,j} & & \text{para } 1 \leq i \leq I, \\
& & n = 0, 1, \ldots, & \quad i = n \bmod I + 1,
\end{aligned}
\tag{3.8}
$$

en dónde $\lambda^{(n)}$ es un parámetro de relajamiento tal que $0.0 < \lambda^{(n)} < 2.0$. Primero inicializamos el sistema de ecuaciones para asegurar una convergencia más rápida de la solución. Usando los parámetros de *blob* seleccionados, para cada *blob* de la rejilla, se le asigna el valor del promedio ponderado de los vóxeles que están dentro del soporte del *blob* j a $c_j^{(0)}$, donde el peso del i-ésimo vóxel al j-ésimo *blob* está dado por a_{ij}. Después de esta asignación inicial, se ejecuta ART (3.8) usando los valores de los vóxeles de la imagen 3D reconstruida como las mediciones (es decir, la y_i de (3.8)) y un $\lambda^{(n)}$ apropiado, para un número fijo de iteraciones o hasta que un error máximo predefinido es alcanzado; en nuestra experiencia, un número máximo de 25 iteraciones (después de ese número de iteraciones no se obtiene una mejora significativa en el resultado ni un cambio significativo en la medición del error) con una $\lambda = 0.1$ provee buenos resultados.

Capítulo 4

Resultados Asignación de Normales

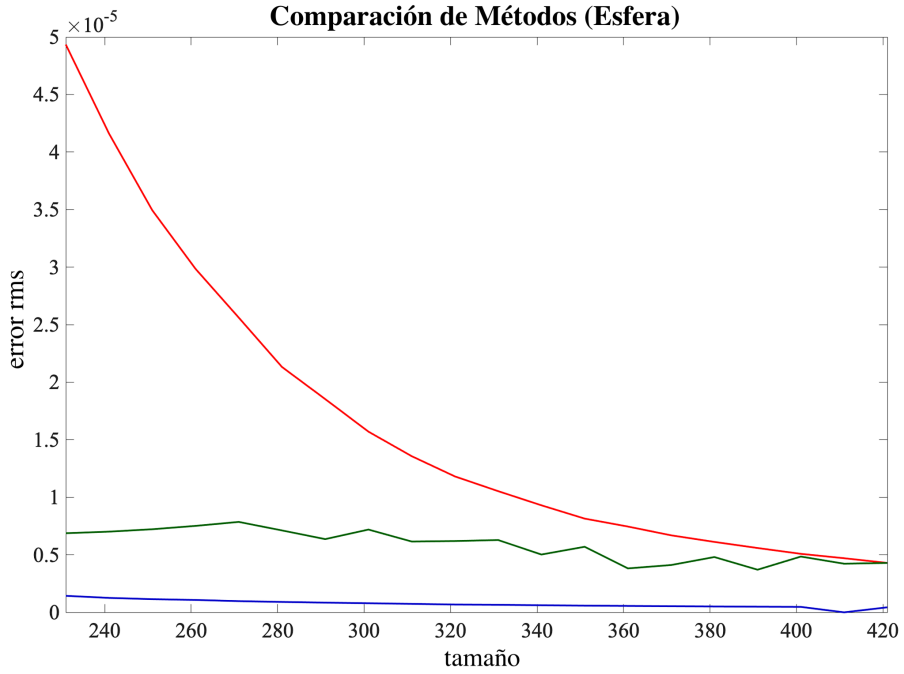
En esta sección reportaremos los experimentos que demuestran el desempeño de los dos métodos presentados en la sección previa. Para nuestros experimentos en esta sección produjimos 1,000 proyecciones 2D con integrales de línea en paralelo con dimensiones 560×560 unidades, en distintas orientaciones distribuidas a lo largo de una esfera unitaria, a partir de objetos 3D diferentes; estas proyecciones fueron generadas por un algoritmo optimizado de *raytracing* implementado en el paquete de *software* Xmipp [109], la principal razón para la elección de dicho *software* es que, a diferencia de la mayoría de programas similares, permite tener acceso a los valores de los coeficientes de los *blobs*. Los diferentes conjuntos de proyecciones fueron usados para producir su función v correspondiente con la implementación de ART usando *blobs* como funciones base incluidas en el mismo paquete Xmipp. Los parámetros de *blob* usados para todas nuestras reconstrucciones usando ART fueron $a = 2.40$, $\alpha = 13.3633$ y $\Delta_\beta = \frac{1}{\sqrt{2}}$ como se sugirió en el trabajo presentado en [33] (ART es un algoritmo bien conocido en el campo de la reconstrucción a partir de proyecciones que produce una aproximación del objeto 3D a partir de sus proyecciones al devolver cada valor en cada una de sus proyecciones 2D al espacio 3D siguiendo la misma trayectoria y orientación de donde se originó; estos valores se corrigen de tal manera que las proyecciones del objeto 3D corresponden a las proyecciones medidas. Para los detalles técnicos referimos al lector a [47].) Al muestrear una función continua v nosotros obtenemos su par correspondiente $\left(G_{\Delta_\gamma},\, f\right)$ con una distancia de muestreo deseada.

A continuación, mostramos que la metodología mostrada en este trabajo produce imágenes superiores, o al menos igual de buenas, que los renderizados producidos por la asignación de normales con un operador de Sobel sobre un objeto 3D discretizado (una forma estándar de obtener vectores gradientes), pero requiere menos tiempo computacional.

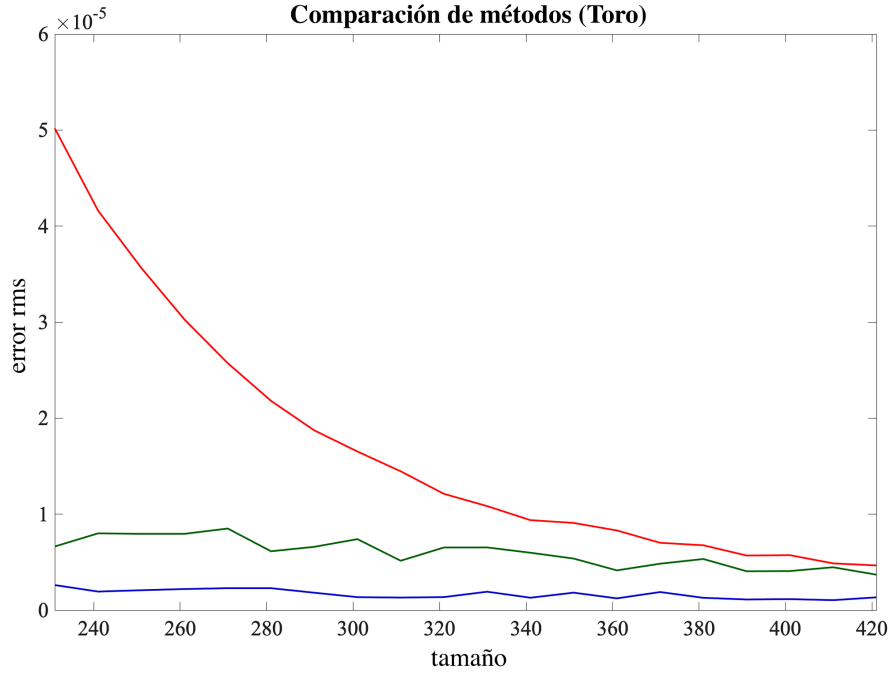
4.1. Comparación de Normales con un Estándar de Oro

En esta subsección presentamos algunos experimentos diseñados para calificar que tan bien se desempeñan nuestros métodos. Para nuestra cuantificación comparamos las normales de formas geométricas analíticas, una esfera y un toro, a aquellos producidos por los métodos de la Subsección 3.1.1 y SMAN de la Subsección 3.1.2 presentados en la Sección 3.1 y el método basado en Sobel usando interpolación tricúbica como estándar de oro (por ejemplo, el gradiente en el centro de cada vóxel en el par $\big(G_{\Delta_\gamma}, \{0,1\}\big)$ es calculado aplicando un filtro Sobel $3 \times 3 \times 3$ vóxeles y el gradiente en la posición de un vértice en V es obtenido haciendo una interpolación tricúbica de los gradientes ubicados en los centros de los vóxeles vecinos). Nosotros seleccionamos una esfera y el toro porque el primero es un ejemplo común de una función convexa y el último representa una función no positiva cuyas normales pueden obtenerse analíticamente sin mucho esfuerzo.

Para llevar a cabo tal comparación seguimos un proceso similar al descrito en la Subsección 3.1.1. Las descripciones continuas de la esfera y el toro están centradas en el origen con radios fijos y discretizados por muestreo usando G_{Δ_γ} de tal manera que a los vóxeles cuyos centros caen dentro de la bola y el tubo doblado se les asigna valor de uno y cero de otra manera. Su superficie discreta S es entonces combinada con la representación discreta para producir los conjuntos V correspondientes. Para cada vértice $\boldsymbol{c}$ en el conjunto de vértices V una normal $\overrightarrow{\boldsymbol{n}}$ es asignada a dichas posiciones por el método basado en Sobel y por los dos métodos de la Sección 3.1. Entonces,



(a)



(b)

Figura 4.1: El error de raíz cuadrática media entre las normales de las superficies analíticas de (a) una esfera y (b) un toro y las normales asignadas a la superficie de varias de sus discretizaciones (el diámetro de la esfera y la distancia desde el centro de un tubo al centro del toro fue muestreada de 231 a 421 puntos). Las gráficas representan el error *rms* para el método basado en Sobel presentado en la Subsección 3.1.1 (rojo), el método GMAN (verde) y el método SMAN (azul).

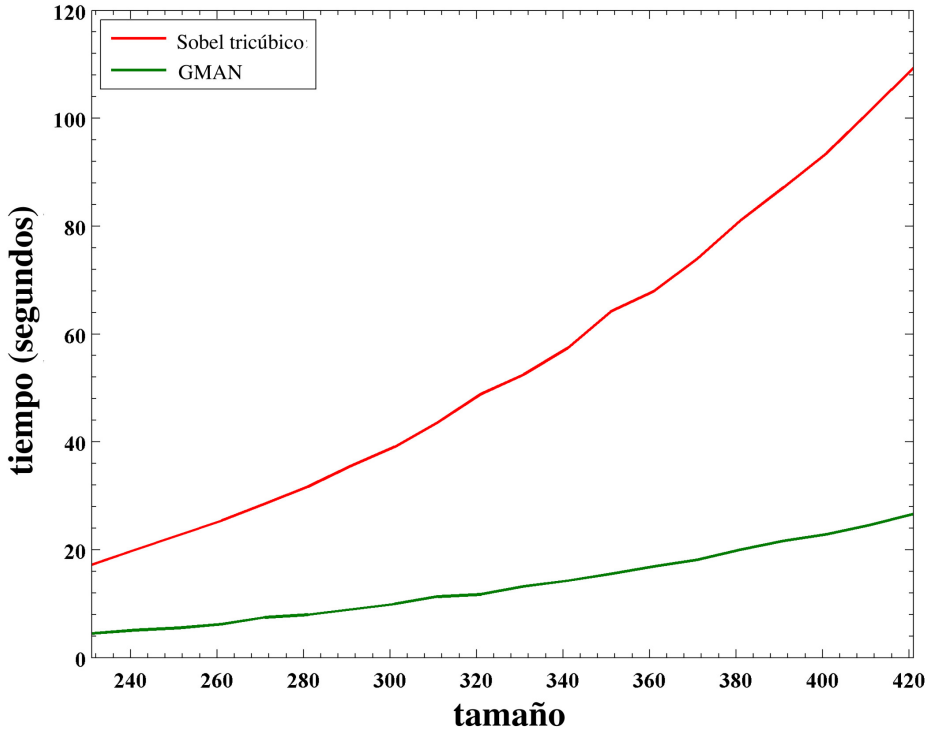


Figura 4.2: Tiempos para calcular las normales para todos los vértices en V con el método basado en Sobel con interpolación tricúbica (rojo) y el método GMAN (verde); no mostramos los tiempos para el método SMAN porque el procedimiento es similar al del GMAN pero con menos *blobs* por vértice.

el error entre una normal $\overrightarrow{\boldsymbol{n}}$ asignada al vértice $\boldsymbol{c} \in V$ y la normal analítica $\overrightarrow{\boldsymbol{d}}_{\boldsymbol{c}}$ en el mismo vértice es calculada usando el producto interno entre estos dos vectores. El error *ms* es calculado para estos tres métodos de asignación de normales usando la raíz cuadrada de (3.5). Al cambiar el número de puntos de discretización para el radio de la esfera y los radios del toro es posible obtener diferentes errores *ms* para cada uno de los métodos de asignación de normales. En estos experimentos se varió la discretización de tal manera que el número de puntos a lo largo del diámetro de la bola varia de 231 a 421 en incrementos de 5; de manera similar, usamos la misma estrategia de discretización para muestrear la distancia a partir del centro del tubo al centro del toro. Mostramos los resultados de estos experimentos en la Figura 4.1. Estos resultados muestran que tanto el método GMAN como el SMAN producen buenas aproximaciones a las normales en las superficies analíticas, también puede verse

que a medida que el tamaño del vóxel cúbico decrece, el método basado en Sobel y los nuestros mejoran la aproximación pero los nuestros tienen mejor desempeño. Como era de esperarse, el método SMAN de la Subsección 3.1.2 supera a los otros dos porque la función continua subyacente υ y su conjunto de coeficientes $\{c_i\}$ fueron aproximados de antemano, por lo tanto, aseguran que las normales son muy cercanas a las normales de las superficies analíticas para cada nivel de discretización. Sin embargo, es importante hacer notar que, para el caso del toro, el método SMAN muestra un peor desempeño que en el caso de la esfera. Después de una investigación cuidadosa, descubrimos que ese comportamiento es resultado de un conjunto de coeficientes que no aproximan adecuadamente el toro, véase la Figura 4.5(d), donde se observan artefactos centrales; tal conjunto de coeficientes fue obtenido por medio de la implementación del método ART mencionado previamente [109], aunque investigamos la razón de raíz de este problema, su discusión y resolución están fuera del alcance de este trabajo. Finalmente, mostramos en la Figura 4.2 que nuestros métodos son más rápidos que el método basado en Sobel en la tarea de asignar las normales a la superficie discreta cuando se usan implementaciones que solo usan *software*, es decir, sin usar paralelismo o aceleración por *hardware*.

4.1.1. Ejemplos Visuales

En esta subsección ilustraremos la salida de los métodos presentados en la Sección 3.1 usando dos conjuntos de datos, ambos obtenidos de [102] y almacenados en formato PVM, representado un angiograma y un árbol bonsai; estos datos contienen los valores de su función f correspondiente muestreada en una rejilla cúbica de $512\times512\times154$ vóxeles. A partir de estos conjuntos de datos obtenemos sus respectivos conjuntos de coeficientes $\{c_j\}$ para producir la función de aproximación υ; posteriormente estas aproximaciones fueron discretizadas para obtener el par $\left(G_{\Delta_\gamma},\, f\right)$ usando una rejilla de $256 \times 256 \times 256$ vóxeles. Para obtener el objeto 3D discreto a partir de $\left(G_{\Delta_\gamma},\, f\right)$, usamos umbralización con valores iguales a 0.5 y 0.6 correspondientes al bonsai y al aneurisma, respectivamente; estos valores fueron seleccionados usando ins-

pección visual. El par resultante $\left(G_{\Delta_\gamma}, \{0,1\}\right)$ es usado como entrada del algoritmo de seguimiento de superficies propuesto por Artzy et al [3]. Para producir un renderizado de las superficies usamos sombreado de Phong con las siguientes variaciones del modelo de iluminación de Phong que no incluye el componente de luz ambiental [30]

$$I = I_d \kappa_d \left\langle \overrightarrow{\boldsymbol{n}}, \overrightarrow{\boldsymbol{\ell}} \right\rangle + I_s \kappa_s \left\langle \overrightarrow{\mathbf{r}}, \overrightarrow{\mathbf{e}} \right\rangle^\gamma . \tag{4.1}$$

En este modelo, las constantes reales no negativas κ_d y κ_s representan las contribuciones de la luz difusa y la luz especular, respectivamente. Para que un punto $\boldsymbol{p}$ en la superficie sea proyectado en la pantalla, el modelo usa los siguientes vectores: el vector normal a la superficie $\overrightarrow{\boldsymbol{n}}$ en $\boldsymbol{p}$, el vector normalizado $\overrightarrow{\boldsymbol{\ell}}$ que va de $\boldsymbol{p}$ a la fuente de luz, el vector normalizado $\overrightarrow{\mathbf{e}}$ que va de $\boldsymbol{p}$ a la cámara y el vector normalizado $\overrightarrow{\mathbf{r}}$ obtenido por medio de $2\left\langle \overrightarrow{\boldsymbol{n}}, \overrightarrow{\boldsymbol{\ell}} \right\rangle \overrightarrow{\boldsymbol{n}} - \overrightarrow{\boldsymbol{\ell}}$. Presentamos los renderizados usando la ley del coseno de Lambert (que indica que la intensidad de la luz observada desde una reflexión difusa ideal depende del coseno del ángulo entre la dirección de la luz incidente y la normal de la superficie) y el modelo de Phong para reflexiones especulares. Seleccionamos los siguientes valores para el segundo tipo de renderizados: $I_d = (0.92,\ 0.92,\ 0.92)$, $I_s = (0.92,\ 0.92,\ 0.92)$, $\kappa_d = 0.8$, $\kappa_s = 0.5$ y $\gamma = 20$.

En la Figura 4.3 mostramos los renderizados del bonsai, en la columna izquierda y del aneurisma, en la columna derecha, usando los métodos de la Sección 3.1. En la imagen superior presentamos, como comparación, renderizados de las superficies rastreadas del bonsai y el aneurisma cuyas normales de vértices fueron calculadas promediando las normales perpendiculares a las caras vecinas. Los renderizados producidos por las normales del método GMAN se presentan en la imagen del medio. Los renderizados producidos por las normales del método SMAN se presentan en la imagen inferior. Podemos observar que los renderizados usando las normales calculadas por nuestros métodos disminuyen bastante la apariencia cuboide (voxelizada) que se da naturalmente de las mallas producidas por el algoritmo de rastreo de superficies. Para ambos casos, el costo de asignar normales no es mucho más alto que el costo de promediar las normales perpendiculares de cara. Desafortunadamente, la

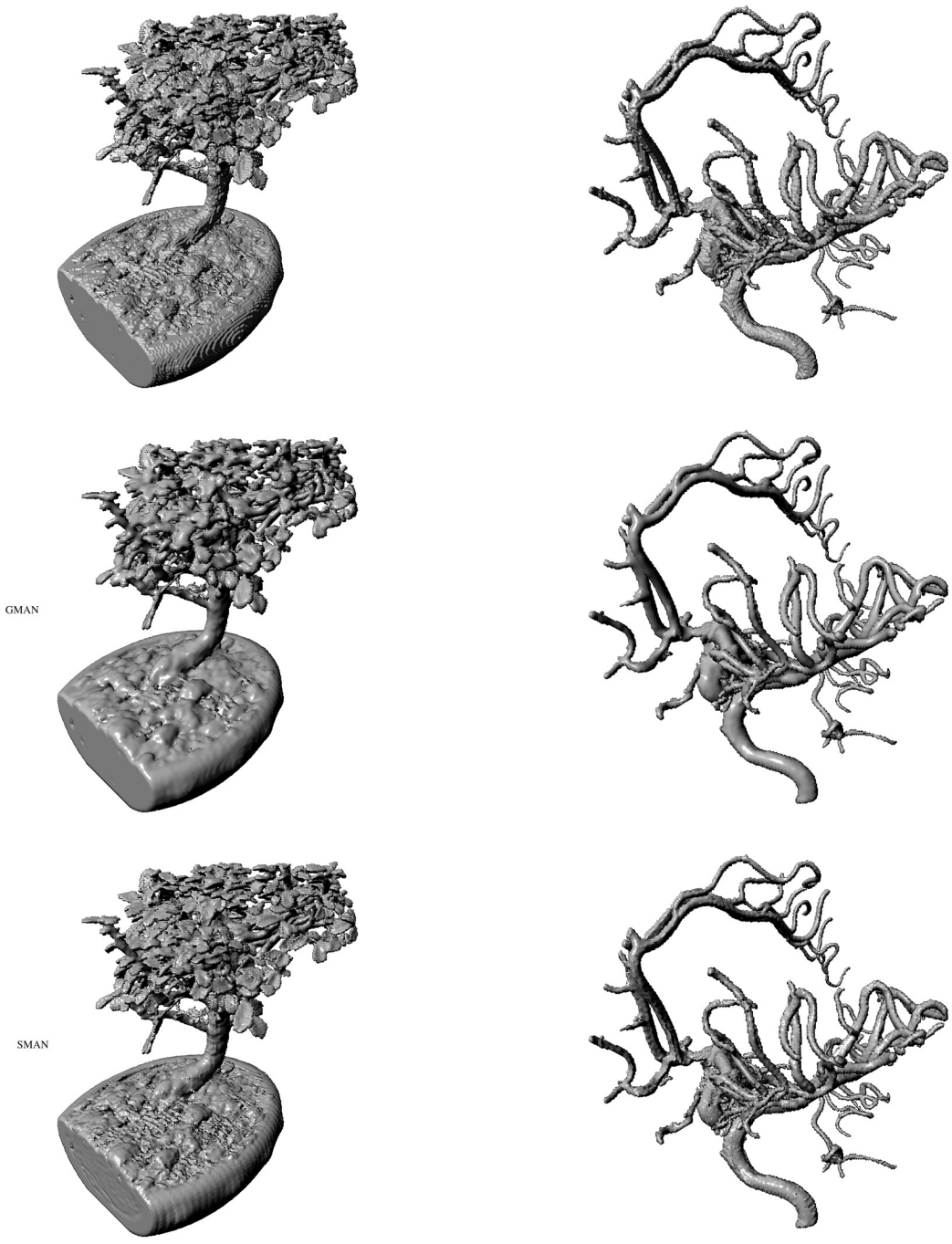


Figura 4.3: Renderizados de mallas cuadrilaterales obtenidas de la representación binaria de la reconstrucción v representando (*izquierda*) un árbol bonsai y (*derecha*) un aneurisma, ambos obtenidos después de ejecutar ART por 25 iteraciones con $a = 2.4000$, $\alpha = 13.3633$ para $\Delta_\beta = \frac{1}{\sqrt{2}}$ que satisface la relación entre α y $\frac{a}{\Delta_\beta}$ para la rejilla *bcc*. Estos renderizados fueron producidos (*arriba*) usando normales de vértices resultantes de promediar las normales de cara perpendiculares vecinas que, (*en medio*) asignando normales a los vértices de superficie con el método GMAN y (*abajo*) al asignar normales a la superficie usando el método SMAN. Los renderizados fueron creados con $I_d = (0.5, 0.5, 0.5)$, $\kappa_d = 0.8$, $\kappa_s = 0.5$ y $\gamma = 20$.

apariencia voxelizada, aunque reducida, es evidente a lo largo de los bordes de los objetos; las normales de los vértices a lo largo de esas regiones son suaves pero el método de sombreado en esas regiones no trabaja a nuestro favor. También observamos que los valores para los parámetros del blob usados en el método GMAN producen renderizados que son más suaves que aquellos producidos por el método SMAN. Es importante hacer notar que este efecto de suavizado puede disminuirse seleccionando un valor de error ms mayor en la metodología GMAN.

La Figura 4.4 muestra un acercamiento de una sección de los renderizados mostrados en la Figura 4.3, usando la ley del coseno de Lambert y el modelo de Phong. Podemos ver que los renderizados producidos al promediar las normales perpendiculares de caras vecinas son altamente sensibles a los efectos de digitalización. Los renderizados producidos por el método GMAN, sin la información acerca de cómo la función continua υ fue creada, mostrados en las imágenes del medio en la Figura 4.3, suaviza los artefactos de digitalización mostrados en la primera hilera con el costo de perder datos significativos, véanse las flechas rojas en la imagen del medio de la Figura 4.4. De manera interesante, en algunos casos el efecto de escalera es más pronunciado usando el método GMAN, véanse las flechas amarillas en la imagen del medio de la Figura 4.4. En la fila inferior podemos ver los renderizados producidos usando el método SMAN, usando el conjunto $\{c_j\}$ para crear la función continua υ, donde podemos ver detalles más pequeños con artefactos de digitalización más suaves.

4.1.2. Comparación con *Digital Integral Invariant Curvature Estimator*

Los autores de [70] publicaron sus métodos originalmente en [18] y proveyeron un *software* con su implementación en [17], nosotros usamos este *software* para comparar visualmente los resultados producidos por esta implementación (DGTal) con los producidos por los métodos GMAN y SMAN. Para esta tarea usamos un toro centrado en el origen discretizado con una rejilla cúbica de dimensiones $401 \times 401 \times 401$ vóxeles. De la Subsección 3.1.1 configuramos los valores de a como 4.9416 y de α como

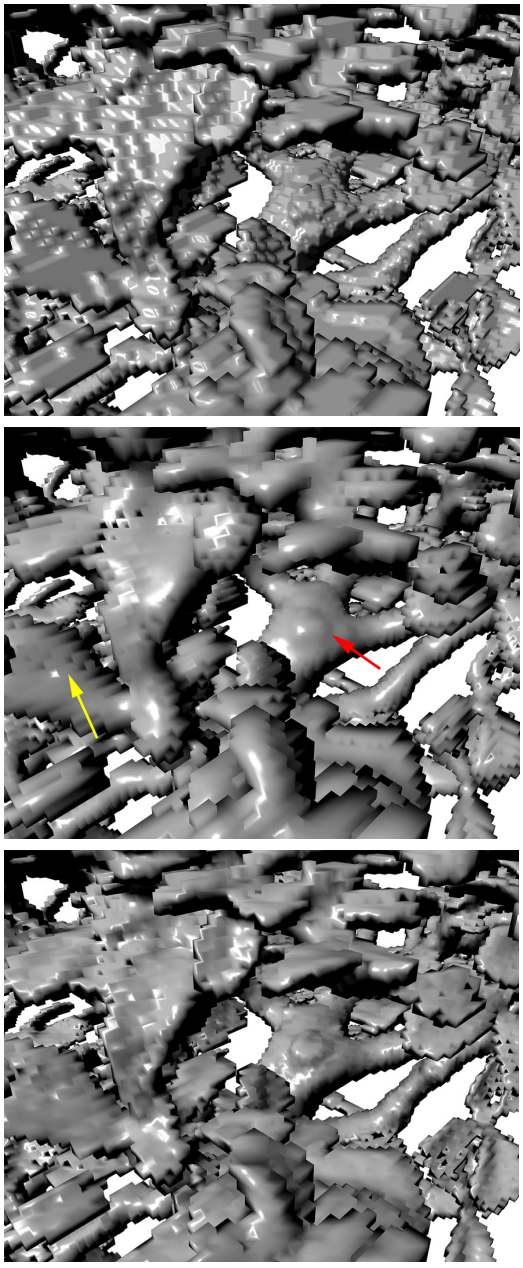


Figura 4.4: Acercamiento de los renderizados mostrados en la Figura 4.3(*izquierda*). Estos renderizados fueron producidos usando (*arriba*) las normales de vértices resultantes de promediar las normales vecinas perpendiculares de cara, (*en medio*) al asignar las normales a los vértices de la superficie usando el método de la Subsección 3.1.1 donde las flechas señalan la perdida de detalles en el modelo por el suavizado, y (*abajo*) al asignar normales a los vértices de la superficie usando el método de la Sección 3.1.2. Los renderizados fueron creados con $I_d = (0.5, 0.5, 0.5)$, $I_s = (0.5, 0.5, 0.5)$, $\kappa_d = 0.8$, $\kappa_s = 0.5$ y $\gamma = 20$.

30.2523 para el método GMAN, sin embargo en esa sección en realidad introdujimos una metodología para seleccionar los valores de los parámetros para obtener distintos grados de renderizados suaves al seleccionar apropiadamente los valores de α y el radio a, así como la consideración de un costo computacional bajo. La discretización obtenida directamente de la descripción analítica del toro fue la entrada del método GMAN. Para el método SMAN usamos los parámetros a y α obtenidos en la Subsección 3.1.2 y los valores de coeficientes producidos por ART al aproximar el toro. La discretización de la descripción analítica del toro fue usada como entrada del método propuesto en [70] con un radio R para el *kernel* de convolución igual a 5 vóxeles y un paso de rejilla $h = 1$. Vale la pena mencionar que la selección de un valor apropiado para el radio del *kernel* R puede ser difícil debido a que la selección de valores muy pequeños (por ejemplo, $R = 1$) podrían producir objetos renderizados con apariencia voxelizada mientras que valores muy grandes podrían suavizar detalles pequeños de los objetos. Además, la esfera de integración podría cubrir múltiples regiones diferentes de la superficie, lo que provocaría malas estimaciones. El tamaño del paso de rejilla también influye en el rendimiento del método, por ejemplo, valores pequeños producen mejores resultados con el costo de un mayor tiempo de cómputo.

En la Figura 4.5 presentamos los renderizados de la superficie S con sus normales asignadas analíticamente, como referencia, y por los distintos métodos GMAN, SMAN y DGTal. De esta figura, los renderizados más atractivos visualmente son los producidos por el método que asigna normales analíticamente y por los métodos SMAN y DGTal. Es evidente de la Figura 4.5(d) que el renderizado producido por SMAN, que tomo 188,064 ms, tiene algunos artefactos a lo largo de la sección central de toro. Tenemos evidencia de que estas bandas son artefactos introducidos por la implementación del método que produce la aproximación al toro a partir de sus proyecciones (es decir, ART en Xmipp); dada una buena aproximación, los renderizados serían comparables y conjeturamos que los producidos por SMAN serían superiores ya que los renderizados no muestran la discretización subyacente en algunas regiones de la imagen (por ejemplo, el lado derecho inferior en DGTal). Por otro lado, el método GMAN produjo renderizados de menor calidad que los renderizados más visualmente

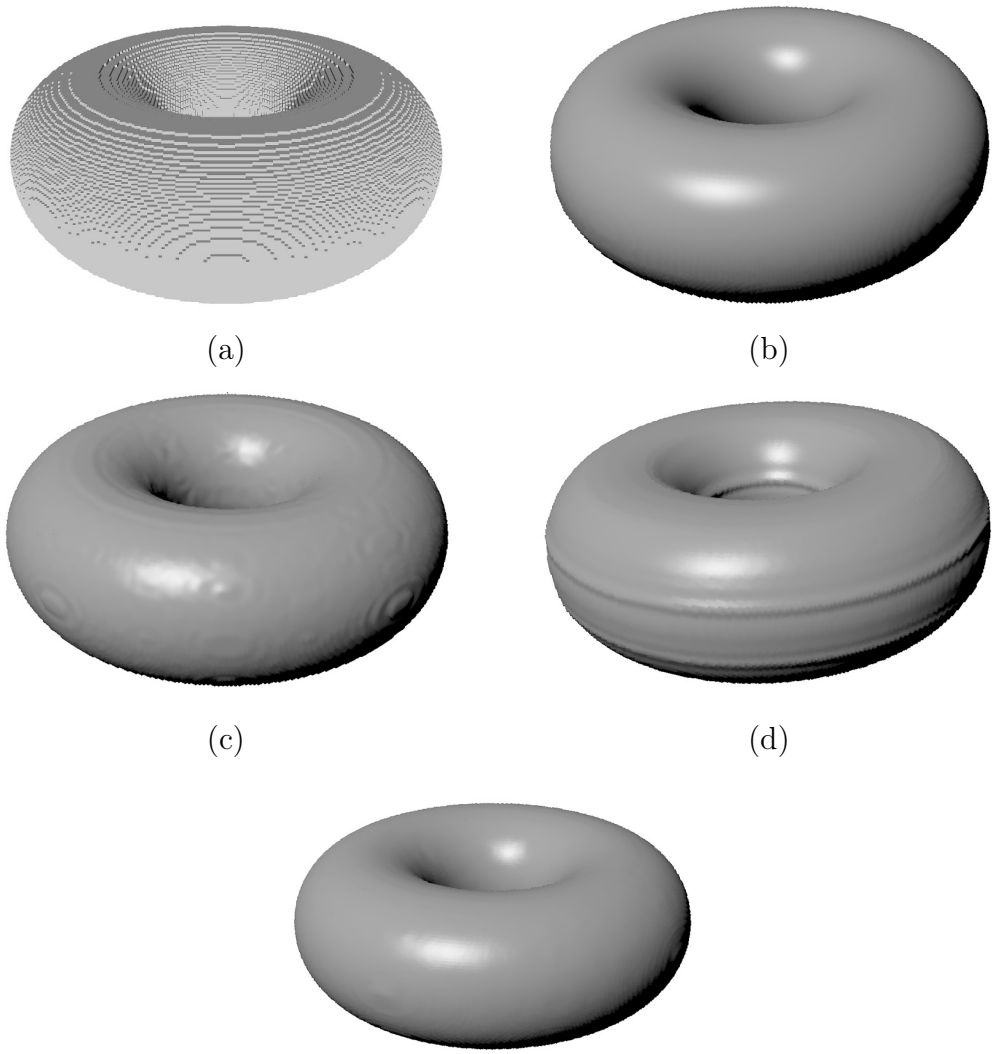
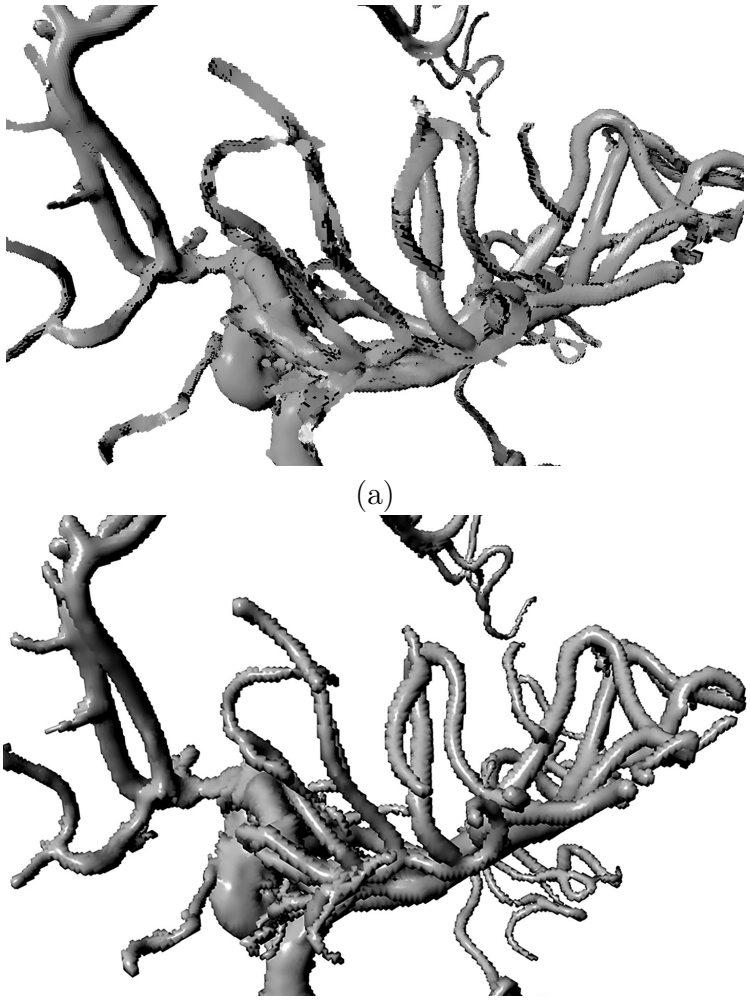


Figura 4.5: Una malla cuadrilateral y renderizado con diferentes métodos: (a) sin normales, (b) con normales calculadas analíticamente, (c) normales obtenidas con el método GMAN, (d) con normales obtenidas con el método SMAN y (e) con el método implementado en [70]. Las renderizaciones fueron creadas con $I_d = (0.5, 0.5, 0.5)$, $I_s = (0.5, 0.5, 0.5)$, $\kappa_d = 0.8$, $\kappa_s = 0.5$ y $\gamma = 20$.

agradables producidos por DGTal, pero tomando 132,773 ms asigno normales casi el doble de rápido que DGTal, el cual tomó un total de 341,453 ms para procesar el conjunto discreto $\left(G_{\Delta_\gamma}, \{0,1\}\right)$ y alrededor de 214,884 ms para estimar las normales (tiempos reportados por la misma biblioteca DGTal).

Como una comparación final, renderizamos en la Figura 4.6 el resultado producido



(a)

(b)

Figura 4.6: (a) Renderizado del resultado producido por DGtal usando y comparado con (b) el conjunto de datos del aneurisma utilizado para producir los renderizados en la Figura 4.3 (*derecha*); los parámetros de renderizado fueron los mismos que los usados en la Figura 4.3.

por DGtal usando el conjunto de datos que representa un aneurisma usado para crear los renderizados en la Figura 4.3 (*derecha*) como entrada y los mismos parámetros usados para producir el renderizado del toro en la Figura 4.5. Opuesto al renderizado del toro, este ejemplo muestra algunos problemas en la asignación de las normales. Nosotros creemos que se debe a la presencia de pequeñas características o imperfecciones en los datos del volumen. Sin embargo, podemos ver en la Figura 4.3 que el método presentado en este trabajo tiene éxito en producir renderizados suaves cuando DGtal tiene algunos problemas, aunque se considera un método robusto a datos con

ruido.

Capítulo 5

Efectos de onda: Antecedentes

Los modelos de iluminación son responsables en buena parte de proporcionar realismo a las escenas resultantes del proceso de graficación por computadora. En este capítulo nos enfocaremos en los modelos de iluminación global debido a que son capaces de reproducir imágenes foto-realistas, sin embargo, la mayoría de estos métodos no modelan los efectos debidos al comportamiento de onda de la luz por simplicidad

En este capítulo se ahondará en las características y los principales modelos de iluminación global, los trabajos relacionados con la modelación de los efectos de interferencia, difracción y polarización en graficación por computadora, así como los conceptos de física necesarios.

5.1. Modelos de iluminación global

Como ya se había mencionado anteriormente, un modelo de iluminación global no sólo calcula la luz emitida, reflejada y transmitida directamente, sino que también toma en cuenta la luz indirectamente reflejada y transmitida. Esta luz indirecta es tratada como constante por los métodos de iluminación local; sin embargo, esta aproximación no permite simular transferencia de color, cáusticas y otros efectos que se dan en escenas reales [29, 97].

La idea básica de los modelos de iluminación global es seguir la luz en su trayectoria desde su fuente hasta el observador pasando por todas las reflexiones y refracciones

que tiene con las superficies de los objetos presentes en una escena. La mayoría de los enfoques usados en iluminación global tratan de resolver la siguiente ecuación de renderizado propuesta por [62],

$$I(\boldsymbol{x},\ \boldsymbol{y}) = g(\boldsymbol{x},\ \boldsymbol{y}) \left[\epsilon(\boldsymbol{x},\ \boldsymbol{y}) + \int_S \rho(\boldsymbol{x},\ \boldsymbol{y},\ \boldsymbol{w}) I(\boldsymbol{y},\ \boldsymbol{w}) d\boldsymbol{w} \right], \tag{5.1}$$

donde $\boldsymbol{x}$, $\boldsymbol{y}$ y $\boldsymbol{w}$ son puntos de la escena, I es una función que relaciona la intensidad de luz pasando de un punto $\boldsymbol{y}$ a otro $\boldsymbol{x}$, g es una función que está relacionada con la geometría y la visibilidad, $\epsilon(\boldsymbol{x},\ \boldsymbol{y})$ es la función que define la intensidad de la luz emitida de $\boldsymbol{y}$ a $\boldsymbol{x}$, el término $\rho(\boldsymbol{x},\ \boldsymbol{y},\ \boldsymbol{w})$ está relacionado con la intensidad de la luz reflejada desde el punto $\boldsymbol{w}$ al punto $\boldsymbol{y}$ y de ahí al punto $\boldsymbol{x}$ (que normalmente se aproxima usando BRDFs y que incluye tanto la reflexión especular como la difusa) y S representa a todos los puntos sobre las superficies de la escena.

La ecuación (5.1) establece una equivalencia entre la intensidad de luz que es reflejada del punto $\boldsymbol{y}$ al punto $\boldsymbol{x}$ con la luz emitida por el mismo punto $\boldsymbol{y}$ como resultado de la contribución de la luz reflejada desde todos los demás puntos de todas las superficies hacia el punto $\boldsymbol{y}$, ese cálculo típicamente se lleva a cabo de forma recursiva, puesto que esos puntos también reciben luz desde los demás puntos superficiales.

Este modelo puede considerarse una aproximación de óptica geométrica a las ecuaciones de Maxwell y por lo tanto no modela los fenómenos ópticos debidos al comportamiento de onda [62].

Una de las definiciones más usuales de la función $g(\boldsymbol{x},\ \boldsymbol{y})$ establece que la función tiene un valor de 0 cuando hay oclusión entre los puntos $\boldsymbol{x}$ e $\boldsymbol{y}$, y de $1/r^2$ cuando no hay obstrucción entre ellos, donde r es la distancia Euclidiana entre ellos. La evaluación inicial de $g(\boldsymbol{x},\ \boldsymbol{y})$ se hace determinando la superficie visible en una esfera alrededor del punto $\boldsymbol{x}$ para un número de superficies incluidas en S y la integral contenida en (5.1) se hace sobre todos los puntos en dichas superficies.

El éxito de cualquier enfoque para resolver la ecuación de renderizado depende en gran parte de como se manejan los términos restantes y la recursión, de que

combinaciones de reflectividad difusa y especular se tomen en cuenta y de que tan bien se modelen las relaciones de visibilidad entre las superficies [29].

Dentro de los enfoques que pueden seguirse para aproximar una solución a (5.1) se encuentran las Series de Neumann, la aproximación por trazado de rayos (*ray tracing*), la aproximación por *radiosity* o la aproximación por *path tracing* y derivados, por ejemplo, *photon mapping*. Es importante señalar que, aunque la iluminación global tiene como resultado imágenes más realistas, el costo computacional de generar una imagen la hace, aún en la actualidad con *hardware* computacionalmente potente, una técnica prohibitiva para aplicaciones en tiempo real.

5.1.1. *Ray tracing*

Los métodos de *ray tracing* se basan en la suposición de que una fuente de luz puede emitir rayos en líneas rectas que representan la trayectoria de sus fotones y que viajan hasta que una superficie interrumpe su avance. En este punto pueden suceder combinaciones de distintos efectos ópticos que dependen del material de la superficie: la luz puede ser absorbida, reflejada, refractada o emitida. En general, los métodos de *ray tracing* tratan de emular las rutas que los rayos de luz seguirían al ser emitidos desde las fuentes de luz y reflejarse sobre las superficies de los objetos en la escena hasta el ojo. Aunque ésta es la forma más natural de explicar el proceso del trazado de rayos, esto genera rayos que nunca llegan al observador. Por lo tanto, una forma más eficiente de llevar a cabo el proceso de trazado de rayos es generar los rayos desde el observador hacia las superficies y las fuentes de luz.

El primer algoritmo de *ray tracing* fue presentado en [2] y es conocido como *ray casting* por ser una versión simplificada del proceso. En esta encarnación, se envía un rayo desde cada píxel en la pantalla y este rayo es usado para determinar si hay una intersección con una superficie. En caso de que exista más de una intersección, sólo se utiliza la más cercana a la pantalla. La intersección obtenida se utiliza para calcular el sombreado del píxel desde donde se origina el rayo.

Posteriormente, el autor de [123] implementó un método recursivo donde el proceso

no termina al encontrar la primera superficie que obstruye la trayectoria del rayo, esto permite generar efectos que con otros métodos son más difíciles o no son posibles de generar, tales como sombras, reflexiones y refracciones. En *ray tracing*, la trayectoria que los rayos puedan tomar se sigue en sus distintas colisiones con las superficies hasta que se cumple algún criterio de paro. Se puede considerar al *ray tracing* como una generalización del *ray casting* que permite generar sombras y efectos de transparencia. Para seguir un rayo se utilizan las leyes de Snell y de reflexión y en cada intersección se pueden generar hasta tres tipos de rayos secundarios (de reflexión, de refracción o de sombra); en cada intersección se acumulan los efectos que modifican la intensidad de la luz en la intersección del rayo con todas las superficies que encuentra en su camino. Cada rayo secundario de reflexión o refracción puede generar recursivamente otro rayo secundario, por lo tanto se puede hablar de un árbol de rayos en dónde cada intersección rayo-superficie se guarda en un nodo y genera una rama nueva, una rama puede terminarse si un rayo ya no encuentra otro objeto en su camino, o bien, si se llega a una profundidad máxima. Al terminar de construir el árbol, éste es evaluado comenzando por los nodos hoja avanzando hacia el nodo raíz, cada valor de intensidad en un nodo se calcula en función de los valores de intensidad de los hijos.

5.1.2. *Radiosity*

El método de radiosidad (*radiosity*) es una aproximación de elemento finito al problema de iluminación global. Se basa en descomponer la escena en un conjunto de pequeños elementos o parches (P) y calcular cuanta energía se transfiere entre dichos elementos. La fracción de energía que se transmite entre cada par de parches $p_i,\ p_j \in P$ es descrita por una ecuación llamada *factor de forma*, cuya versión más sencilla es

$$F(p_i,\ p_j) = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V(p_i,\ p_j), \tag{5.2}$$

en la cual existen un término de visibilidad $V(p_i,\ p_j)$, que determina que tan visibles son los elementos p_i y p_j entre ellos, y un término geométrico, que establece la fracción

de energía que se transfiere entre los dos elementos p_i y p_j y depende de la distancia r entre ellos y de su orientación relativa, dada por los ángulos θ_i y θ_j.

En el algoritmo clásico presentado en [38], los factores de forma se usan para construir y resolver un sistema de ecuaciones lineales para realizar el renderizado. Estas ecuaciones describen la radiosidad de un parche p_i en función de la energía desde cualquier otro parche p_j, de los factores de forma (como ponderadores) y la reflectancia del parche p_i. Resolviendo el sistema de ecuaciones se obtiene la radiosidad (brillo) de cada parche tomando en cuenta las interreflexiones difusas y sombras suaves.

Existe un método de refinamiento progresivo [40], éste resuelve el sistema de forma iterativa con valores intermedios de radiosidad para cada parche, donde cada iteración corresponde a un rebote de la luz. Cada parche en la escena mantiene dos valores de energía: la acumulada y la residual [97]. En este esquema, primero se elige un elemento p_i de la escena, el cual será considerado como emisor de energía, para después revisar la visibilidad entre este elemento y todos los demás parches. Si un parche p_j es visible, se calcula la cantidad de energía transferida desde el elemento p_i con base en su energía residual y en el factor de forma correspondiente. Posteriormente, el resultado se multiplica por la reflectancia del elemento p_j (la fracción de la energía entrante que es reflejada de nuevo). El valor resultante es agregado al valor acumulado y al residual de p_j. Cuando la energía residual del elemento p_i se vuelve cero, entonces se selecciona un nuevo elemento para emitir energía.

Entre las ventajas que tiene este método es que es independiente del punto de vista, lo que incrementa el número de cálculos necesarios pero, una vez hechos, los cálculos son útiles para cualquier punto de vista. Generalmente, el método no se usa para resolver la ecuación de renderizado completa, ya que por sí mismo no presenta efectos de iluminación especular y reflexiones brillantes.

5.1.3. *Photon Mapping*

Un método más reciente, y que también es de interés, es el llamado mapeo de fotones (*photon mapping*), introducido en [56, 57, 58], el cuál consiste en simular el

envío de paquetes de fotones hacia los objetos de la escena desde las fuentes de luz, en dónde se pre-procesa una primera estimación de la luz entrante. Estos estimados burdos son usados en un segundo paso para hacer una mejor estimación de la contribución de la luz. De esta forma, se concentran muestras en las partes importantes de la escena.

Durante la primera etapa, el algoritmo funciona emitiendo "fotones", cada que un fotón golpea una superficie se guarda en un *kd-tree* [65] (estructura de datos que divide y organiza puntos en un espacio k dimensional), denominado mapa de fotones. Típicamente se usan dos mapas, uno especial para caústicas y otro global. El mapa de fotones cáusticos se usa sólo para almacenar reflexiones en superficies especulares antes de golpear superficies difusas. Por otro lado, el mapa global de fotones es usado como una aproximación burda al flujo de la luz y es creado emitiendo fotones hacia todos los objetos.

Una vez que hay una intersección rayo-superficie se usa un método de ruleta rusa (técnica de reducción de variancia del método Monte Carlo [60], la cuál es una técnica de integración numérica que utiliza muestreo aleatorio repetido para aproximar un resultado [82, 63]) para determinar si el fotón es reflejado o absorbido. Si el fotón se refleja, la nueva dirección del fotón se calcula con la BRDF de la superficie. De igual forma, se crean rayos de sombra utilizando la primera intersección con un objeto.

En el segundo paso, se renderiza la imagen usando técnicas estadísticas para calcular un estimado de irradiancia basado directamente en el mapa de fotones previamente calculado, por lo que requiere que la cantidad de los mismos sea alta.

Para ello se usa *ray tracing* basado en Monte Carlo, método por el cual se seleccionan aleatoriamente las muestras de los posibles caminos de los rayos de luz. La irradiancia de cada píxel se calcula promediando un número de muestras. Cada muestra consiste en lanzar un rayo desde la pantalla a través del píxel hacia la escena. La irradiancia regresada por cada rayo es calculada en la primera superficie intersecada y se asume que equivale a la irradiancia de la superficie.

5.2. Fenómenos de onda

En esta sección, introduciremos lo más relevante del estado del arte para nuestro trabajo sobre el modelado de onda, dejando de lado los modelos de iluminación global más generales. Un estudio exhaustivo de diversos métodos e implementaciones de iluminación global puede encontrarse en [15].

5.2.1. Trabajos relacionados

Como ya se había mencionado, el objetivo principal de cualquier modelo de iluminación es encontrar el valor de la intensidad de luz en cada punto de la escena. Sin embargo, no existe un consenso en la definición de intensidad [139]. Por ejemplo, puede asociarse a la radiancia, que indica la potencia de la luz captada por unidad de ángulo sólido por unidad de área proyectada en un sistema óptico, pero es un concepto que no se puede definir bien en la óptica de ondas y que no puede medirse directamente; también, se puede asociar a la irradiancia, que es la potencia de luz recibida, o flujo radiante recibido, por unidad de área en una superficie; o puede asociarse a la intensidad radiante, que es el flujo radiante emitido, reflejado o transmitido por unidad de ángulo sólido. Por eso, hay ocasiones en que se prefiere el uso de conceptos como la reflectancia, que define la porción de potencia electromagnética incidente que es reflejada en un punto de la superficie, normalmente modelada a través de BRDFs [91] (es decir, relacionan la irradiancia entrante con la radiancia saliente [90]).

Existen varias implementaciones computacionales para el cálculo de efectos de onda, los esquemas generales son el rastreo del camino óptico (el camino que sigue la luz mientras atraviesa diversos elementos ópticos, por ejemplo [84]), modelado de variaciones de altura en las superficies (por ejemplo, [111]), o bien, la resolución o aproximación de ecuaciones que modelan las ecuaciones de Maxwell para electromagnetismo [46] (por ejemplo, aproximación de Huygens-Fresnel o Kirchhoff, [86]).

La inquietud de modelar los efectos de onda en graficación por computadora se encuentra presente desde 1981, los autores de [84] propusieron un modelo que toma en cuenta el comportamiento de onda en el problema de la iluminación global, re-

presentando a la luz como frentes de onda almacenados en arreglos bidimensionales de números complejos. El patrón de interferencia se genera llevando registro de la distancia y la fase de los frentes de onda.

Aunque no es una aplicación para graficación por computadora, la aproximación mostrada en [35] es notable por su sencillez para generar patrones de difracción y *speckle* en 2D. En este caso se calculan los patrones de difracción y *speckle* sobre un plano (plano difractante) por medio de la Transformada Discreta Rápida de Fourier (DFFT, por sus siglas en Inglés). Para obtener un patrón de difracción se hace la suposición de que la luz llega perpendicularmente a una abertura que actúa como elemento difractante, el patrón resultante se obtiene calculando la DFFT de la abertura difractante, mientras que el *speckle* se reproduce con la DFFT de un arreglo de números aleatorios con una distribución Gaussiana que se multiplican por una constante que representa el factor de rugosidad de la superficie, el camino óptico y la longitud de onda. Con un método similar, [6] propone crear un patrón bidimensional que se usa como tabla de búsqueda (conocida como LUT, el acrónimo de *look up table*) para modelar el *speckle* en la iluminación con luz láser. Este modelo puede tomar en cuenta la decorrelación para cambiar gradualmente el patrón al modificar la posición de la cámara. También, el trabajo presentado en [25] obtiene un patrón de *speckle* realizando la Transformada Rápida de Fourier (FFT) de un arreglo bidimensional de números complejos y multiplicando el resultado punto a punto con el complejo conjugado del arreglo.

En [48] se presenta un método para modelar efectos ópticos causados por delgadas películas multicapas (característica en los caparazones de caracoles, conchas y lentes multicapa), estos efectos incluyen fenómenos físicos como interferencia y absorción dentro de cada capa y reflexión, así como transmisión en cada unión entre capas mediante el cálculo iterativo de rayos múltiples. De forma similar, la propuesta de [54] se enfoca en los efectos ópticos en las burbujas de jabón, ya que el grosor de la burbuja provoca interferencia de las ondas de luz. Las reflectividades de la fina película (resultado de la interferencia) son calculadas de antemano y guardadas en memoria gráfica lo que permite el renderizado en tiempo real.

El modelo propuesto en [42] describe el transporte de luz en superficies con capas usando BSDFs en vez de BSSRDFs al hacer la suposición de que si las capas son delgadas y la luz está significativamente lejos, las ubicaciones de entrada y salida del rayo de luz se encontrarán cerca la una de la otra y, por lo tanto, es aceptable ignorar ese desplazamiento. La idea clave es definir la superficie como una losa plana infinita con una BSDF en la interfaz de la capa superior, otra BSDF en la interfaz de la capa inferior y una función de fase en el interior de la capa. Además, en la losa sólo la posición vertical (profundidad) y las direcciones de los vértices son relevantes a la integral de transporte de luz; para calcular la contribución del rayo de luz se crea un camino de direcciones y vértices.

Los autores de [137] introducen un método para modelar las propiedades reflectivas y transmisivas de estructuras de materiales anisotrópicos con capas arbitrarias. Los datos de reflectancia (incluyendo datos medidos y modelos de microfacetas) se expanden a una representación de frecuencia-espacio en matrices, esta representación permite la composición aditiva a través de un proceso no lineal que calcula las propiedades agregadas de una capa observada a través de otra capa. Además, se permite la composición substractiva, lo que reconstruye la BSDF de un material que se observa indirectamente a través de otra capa con propiedades de reflectancia conocidas. La aproximación depende fuertemente de las bases de Fourier direccionales, cuyas propiedades numéricas permiten cálculos para los parámetros de materiales que van desde la anisotropía leve a la pronunciada.

El trabajo de [77] presenta dos modelos paramétricos de BRDFs, uno inspirado en la teoría de Rayleigh-Rice para la dispersión de luz desde superficies ópticamente suaves y otro inspirado en la teoría de microfacetas. De forma similar, el trabajo de [64] renderiza efectos de difracción usando el mismo modelo de distribución de microfacetas para los diferentes canales de color, los colores iridiscentes se pueden producir perturbando las normales de la microfaceta para cada color.

En el trabajo propuesto en [26] se usa una distribución espectral en lugar de tres canales de color (uno para el espectro rojo, otro para el verde y un último para el azul, como es tradicional) para representar fuentes de luz y propiedades de la superficie en

el modelo de Phong, de esta forma se simulan los efectos dependientes de la longitud de onda (interferencia, dispersión y difracción en capas delgadas) pero también se hace un mejor cálculo de color al incluir la reflectancia en el modelo de Phong.

El trabajo de [111] hace uso del análisis de Fourier para calcular las ondas reflejadas de superficies anisotrópicas (en dónde sí importa la dirección en la que se realicen las mediciones), extendiendo el modelo de He-Torrance [45] e introduciendo un modelo analítico de reflexión para simular efectos de difracción. Los autores derivaron varios modelos de reflexión anisotrópica usando la teoría de onda escalar de Kirchhoff y la teoría de procesos aleatorios. Esto permite describir la estructura de las superficies como un campo de alturas que cambia la fase de las ondas reflejadas y correlacionar dicho campo con las direcciones de entrada y salida de la luz.

Por otra parte, los autores de [138] muestran una forma de describir el campo de luz desde la óptica de ondas usando la intensidad radiante y demuestran la equivalencia entre el campo de luz y una Función de Distribución de Wigner (WDF, por sus siglas en Inglés) suavizada. Esto implica que se puede obtener una medición de la intensidad radiante calculando varias transformadas de Fourier, y sus inversas, del campo escalar en el plano difractante. Esta operación, equivale a descomponer el campo escalar en un conjunto de planos de onda propagándose en diferentes direcciones, entonces se obtiene la magnitud cuadrada de la amplitud de cada plano de onda para calcular la potencia a lo largo de cierta dirección. Posteriormente, los trabajos de [20, 94, 95] implementaron métodos que aprovechan dicha equivalencia en dónde, esencialmente, proponen usar la WDF para representar los frentes de onda complejos como una serie de planos de onda con valores reales y con posibles amplitudes negativas, lo que genera la interferencia. Sin embargo, la WDF de la estructura de una superficie 2D es una función con argumentos tetra dimensionales.

En los trabajos presentados en [94] y [95] se define un marco de trabajo denominado *Augmented Light Field* (ALF) para incluir el modelado de fenómenos de onda en representaciones basadas en rayos. El ALF imita el análisis de la WDF y usa programas *shaders* para simular varios fenómenos de difracción. (Un *shader* es un programa de graficación por computadora usado originalmente y principalmente para cálculos

de sombreado durante el proceso de renderizado; sin embargo, este tipo de programas han evolucionado para permitir la realización de más tareas de renderizado y de cómputo general.)

De manera similar, el artículo [20] introduce un método para crear *Wave Bidirectional Scattering Distribution Functions* (WBSDFs), derivadas de la WDF, que simulan difracción e interferencia en métodos de renderizado basados en rayos. En vez de calcular los efectos para una dirección de entrada y una dirección de salida instantáneamente en el lugar de la reflexión, se difieren los cálculos para etapas posteriores. Una WBSDF codifica indirectamente la información de fase en las radiancias y direcciones reflejadas, también codifica la microestructura de la superficie en una función de reflectancia independiente de los otros elementos o de la iluminación en la escena.

Los autores de [130, 131] presentan un método inspirado en los *shaders* de difracción de [111] extendiendo dicho modelo para construir una BRDF que tome en cuenta simultáneamente variaciones en la fase y la amplitud de la luz. Al igual que los trabajos anteriores, incorpora una WDF para modelar los efectos de la difracción después de varios rebotes y deja al final el cálculo de la fase. Su modelo toma en cuenta la variable del coeficiente de Fresnel, que depende de la longitud de onda, e introduce una métrica para la dispersión de microfacetas. Las variaciones de amplitud y fase se encapsulan en forma de intensidad radiante de los rayos reflejados para explicar la absorción, el ensombrecimiento o la interreflexión de los fotones usando un método Monte Carlo para resolver la ecuación de onda.

El trabajo presentado en [129] introduce el uso de coeficientes de Fresnel para películas dieléctricas y metálicas, respectivamente, para renderizar de forma foto-realista las propiedades de onda de las películas multicapa. El modelo aplica un coeficiente de dispersión de una BSDF dependiente de la longitud de onda para describir las distribuciones espaciales y espectrales de la luz transmitida o reflejada desde una superficie, todo implementado en un algoritmo de *ray tracing*. Se introdujeron ecuaciones de multirayo para representar la reflexión múltiple y la transmisión de la luz dentro de estructuras multicapa cuyos valores pueden usarse para realizar un muestreo Mon-

te Carlo ponderado. Posteriormente, [132] presenta un método para la visualización de colores iridiscentes en estructuras multicapa usando una ecuación de interferencia multirayo y las fórmulas de Fresnel para tomar en cuenta reflexión múltiple, interferencia y absorción.

En aplicaciones ópticas (que por lo tanto no generan imágenes renderizadas), [86] presenta un método para simular difracción múltiple en sistemas de imagenología basados en el principio de Huygens-Fresnel. El campo eléctrico es propagado usando la óptica geométrica y al encontrar una superficie difractante es usado como el campo de entrada para realizar una integral de difracción. El campo eléctrico en el plano de salida se determina con la suma compleja de todos los campos elementales. Posteriormente, en [85] se basan en el trabajo presentado en [95] para proponer dejar que el rayo inicial pueda crear un conjunto de nuevos rayos con la misma posición inicial pero desde diferentes direcciones determinadas usando el valor de la WDF, los rayos secundarios se crean con el método Monte Carlo. La intensidad de un píxel se calcula sumando los valores de WDF acarreados por el rayo.

El trabajo presentado en [89] utiliza un método de diferencias finitas en el dominio del tiempo (FDTD, por sus siglas en Inglés) para propagar los campos eléctricos y magnéticos en una escena tridimensional con la finalidad de generar imágenes por computadora de la mariposa *Morpho*, elegida por su propiedad de generar iridiscencia [108]. Para ello, se crean superficies con materiales nano-estructurados, luego se calcula la interacción de las ondas electromagnéticas con dichas superficies y se modelan los efectos de difracción e interferencia de subsuperficie.

El trabajo presentado en [22] describe una técnica para renderizado interactivo de efectos de difracción producidos por nanoestructuras biológicas, particularmente la piel de dos especies de serpientes, la *Elaphe guttata* y la *Xenopeltis unicolor*, ya que exhiben iridiscencia de moderada a intensa. Para ello, construyeron una BRDF directamente de mediciones de un espécimen y LUTs para evaluar la BRDF en tiempo real basándose en el trabajo realizado por [111] y proponiendo el uso de una expansión de series de Taylor de la función exponencial auxiliar (que depende del campo de altura de la superficie, de las direcciones de entrada y reflexión, así como de la

dirección de vista) para precalcular su transformada de Fourier. La BRDF para un campo de altura discreto se obtiene reemplazando la transformada de Fourier de un campo continuo con la transformada de Tiempo Discreto de Fourier (DTFT, por sus siglas en Inglés). Posteriormente, en [21] se propone un método que modela los coeficientes de la transformada discreta de Fourier (DFT) de un campo aleatorio de alturas como un polinomial complejo, usando funciones base de Chebyshev. Se precalculan los coeficientes del polinomio de Chebyshev y son almacenados en LUTs que se usan para calcular la aproximación final en un *shader*. Recientemente, los autores de [119] presentan un enfoque de factorización para aproximar una LUT de difracción bidimensional con el producto externo de dos matrices de rango bajo que son más sencillas. Su trabajo anterior [118], propone un método de medición directa de efectos de difracción en reflectancia de superficies y una técnica de renderizado que almacena irradiancia difractiva en una LUT. Las reflexiones difusa y especular se calculan con modelos estándar y, para superficies difractivas, se emplea un modelo de difracción basado en una aproximación de Taylor de primer orden del modelo de Stam [111].

En [4] se introduce una extensión a la teoría de microfacetas para renderizado de efectos iridiscentes causados por películas delgadas de grosor variante. Se reemplaza el término de Fresnel del modelo de microfacetas por un término que toma en cuenta reflexiones internas dentro de la película, tomando ventaja de las formas simples de la reflectancia y transmisión de Airy. La interferencia no se causa por variaciones en la microgeometría si no por las diferencias de camino óptico entre las capas.

En el trabajo de [49] se presenta un modelo de reflectancia que aproxima mediciones de reflectancias reales. El estudio muestra que la reflectancia de un material modelada con una BRDF es la suma de tres lóbulos: difuso, difracción y reflexión. El hecho de que la reflectancia sea una combinación de dos fenómenos físicos diferentes permite diseñar funciones base diferentes para cada uno, una función de difracción para la dispersión de gran angular y una función de reflexión de microfacetas cerca de la dirección especular. Un trabajo posterior [50] propone un modelo paramétrico para predecir explícitamente la repartición de energía entre la reflexión especular y los efectos de difracción por medio de una combinación de reflexión de microfacetas

y difracción que se calcula usando la transformada de Fourier de la función de autocovarianza de la altura de la superficie. Además, en [51] se propone la separación de la geometría de la superficie en dos niveles de escala, uno para detalles mayores a la longitud de onda y otro para detalles del mismo orden de magnitud que la longitud de onda. En este caso, la respuesta del material es una combinación de los dos niveles usando una combinación lineal de un lóbulo estándar del modelo Cook-Torrance y un lóbulo de difracción (convolución de respuesta Cook-Torrance y respuesta de difracción).

El trabajo desarrollado en [23] presenta una aproximación para capturar la apariencia de superficies metálicas usando mediciones de la topografía de la superficie en vez de mediciones de reflectancia. Para ello se introduce una función de distribución de normales que se estima de la medición directa de la superficie microscópica (usando el espectro de la FFT y su inverso para obtener algunos parámetros de la distribución) que finalmente se usa para predecir una BRDF.

Los autores de [122] proponen un modelo de iluminación óptica de onda basado en la teoría no-paraxial escalar de la difracción para representar iridiscencia debida a rasguños en las superficies. Cada rasguño se expresa como una colección de segmentos de línea, y para cada segmento se calculan analíticamente los patrones de difracción locales con la transformada de Fourier unidimensional. La formulación exhibe un comportamiento multiescala, la interferencia de un gran número de rasguños causa que a la distancia el resultado se parezca a los modelos geométricos típicos mientras que ampliaciones revelan reflexiones iridiscentes.

En [135] se introduce un modelo de dispersión de luz en fibras pequeñas que permite simular interferencia y difracción. Para ello proponen una función de dispersión azimutal, resolviendo las ecuaciones de Maxwell para una sección transversal constante de una fibra con el BEM (*Boundary Element Method*) [44].

En [106] se propone una técnica basada en la solución de forma cerrada de la transformada de Fourier continúa para primitivas de vector simples con evaluación jerárquica y progresiva que resuelve la difracción del campo lejano y cercano. Se aprovecha de que la imagen de la abertura difractante es relativamente simple en

color y forma para acelerar los cálculos de la transformada de Fourier. La idea es descomponer la señal de entrada en una suma de primitivas (polígonos) con intensidad constante y usar el principio de la superposición, lo que permite que la transformada de Fourier de cada primitiva se calcule independientemente. Sin embargo, para el cálculo del campo cercano, involucra la evaluación de integrales de Fresnel que no tienen solución analítica.

Los autores de [73] revisan un modelo de microfacetas para derivar una solución analítica para dispersión múltiple en superficies rugosas. Se modelan microfacetas asimétricas, reales y virtuales, que permiten calcular dispersión de orden mayor de forma analítica.

En [41] se presenta un método para simular la óptica de materiales nacarados, modelando la dispersión de la luz desde microestructuras 3D ubicadas en varias capas, estas microestructuras están caracterizadas por su grosor y un índice de refracción complejo dependiente de las longitudes de onda.

Más recientemente, en [112] generalizan un método de transporte de luz basado en radiometría modelando la densidad espectral transversal como una función, indicando la distribución probable de la potencia a lo largo del espectro. Este método describe el estado de coherencia y correlación entre dos puntos del espacio-tiempo.

El primer trabajo en llevar a cabo la implementación de un sistema capaz de manejar los efectos de la polarización fue [128]. En dicho trabajo se demostraron varios efectos, tales como cambios en los colores de las luces reflejadas. Por otra parte, el trabajo presentado en [113] se enfocó en renderizar cristales anisotrópicos con más de un eje óptico y extendió las técnicas utilizadas por [128]. La meta principal de los dos trabajos anteriores fue encontrar una forma apropiada para describir y realizar cálculos con luz polarizada, para ello utilizaron el formalismo de la matriz de coherencia [8]. El modelo propuesto por [45] considera los efectos de polarización y los define teóricamente, pero no lleva a cabo su implementación.

Más recientemente, el trabajo presentado en [124] presenta una propuesta para incluir efectos de polarización y fluorescencia en un sistema de renderizado, utilizando los vectores de Stokes [8] como una descripción del estado de polarización, lo cual

permite usar ambos efectos simultáneamente; cabe mencionar que el primer trabajo en sugerir el uso del formalismo de los vectores de Stokes en lugar de las matrices de coherencia fue [32].

En [43] se presenta un modelo para renderizar piedras preciosas basado en el análisis de fenómenos ópticos, entre ellos los efectos de la polarización, ya que estos afectan significativamente la apariencia de dichos objetos. Para ello, se utilizan matrices de coherencia para representar la intensidad y el estado de polarización del campo eléctrico a lo largo de un rayo de luz y se reagrupan las facetas de las piedras para construir un árbol de facetas usado para el cálculo de la contribución de una faceta a la imagen final.

Por otro lado, [125] propone un modelo analítico para simular la polarización de la luz atmosférica basado en el método propuesto en [100]. Su método renderiza escenas al aire libre incluyendo luz del Sol y los efectos de perspectiva aérea (desaturación y cambio de color en objetos distantes). La idea básica es calcular la función de irradiancia espectral del cielo para un conjunto de posiciones del Sol y para algunas turbulencias dadas.

En el trabajo propuesto en [126] se discuten los problemas de la visualización de la polarización y sus fundamentos físicos. Además, se propone un conjunto de visualizaciones estandarizadas para facilitar las comparaciones del trabajo de distintos investigadores; esto se propuso con el objetivo de ayudar al desarrollo de sistemas de renderizado con la capacidad de representar la polarización de la luz. Su propuesta cubre aspectos de la polarización que son relevantes para la investigación en graficación por computadora pero también para tareas de diseño óptico.

El trabajo presentado en [72] se enfoca en formular y resolver un sistema algebraico no lineal que se obtiene cuando el proceso de refracción es simulado usando el modelo geométrico de Huygens [8]. Su principal contribución se enfoca en el caso de birrefringencia y medios biaxiales; sin embargo, no toma en cuenta los problemas de polarización.

El trabajo de [103] presenta una técnica para simular fenómenos relacionados con los arcoíris, su técnica está basada en las técnicas de *ray tracing*, que extiende para

tomar en cuenta dispersión, interferencia y difracción utilizando algo similar a vectores de Jones [19].

El objetivo principal de muchos de estos trabajos ha consistido en encontrar una forma apropiada para describir y realizar los cálculos relativos al renderizado usando luz polarizada; desarrollos que incluyen formalismos matemáticos o estructuras de datos. Puede notarse la existencia de dos grandes grupos, aquellos que se quedaron con la notación sugerida por las matrices de coherencia y aquellos que se quedaron con la notación de los vectores de Stokes. Ambos formalismos son equivalentes y pueden ser convertidos el uno en el otro [1].

Más recientemente, en [55] se presenta una extensión de los métodos de *path tracing* y de *photon mapping* para soportar polarización. Se propone una integral de camino de vector que requiere mantener la secuencia de eventos para que las operaciones se lleven a cabo de forma adecuada (desde la pantalla hacia la luz y viceversa), para después fusionar ambos subcaminos. De esta forma se conserva la simetría de operaciones necesaria para ambos métodos de renderizado mientras se respeta la direccionalidad de efectos como la polarización y fluorescencia.

La mecánica cuántica establece que la luz posee un comportamiento dual y que puede comportarse a la vez como onda y como partícula, dicho comportamiento afecta la manera en que la luz interactúa con la materia y, por lo tanto, también afecta la manera en que percibimos esa interacción.

A pesar de ello, en los métodos descritos anteriormente no se modela el comportamiento de onda de la luz, esto se debe principalmente a la dificultad y el costo de simularlos [111]. Además, en la literatura han sido pocos los esfuerzos y la atención que se le ha brindado a la generación de efectos ópticos debidos al comportamiento de onda de la luz. Esto se debe principalmente a que, en general, en graficación por computadora esos efectos pueden considerarse como poco relevantes. Sin embargo, hay circunstancias particulares donde modelar los fenómenos de onda es importante, tales como la interferencia, la difracción y la polarización, cuyos efectos en la percepción de los materiales es de gran importancia y de los cuales se hará una breve descripción a continuación.

5.2.2. Interferencia

La interferencia óptica corresponde a la interacción de dos o más ondas de luz, cuyas características pueden ser diferentes, que produce una irradiancia derivada de la suma de las componentes de irradiancias de cada onda [46]. En su forma más sencilla, la interferencia se da cuando dos ondas se superponen al incidir en el mismo punto y generan una nueva onda cuya amplitud es igual a la suma de amplitudes de las ondas individuales, por lo que la interferencia puede ser constructiva o destructiva. Un ejemplo de este fenómeno puede observarse en los patrones de colores de una mancha de aceite en un charco de agua y en los arcos supernumerarios de los arcoíris.

La irradiancia en un punto $\boldsymbol{x} \in \mathbb{R}^3$ del espacio para dos campos eléctricos $E_1(\boldsymbol{x})$ y $E_2(\boldsymbol{x})$ correspondientes a dos fuentes de luz que interfieren entre ellas y que tienen amplitudes A_1 y A_2, respectivamente, se calcula como

$$I(\boldsymbol{x}) = I_1(\boldsymbol{x}) + I_2(\boldsymbol{x}) + I_{12}(\boldsymbol{x}), \tag{5.3}$$

tomando en cuenta que

$$\begin{aligned} I_1(\boldsymbol{x}) &= \langle (A_1(\boldsymbol{x}))^2 \rangle_T , \\ I_2(\boldsymbol{x}) &= \langle (A_2(\boldsymbol{x}))^2 \rangle_T , \\ I_{12}(\boldsymbol{x}) &= 2 \langle A_1(\boldsymbol{x}) A_2(\boldsymbol{x}) \rangle_T \quad = 2\sqrt{I_1(\boldsymbol{x}) I_2(\boldsymbol{x})} \cos\phi. \end{aligned} \tag{5.4}$$

En las ecuaciones anteriores $\langle (A_j(\boldsymbol{x}))^2 \rangle_T$, para $j \in \{1, 2\}$, representa el promedio en tiempo del cuadrado de la amplitud del campo eléctrico correspondiente, $I_{12}(\boldsymbol{x})$ es el término de interferencia y ϕ es la diferencia de fase. Aunque la ecuación (5.4) fue enunciada para luz linealmente polarizada, también puede usarse para cualquier estado de polarización ya que cualquier estado de polarización puede sintetizarse con dos ondas ortogonales linealmente polarizadas [46].

5.2.3. Difracción

La difracción es un efecto característico de los fenómenos de onda que ocurre cuando una porción de un frente de onda se ve obstruido, alterando una región de la onda en fase o amplitud [46]. Los segmentos de onda que se propagan más allá del obstáculo interfieren entre sí causando una distribución de energía-densidad particular llamada *patrón de difracción*. En realidad, no existe una distinción física entre interferencia y difracción, sin embargo, se ha vuelto costumbre hablar de interferencia cuando se considera la superposición de pocas ondas (conjunto discreto de fuentes o frentes de onda originados de la misma fuente) y difracción cuando se trata de un gran número de ondas (conjunto continuo de fuentes o diferentes partes del mismo frente de onda) [46].

5.2.3.1. Principio de Huygens-Fresnel

El principio de Huygens-Fresnel dice que cada punto de un frente de onda que no ha sido obstruido puede, en un momento dado, verse como la fuente de ondas esféricas secundarias con la misma frecuencia de la onda primaria. La amplitud del campo óptico resultante en cualquier punto posterior se puede considerar como la superposición de todas estas ondas, considerando sus amplitudes y fases relativas.

Suponiendo que se tiene una configuración que consiste de una fuente de luz distante, una pantalla de proyección y una pantalla opaca con una abertura entre ambas y muy cerca de la pantalla de proyección, se puede observar una proyección de la abertura sobre la pantalla de observación, reconocible a pesar de la aparición de unas franjas alrededor de la misma. Si la longitud de onda es grande comparada con la abertura, las ondas se esparcirán en ángulos mayores al pasar la región de la abertura. Si el plano de observación se mueve lejos de la pantalla con la abertura, la imagen de la abertura se vuelve más estructurada y las franjas más prominentes. Este fenómeno recibe el nombre de difracción de Fresnel o difracción del campo cercano. A una distancia muy grande de la pantalla, el patrón proyectado ya no guardaría mucha relación con la abertura y sólo cambiaría en tamaño pero no en forma aunque

se siga alejando el plano de observación. Ésta es la difracción de Fraunhofer o de campo lejano. Si en ese punto pudiera reducirse la longitud de onda de la radiación entrante el patrón se revertiría al patrón de Fresnel.

5.2.3.2. Difracción de Kirchhoff

La fórmula de la difracción escalar de Kirchhoff en el vacío para un punto $\boldsymbol{x}$ está dada por

$$E(A;\ \lambda) = \frac{1}{4\pi}\int_S \left[\frac{e^{iks}}{r}\frac{\partial A}{\partial \overrightarrow{n}} - A\frac{\partial}{\partial \overrightarrow{n}}\left(\frac{e^{iks}}{r}\right)\right] ds, \tag{5.5}$$

donde $k = \frac{2\pi}{\lambda}$, denotando el recíproco de la longitud de onda λ o número de propagación, r es la distancia de la abertura al punto $\boldsymbol{x}$, A es un fasor que representa la amplitud compleja de la perturbación en la superficie S y $\partial A/\partial \overrightarrow{n}$ es la derivada direccional de A en dirección de la normal $\overrightarrow{n}$ en el elemento de la superficie ds. Es importante hacer notar que se puede encontrar una equivalencia entre la difracción de Kirchhoff y el principio de Huygens-Fresnel. A pesar de que la fórmula es adecuada para describir varias configuraciones de problemas de difracción no existe una solución analítica para la mayoría de ellas, así que la distancia r determina si puede utilizarse la aproximación de Fresnel o la aproximación de Fraunhofer, ya que dan lugar a distintos fenómenos de difracción.

5.2.3.3. Difracción de Fraunhofer

La difracción de Fraunhofer es una versión simplificada de la fórmula de difracción de Kirchhoff. Estrictamente sólo se puede usar cuando el plano de observación está en el infinito donde se pueden considerar frentes de onda planos, pero prácticamente se habla de que el fenómeno ocurrirá en una pantalla con una abertura rectangular de ancho b cuando

$$R > b^2/\lambda,$$

donde R es la menor de las distancias entre la distancia fuente de luz-abertura o la distancia abertura-plano de proyección.

Por otro lado, hacemos la suposición de que entre la fuente de luz y el plano

de proyección se tiene una pantalla que cuenta con varias aberturas rectangulares, cada una de ellas con el mismo ancho b y con una longitud infinita, tomando por simplicidad que $\beta_\theta \equiv (kb/2)\sin\theta$ con un número de propagación k y un ángulo de difracción θ, entonces la irradiancia resultante de una fuente coherente que tiene una intensidad I_i es

$$I(\theta) = I_i \left(\frac{\sin \beta_\theta}{\beta_\theta} \right)^2. \tag{5.6}$$

Cuando se tienen N aberturas rectangulares, la función de flujo-densidad se vuelve

$$I(\theta) = I_i \left(\frac{\sin \beta_\theta}{\beta_\theta} \right)^2 \left(\frac{\sin N\alpha}{\sin \alpha} \right)^2, \tag{5.7}$$

donde $\alpha \equiv ka/2$, a es la separación entre aberturas rectangulares y b en β_θ sigue siendo el ancho de cada abertura. En este caso, la función sinc definida por (5.6) actúa como envolvente para los siguientes valores de intensidad de las franjas de difracción.

La expresión para la perturbación óptica en un punto $\boldsymbol{x} \in \mathbb{R}^3$ para una abertura arbitraria $\mathcal{A}$ está dada por

$$E(\boldsymbol{x}) = \frac{\varepsilon_{\mathcal{A}}}{\|\boldsymbol{x}\|} e^{i(\omega t - k\|\boldsymbol{x}\|)} \int\int_{\mathcal{A}} e^{ik(x_2 x_2' + x_3 x_3')/\|\boldsymbol{x}\|} dS, \tag{5.8}$$

donde $\varepsilon_{\mathcal{A}}$ es la potencia de la fuente por unidad de área (asumiendo que es constante sobre toda la abertura), $\|\boldsymbol{x}\|$ representa la norma ℓ_2, o la distancia Euclidiana, entre el punto $\boldsymbol{x}$ y el origen del sistema coordenado del plano de la abertura, la frecuencia angular temporal es $\omega = 2\pi v$ y v es la frecuencia temporal.

5.2.3.4. Difracción de Fresnel

La difracción de Fresnel es una aproximación de la integral de difracción de Kirchhoff-Fresnel usada para calcular el patrón de difracción generado en el campo cercano. Es más complicada que la difracción de Fraunhofer porque al encontrarse la pantalla de proyección a poca distancia de la abertura, las ondas ya no pueden considerarse planas. En este caso, la perturbación en el punto $\boldsymbol{x}$ está dada por

$$E(\boldsymbol{x}) = \frac{e^{-ikx_3}}{i\lambda} \int\int E(x_1,\ x_2,\ 0) e^{\left[-\frac{ik}{2x_3}((x_1-u)^2+(x_2-v)^2)\right]} dx_1\, dx_2, \tag{5.9}$$

donde k sigue siendo el número de propagación y $E(x_1,\ x_2,\ 0)$ es el campo eléctrico en el punto $(x_1,\ x_2,\ 0)$ de la abertura.

Sin embargo, el problema con esta expresión es que sólo tiene solución analítica en casos especiales, es por ello que se han buscado alternativas de representación, por ejemplo, usando las integrales de Fresnel o resolviendo el problema como una convolución. Para ello se define la siguiente función

$$h(\boldsymbol{x}) = \frac{e^{ikx_3}}{i\lambda x_3} e^{\frac{ik}{2x_3}(x_1^2+x_2^2)},$$

de esta forma, la integral (5.9) puede expresarse en términos de una convolución como

$$E(\boldsymbol{x}) = E(x_1,\ x_2,\ 0) * h(\boldsymbol{x}).$$

5.2.3.5. Relación entre Transformada de Fourier y la difracción

Podemos expresar un rayo de luz monocromática como la suma de varias ondas planas que viajan en diferentes ángulos, cada una con la misma frecuencia pero con su propia amplitud y fase. Cada una de estas ondas planas se puede considerar como una frecuencia espacial y cuando pasan a través de una lente o abertura son transformadas en posiciones. Esta descomposición es equivalente a realizar la transformada de Fourier de la distribución de la luz que atraviesa la abertura o lente, el resultado de la transformación indica la distribución de la luz en la pantalla de proyección (por ejemplo, la forma del patrón de difracción).

Para explicar más a detalle la relación entre la transformada de Fourier y la difracción, se puede tomar el ejemplo de la transparencia fotográfica descrito en [46]. Podemos suponer que una transparencia o diapositiva equivale a un registro bidimensional de la distribución de la luz que formaba la imagen original. La información almacenada en dicha transparencia puede leerse como una señal al iluminarla con alguna fuente de luz. Si cada punto en la superficie de la transparencia se ilumina

con ondas planas monocromáticas, ésta actúa como un dispersor y los rayos de luz emergerán de ella en un amplio rango de direcciones. Cada plano de onda o conjunto de rayos paralelos viajando en una dirección particular puede considerarse una componente de frecuencia espacial de Fourier. El conjunto de todos estos planos de onda se puede interpretar como la transformada de Fourier del campo óptico transmitido en la transparencia. La transformada de Fourier del campo eléctrico en la transparencia es una función de ponderación que representa la contribución relativa de cada componente de frecuencia en el campo y, por lo tanto, de cada plano de onda dejando la abertura.

Consecuentemente, la amplitud del campo eléctrico en cualquier parte del patrón de difracción de Fraunhofer corresponde a la transformada de Fourier de la señal de entrada, es decir, la distribución del campo eléctrico sobre la abertura, aunque es importante enfatizar que ninguna es directamente medible. El único fenómeno observable es la distribución bidimensional de la irradiancia, que es idéntica al cuadrado del valor absoluto de la transformada de Fourier del campo de entrada [46].

En [138] se afirma que usando la técnica del espectro angular se puede descomponer un campo escalar arbitrario expandiendo el campo de onda en un número infinito de planos de onda que se propagan en varias direcciones. Esa descomposición es equivalente a obtener la transformada de Fourier del campo escalar en ese plano.

El método del espectro angular es una técnica para el modelado de la propagación del campo de onda. Involucra la expansión de un campo de onda complejo como la suma de un número infinito de ondas planas que viajan con diferentes amplitudes y direcciones de propagación. Al aplicar la transformada de Fourier sobre un campo de onda se obtiene el espectro angular de las ondas planas componentes, cada una viajando en una sola dirección.

5.2.3.6. Láser

Un láser es un dispositivo que utiliza la emisión inducida de fotones para generar un haz de luz coherente. La coherencia es una propiedad de las ondas que permite interferencia estacionaria (temporal o espacial), ésta ocurre si la diferencia de fase

es constante entre campos eléctricos en diferentes posiciones y describe la correlación entre cantidades físicas en una o más ondas; esto significa que los fotones tienen frentes de onda organizados. La luz láser es monocromática, es decir contiene una longitud de onda específica y muy direccional y tiene un haz de luz muy fuerte y concentrado.

Cuando un rayo de luz coherente ilumina una superficie rugosa aparece el fenómeno llamado *speckle*. Este fenómeno se atribuye al hecho de que cuando una onda electromagnética monocromática es reflejada desde una superficie rugosa (en términos de la escala de la longitud de onda de la luz) la onda resultante en un punto de observación consiste de componentes que surgen de un elemento microscópico diferente de la superficie y, por lo tanto, cada una tiene un camino óptico diferente. Cuando la superficie es irregular, la reflectancia compleja es una función aleatoria, por lo que existe la probabilidad de que en algún punto en el espacio y sus alrededores exista una intensidad máxima debido a la interferencia constructiva. Estos puntos brillantes tienen forma y distribución irregular y proveen *speckle* en una pantalla de observación [35].

El *speckle* juega un papel importante en varios fenómenos físicos [37], tanto como un problema a resolver por sí mismo como en una variedad de aplicaciones, por ejemplo, interferometría holográfica.

5.2.4. Polarización

En diferentes campos del conocimiento basta con describir la luz como una onda electromagnética con cierta frecuencia que viaja linealmente a través del espacio como un conjunto de rayos discretos; sin embargo, varios experimentos revelan que el tren de onda también oscila alrededor de su dirección de propagación en un plano perpendicular a dicha dirección, esta propiedad es conocida como polarización.

Es raro que la luz se emita en forma polarizada, en la mayoría de los casos este fenómeno es el resultado de la interacción con superficies o medios de transmisión y, por esa razón, no suele modelarse en graficación por computadora. Sin embargo, la polarización es esencial para configuraciones artificiales, como aquellas que usan filtros

polarizantes. Esas escenas aún son problemáticas para los métodos de renderizado foto-realistas (en parte debido a la complejidad de dichos métodos). De igual forma, el uso de filtros polarizados en escenas al aire libre puede hacer una diferencia en las imágenes obtenidas (por ejemplo, mejorando el color y el contraste), debido a la fuerte presencia de polarización en la luz atmosférica. Otros efectos que dependen de la polarización son patrones de oscurecimiento o decoloración en objetos metálicos y sus reflexiones, así como el oscurecimiento de ciertas caras de cristales y la birrefringencia en objetos transparentes [43].

5.2.5. Representación de polarización

Existen diversos formalismos para describir el estado de polarización dada una configuración de luz con longitud de onda λ, para este trabajo se realizó un análisis de diversas implementaciones de polarización en la literatura de las áreas de graficación y visión por computadora. Las representaciones más usadas son los vectores de Jones, los vectores de Stokes y las matrices de coherencia [8]. De estos formalismos, los vectores de Jones solamente son aplicables a la luz totalmente polarizada. Por otra parte, para los dos formalismos restantes los autores de [126] dan las siguientes como razones principales para darle preferencia a la representación usando vectores de Stokes sobre las matrices de coherencia:

- Consideran que el significado de los componentes de los vectores de Stokes es más intuitivo (ayudando a la depuración y a la interacción con el sistema de cómputo).
- Sólo involucran aritmética de números reales.

Por lo tanto, para este trabajo utilizaremos los vectores de Stokes junto con las matrices de Mueller para describir la polarización; adicionalmente a las razones anteriormente mencionadas, en [127] se presenta una explicación general de como implementar estos formalismos en un programa basado en *ray tracing*. Es importante hacer notar que existen métodos de conversión entre los vectores de Stokes y las matrices de coherencia [1]; por lo que, en teoría, nuestra propuesta es general.

5.2.5.1. Vectores de Stokes

Los vectores de Stokes (llamados así a pesar de no ser formalmente vectores desde el punto de vista de la física) permiten describir el estado de polarización de una onda transversal con solo cuatro parámetros dada una longitud de onda. Las componentes de un vector de Stokes $\bar{\mathbf{s}} \in \mathbb{R}^4$ se definen como

$$\begin{aligned} s_0 = &\quad \kappa\left(A_x^2 + A_y^2\right), \\ s_1 = &\quad \kappa\left(A_x^2 - A_y^2\right), \\ s_2 = &\quad \kappa\left(2A_x \cdot A_y \cdot \cos\delta\right) \text{ y} \\ s_3 = &\quad \kappa\left(2A_x \cdot A_y \cdot \sin\delta\right). \end{aligned}$$

Dónde A_x y A_y son las amplitudes de las componentes del campo eléctrico E en los planos perpendiculares a los vectores $\overrightarrow{y}$ un plano Cartesiano $\overrightarrow{\boldsymbol{x}}\overrightarrow{\boldsymbol{z}}$) y $\overrightarrow{\boldsymbol{x}}$ (un plano Cartesiano $\overrightarrow{\boldsymbol{y}}\overrightarrow{\boldsymbol{z}}$), respectivamente, δ es la diferencia entre las fases en el eje $\overrightarrow{\boldsymbol{x}}$ y el eje $\overrightarrow{\boldsymbol{y}}$, y κ es el grado de polarización. La componente $s_0 \geq 0$ representa la intensidad radiante no polarizada (es decir, la misma cantidad que utilizan los renderizadores que no implementan polarización), las componentes s_1 y s_2 describen la preferencia de la onda por la polarización lineal a 0° y 45°, respectivamente, mientras que la cuarta componente, s_3, codifica la preferencia por la polarización circular en sentido de las manecillas del reloj. Las componentes se encuentran restringidas por las siguientes condiciones $s_1,\ s_2,\ s_3 \in [-s_0,\ s_0]$ y $s_0 \geq \sqrt{s_1^2 + s_2^2 + s_3^2}$.

Sin embargo, es frecuente normalizar el vector de Stokes de la siguiente manera

$$\bar{\mathbf{s}} = I \begin{bmatrix} 1 \\ s_1/I \\ s_2/I \\ s_3/I \end{bmatrix},$$

donde el factor de normalización es la intensidad I de la luz en el punto donde se calcula la polarización.

A continuación, se describirán algunos de los vectores de Stokes para ciertas con-

figuraciones de polarización [105]. Para la luz linealmente polarizada de forma horizontal se tiene que $A_y = 0$, por lo tanto la intensidad $I = A_x^2$, por lo que su vector de Stokes es

$$\bar{\mathbf{s}} = I \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

De forma similar, para la luz linealmente polarizada de forma vertical se tiene que $A_x = 0$ e $I = A_y^2$, lo cual genera el siguiente vector de Stokes

$$\bar{\mathbf{s}} = I \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}.$$

La luz linealmente polarizada a 45°, donde $A_x = A_y$ e $I = 2A_x^2$, tiene como vector de Stokes

$$\bar{\mathbf{s}} = I \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Como último ejemplo, la luz circularmente polarizada hacia la derecha se representa con $A_x = A_y$ e $I = 2A_x^2$, por lo que tiene la siguiente representación como vector de Stokes

$$\bar{\mathbf{s}} = I \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

5.2.5.2. Matrices de Mueller

Desde el punto de vista computacional, además de contar con una estructura para guardar la información sobre la luz, también es necesario tener una estructura para describir la atenuación de la misma, para ello se usan las matrices de Mueller. Éstas son matrices reales de dimensiones 4×4 que pueden describir todos los cambios de la intensidad y el estado de polarización de la luz; como resultado, esta representación permite afectar los vectores de Stokes utilizando álgebra de matrices.

A continuación, se describirán las matrices de Mueller más comunes que se necesitan en el contexto de graficación por computadora como se presentan en [127].

Atenuación simple La atenuación simple representa una interacción idealizada donde la luz pasa en línea recta a través de un objeto sin alterar su estado de polarización aunque su intensidad puede verse alterada. Por ejemplo, si observamos la luz que pasa a través de una delgada capa de material transparente, la intensidad de la luz se atenúa uniformemente pero sus otras características permanecen sin cambio. Considerando que la atenuación se representa con un factor σ, la matriz de Mueller para este comportamiento se define como:

$$\mathbf{M}_{A_\sigma} = \begin{bmatrix} \sigma & 0 & 0 & 0 \\ 0 & \sigma & 0 & 0 \\ 0 & 0 & \sigma & 0 \\ 0 & 0 & 0 & \sigma \end{bmatrix}.$$

Despolarizador Las superficies perfectamente difusas y los materiales transmisivos actúan como despolarizadores, es decir, no sólo la intensidad de la luz incidente es atenuada si no que cualquier polarización presente en dicha luz es destruida. Por ejemplo, este tipo de interacción se da en las BRDF Lambertianas [127]. Para un factor de atenuación σ, la matriz de Mueller para este comportamiento está dada

por:

$$\mathbf{M}_{D_\sigma} = \begin{bmatrix} \sigma & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Polarización lineal ideal Para un plano de polarización lineal perfecta (que contiene el punto $\boldsymbol{r}$, con normal $\overrightarrow{\boldsymbol{n}}_p$ y $\mathbf{z} = \mathbf{r} \times \overrightarrow{\boldsymbol{n}}_p$) su orientación puede representarse a través del ángulo ϕ con respecto al eje $\overrightarrow{\boldsymbol{x}}$ y su representación matricial es la siguiente:

$$\mathbf{M}_{P_\phi} = \frac{1}{2} \begin{bmatrix} 1 & \cos(2\phi) & \sin(2\phi) & 0 \\ \cos(2\phi) & \cos^2(2\phi) & \sin(2\phi)\cos(2\phi) & 0 \\ \sin(2\phi) & \sin(2\phi)\cos(2\phi) & \sin^2(2\phi) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Reflectancia de Fresnel Sólo existen dos casos en los que existe una fórmula exacta para la matriz de Mueller, la superficie Lambertiana como despolarizador perfecto y la reflexión de Fresnel de un límite de fase perfectamente suave. La fórmula para determinar la matriz de Mueller en estos casos es

$$\mathbf{M}_{F_{F_\perp},\ F_\parallel,\ \varphi_\parallel,\varphi_\perp} = \begin{bmatrix} \frac{F_\perp + F_\parallel}{2} & \frac{F_\perp - F_\parallel}{2} & 0 & 0 \\ \frac{F_\perp - F_\parallel}{2} & \frac{F_\perp + F_\parallel}{2} & 0 & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -D & C \end{bmatrix},$$

donde $F_\parallel$ y $F_\perp$ son los términos de Fresnel paralelo y vertical, respectivamente, que determinan la cantidad de luz incidente que se refleja para las componentes de una onda incidente, para las constantes C y D definidas como

$$C = \cos\left(\varphi_\perp - \varphi_\parallel\right) \cdot \sqrt{F_\perp F_\parallel} \text{ y}$$
$$D = \sin\left(\varphi_\perp - \varphi_\parallel\right) \cdot \sqrt{F_\perp F_\parallel}.$$

La diferencia $\varphi_\perp - \varphi_\parallel$ representa el retardo total al que está sujeto el tren de onda incidente (por ejemplo, el desplazamiento de fase relativo entre los componentes vertical y horizontal que el tren de onda ha experimentado durante la reflexión).

5.2.5.3. Operaciones de polarización

Un problema de implementación asociado a la polarización es que el sistema de coordenadas de cada luz o, alternativamente, el valor de atenuación que se está describiendo debe ser seguido en todo momento. Es decir, los vectores de Stokes que se usan para describir cuantitativamente un estado de oscilación no tienen sentido sin el sistema coordenado en el que están definidos, por lo que debe guardarse tal marco de referencia para cada vector de Stokes que se almacene. Además, valores de atenuación como las matrices de Mueller requieren dos marcos de referencia: uno para la luz entrante y otro para la luz reflejada o saliente.

La única operación entre vectores de Stokes que involucra dos fuentes de luz L_0 y L_1, y que tiene sentido desde el punto de vista físico, es la adición. Debido al principio de superposición, la suma de dos valores de luz puede hacerse de forma simple, componente a componente para cada $s_i \in \bar{\mathbf{s}}$ de cada fuente para obtener la componente de la luz resultante $s_i(L_r)$, de la siguiente forma:

$$s_i(L_r) = s_i(L_0) + s_i(L_1).$$

La suma de vectores de Stokes es conmutativa pero se permite sólo si los marcos de referencia de ambos valores de luz coinciden exactamente. Sin embargo, si los dos marcos de referencia son colineales pero no están alineados, pueden cambiarse a un marco de referencia común y entonces sumarse.

La única forma de combinar n matrices de Mueller $\mathbf{M}_0$, $\mathbf{M}_1$, ..., $\mathbf{M}_n$ en una sola matriz resultante $\mathbf{M}_r$ es a través de su multiplicación:

$$\mathbf{M}_r = \mathbf{M}_0 \times \mathbf{M}_1 \times \ldots \times \mathbf{M}_n.$$

Al igual que en el caso de los vectores de Stokes, la multiplicación sólo puede hacerse si los marcos de referencia coinciden. Sin embargo, al tener dos marcos de referencia, uno de entrada y otro de salida, la multiplicación es más complicada: el marco de salida de la primera matriz debe ser el marco de referencia de entrada de la segunda, de otra forma, el resultado no tiene significado.

Adicionalmente, existe una operación que involucra los vectores de Stokes y las matrices de Mueller, una multiplicación del vector de Stokes $\bar{\mathbf{s}}$ con la matriz de Mueller $\mathbf{M}_0$ se lleva a cabo teniendo como resultado el vector de la luz atenuada $\bar{\mathbf{s}}_r$

$$\bar{\mathbf{s}}_r = \mathbf{M}_0\bar{\mathbf{s}}.$$

Nuevamente, el marco de referencia del vector de Stokes debe coincidir con el marco de entrada de la matriz de Mueller y el vector de Stokes resultante tiene el marco de referencia de la matriz.

En el siguiente capítulo se presenta la segunda propuesta de este trabajo, que consiste en un método que pueda incorporar los efectos de onda debidos a la difracción y a la polarización para generar patrones de difracción que permitan ver el cambio de tonalidad en los materiales, usando las propiedades de la difracción de Fraunhoffer que permite calcular el patrón de difracción resultante de la interacción entre una apertura difractante y una fuente de luz por medio de la transformada de Fourier de dicha apertura, así como implementando los vectores de Stokes operados con matrices de Mueller para manejar la polarización.

La principal ventaja de la propuesta es que sólo requiere una transformada de Fourier para generar los efectos de difracción que otros métodos generan y que, aunque el resultado no es físicamente exacto, sí tiene un fundamento físico para su funcionamiento.

Capítulo 6

Modelo para el Comportamiento de Onda en Iluminación Global

En este capítulo describiremos el modelo de transporte de la luz propuesto para incorporar el comportamiento de onda a un sistema de renderizado con iluminación global basado en *ray tracing*. Primero nos enfocaremos en la parte del modelo que produce efectos de difracción e interferencia para posteriormente describir la parte del modelo que maneja el estado de polarización.

6.1. Modelo de luz incorporando ondas

Como mencionamos en los Capítulos 1 y 5, muchos de los modelos de iluminación y sombreado que son usados tradicionalmente en graficación por computadora incluyen una variedad de "atajos" y simplificaciones que no tienen una base sólida en la teoría pero funcionan en la práctica, ya que muchos de ellos pueden producir resultados visuales atractivos reduciendo el costo computacional [29]; un ejemplo claro es la ecuación (1.1), que modela materiales que parecen plásticos pero tiene problemas para presentar otros materiales de forma realista.

A pesar de que existen métodos de iluminación más apegados a la simulación física (por ejemplo, [89]) o que son resultado de mediciones directas (por ejemplo, [118]), es más común aproximar las reglas de la óptica y la radiación térmica necesarias para modelar la interacción de la luz, por ejemplo, para simplificar el cómputo, porque los modelos precisos no se conocen, o bien, tienen tiempos de cálculo prohibitivos.

Nuestro modelo para la incorporación de ondas sigue la ruta de realizar una aproximación que permita generar efectos de onda utilizando bases físicas sin llegar a ser una simulación y tomando ventaja de las implementaciones de gráficos por computadora existentes.

6.1.1. Luz estocástica

Del Capítulo 5 podemos recordar que para poder incorporar efectos de comportamiento de onda en un modelo de transporte de luz es necesario poder seguir o predecir la fase de la onda mientras ésta interactúa con otras ondas u objetos en una escena. Para ese propósito, asumimos que cuando luz llega a un punto $\boldsymbol{x}$ sobre una superficie, existe un desfase aleatorio (debido a que existe algún elemento difractante, ya sea abertura o microestructura) que altera el camino óptico de las ondas lo que resulta en la generación de un patrón de difracción por la interferencia constructiva y destructiva entre las diferentes ondas. Podemos definir el campo de luz incidente sobre el punto $\boldsymbol{x}$ (representado por el vector que va desde el origen del sistema coordenado de referencia al punto x, denominado $\mathbf{x}$) de una superficie como

$$E(\bar{\mathbf{x}}) = Ae^{i\langle \mathbf{k},\mathbf{x}\rangle}e^{i\omega t},$$

donde A es la amplitud del campo eléctrico, $\mathbf{k}$ es el vector de propagación de onda y ω es la frecuencia de la oscilación del campo. Sin embargo, la ecuación anterior no incorpora el comportamiento aleatorio responsable de las variaciones de fase en las ondas, este aspecto se puede conseguir si se incorpora un factor de fase $e^{i\phi(t)}$ de la siguiente manera

$$E(\bar{\mathbf{x}}) = Ae^{i\omega t}e^{i\langle \mathbf{k},\mathbf{x}\rangle}e^{i\phi(t)},$$

donde $\phi(t)$ representa la fase de la onda en un tiempo t y puede manejarse con una variable aleatoria con distribución Gaussiana y $\langle\phi(t)\rangle = 0$ (media cero). Aunque podríamos usar dicho factor para generar un valor que pondere el efecto de la diferencia

de fase para cada rayo individual en la formación del patrón de difracción final, sabemos que el ojo humano y los sensores actuales no pueden detectar directamente las variaciones en las frecuencias altas del campo eléctrico de luz por lo que solo puede observarse la intensidad del patrón de difracción (es decir, su amplitud) [46]. Asimismo, el patrón de difracción producido por una estructura difractante depende de varios factores, pero está especialmente relacionado a la geometría de la estructura; esta estructura puede caracterizarse por medio de una función de transmitancia $T \in [0, 1]$, que indica si la luz puede pasar o reflejarse desde cierto punto y en qué medida. La teoría de difracción de Fraunhoffer dice que la amplitud de la onda difractada es proporcional al módulo cuadrado de la transformada de Fourier de la función de transmitancia $T(x)$ [35], es decir, la intensidad de la luz en un punto dado del patrón es proporcional al cuadrado de la amplitud promedio de la onda.

Adicionalmente, podemos hacer una suposición similar a la hecha en [22], donde se asume que el área de integración es lo suficientemente amplia para que un parche discretizado sea capaz de capturar las estadísticas completas del campo de altura (o en este caso, la abertura o estructura difractante) y por lo tanto se puede usar el resultado de la transformada de Fourier directamente en lugar de calcularlo como variable aleatoria y usar su valor esperado.

Durante este proceso se debe tener en cuenta que usar la transformada de Fourier para calcular el patrón de difracción produce como resultado valores de la radiancia difractada (potencia/ángulo sólido) y no de la irradiancia (potencia/área) [69], considerando que, además, sólo funciones simples tienen transformada de Fourier analítica.

Para funciones más complejas se usan soluciones o aproximaciones por medio de métodos numéricos. Por ejemplo, la DFT evalúa puntos discretos de la función, los periodos de muestreo definen la resolución con que el objeto es muestreado y la resolución de la imagen producida por la transformada de Fourier. Estos periodos no son independientes: a mayor resolución en el espacio real hay una reducción de la resolución en el espacio de Fourier, lo que puede hacer que algunos detalles del espectro angular sean difíciles de identificar. La solución a ese problema es incrementar el tamaño y el número de muestras y, por lo tanto, el tiempo de cómputo.

Bajo las consideraciones mencionadas, nuestra propuesta consiste en definir la función de transmitancia $T(x)$ como una función escalar que producirá una imagen mono-canal (o sea, con niveles de gris), en dónde el color negro indica que no hay transmisión y el blanco que hay transmisión total. Posteriormente, usamos la DFT de la imagen producida para obtener el patrón de difracción en el campo lejano, resultado que utilizaremos para definir la distribución y contribución de los rayos.

La ventaja de usar esta aproximación es que sólo se usa una transformada de Fourier, a diferencia de las aproximaciones basadas en el espectro angular para las que se requiere aplicar la transformada dos veces y además requieren la aplicación de la transformada inversa.

Una limitación de esta aproximación es que sólo funciona para patrones formados en la zona lejana (Fraunhoffer), aunque existe una aproximación similar para la zona cercana que requiere algunos cálculos extras [120].

Una razón común por la que se evita la incorporación del modelado de ondas en los renderizadores basados en rayos es que la interferencia y la difracción rompen con la conservación de energía. Aunque esa afirmación es cierta a nivel local (la intensidad de luz resultante para un rayo individual no cumple con la conservación de energía), de manera global la energía sí se conserva debido a que la energía de luz es redistribuida. Si se reduce en una región en dónde se produce una franja oscura, entonces se incrementa en otra región produciendo una franja brillante. No hay perdida ni ganancia de energía, consistente con el principio de conservación de energía [27].

6.2. Implementación patrón de difracción

Para realizar la implementación de nuestro modelo se modificó un sistema de renderizado que utiliza el método de *ray tracing* ya que es un método estándar para la generación de imágenes foto-realistas. El programa elegido para la implementación es conocido como *pbrt* (siglas para *physically based ray tracer*) [98], este programa fue realizado con la intención de proveer un marco de trabajo que facilite la enseñanza de conceptos básicos de graficación e iluminación global así como la implementación

y comparación de diversos métodos de renderizado foto-realista.

De esta forma, *pbrt* pretende usar las bases físicas como un estándar concreto para la medición de la exactitud de un programa, razón por la cual incluye una serie de módulos, clases y otras facilidades para añadir funcionalidades e implementar nuevos métodos. El programa, implementado con el lenguaje C++, también provee una forma de diseñar distintas escenas e incluye paralelismo de ciertas tareas. De hecho, varios trabajos mencionados en el Capítulo 5 fueron implementados en este programa o en programas inspirados en el mismo (por ejemplo, [20, 130]).

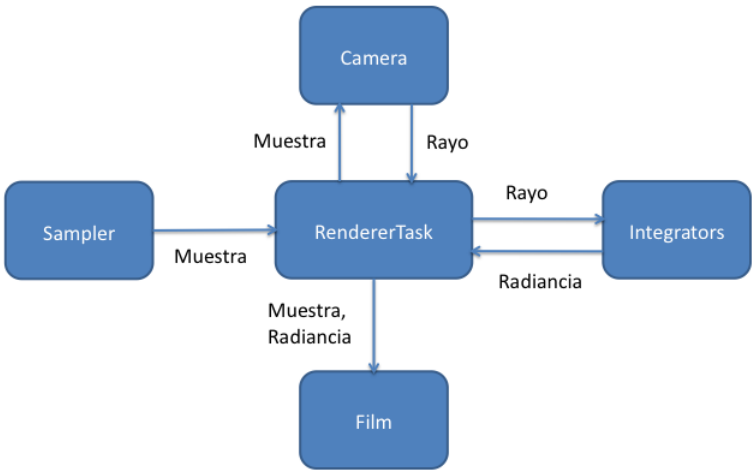


Figura 6.1: Diagrama que muestra el proceso de renderizado básico en *pbrt*, se genera una muestra que se envía a la tarea de renderizado. El ciclo de renderizado utiliza la posición de la muestra y la cámara para calcular un rayo, usando los integradores determina la cantidad de luz llegando al plano de imagen con ese rayo. Ese valor se guarda como contribución de luz para generar la imagen final.

El programa *pbrt* genera imágenes en dos fases de ejecución: analiza gramaticalmente el archivo de descripción de la escena y ejecuta el ciclo de renderizado. Durante la primera etapa se determinan las formas geométricas, las propiedades de los materiales, las posiciones y características de las luces y las cámaras, así como los parámetros de los algoritmos individuales.

La etapa de renderizado depende de la implementación de varios integradores de

luz, cada uno de ellos incluye métodos y atributos que permiten conocer la radiancia incidente a lo largo de un rayo y conocer la transmitancia (luz atenuada por la dispersión volumétrica a lo largo del rayo) de acuerdo a métodos específicos. Los integradores simulan la propagación de la luz en la escena para calcular cuanta luz llega a las posiciones de cada muestra. De igual forma, evalúan numéricamente las integrales y las ecuaciones de transporte de luz sobre los puntos de muestreo sobre la superficie.

El ciclo de renderizado utiliza la posición de la muestra y la cámara para calcular un rayo, usando los integradores determina la cantidad de luz que arriba al plano de imagen con ese rayo. Ese valor se guarda como contribución de luz para generar la imagen final. Un diagrama de este proceso y de los elementos básicos que son enviados entre clases se muestra en la Figura 6.1.

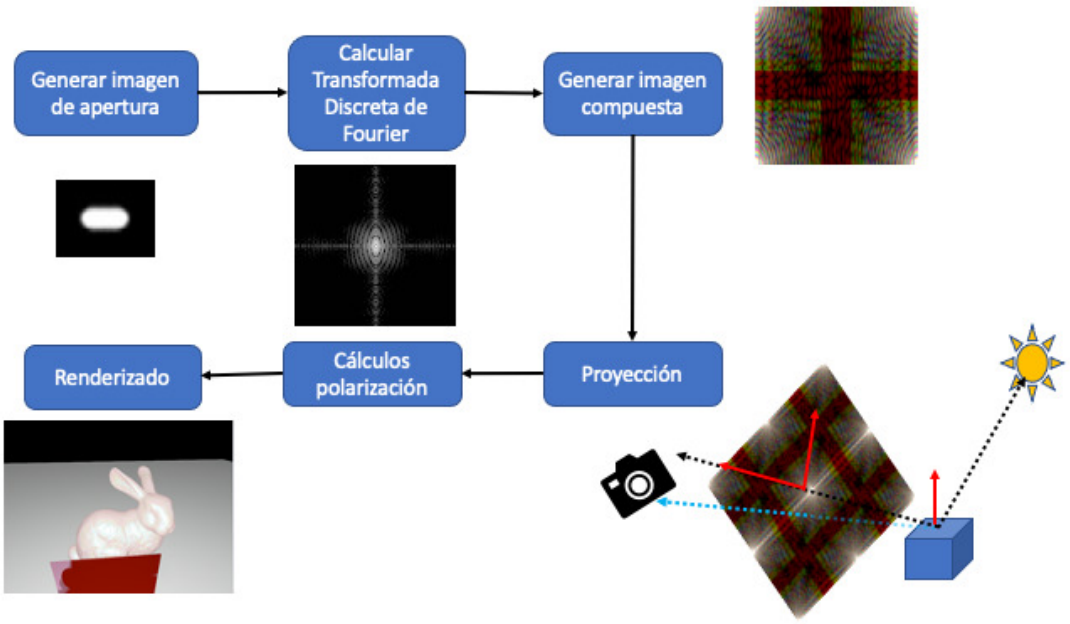


Figura 6.2: Diagrama del proceso general para realizar el renderizado del patrón de difracción.

El proceso general de la generación de efectos de onda puede observarse en el diagrama de la Figura 6.2. En primer lugar, se crea una imagen que representa la función de transmitancia de la apertura difractante que se usará para calcular la DFT, la imagen del espectro resultante se escala para cada longitud de onda y se

usa para generar una imagen a color que representa la distribución de la luz en el patrón de difracción. Posteriormente, se lee el patrón de difracción como textura en *pbrt*, donde el patrón es proyectado sobre el objeto y mezclado con las propiedades de material, donde se incluye el cálculo de la polarización para generar un renderizado final. A continuación, ahondaremos en los métodos usados para la incorporación del modelo de comportamiento de ondas al programa.

6.2.1. Creación del patrón de difracción

Como mencionamos anteriormente, el patrón de difracción se puede crear usando la DFT de la abertura difractante, este patrón se usará como distribución de la intensidad de la luz. Para crear dicho patrón primero definimos una imagen, ya sea binaria o en escala de grises, que representa la función de transmitancia $T(x)$ de la abertura, o estructura difractante, asociada a un material que caracteriza la superficie. Decidimos crear la imagen para $T(x)$ y realizar el cálculo del patrón de difracción fuera del programa de renderizado, aunque existe la posibilidad de llevarlo a cabo dentro del programa con su aumento en costo y tiempo computacional, ya que se agregaría al proceso el cálculo de la transformada de Fourier y la composición de la imagen a color que actualmente se hacen en una etapa de preprocesamiento previa.

Hasta el momento, siempre que hablamos de difracción e interferencia lo hemos hecho para una longitud de onda específica, sin embargo, es posible ver franjas de interferencia y patrones de difracción usando luz blanca. Se puede considerar que para cada longitud de onda se tiene un patrón, cada uno con espaciado ligeramente diferente. Si todos los patrones de franjas están en fase en el centro entonces éste seguirá siendo de luz blanca, las franjas incrementarán de tamaño con la longitud de onda y se verán algunos de los colores variantes dispersados. Las longitudes de onda largas de luz (rojo) son desviadas a ángulos más grandes que longitudes de onda medias (verde) que a su vez son desviadas a ángulos más grandes que longitudes de onda cortas (azul), es decir, el rojo se esparce más pero con menos intensidad.

Por lo tanto, con la finalidad de crear una imagen a color, nosotros decidimos usar

el estándar de tres canales de color asociados con las longitudes de onda rojo, verde y azul, aunque se podría seguir una metodología similar para imágenes con un muestreo espectral diferente; algo que potencialmente aumentaría el costo computacional. Una vez que se obtiene la DFT de la imagen que representa la función de transmitancia $T(x)$, ese patrón de difracción es asociado a una longitud de onda de referencia, en este caso, rojo; los patrones asociados a las longitudes de onda verdes y azules se calculan escalando el patón original utilizando la función $y \approx D\frac{\mu\lambda}{d}$, que describe la distancia entre la franja central y la franja de difracción de orden μ cuando se tiene una rejilla de difracción y dónde y es la distancia entre franjas, D la distancia al plano de proyección, d el tamaño de la apertura, λ la longitud de onda y μ el orden de difracción. Para los experimentos se eligieron valores relativamente arbitrarios para D, d y μ asociados a las λ específicas y el valor de y resultante se usó para escalar el patrón. Para D se eligió una distancia de un metro, para d se eligió el largo mínimo estándar de una perforación de CD (equivalente a 833 nm), con $\mu = 1$ y los valores de 645 nm, 526 nm y 444 nm para λ_r, λ_g y λ_b , respectivamente. Los valores obtenidos de y para cada λ son usados como referencia para obtener un valor de escalamiento para cada longitud de onda.

Finalmente, se crea una imagen final a color compuesta de las tres bandas resultantes de escalar el espectro de Fourier de la abertura, cada imagen escalada se asocia a la longitud de onda que se utilizó para calcular el factor de escalamiento. Es importante hacer notar que el patrón se almacena reacomodando la frecuencia cero y frecuencias bajas hacia las esquinas en lugar de que se encuentren en el centro de la imagen; un ejemplo del patrón de difracción resultante puede verse en la Figura 6.3. La razón para este acomodo es aprovechar el método de generación de coordenadas de textura que se usará posteriormente para colocar el patrón sobre el objeto hecho de un material difractante, de esta forma el centro del sistema coordenado usado para la colocación de textura coincide con el centro del patrón.

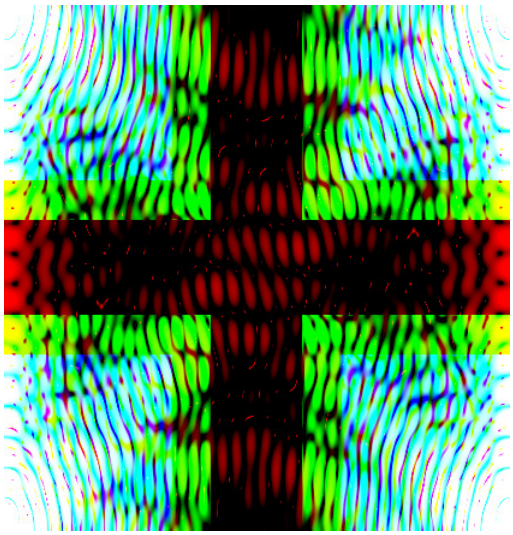


Figura 6.3: Patrón de difracción resultado de la composición de la DFT de la función de transmitancia escalada para tres diferentes longitudes de onda, rojo, verde y azul. Cada componente fue acomodada de forma que las frecuencias bajas de cada una coincidan en las esquinas de la imagen final y no en el centro.

6.2.2. Colocación del patrón de difracción

En *pbrt* se lee el patrón de difracción como una textura (un arreglo de valores escalares o vectoriales) asociada a un material. El patrón se usa de manera similar a una máscara que modifica la intensidad de cada rayo que interseca con él durante el proceso de integración de la luz.

Para realizar dicha tarea se creó una subrutina (un método en C++) para manejar efectos de difracción que define una BRDF, existe una subrutina específica para cada tipo de material. Esta BRDF se encarga de definir la distribución de la luz para un par $(\omega_i,\ \omega_o)$ de direcciones, donde ω_i es la dirección de la luz (el vector de entrada) y ω_o es la dirección de observación (el vector de salida). Estos vectores son transformados a las coordenadas del sistema local del polígono que está siendo iluminado, lo cual facilita los cálculos posteriores. El vector transformado ω_i se utiliza para calcular su vector de reflexión especular $\mathbf{r}$ con respecto a la normal de la superficie en el punto que está siendo iluminado de forma que $\mathbf{r} = (-x_{\omega_\mathbf{i}}, -y_{\omega_i}, z_{\omega_i})$.

El vector de reflexión es necesario porque según [50], la reflectancia puede separarse en dos lóbulos diferentes, con el pico de difracción en dirección de la reflexión especular

(correspondiente a la función Delta de Dirac) rodeada por un halo de luz dispersada.

Posteriormente, se utiliza el vector de reflexión $\mathbf{r}$ para crear un nuevo sistema coordenado. Este sistema se construye con un vector perpendicular $\mathbf{v_2}$, que se genera cambiando uno de los componentes de $\mathbf{r}$ a 0, intercambiando los dos componentes restantes y negando uno de ellos; el producto vectorial entre $\mathbf{r}$ y $\mathbf{v}_2$ se usa para generar el vector $\mathbf{v_3}$. Además, se crea la siguiente matriz de transformación afín para llevar puntos del sistema local del polígono a este nuevo sistema coordenado

$$\begin{bmatrix} x_{\mathbf{v}_2} & y_{\mathbf{v}_2} & z_{\mathbf{v}_2} & -\langle \mathbf{r}, \mathbf{v}_2 \rangle \\ x_{\mathbf{v}_3} & y_{\mathbf{v}_3} & z_{\mathbf{v}_3} & -\langle \mathbf{r}, \mathbf{v}_3 \rangle \\ x_{\mathbf{r}} & y_{\mathbf{r}} & z_{\mathbf{r}} & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

La idea es hacer que el patrón se encuentre centrado en el plano formado por los vectores $\mathbf{v}_2$ y $\mathbf{v}_3$ del sistema coordenado, lo cual se asegura por la forma en que fue almacenada la imagen. Posteriormente, se transforma el vector en dirección al observador ω_o al sistema coordenado de reflexión para proyectarlo sobre el plano donde se encuentra el patrón de difracción y obtener las coordenadas de textura correspondientes al elemento de textura (*texel*) en dicho punto. Por último, el valor del *texel* es ponderado usando una función trapezoidal para disminuir la intensidad del patrón conforme nos alejamos del centro. De esta manera, nos aseguramos de que el pico del patrón corresponde a la dirección especular.

6.3. Implementación polarización

La mayoría de trabajos que implementan los efectos de difracción omiten, por simplicidad, la polarización de los rayos de luz. Sin embargo, la polarización es relevante en la formación de los patrones de difracción porque la orientación de la polarización puede cancelar la interferencia si se trata de dos ondas con polarización ortogonal entre ellas, además de generar sus propios efectos distintivos.

En el modelo con el que trabajamos en este capítulo, la polarización se puede

manejar por separado usando los vectores de Stokes y las matrices de Mueller descritas en la Subsección 5.2.5, ya que no haremos un seguimiento momento a momento de la fase de la onda de luz.

Para implementar polarización en un programa de *ray tracing* o *path tracing* seguimos las modificaciones generales propuestas por [127]. Por lo tanto, las estructuras de datos usadas para la luz y los cálculos de la atenuación se reemplazaron con estructuras de datos que contienen vectores de Stokes y matrices de Mueller. También se incluyeron optimizaciones sencillas, tales como una bandera que indica si la instancia en cuestión es polarizadora o no. Esta información puede aumentar la velocidad del programa al evitar multiplicaciones innecesarias de matrices durante la concatenación de filtros. Se implementó el seguimiento de marcos de referencia para todos los caminos trazados en los integradores. Se cambió la BRDF para generar y evaluar la matriz de Mueller correcta durante el proceso de renderizado.

La idea es usar los rayos generados por el programa de *ray tracing* como vectores de propagación de la onda, la intensidad de la luz modificada por el patrón de difracción y la polarización sólo en las intersecciones de los rayos de luz con otros elementos en la escena.

6.3.1. Marcos de referencia

Para implementar la polarización se crearon clases *ad hoc* (en C++) y estructuras de datos para el manejo de los vectores de Stokes, las matrices de Mueller y las operaciones entre ellos. De igual forma, se agregó una clase *ad hoc* para el manejo del espectro de luz que incluye un vector de Stokes por cada canal, un vector para almacenar el marco de referencia y subrutinas (en forma de métodos de C++) para realizar operaciones con ellos. También se realizaron los cambios en una implementación del integrador basada en el método de [123], ya que es la clase encargada de calcular la radiancia y la atenuación y, por lo tanto, de modificar los estados de polarización.

Se modificó el programa *pbrt*, la clase de C++ a cargo de la *BSDF*, para almacenar las matrices de Mueller correspondientes (una por cada longitud de onda y definidas

en implementaciones polarizadas de las clases de C++ en *pbrt* que definen el material) y poder tener acceso a ellas, así como un vector para almacenar el marco de referencia.

La parte más complicada de la implementación del proceso de polarización es el manejo de los marcos de referencia. El vector de Stokes resultante de una operación indica el estado de polarización con respecto al plano de incidencia, definido por la dirección de incidencia y el vector de reflexión de la luz; el vector de referencia ayuda a definir el plano y, por lo tanto, cambia en cada reflexión que se calcula.

Si el vector de Stokes incidente es $\bar{\mathbf{s}}_i$ y el eje asociado al marco de referencia es $\bar{\mathbf{s}}_\mathbf{i}$, entonces el vector de Stokes reflejado $\bar{\mathbf{s}}_r$ se obtiene como

$$\bar{\mathbf{s}}_r = \mathbf{M}_r \mathbf{M}_{rot}(\theta) \bar{\mathbf{s}}_i,$$

donde $\mathbf{M}_r$ es la matriz de Mueller para la reflexión y $\mathbf{M}_r(\theta)$ es la matriz de rotación para el ángulo θ entre el vector de marco de referencia incidente y el marco de referencia de la normal. La matriz de rotación está definida de la siguiente forma

$$\mathbf{M}_{rot}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\theta & \sin 2\theta & 0 \\ 0 & -\sin 2\theta & \cos 2\theta & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

El eje del marco de referencia de la normal se calcula como [105]

$$\bar{\mathbf{s}}_i^{\vec{\mathbf{n}}} = \frac{\mathbf{k} \times \vec{\mathbf{n}}}{|\mathbf{k} \times \vec{\mathbf{n}}|},$$

en dónde $\mathbf{k}$ es la dirección de propagación y $\vec{\mathbf{n}}$ es la normal a la superficie en el punto de intersección. El marco de referencia del vector reflejado se calcula de forma similar.

Después de rotar el vector de referencia y multiplicarlo por la matriz de Mueller, el vector resultante será el nuevo vector de referencia, de esta manera podemos propagar los cambios de polarización en la onda. Aunque este método no funciona en métodos que utilizan simetría y bidireccionalidad para calcular los caminos de los rayos, pero

ya se han desarrollado métodos para sortear dicha dificultad [55].

Capítulo 7

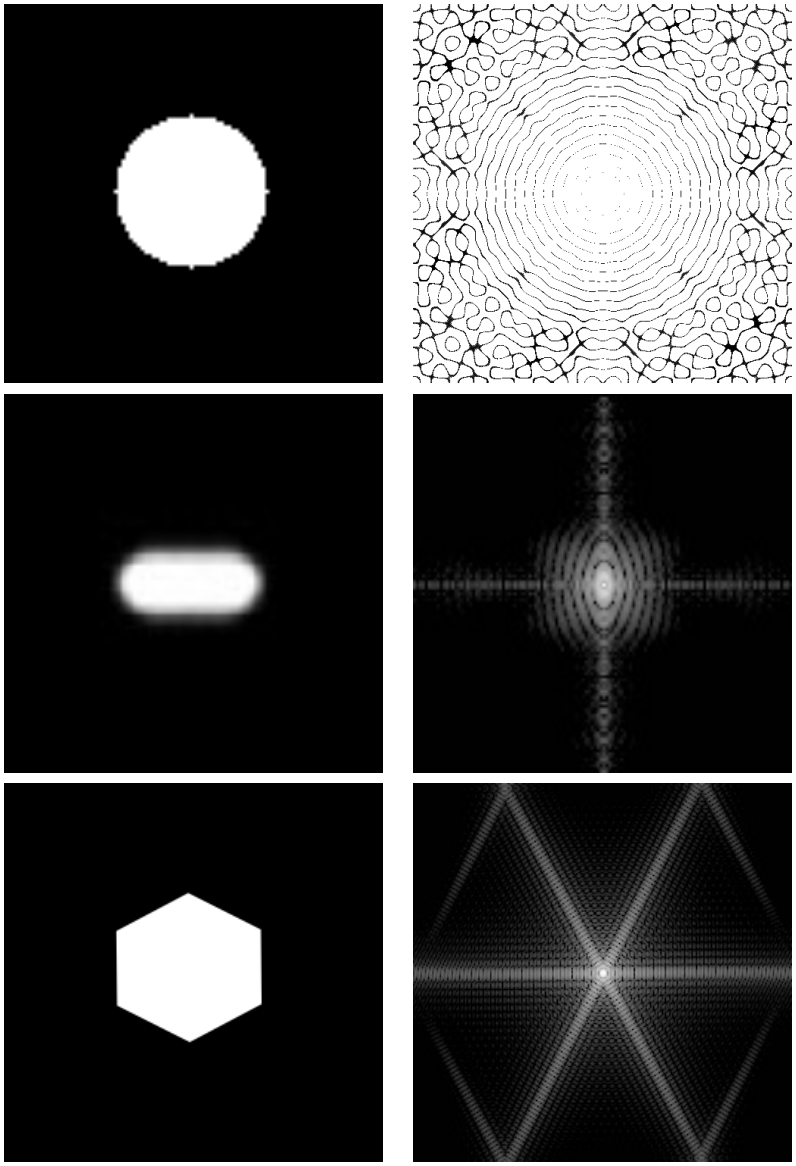
Resultados Efectos de Onda

En este capítulo mostraremos los resultados de la implementación del modelo para la reproducción de efectos de onda descrito en el capítulo anterior. Primero, mostraremos los resultados obtenidos con la implementación de patrones de difracción y, posteriormente, los resultados obtenidos con la implementación de polarización, ambos en el programa de renderizado *pbrt*, versión 2, basado en emisión y rastreo de rayos [98]. Como se mencionó anteriormente, la generación de los patrones de difracción se llevó a cabo por fuera del sistema de renderizado y la implementación del modelo y la generación de las imágenes se realizó en el programa *pbrt*. Las modificaciones se hicieron sólo en algunos tipos de materiales y en el integrador de luz basado en el original de Whitted.

7.1. Difracción

7.1.1. Patrones de difracción

Como se describió en la Subsección 6.2.1 del capítulo anterior, para la implementación que genera efectos de difracción es necesario obtener un patrón de difracción de una abertura u objeto difractante. Como primer paso para llevar a cabo esa tarea, definimos imágenes en blanco y negro que representan las funciones de transmitancia de la abertura, estás imágenes fueron generadas siguiendo las descripciones de aberturas básicas. Las aberturas usadas pueden observarse en la columna izquierda de la



(izquierda) (derecha)

Figura 7.1: (izquierda) Aberturas difractantes, (arriba) abertura circular, (en medio) hueco de un CD, (abajo) abertura hexagonal, y (derecha) sus espectros correspondientes obtenidos por medio de la DFT.

Figura 7.1, abertura circular (arriba), hueco de un disco compacto o CD (en medio), abertura hexagonal (abajo).

Una vez definida la abertura es necesario calcular su espectro, para lo cuál usamos tanto la implementación de la DFT de MATLAB® [53] como la implementación de la DFT del programa *ImageJ* [107]. La razón para usar ambas implementaciones consiste en que generan resultados ligeramente diferentes debido a la elección de la función para visualizar el resultado (por ejemplo, logaritmo del valor absoluto del espectro, ponderado con diferentes valores). La transformada de Fourier correspondiente a cada abertura puede observarse en la columna derecha de la Figura 7.1. Una vez obtenido el espectro de la imagen de la función de transmitancia, el resultado se escaló siguiendo la metodología descrita en la Subsección 6.2.1; para ello, definimos $D = 1$ m, $d = 833$ nm, $\mu = 1$, $\lambda_r = 645$ nm, $\lambda_g = 526$ nm y $\lambda_b = 444$ nm para las diferentes longitudes de onda en la función $y \approx D\frac{\mu\lambda}{d}$. Posteriormente, podemos tomar los valores de y para cada longitud de onda (o color) y usarlos para calcular los valores de escalamiento apropiados para cada longitud de onda. Para ello, tomamos el valor de y definido para λ_r como referente del tamaño del patrón original y los valores y correspondientes a λ_g y λ_b se usan para calcular un escalamiento que sea proporcional al espacio de separación esperado para ambas longitudes de onda con respecto a λ_r.

Todos los patrones de difracción a color se generaron usando un *script* en MATLAB® y fueron almacenados en formato TGA (un formato de rasterización gráfica creado por Truevision Inc. y diseñado para proveer soporte de color verdadero y de alta gama en las tarjetas *Truevision Advanced Raster Adapter*, TARGA, y para computadoras compatibles con IBM-PCs [9]); adicionalmente, fueron separados en cuadrantes para aprovechar los métodos de mapeo de texturas en *pbrt*. Los patrones generados pueden observarse en la Figura 7.2, las diferencias entre patrones de la misma abertura se deben a que en algunos casos se agregó relleno a la imagen para que la DFT tuviera mayor resolución, ver la abertura circular en la Figura 7.2(a) y la Figura 7.2(b), se sacó el logaritmo al módulo del cuadrado de la transformada de Fourier correspondiente y se pondero por un valor real, ver el hueco de CD en la Figura 7.2(d) y la Figura 7.2(e), o se utilizó la implementación de DFT del programa *ImageJ*, ver los

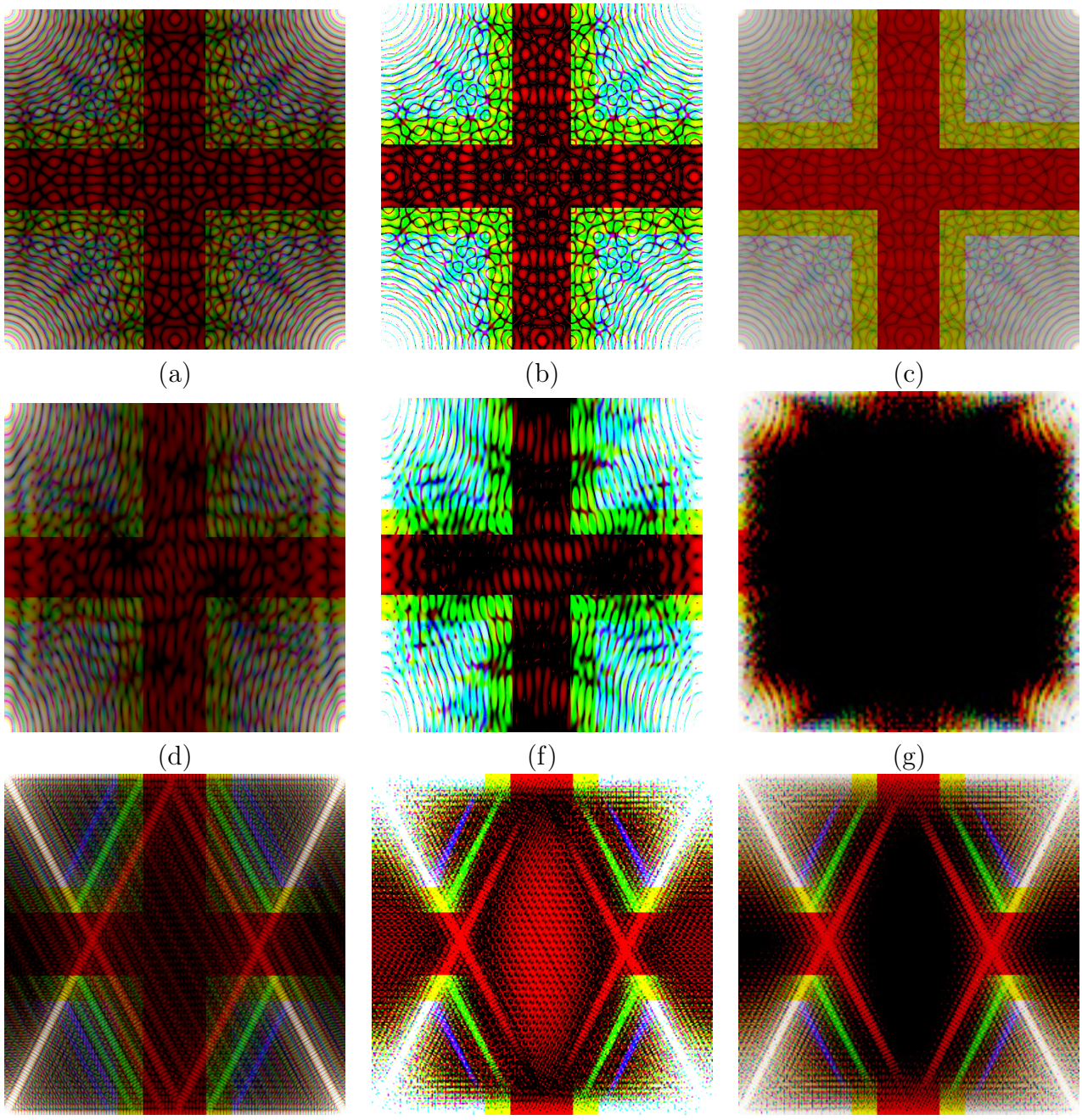


Figura 7.2: Patrones de difracción de cada tipo de abertura de la Figura 7.1, cada imagen fue compuesta a partir del escalamiento de la DFT generada por diferentes programas (MATLAB® e *ImageJ*) para cada banda de color, (a), (b) y (c) apertura circular, (d), (e) y (f) hueco de CD y (g), (h) e (i) apertura hexagonal.

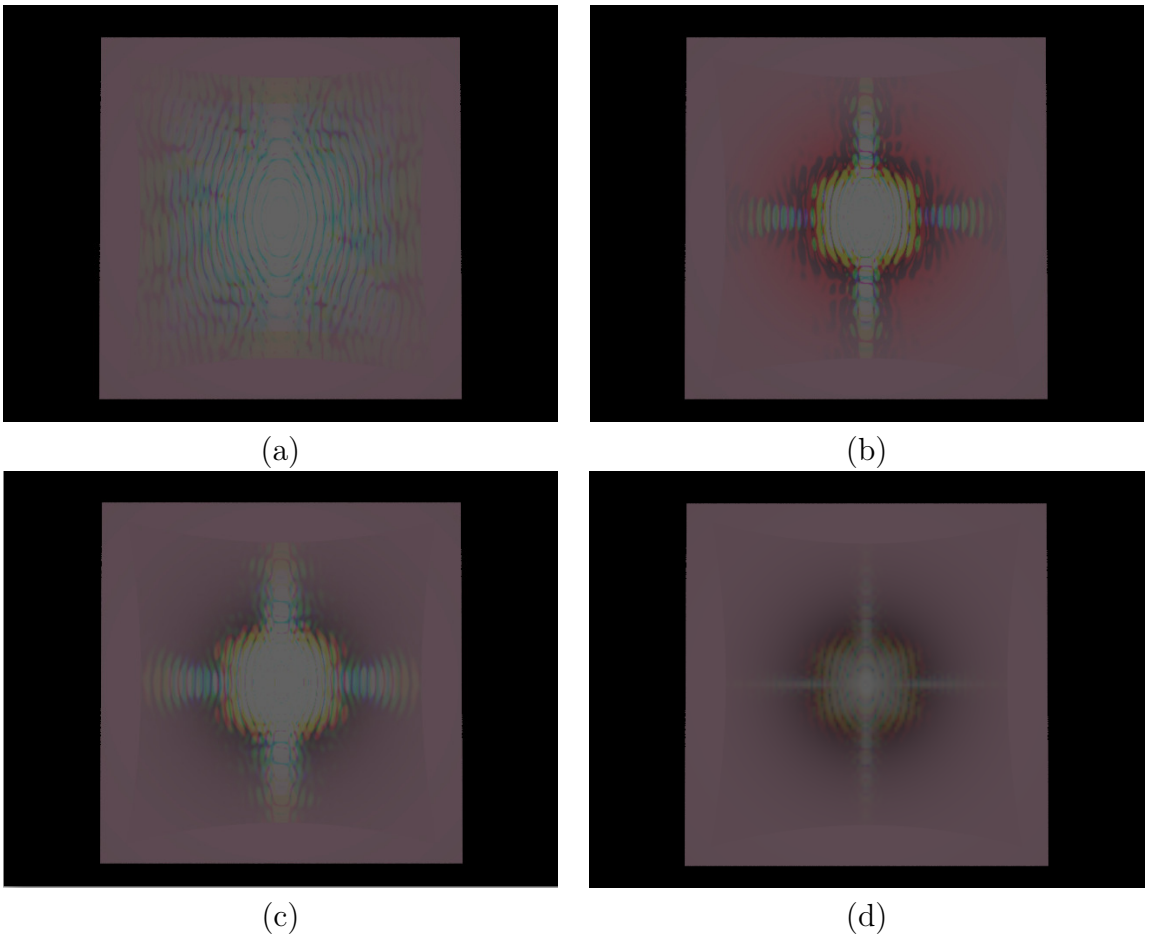
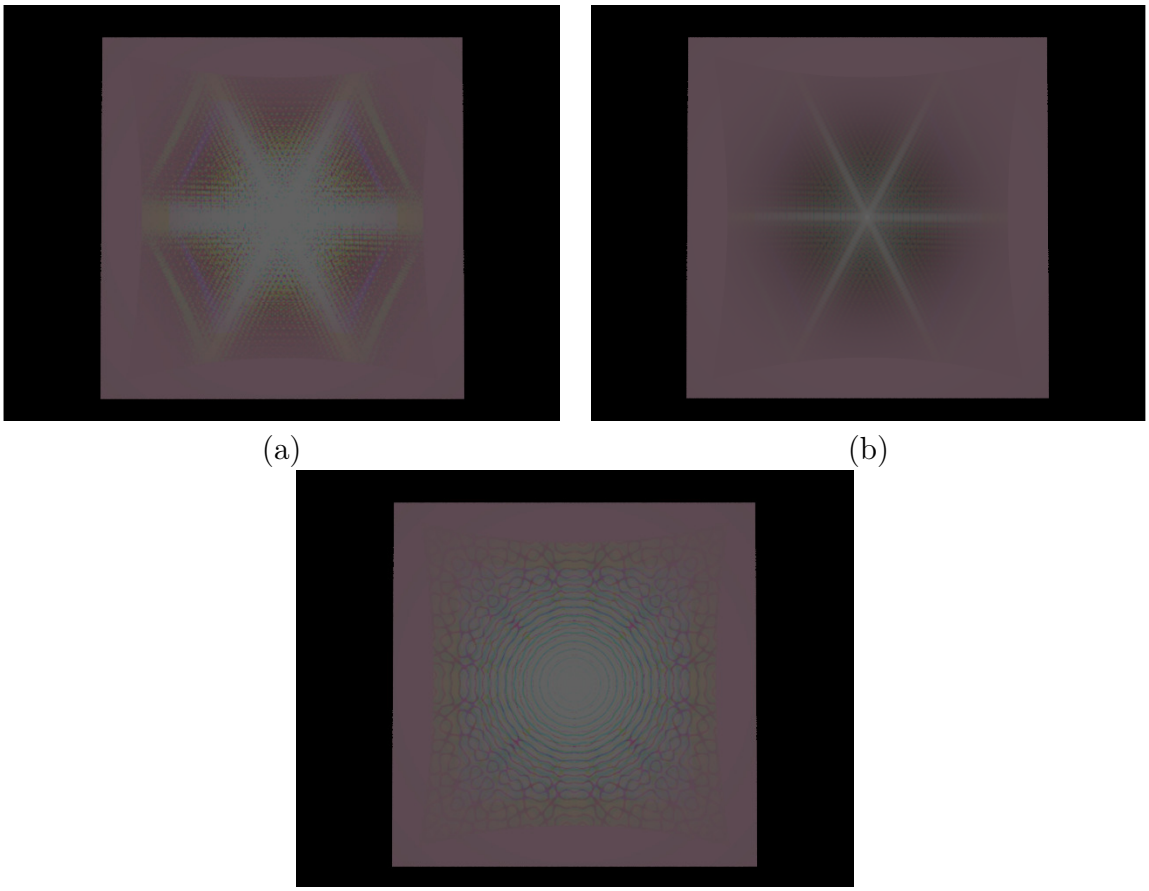


Figura 7.3: Diferentes patrones de difracción correspondientes al hueco de un CD usando (a) el módulo del espectro calculado con MATLAB®, (b) el módulo al cuadrado o (c) el logaritmo del módulo al cuadrado (d) el módulo cuadrado del espectro calculado con *ImageJ*.



(a) (b)

(c)

Figura 7.4: Patrones de difracción correspondientes para (arriba) una apertura hexagonal y (c) una circular. Para las imágenes (a) y (c) se usa el módulo del espectro calculado con MATLAB® y para (b) el módulo del espectro calculado con *ImageJ*.

patrones para el CD y la abertura hexagonal, en la última columna de la Figura 7.2.

La validez de los patrones de difracción ya está verificada por la relación de la transformada de Fourier y el patrón de difracción, recordando que hay ciertas limitaciones a su uso.

7.1.2. Ejemplos visuales

En esta subsección presentamos la salida de la implementación del modelo presentado en la Sección 6.2 en el *software* de renderizado *pbrt*, usando los patrones mostrados en la sección anterior. La imagen del patrón de difracción deseado es leída por *pbrt* como una textura que describe una propiedad asociada a un material. Los

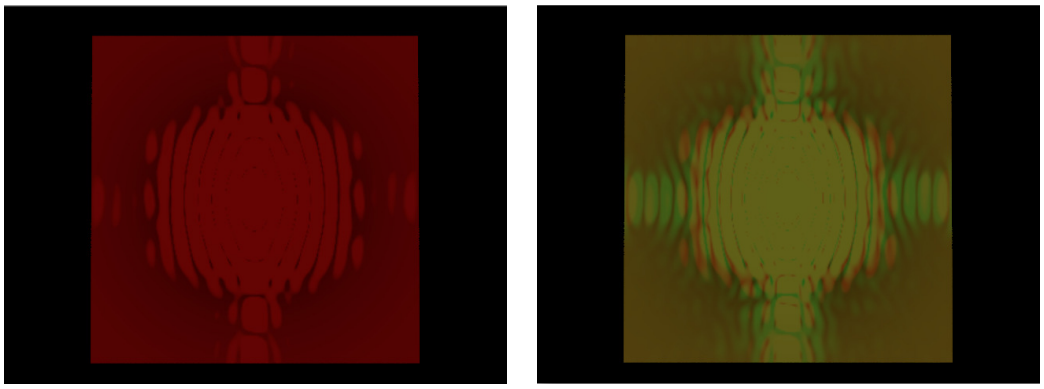


Figura 7.5: Cambios en el renderizado si usamos luz monocromática roja (izquierda) o luz con componentes roja y verde (derecha).

primeros conjuntos de imágenes de las Figuras 7.3 y 7.4 sólo muestran los diferentes patrones de difracción colocados sobre un plano, con una luz puntual que se encuentra colocada directamente en el centro de la escena, en la misma posición de la cámara. Puede observarse como el patrón se va difuminando hacia las orillas y como se va mezclando con el color del material mate del plano. En el caso de la Figura 7.3, los patrones son los diferentes resultados que se obtuvieron del hueco de CD, variando el programa para obtener la DFT y usando el valor directo (el módulo del espectro), el módulo al cuadrado o el logaritmo del módulo al cuadrado. La Figura 7.4 muestra los patrones correspondientes para la apertura hexagonal y la circular.

No se realizó una comparación con alguno de los métodos mencionados en el Capítulo 5 principalmente por dos razones, la primera, la dificultad de implementar dichos métodos y la segunda es que no sería una comparación justa debido a que nuestro método hace una aproximación a los fenómenos de difracción para las condiciones específicas de la escena, mientras otros métodos realizan un cálculo más exacto del fenómeno. Por último, no contamos con métodos para medir o generar imágenes reales que puedan compararse con los renderizados obtenidos.

En la Figura 7.5 mostramos el comportamiento del patrón de difracción cuando utilizamos luz monocromática o una luz con prevalencia de un tinte en vez de luz blanca. Una limitación de esta aproximación es que en estos casos existen zonas iluminadas que no deberían estarlo.

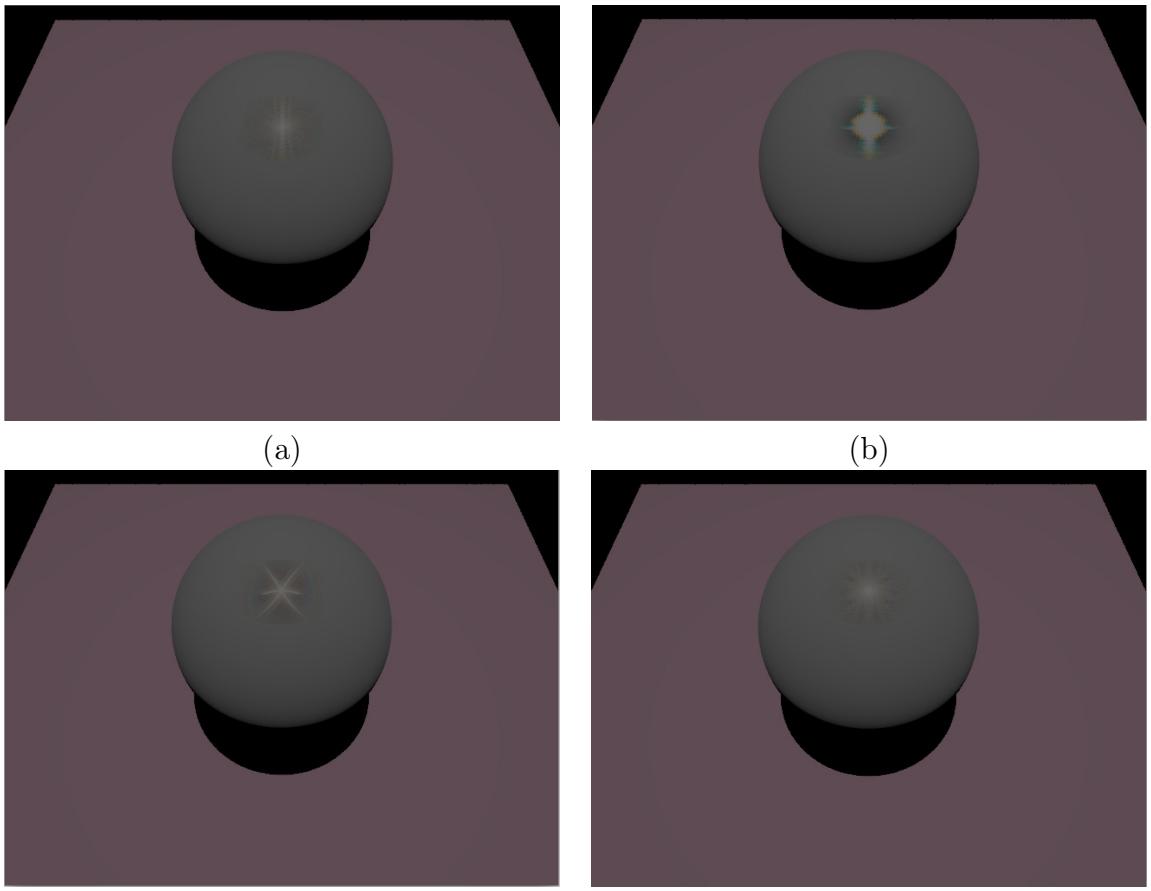
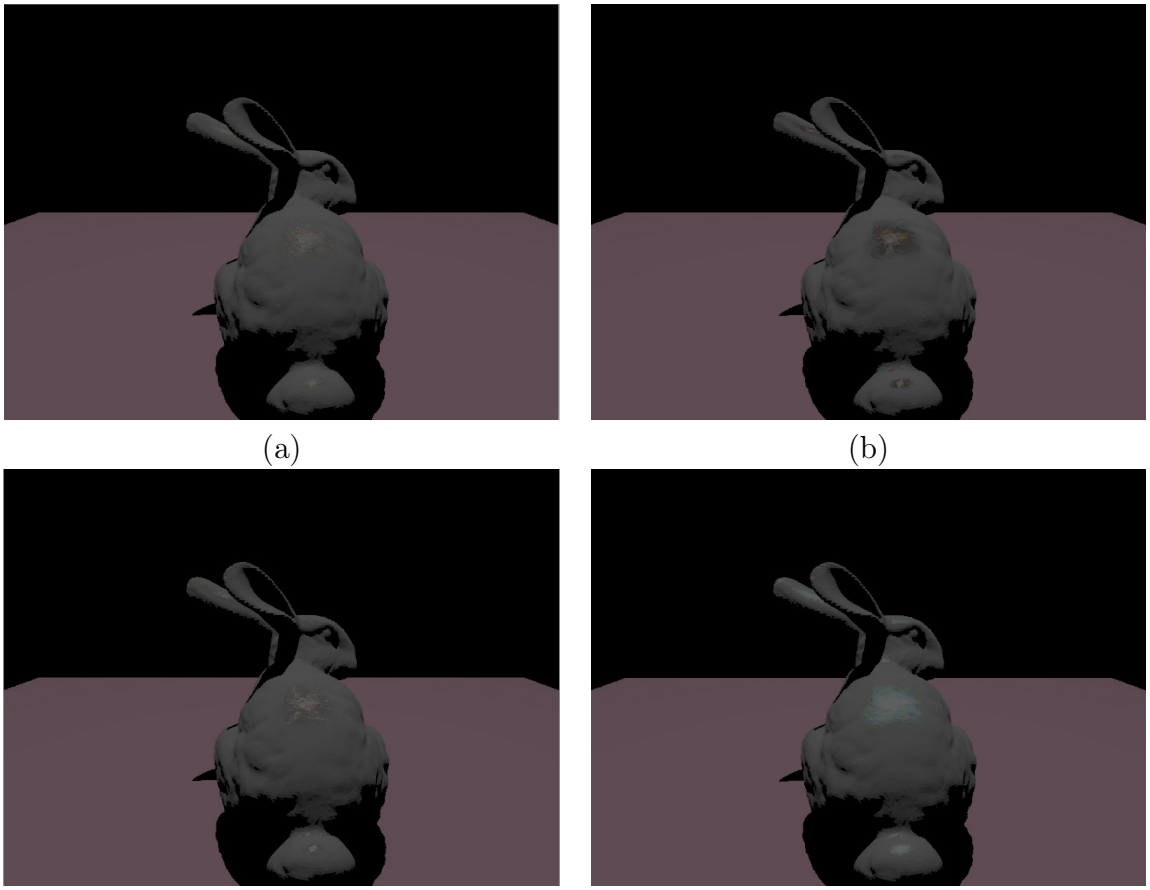


Figura 7.6: Patrón de difracción de (a) y (b) hueco de un CD, (c) una abertura hexagonal y (d) una abertura circular proyectados sobre una esfera. El patrón de difracción del hueco de un CD se proyecta usando (a) su módulo calculado con MATLAB® y (b) el logaritmo del módulo calculado con *ImageJ*.

También realizamos experimentos para ver cómo se proyectaban los distintos patrones de difracción sobre objetos un poco más complejos, una esfera (véase la Figura 7.6) y el modelo de un conejo (véase la Figura 7.8). Puede notarse que la distancia del objeto a la fuente de luz y la complejidad de la geometría del objeto generan efectos más interesantes ya que la proyección del patrón de difracción empieza a mezclarse entre los polígonos que se encuentran cercanos al punto de reflexión especular y eso genera otras variaciones de color, un acercamiento al modelo donde se aprecian mejor los cambios de coloración se puede ver en la Figura 7.7.

Adicionalmente, en la Figura 7.9 se puede observar el cambio en el patrón de difracción en posición, tamaño y variación de colores cuando se cambia la posición de

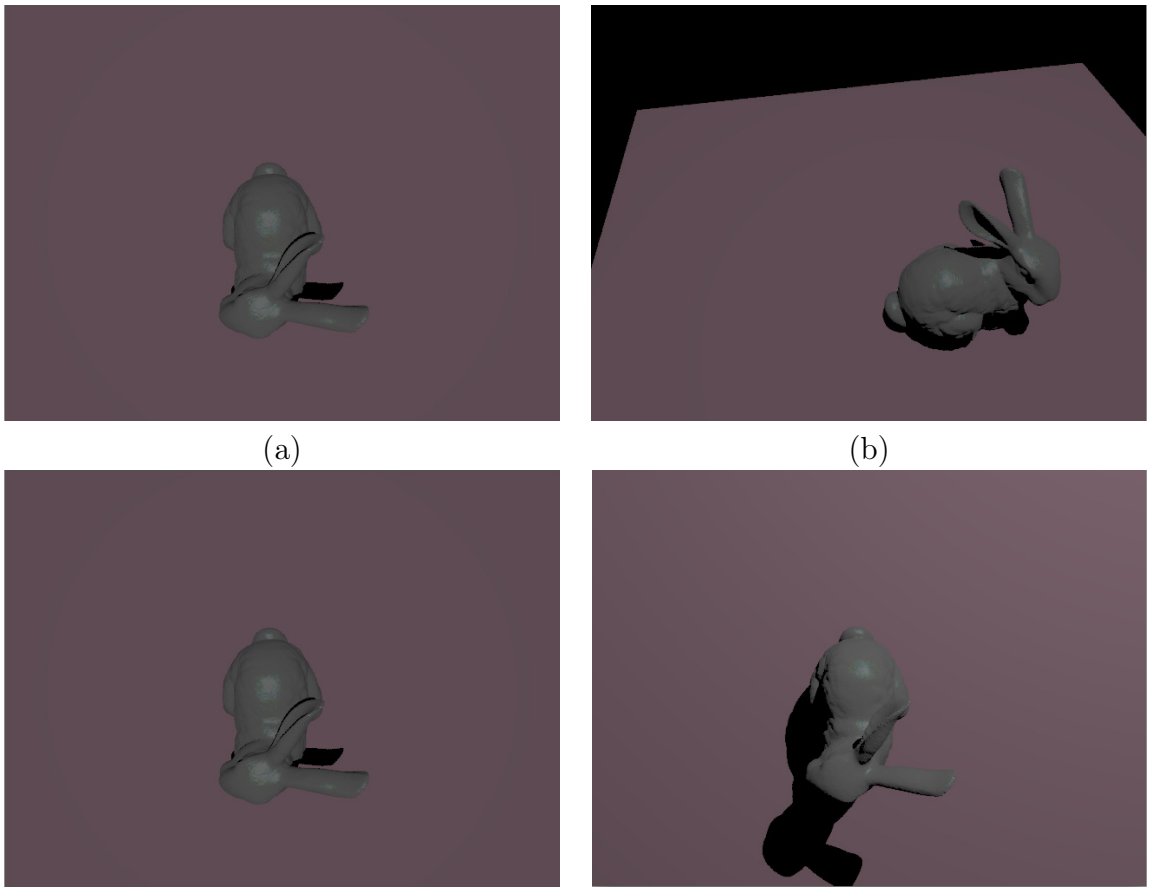
Figura 7.7: Acercamiento al patrón de difracción, donde puede apreciarse el cambio de colores.



(a) (b)

(c) (d)

Figura 7.8: Patrón de difracción de (a) y (b) hueco de un CD, (c) una abertura hexagonal y (d) una abertura circular proyectados sobre el lomo de un conejo. El patrón de difracción del hueco de un CD se proyecta (a) su módulo calculado con MATLAB® y (b) logaritmo del módulo calculado con *ImageJ*.



(a) (b)

(c) (d)

Figura 7.9: Cambios en el patrón de difracción si (a-b) se mueve la posición de la cámara y (c-d) se mueve la posición de la luz. Se puede notar como la proyección del patrón de difracción de una abertura circular cambia en tamaño, posición y en gama de colores.

la cámara, o bien, la posición de la luz.

Otra prueba que realizamos fue escalar las coordenadas del vector en dirección del observador, lo cual equivale a hacer un escalamiento al patrón de difracción antes de elegir la coordenada de textura adecuada, con ello se puede variar el tamaño de la proyección del patrón de difracción y del área que abarca, ejemplos de los resultados pueden verse en la Figura 7.10.

Finalmente, realizamos un experimento en que se utilizan las luces de área (luces que están asociadas a una forma geométrica y por lo tanto requieren ser muestreadas para determinar su dirección) y no sólo luces puntuales; en la Figura 7.11 podemos ver el resultado de usar tanto sólo una luz de área (donde los efectos del muestreo de

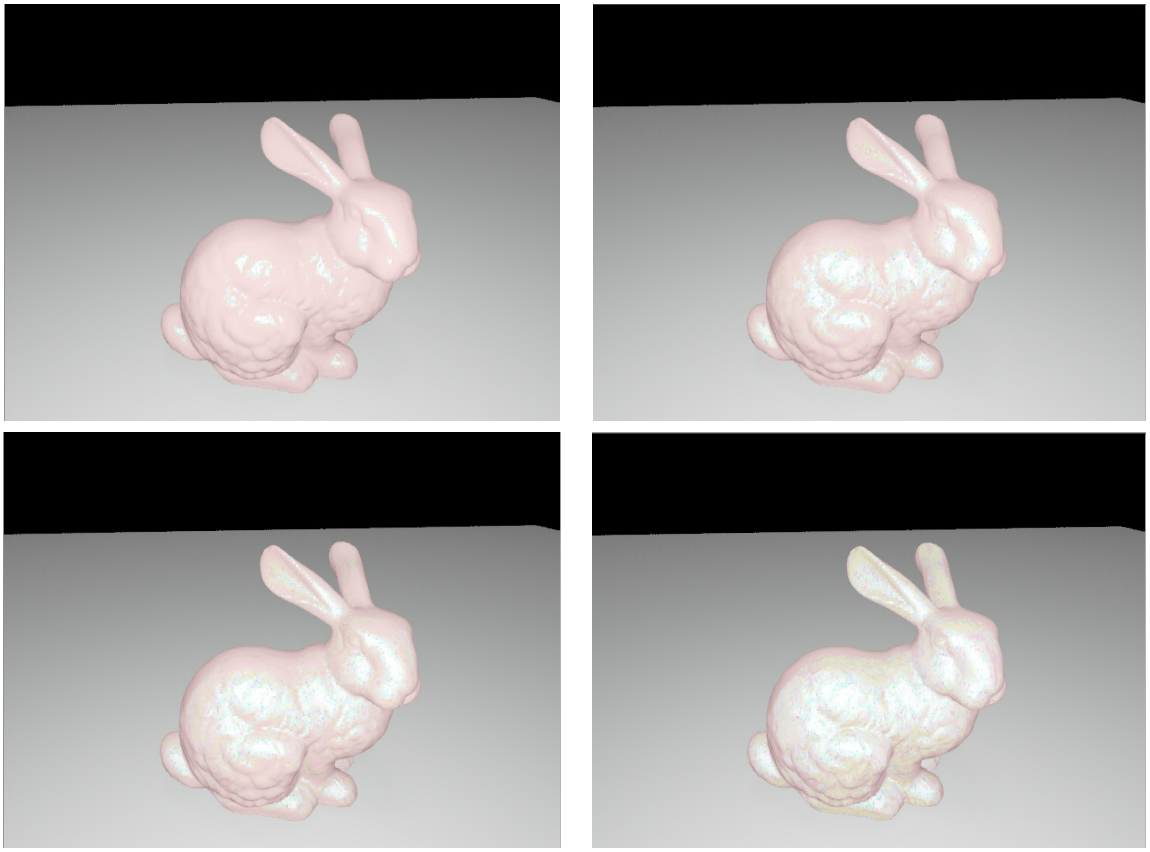
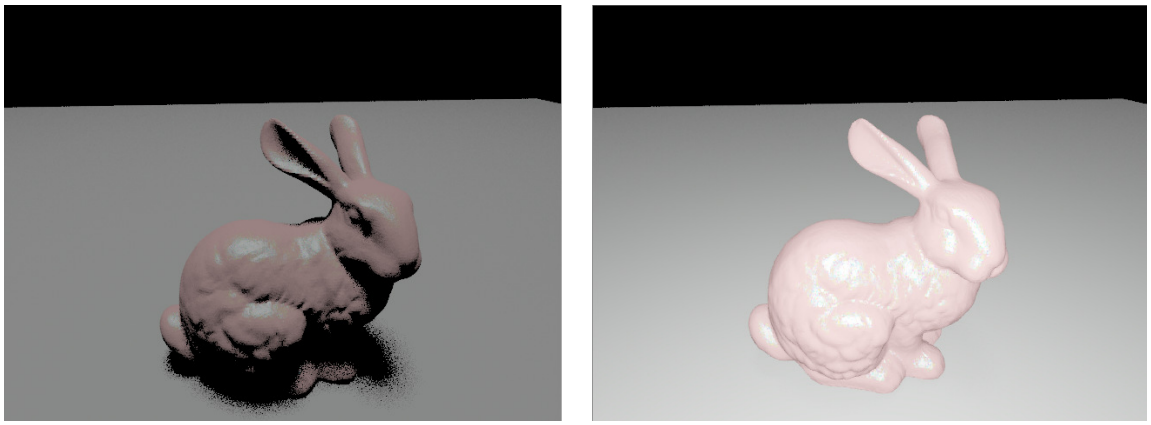


Figura 7.10: Cambios en el renderizado si cambiamos el escalamiento del patrón de difracción para proyectar las coordenadas en el objeto, entre más grande es el patrón más grande es el área de brillo del objeto.



(a) (b)

(c)

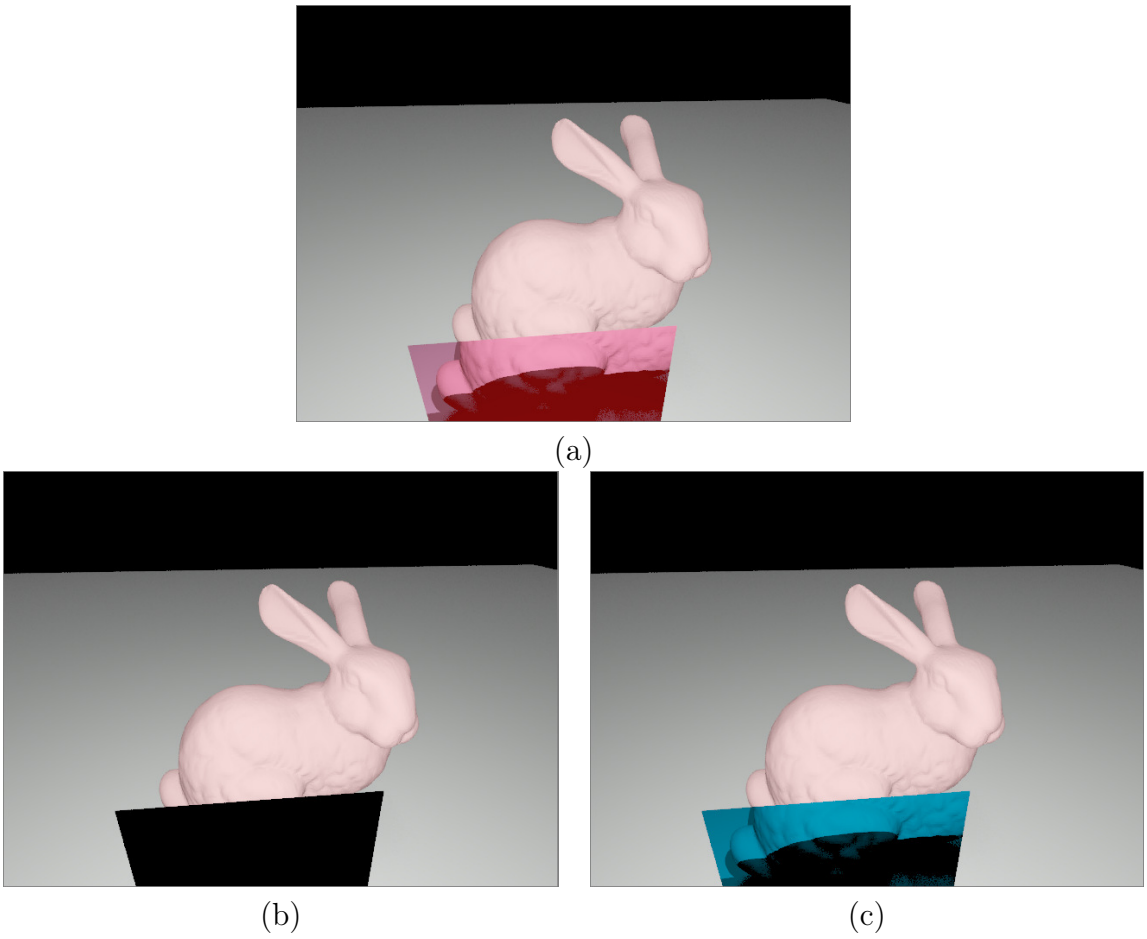
Figura 7.11: Cambios en el renderizado si usamos (a) luz de área, (b) luz puntual o (c) ambas luces.

la luz de área son visibles), sólo una luz puntual y la combinación de ambas luces, nótese que a pesar de la combinación de luces "gana" la aportación de la luz puntual, lo que mejora la apreciación final de la imagen al disminuir los artefactos debidos a un muestreo deficiente. Es importante mencionar que el programa *pbrt* utiliza muestreo e integración Monte Carlo para calcular las integrales de luz sobre la superficie, lo cual implica un costo computacional extra cuando se incluyen luces de área.

En términos generales, la implementación del modelo de difracción no aumento significativamente el tiempo de cómputo, exceptuando aquellas imágenes en dónde el área del patrón era muy amplia, ver la Figura 7.3.

7.2. Polarización

Como se mencionó en la Sección 6.3, la implementación de polarización se realizó modificando la forma en que se lleva a cabo la integración de la luz, las estructuras de datos que manejan el espectro de luz y los materiales.



(a)

(b) (c)

Figura 7.12: La imagen en (a) fue generada sin cálculos de polarización, las imágenes restantes fueron generadas con luz polarizada linealmente a 0^{o} y un elemento transparente con un filtro a 90^{o}. En la imagen en (c) se filtran todas las longitudes de onda mientras que en la imagen de (b) se filtra únicamente la longitud de onda roja.

Como resultado de la implementación del manejo de la polarización se generaron las imágenes de la Figura 7.12, para las cuales se definió una luz puntual con polarización lineal a 0^{o} y un material transparente pero con coloración que funciona como un filtro que sólo deja pasar luz linealmente polarizada a 90^{o} con dos variaciones. Se puede ver en la imagen de la Figura 7.12(b) como el material transparente filtra todas

las longitudes de onda y, por lo tanto, no pasa luz, ya que no hay luz polarizada a 90^{o}, por otra parte, en la imagen de la Figura 7.12(c) sólo se filtra la longitud de onda roja y, por lo tanto, las longitudes de onda correspondientes al verde y al azul pasan sin problema.

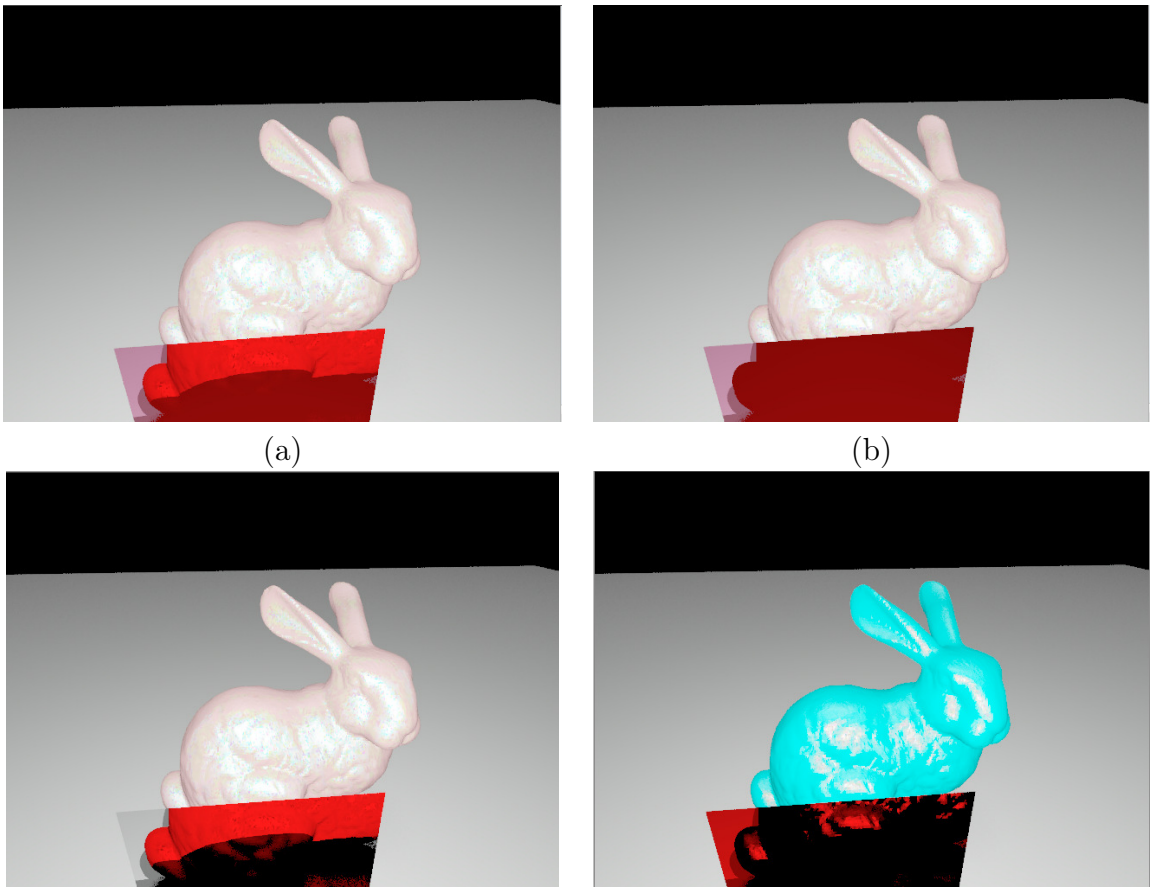
En la Figura 7.13 se pueden observar renderizados en los que se utilizó un material difractante para la superficie del conejo. Para la Figura 7.13(a) se utilizó una luz que no está polarizada y el conejo tampoco tiene la propiedad de polarizar la luz, aunque el filtro sigue filtrando ondas a 90^{o}, pero está vez funciona para luz azul y verde dejando pasar sólo la luz roja. La imagen de la Figura 7.13(b) fue generada con el conejo modelado con un material que además es polarizador lineal a 180^{o}, por lo tanto, nada de la luz que es reflejada de él puede pasar por el filtro (se percibe color, aunque no es negro, por la luz que no es reflejada del objeto). La imagen de la Figura 7.13(c) representa un filtro a 45^{o} transparente y sin color para luces verde y azul, se utiliza una luz puntual polarizada a 45^{o} y el conejo es de material que polariza a 180^{o}; en este caso, es más fácil observar al filtro en acción porque no afecta los colores del resto de la escena. Por último, la imagen de la Figura 7.13(d) presenta al conejo cuyo color no contiene componente roja, el filtro solo deja pasar la longitud de onda roja, por lo tanto, solo esa longitud de onda del patrón de difracción es visible a través del filtro.

Cuando se utiliza el patrón de difracción junto a los cálculos para incluir polarización sí se afecta el tiempo de cómputo, aumentando varios segundos al tiempo de renderizado.

7.3. Comparación de tiempos de cómputo

Todos los renderizados se llevaron a cabo en una computadora MacBook Pro®, con un procesador Intel® Core 2 Duo a 2.26 GHz, 2 GB de RAM y sistema operativo versión 10.6.8., los tiempos mostrados son los tiempos reportados por el mismo programa *pbrt*, las imágenes generadas tienen un tamaño de 640 × 420 píxeles.

Como se mencionó anteriormente, se puede observar en la Tabla 7.1 que en el caso



(a) (b)

(c) (d)

Figura 7.13: Para la imagen (a) el material del conejo y la luz se encuentran sin polarizar, el elemento transparente filtra longitudes de onda verdes y azules a 90^{o}, para la imagen (b) el material del conejo es polarizador lineal a 180^{o} (en el elemento transparente solo se percibe la luz que no es reflejada del objeto), en la imagen (c) el material del conejo polariza a 180^{o}, la luz puntual está polarizada a 45^{o}, el elemento transparente filtra las longitudes de onda verde y azul (el filtro solo afecta al conejo). Por último, la imagen (d) solo deja pasar la longitud de onda roja, por lo que solo esa parte del patrón de difracción es visible a través del filtro.

del patrón de difracción el factor que determina el impacto en el tiempo de cómputo es el tamaño del área en el que se mapea el patrón, ya sea que es muy grande o estamos muy cerca del objeto. Se presentan los tiempos de escenas representativas y diferentes, ya que cambiar el patrón de difracción no hace una diferencia en el tiempo de cómputo.

Escena/Efectos	Sin efectos	Difracción
Figura 7.11 (c) escalando patrón de difracción (mayor tamaño)	25.5 s	27.5 s
Figura 7.11 (c) sin escalar patrón de difracción	25.5 s	27.0 s
Figura 7.9(d) escalando patrón de difracción (mayor tamaño)	18.2 s	18.7 s
Figura 7.9(d) sin escalar patrón de difracción	18.2 s	18.6 s
Figura 7.4(a) escalando patrón de difracción (mayor tamaño)	11.2 s	16.7 s
Figura 7.4(a) sin escalar patrón de difracción	11.2 s	15.5 s

Tabla 7.1: Comparación de tiempos para diferentes escenas y escalamientos del patrón de difracción.

Finalmente, la Tabla 7.2 muestra los tiempos de ejecución comparando entre una imagen sin efectos de onda, una imagen que se generó sólo usando difracción, una imagen que se generó sólo usando polarización y una imagen con ambos efectos.

Escena/Efectos	Sin efectos	Difracción	Polarización	Difracción+Polarización
Escalando patrón	27.5 s	30.4 s	28.5 s	30.5 s
Sin escalar patrón	27.5 s	29.7 s	28.5 s	29.9 s

Tabla 7.2: Comparación de tiempos utilizados para generar las diferentes versiones de la Figura 7.13.

Capítulo 8

Conclusiones

En general, se puede decir que el interés y la forma de llevar a cabo el proceso de generación de imágenes por computadora depende de la aplicación y la razón por la que se quiere generar la imagen; sin embargo, en la mayoría de los casos se buscan imágenes que además de cumplir con su propósito también sean agradables a la vista. Hacer un modelado y un renderizado exacto ha probado ser un tema complicado debido a las limitaciones inherentes de *hardware* y *software*, sin embargo, diversas aproximaciones pueden generar imágenes adecuadas, sobre todo si tienen una base física en su implementación. Se puede concluir que haciendo pequeñas adecuaciones a los procedimientos estándar de generación de imágenes por computadora, principalmente al proceso que modela la iluminación, podemos obtener mejoras visuales en el renderizado de dichas imágenes.

Debido a la gran variedad de usos que se le pueden dar a las imágenes, las razones para buscar mejoras visuales pueden ser igual de diversas, ya sea porque queremos conservar las propiedades de la topología del objeto geométrico a representar, suavizando su visualización final, o porque queremos modelar más efectos con la interacción de la luz para darle más realismo a las imágenes, entre otras.

En el presente trabajo se mostraron dos métodos distintos para mejorar imágenes generadas por computadora a través de realizar modificaciones al proceso de renderizado, la idea principal de ambos métodos es que se pueden generar mejoras o introducir efectos más realistas en las imágenes generadas por computadora haciendo cambios al modelo de iluminación. Esto es posible debido a que este proceso juega

un papel central en la determinación del valor final de cada píxel de una imagen renderizada. El primer método consiste en asignar normales apropiadas a superficies voxelizadas para suavizar el renderizado final y el segundo método consiste en generar patrones de difracción usando óptica de Fourier y mapeo de texturas, así como una implementación de polarización, para tratar de reproducir los efectos de onda.

A continuación, abundaremos en la discusión de los resultados obtenidos en este trabajo y en el posible trabajo futuro para cada método.

8.1. Asignación de Normales

Actualmente hay muchos dispositivos, principalmente para imagenología, que producen versiones discretizadas de funciones que aproximan objetos del mundo real por medio de un proceso conocido como muestreo. Una forma común de muestrear funciones 3D es evaluar la función en las posiciones de una rejilla cúbica simple, resultando en una función discreta de elementos pequeños de volumen cúbico colindantes (es decir, vóxeles cúbicos o simplemente vóxeles). Normalmente, es necesario procesar las funciones discretizadas para visualizar los objetos representados por dichas funciones y de esta forma obtener una malla que aproxima una isosuperficie del objeto.

En este trabajo usamos un método para extraer la gráfica que identifica los elementos del borde (es decir, caras) entre los vóxeles con valor de uno (que se encuentran dentro del objeto) y los vóxeles con valor de cero (que se encuentran fuera del objeto), produciendo una malla de polígonos cuadrilaterales. Una malla producida por este método preserva la topología de los objetos voxelizados pero su renderizado produce imágenes del objeto con bordes muy cuadrados. En este trabajo presentamos dos métodos para asignar normales a los vértices de una malla, hecha de polígonos cuadrilaterales, de tal manera que los renderizados de dichas mallas reducen significativamente la percepción de la voxelización inherente. La idea detrás de este método es que al asignar normales adecuadamente se puede cambiar la apariencia final de un objeto, esto se debe a que las normales a las superficies son fundamentales en el modelo de iluminación y en el proceso de renderizado.

Debido a que la naturaleza de la función subyacente que representa a los objetos no siempre es conocida, propusimos un método que usa una combinación lineal de funciones base suaves para asignar las normales en los vértices de la malla de cuadriláteros. Para este método usamos el conjunto de datos binario para ubicar los centros de las funciones base y asignar los valores de sus coeficientes. Además, propusimos otro método que supone que la función subyacente puede aproximarse por medio de una combinación lineal de funciones base suaves, conocidas como *blobs*. El gradiente de dicha función varía suavemente a lo largo de los puntos usados para obtener su discretización. En nuestro trabajo demostramos que, al evaluar los gradientes de dicha combinación lineal de *blobs* en los puntos de la malla, se producen renderizados que son suaves sin modificar la malla subyacente. La suavidad del renderizado producido por este método puede controlarse por medio de la selección de diferentes valores del error ms en el método de la Subsección 3.1.1.

También comparamos visualmente algunos renderizados producidos por nuestros dos métodos contra aquellos producidos por la implementación de *Digital Integral Invariant Curvature Estimator* (DGtal). Encontramos que nuestro método GMAN puede producir renderizados de calidad similar a los producidos por el método DGtal y aunque nuestro método SMAN produce renderizados relativamente inferiores, puede obtenerlos en un tiempo menor. Estamos trabajando en un método que permita producir mejores renderizados usando vóxeles asociados a la rejilla cúbica centrada en caras o *fcc* (es decir, dodecaedros rómbicos), como se ha propuesto en [34]. Ahí se demostró que estos vóxeles producen renderizados similares a los producidos por los vóxeles cúbicos pero con un menor número de puntos de muestra. Con un número similar de vóxeles, los renderizados de dodecaedros rómbicos se ven más suaves. Incidentalmente, la metodología presentada en este trabajo puede ser aplicada, con muy pocas modificaciones, a otras voxelizaciones regulares tal como las rejillas cúbicas centradas en cuerpo (*bcc*) y centradas en cara (*fcc*) que ayudarían a minimizar los errores de "escalera"; lamentablemente, la mayoría de los equipos de imagenología solo digitalizan las funciones 3D usando vóxeles cúbicos simples.

Finalmente, los dos métodos presentados pueden usarse como puntos de inicio

para un método de remallado. Los puntos iniciales pueden obtenerse evaluando la envoltura formada por la combinación de *blobs*; además, debido a que la envoltura es continua es posible crear y evaluar nuevos puntos en la malla.

8.2. Efectos de onda

Un aspecto fundamental para brindar realismo a una imagen generada por computadora consiste en modelar el comportamiento de la luz en su interacción con los diferentes objetos en una escena de manera cercana a su comportamiento físico. A pesar de que se han desarrollado varios métodos con el fin de generar un modelo más apegado al comportamiento físico de la luz suelen omitirse los efectos debidos al comportamiento ondulatorio de la misma, tales como la difracción y la polarización. En este trabajo presentamos un método para tomar en cuenta dicho comportamiento y consiste en dos aproximaciones que pueden usarse en conjunto: la primera, calcular el patrón de difracción usando óptica de Fourier para luego colocarlo usando mapeo de texturas, y la segunda, una implementación del seguimiento de la polarización de la luz basado en [127].

Para generar el patrón de difracción es necesario calcular la transformada de Fourier discreta de la función de transmitancia de una apertura difractante, representada por una imagen, a fin de producir el patrón de difracción para luz monocromática. Con el objetivo de generar una imagen con un patrón para luz blanca que, por lo tanto, produzca ciertos cambios de coloración tipo arcoíris, es necesario escalar el patrón original para cada una de las bandas con las que se quiere conformar una imagen compuesta. Se aprovecharon las capacidades de un renderizador basado en rayos, tal como *pbrt*, para colocar el patrón de manera similar a como se coloca una textura. Debido a que se conoce la dirección, como un vector, de reflexión especular para la luz incidente en un punto dado de la superficie, se puede usar dicho vector para generar un sistema coordenado nuevo y colocar el patrón en este sistema; por ello, se pueden usar las coordenadas de la cámara o de los puntos de vista (transformados al sistema coordenado) para determinar qué punto del patrón corresponde al píxel

a renderizar. De esta manera el centro del patrón siempre corresponde a los puntos donde se presentaría la reflexión especular y, por lo tanto, el patrón se traslada si movemos la ubicación de la fuente de luz o si movemos la cámara de forma similar a la que se movería un reflejo real. En este trabajo hacemos la suposición de que toda la superficie está conformada con un material con las características de la apertura difractante y por lo tanto en cualquier punto puede producirse el patrón. Nuestros resultados muestran que en objetos simples el patrón de difracción se muestra completo y puede observarse el patrón de colores característico en la difracción en sistemas ópticos reales, en objetos más complejos se consigue un efecto aperlado, debido a que se aglomeran los patrones. Nuestro método está limitado a la aproximación paraxial y a la zona lejana, pero es una limitación que comparte con otros métodos en la literatura para generar difracción. Sin embargo, pueden añadirse cálculos de fase y transformadas de Fourier adicionales para generar patrones en la zona cercana con el correspondiente aumento en la complejidad y el costo de cómputo. El método que hemos propuesto también puede implementarse en un sistema renderizador basado en iluminación local, aunque estaría limitado por las restricciones naturales de dichos métodos ya que no puede acarrearse el valor de la intensidad de luz en una reflexión. Aunque se puede modificar el tamaño del patrón y la distancia a la que se proyecta en el objeto esto no puede llevarse a cabo de forma dinámica, probablemente sería preferible que fuera dependiente de alguna característica del material y dependiente de la distancia del objeto al observador. De forma similar, por el momento no hay variaciones en los factores de escala usados para generar las distintas componentes del patrón, idealmente tendrían que calcularse de forma dinámica tomando en cuenta el tamaño de la abertura difractante y la distancia del objeto al observador. Adicionalmente, una mejora en cuanto a memoria consistiría en usar el patrón de difracción de una sola longitud de onda y generar la imagen a color muestreando la misma imagen en tres ubicaciones distintas (correspondientes a las componentes rojo, azul y verde).

Los efectos de polarización se agregaron por separado usando vectores de Stokes y matrices de Mueller, para ello debieron modificarse las clases, estructuras de datos y métodos que integran la luz. Las imágenes resultantes muestran cómo podemos usar

los materiales como polarizadores y filtros para generar variaciones de color; también, los resultados demuestran como los efectos de difracción y polarización pueden combinarse sin inconvenientes. Sin embargo, las pruebas se hicieron usando ejemplos de polarización sencillos ya que es complicado generar las matrices de Mueller que generen efectos más interesantes y físicamente posibles. En un sentido similar, falta implementar direccionalidad en la relación del material con el objeto, para que al girar el objeto también se haga una rotación de la matriz de Mueller correspondiente y permita simular los cambios de orientación de un filtro. Nuestro método también permite implementar otros efectos de polarización como la birrefringencia, incluyendo la creación de rayos extras que permitan generar la imagen doble. Otra desventaja de nuestro método es que no se está llevando a cabo el seguimiento punto a punto del cambio de fase ni de la polarización de la luz, sólo se toma en cuenta el resultado de la interacción y éste es el que se propaga, sin embargo, esto simplifica la implementación y aunque aumenta el costo computacional lo hace de manera moderada.

Bibliografía

[1] J. Almeida, "Radiative transfer for polarized light - equivalence between stokes parameters and coherency matrix formalisms," *Solar Physics*, pp. 1 – 14, 1992.

[2] A. Appel, "Some techniques for shading machine renderings of solids," in *Proceedings of the April 30–May 2, 1968, Spring joint Computer Conference*, 1968, pp. 37 – 45.

[3] E. Artzy, G. Frieder, and G. T. Herman, "The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm." *Computer Graphics and Image Processing*, vol. 15, no. 1, pp. 1 – 24, 1981.

[4] L. Belcour and P. Barla, "A practical extension to microfacet theory for the modeling of varying iridescence," *ACM Transactions on Graphics*, vol. 36, pp. 65:1 – 65:14, 2017.

[5] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva, "State of the art in surface reconstruction from point clouds." in *Eurographics 2014 - State of the Art Reports*, S. Lefebvre and M. Spagnuolo, Eds. Strasbourg, France: The Eurographics Association., April 2014, pp. 161 – 185.

[6] S. Bergmann, M. Mohammadikaji, S. Irgenfried, H. Wörn, J. Beyerer, and C. Dachsbacher, "A phenomenological approach to integrating gaussian beam properties and speckle into a physically-based renderer," in *Proceedings of the Conference on Vision, Modeling and Visualization*, 2016.

[7] J. F. Blinn, "A generalization of algebraic surface drawing," *ACM Transactions on Graphics*, vol. 1, pp. 235 – 256, 1982.

[8] M. Born and E. Wolf, *Principles of Optics*. Cambridge University Press, 2005.

[9] P. Bourke. (1996, September) Creating TGA image files. [Online]. Available: http://www.paulbourke.net/dataformats/tga/

[10] A. J. P. Braquelaire and A. Pousset, "Euclidean nets: An automatic and reversible geometric smoothing of discrete 3D object boundaries." in *Discrete Geometry for Computer Imagery*, G. Borgefors, I. Nyström, and G. Sanniti di Baja, Eds., vol. 1953. Springer, 2000, pp. 198 – 209.

[11] B. M. Carvalho and G. T. Herman, "Helical CT reconstruction from wide cone-beam angle data using ART," in *XVI Brazilian Symposium on Computer Graphics and Image Processing*, 2003, pp. 363 – 370.

[12] ——, "Low-dose, large-angled cone-beam helical CT data reconstruction using algebraic reconstruction techniques," *Image and Vision Computing*, vol. 25, no. 1, pp. 42 – 61, 2007.

[13] T. J. Cashman, "Beyond Catmull–Clark? A survey of advances in subdivision surface methods," *Computer Graphics Forum*, vol. 31, no. 1, pp. 42 – 61, 2012.

[14] C. Ceja, C. Rascon, E. Garduño, B. M. Carvalho, and G. T. Herman, "Smooth normals with blobs for surfaces from 3d binary images," *Topology Proceedings*, vol. 61, pp. 239 – 267, 2023.

[15] E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion, "A survey on participating media rendering techniques," *The Visual Computer*, vol. 21, pp. 303 – 328, 2005.

[16] L.-S. Chen, G. T. Herman, R. A. Reynolds, and J. K. Udupa, "Surface shading in the cuberille environment," *IEEE Computer Graphics and Applications*, vol. 5, no. 12, pp. 33 – 43, Diciembre 1985.

[17] D. Coeurjolly and J.-O. Lachaud. (2017, October) DGTAL: Digital Geometry tools and algorithms library. [Online]. Available: http://dgtal.org

[18] D. Coeurjolly, J.-O. Lachaud, and J. Levallois, "Multigrid convergent principal curvature estimators in digital geometry," *Computer Vision and Image Understanding*, vol. 129, pp. 27 – 41, 2014.

[19] E. Collett, *Field Guide to Polarization*. SPIE Press, 2005.

[20] T. Cuypers, T. Haber, P. Bekaert, S. B. Oh, and R. Raskar, "Reflectance model for diffraction," *ACM Transactions on Graphics*, pp. 122:1 – 122:11, 2012.

[21] D. S. J. Dhillon and A. Ghosh, "Efficient surface diffraction renderings with chebyshev approximations," in *SIGGRAPH ASIA 2016 Technical Briefs*, 2016.

[22] D. Dhillon, J. Teyssier, M. Single, I. Gaponenko, M. Milinkovitch, and M. Zwicker, "Interactive diffraction from biological nanostructures," *Computer Graphics Forum*, vol. 33, pp. 177 – 188, 2014.

[23] Z. Dong, B. Walter, S. Marschner, and D. P. Greenberg, "Predicting appearance from measured microgeometry of metal surfaces," *ACM Transactions on Graphics*, vol. 35, pp. 9:1 – 9:13, 2015.

[24] J. Dorsey, H. Rushmeier, and F. Sillion, *Digital Modeling of Material Appearance*. Morgan Kaufmann Publishers Inc., 2008.

[25] D. D. Duncan and S. J. Kirkpatrick, "Algorithms for simulation of speckle (laser and otherwise)," in *Complex Dynamics and Fluctuations in Biomedical Photonics V*, V. V. Tuchin and L. V. Wang, Eds., vol. 6855, International Society for Optics and Photonics. SPIE, 2008, p. 685505.

[26] R. Durikovic and R. Kimura, "GPU rendering of the thin film on paints with full spectrum," in *Proceedings of the conference on Information Visualization*, 2006, pp. 751 – 756.

[27] R. P. Feyman and M. L. Leighton, Robert B. an Sands, *The Feyman lectures in Physics; Vol. I*, R. Mass, Ed. Addison-Wesley, 1965, vol. 33, no. 9.

[28] F. Flin, J.-B. Brzoska, D. Coeurjolly, R. A. Pieritz, B. Lesaffre, C. Coléou, P. Lamboley, O. Teytaud, G. L. Vignoles, and J.-F. Delesse, "Adaptive estimation of normals and surface area for discrete 3-D objects: Application to snow binary data from X-ray tomography," *IEEE Transactions on Image Processing*, vol. 14, no. 5, pp. 585 – 596, 2005.

[29] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips, *Introduction to Computer Graphics*. Addison-Wesley, 1993.

[30] J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed. Addison-Wesley, 1996.

[31] S. Fourey and R. Malgouyres, "Normals estimation for digital surfaces based on convolutions," *Computer Graphics*, vol. 33, pp. 2 – 10, 2009.

[32] E. R. Freniere, G. Gregory, and R. A. Hassler, "Polarization models for Monte Carlo ray tracing," in *Proceedings of the SPIE*, 1999.

[33] E. Garduño and G. T. Herman, "Optimization of basis functions for both reconstruction and visulization," *Discrete Applied Mathematics*, vol. 139, pp. 95 – 111, Abril 2004.

[34] E. Garduño, G. T. Herman, and H. Katz, "Boundary tracking in 3D binary images to produce rhombic faces for a dodecahedral model," *IEEE Transactions on Medical Imaging*, vol. 17, pp. 1097 – 1100, 1998.

[35] F. Gascón and F. Salazar, "A simple method to simulate diffraction and speckle patterns with a PC," *Optik - International Journal for Light and Electron Optics*, vol. 117, pp. 49 – 57, 2006.

[36] R. Goldman, *An Integrated Introduction to Computer Graphics and Geometric Modeling*. CRC Press, Inc., 2009.

[37] J. W. Goodman, "Statistical properties of laser speckle patterns," in *Laser Speckle and Related Phenomena*, 1984.

[38] C. Goral, K. Torrance, D. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," in *ACM SIGGRAPH Computer Graphics*, 1984, pp. 213 – 222.

[39] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers*, vol. C-20, no. 6, pp. 623 – 629, 1971.

[40] D. Greenberg, M. Cohen, and K. Torrance, "Radiosity: A method for computing global illumination," *The Visual Computer*, vol. 2, pp. 291 – 297, 1986.

[41] I. Guillén, J. Marco, D. Gutierrez, W. Jakob, and A. Jarabo, "A general framework for pearlescent materials," *ACM Transactions on Graphics*, vol. 39, no. 6, nov 2020.

[42] Y. Guo, M. Hašan, and S. Zhao, "Position-free monte carlo simulation for arbitrary layered bsdfs," *ACM Trans. Graph.*, vol. 37, pp. 279:1 – 279:14, 2018.

[43] S. Guy and C. Soler, "Graphics gems revisited: fast and physically-based rendering of gemstones," in *ACM Transactions on Graphics (TOG)*, 2004, pp. 231 – 238.

[44] F. Hartmann, "Boundary element methods," in *Encyclopedia of Vibration*, S. Braun, Ed. Oxford: Elsevier, 2001, pp. 192 – 202.

[45] X. He, K. Torrance, F. Sillion, and D. Greenberg, "A comprehensive physical model for light reflection," in *ACM SIGGRAPH Computer Graphics*, 1991, pp. 175 – 186.

[46] E. Hecht, *Optics*. Addison Wesley, 2002.

[47] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, 2nd ed., ser. Advances in Pattern Recognition. Springer, 2009.

[48] H. Hirayama, K. Kaneda, H. Yamashita, and Y. Moden, "An accurate illumination model for objects coated with multilayer films," *Computer & Graphics*, vol. 25, pp. 391 – 400, 2001.

[49] N. Holzschuch and R. Pacanowski, "A physically accurate reflectance model combining reflection and diffraction," INRIA, Tech. Rep., 2015.

[50] ——, "A physically-based reflectance model combining reflection and diffraction," INRIA, Tech. Rep., 2016.

[51] ——, "A two-scale microfacet reflectance model combining reflection and diffraction," *ACM Trans. Graph.*, vol. 36, pp. 66:1 – 66:12, 2017.

[52] D. Hong, G. Ning, T. Zhao, M. Zhang, and X. Zheng, "Method of normal estimation based on approximation for visualization," *Journal of Electronic Imaging*, vol. 12, no. 3, pp. 470 – 477, 2003.

[53] T. M. Inc., "Matlab version 9.1.0.441655 (r2016b)," Natick, Massachusetts, United States, 2016. [Online]. Available: https://www.mathworks.com/help/stats/index.html

[54] K. Iwasaki, K. Matsuzawa, and T. Nishita, "Real-time rendering of soap bubbles taking into account light interference," in *Proceedings of the Computer Graphics International*, 2004, pp. 344 – 348.

[55] A. Jarabo and D. Gutierrez, "Bidirectional rendering of polarized light transport," in *Proceedings of the XXVI Spanish Computer Graphics Conference*, 2016.

[56] H. Jensen, "Global illumination using photon maps," *Rendering Techniques*, vol. 96, pp. 21 – 30, 1996.

[57] H. Jensen and N. Christensen, "Photon maps in bidirectional monte carlo ray tracing of complex objects," *Computers & Graphics*, vol. 19, pp. 215 – 224, 1995.

[58] H. Jensen *et al.*, "Importance driven path tracing using the photon map," *Rendering Techniques*, vol. 95, pp. 326 – 335, 1995.

[59] T. Jianlei, L. Xumin, and G. Yong, "Anisotropic smoothing algorithm for triangular mesh models," in *International Forum on Computer Science-Technology and Applications, IFCSTA'09*, vol. 1, 2009, pp. 126 – 130.

[60] H. Kahn, "Use of different monte carlo sampling techniques," in *Symposium on Monte Carlo Methods*, H. A. Mayer, Ed. Wiley, 1956, pp. 146 – 190.

[61] J. F. Kaiser and R. W. Schafer, "On the use of the I0-sinh window for spectrum analysis," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 1, pp. 105 – 107, 1980.

[62] J. T. Kajiya, "The rendering equation," *ACM SIGGRAPH Computer Graphics*, vol. 20, pp. 143 – 150, 1986.

[63] P. A. Kalos, Malvin H.; Whitlock, *Monte Carlo Methods*. Wiley-VCH, 2008.

[64] Y.-M. Kang, D.-H. Lee, and H.-G. Cho, "Multipeak aniostropic microfacet model for iridescent surfaces," *Multimedia Tools Appl.*, vol. 74, pp. 6229 – 6242, 2015.

[65] M. R. Kaplan, "The uses of spatial coherence in ray tracing," ACM SIGCRAPH '85 Course Notes 11, July 1985.

[66] B. Kerautret and A. Braquelaire., "A statistical approach for geometric smoothing of discrete surfaces," in *Discrete Geometry for Computer Imagery*, ser. Lecture Notes in Computer Science, E. Andres, G. Damiand, and P. Lienhardt, Eds., vol. 3429. Springer, 2005, pp. 404 – 413.

[67] R. Klette and A. Rosenfeld, *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Morgan Kaufmann, 2004.

[68] T. Y. Kong and A. Rosenfeld, "Digital topology: Introduction and survey," *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 3, pp. 357 – 393, 1989.

[69] A. Krywonos, "Predicting surface scatter using a linear system formulation of non-paraxial scalar diffraction," Ph.D. dissertation, University of Central Florida, 2006.

[70] J.-O. Lachaud, D. Coeurjolly, and J. Levallois, "Robust and convergent curvature and normal estimators with digital integral invariants." in *Modern Approaches to Discrete Curvature*, ser. Lecture Notes in Computer Science, L. Najman and e. Pascal Romon, Eds., vol. 2184. Springer, 2017, pp. 293 – 348.

[71] S. Laine and T. Karras, "Efficient sparse voxel octrees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 8, pp. 1048 – 1059, 2011.

[72] P. Latorre, F. Seron, and D. Gutierrez, "Birefringence: calculation of refracted ray paths in biaxial crystals," *The Visual Computer*, vol. 28, no. 4, pp. 341 – 356, 2012.

[73] J. H. Lee, A. Jarabo, D. S. Jeon, D. Gutierrez, and M. H. Kim, "Practical multiple scattering for rough surfaces," *ACM Transactions on Graphics*, vol. 37, no. 6, dec 2018.

[74] A. Lenoir, R. Malgouyres, and M. Revenu, "Fast computation of the normal vector field of the surface of a 3-D discrete object," in *Discrete Geometry for Computer Imagery*, S. Miguet, A. Montanvert, and S. Ubéda, Eds. Springer, 1996.

[75] R. M. Lewitt, "Multidimensional digital image representations using generalized Kaiser–Bessel window functions," *Journal of the Optical Society of America A*, vol. 7, pp. 1834 – 1846, 1990.

[76] ——, "Alternatives to voxels for image representation in iterative reconstruction algorithms," *Physics in Medicine and Biology*, vol. 37, pp. 705 – 716, 1992.

[77] J. Löw, J. Kronander, A. Ynnerman, and J. Unger, "Brdf models for accurate and efficient rendering of glossy surfaces," *ACM Trans. Graph.*, vol. 31, pp. 9:1 – 9:14, 2012.

[78] L.-M. Luo and J.-L. Coatrieux, "Surface normal for 3D object display in cuberille environment," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 1, 1988, pp. 420 – 421.

[79] R. Marabini, G. T. Herman, and J. M. Carazo, "3D reconstruction in electron microscopy using ART with smooth spherically symmetric volume elements (blobs)," *Ultramicroscopy*, vol. 72, pp. 53 – 65, 1998.

[80] S. Matej and R. Lewitt, "Practical considerations for 3-d image reconstruction using spherically symmetric volume elements," *IEEE Transactions on Medical Imaging*, vol. 15, pp. 68 – 78, 1996.

[81] S. Matej and R. M. Lewitt, "Efficient 3D grids for image-reconstruction using spherically-symmetrical volume elements." *IEEE Transactions on Nuclear Science*, vol. 42, pp. 1361 – 1370, 1995.

[82] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335 – 341, 1949, pMID: 18139350.

[83] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "Evaluation and design of filters using a Taylor series expansion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 184 – 199, 1997.

[84] H. P. Moravec, "3D graphics and the wave theory," *ACM SIGGRAPH Computer Graphics*, vol. 15, pp. 289 – 296, 1981.

[85] M. Mout, M. Wick, F. Bociort, J. Petschulat, and P. Urbach, "Ray tracing the wigner distribution function for optical simulations," *Optical Engineering*, vol. 57, 2018.

[86] M. Mout, M. Wick, F. Bociort, J. Petschulat, and P. Urbach, "Simulating multiple diffraction in imaging systems using a path integration method," *Applied Optics*, vol. 55, no. 14, pp. 3847 – 3853, May 2016.

[87] S. Muraki, "Volumetric shape description of range data using "Blobby Model"," *Computer Graphics*, vol. 25, pp. 227 – 235, 1991.

[88] ——, "Multiscale volume representation by a DoG wavelet," *IEEE Transactions on Visualization and Computer Graphics*, pp. 109 – 116, 1995.

[89] A. Musbach, G. W. Meyer, F. Reitich, and S. H. Oh, "Full wave modelling of light propagation and reflection," *Computer Graphics Forum*, vol. 32, no. 6, pp. 24 – 37, 2013.

[90] F. Nicodemus, J. Richmond, J. Hsia, I. Ginsberg, and T. Limperis, "Geometrical considerations and nomenclature for reflectance (national bureau of standards, u. s. a.)," *Monograph*, 01 1977.

[91] F. E. Nicodemus, "Directional reflectance and emissivity of an opaque surface," *Appl. Opt.*, vol. 4, no. 7, pp. 767 – 775, Jul 1965.

[92] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura, "Object modeling by distribution function and a method of image generation," *Transactions of the Institute of Electronics and Communications Engineers of Japan, D*, vol. 68, no. 4, pp. 718 – 725, 1985.

[93] P. Novotny, L. I. Dimitrov, and M. Sramek., "Enhanced voxelization and representation of objects with sharp details in truncated distance fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 484 – 498, 2010.

[94] S. B. Oh, S. Kashyap, R. Garg, C. S., and R. Raskar, "Rendering wave effects with augmented light field," *Computer Graphics Forum*, vol. 29, pp. 507 – 516, 2010.

[95] S. B. Oh, G. Barbastathis, and R. Raskar, "Augmenting light field to model wave optics effects," *CoRR*, 2009.

[96] L. Papier, "Polyédrisation et visualisation d'objets discrets tridimensionnels," Ph.D. dissertation, Université Louis Pasteur, Strasbourg, France, 1999.

[97] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.

[98] M. Pharr and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2010.

[99] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, pp. 311 – 317, June 1975.

[100] A. J. Preetham, P. Shirley, and B. Smits, "A practical analytic model for daylight," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '99, 1999, pp. 91 – 100.

[101] A. Reshetov, A. Soupikov, and W. R. Mark, *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 142:1 – 142:8, 2010.

[102] S. Roettger, "The volume library." [Online]. Available: http://www9.informatik.uni-erlangen.de/External/vollib/

[103] I. Sadeghi, A. Munoz, P. Laven, W. Jarosz, F. Seron, D. Gutierrez, and H. W. Jensen, "Physically-based simulation of rainbows," *ACM Transactions on Graphics*, pp. 3:1 – 3:12, 2012.

[104] D. Salomon, *Computer Graphics and Geometric Modeling*. Springer, 1999.

[105] S. Sankaranarayanan, "Modelling polarized light for computer graphics," Ph.D. dissertation, Iowa State University, 1997.

[106] L. Scandolo, S. Lee, and E. Eisemann §, "Quad-based fourier transform for efficient diffraction synthesis," in *Computer Graphics Forum*, vol. 37, no. 4. Wiley Online Library, 2018, pp. 167 – 176.

[107] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "Nih image to imagej: 25 years of image analysis," *Nature Methods*, vol. 9, no. 7, pp. 671 – 675, 2012.

[108] R. H. Siddique, S. Diewald, J. Leuthold, and H. Hölscher, "Theoretical and experimental analysis of the structural pattern responsible for the iridescence of morpho butterflies," *Opt. Express*, vol. 21, no. 12, pp. 14 351 – 14 361, Jun 2013.

[109] C. O. S. Sorzano, R. Marabini, J. A. Velázquez-Muriel, J. R. Bilbao-Castro, S. H. W. Scheres, J. M. Carazo, and A. Pascual-Montano, "XMIPP: a new generation of an open-source image processing package for electron microscopy," *Journal of Structural Biology*, pp. 194 – 204, 2004.

[110] M. Sramek and A. E. Kaufman, "Alias-free voxelization of geometric objects," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, pp. 251 – 267, 1999.

[111] J. Stam, "Diffraction shaders," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 101 – 110.

[112] S. Steinberg and L.-Q. Yan, "A generic framework for physical light transport," *ACM Transactions on Graphics*, vol. 40, no. 4, jul 2021.

[113] D. Tannenbaum, P. Tannenbaum, and M. Wozny, "Polarization and birefringency considerations in rendering," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 221 – 222.

[114] G. Tanski. Global illumination illustration. [Online]. Available: https://commons.wikimedia.org/wiki/File:Global_illumination.JPG

[115] ——. Local illumination illustration. [Online]. Available: https://commons.wikimedia.org/wiki/File:Local_illumination.JPG

[116] P. Tellier and I. Debled-Rennesson, "3D discrete normal vectors," in *Proceedings of the 8th International Conference on Discrete Geometry for Computer Imagery*, ser. DCGI'99 in Lecture Notes in Computer Science, G. Bertrand, M. Couprie, and L. Perroton, Eds., vol. 1568. Springer, 1999, pp. 447 – 458.

[117] G. Thürmer, "Smoothing normal vectors on discrete surfaces while preserving slope discontinuities," *Computer Graphics Forum*, vol. 20, no. 2, pp. 103 – 114, 2001.

[118] A. Toisoul and A. Ghosh, "Practical acquisition and rendering of diffraction effects in surface reflectance," *ACM Transactions on Graphics*, vol. 36, 2017.

[119] ——, "Real-time rendering of realistic surface diffraction with low rank factorisation," in *Proceedings of the 14th European Conference on Visual Media Production (CVMP 2017)*, 2017.

[120] S. Trester, "Computer-simulated fresnel diffraction using the fourier transform," *Computing in Science & Engineering*, vol. 1, no. 5, pp. 77 – 83, 1999.

[121] G. Turk and J. F. O'Brien, "Modelling with implicit surfaces that interpolate," *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 855 – 873, 2002.

[122] S. Werner, Z. Velinov, W. Jakob, and M. B. Hullin, "Scratch iridescence: Wave-optical rendering of diffractive surface structure," *ACM Trans. Graph.*, vol. 36, pp. 207:1 – 207:14, 2017.

[123] T. Whitted, "An improved illumination model for shaded display," *Communications of the ACM*, vol. 23, no. 6, pp. 343 – 349, Jun 1980.

[124] A. Wilkie, R. Tobler, and W. Purgathofer, "Combined rendering of polarization and fluorescence effects," in *12th Eurographics Workshop on Rendering*, 2001, pp. 197 – 204.

[125] A. Wilkie, C. Ulbricht, R. F. Tobler, G. Zotti, and W. Purgathofer, "An analytical model for skylight polarisation," in *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, 2004, pp. 387 – 397.

[126] A. Wilkie and A. Weidlich, "A standardised polarisation visualisation for images," in *Proceedings of the 26th Spring Conference on Computer Graphics*, 2010, pp. 43 – 50.

[127] ——, "How to write a polarisation ray tracer," in *SIGGRAPH Asia 2011 Courses*, 2011.

[128] L. Wolff and D. Kurlander, "Ray tracing with polarization parameters," *IEEE Computer Graphics and Applications*, vol. 10, pp. 44 – 55, 1990.

[129] F.-k. Wu and C.-w. Zheng, "Microfacet-based interference simulation for multilayer films," *Graphical Models*, vol. 78, no. 0, pp. 26 – 35, 2015.

[130] F. Wu and C. Zheng, "A comprehensive geometrical optics application for wave rendering," *Graphical Models*, vol. 75, pp. 318 – 327, 2013.

[131] ——, "Simulation of wave effects based on ray tracing," in *International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics), 2013*, 2013, pp. 9 – 15.

[132] ——, "Interference shader for multilayer films," in *Computer Vision, Imaging and Computer Graphics Theory and Applications*, J. Braz, J. Pettré, P. Richard, A. Kerren, L. Linsen, S. Battiato, and F. Imai, Eds. Cham: Springer International Publishing, 2016, pp. 62 – 74.

[133] B. Wyvill and G. Wyvill, "Field functions for implicit surfaces," *The Visual Computer*, vol. 5, no. 1-2, pp. 75 – 82, 1989.

[134] G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," *The Visual Computer*, vol. 2, pp. 227 – 234, 1986.

[135] M. M. Xia, B. Walter, E. Michielssen, D. Bindel, and S. Marschner, "A wave optics based fiber scattering model," *ACM Transactions on Graphics*, vol. 39, no. 6, nov 2020.

[136] R. Yagel, D. Cohen, and A. E. Kaufman, "Normal estimation in 3D discrete space," *The Visual Computer*, vol. 8, pp. 278 – 291, 1991.

[137] T. Zeltner and W. Jakob, "The layer laboratory: A calculus for additive and subtractive composition of anisotropic surface reflectance," *ACM Transactions on Graphics*, vol. 37, pp. 1 – 14, 07 2018.

[138] Z. Zhang and M. Levoy, "Wigner distributions and how they relate to the light field," in *Proc. IEEE International Conference on Computational Photography*, 2009.

[139] Y. Zong, "From candle to candela," *Nature Physcs*, vol. 12, p. 614, 2016.