Universidad Zaragoza
1542

Trabajo Fin de Grado

Generación de imágenes sintéticas optimizadas para representaciones masivas de apariencia

Optimized generation of synthetic images for massive appearance representation

Autor

Adrián Alejandre Escriche

Director

Adolfo Muñoz Orbañanos

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Adrián Alejandre Escriche,

con nº de DNI 17458701E en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Generación de imágenes sintéticas optimizadas para representaciones masivas de apariencia,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24/11/2017

Fdo: Adrián Alejandre Escriche

Título del resumen

RESUMEN

La generación de imágenes fotorrealistas es una de las principales disciplinas de la Informática Gráfica que más repercusión tiene hoy en día. Conseguir estas imágenes es un proceso complicado y pesado que requiere simular el transporte que realiza la luz por la escena. Para conseguir una simulación físicamente correcta hay que utilizar un algoritmo adecuado para recorra todos los caminos posibles que realiza la luz y modelos para representar la apariencia de los objetos que forman la escena.

Los modelos que definen las propiedades de apariencia suelen estar representados por funciones n-dimensionales que pueden depender de modelos heurísticos o pueden ser datos tabulados provenientes de capturas o simulaciones. Estas funciones describen como interacciona la luz con el material y debido a la dimensionalidad de estas, tabuladas pueden requerir un alto espacio en memoria. Ademas cuando los datos están tabulados, la producción de una imagen es muy costosa y ralentiza el ya de por si lento proceso de renderizar una imagen.

En este proyecto, con el objetivo de acelerar la convergencia de generación de imágenes sintéticas que requieren de datos tabulados, se propone una adaptación del algoritmo *Expectation-Maximization* para aproximar de forma eficiente y eficaz datos físicos de la apariencia de los materiales. Generando un *Gaussian Mixture Model* que adapta la función tabulada mediante la suma de un conjunto de Gaussianas. Permitiendo así, realizar *Importance Sampling*, una técnica de muestreo que concentra los cálculos en las partes con mas contribución a la imagen final, reduciendo el tiempo que necesario para obtener resultados similares.

Índice

Capítulo 1

Introducción

La Informática Gráfica es el campo de la informática que se encarga de generar imágenes sintéticas. Aunque donde más expectación ante el publico genera sea en el ámbito cinematográfico y en los videojuegos, sus aplicaciones y avances van mucho más allá. Se utiliza en diversos sectores profesionales, desde crear una pre-visualización de un producto antes de entrar en la costosa etapa de producción, estimar la radianza incidente en ciertas estructuras o hasta deducir objetos escondidos detrás de esquinas. Por ejemplo, si se esta diseñando un edificio es útil antes de instalar las luces simular la localización de las fuentes de luz y como estas van a actuar sobre el ambiente.

Una de las principales tareas de la Informática Gráfica es la de generar imágenes sintéticas, imágenes digitales obtenidas mediante la simulación del paso de la luz por una escena 3D. Conforme avanza la investigación, estas imágenes son cada vez más indistinguibles de imágenes reales tomadas con cámara.

1.1. Generación de imágenes sintéticas

En el proceso de generar imágenes sintéticas se intenta recrear de la forma más físicamente correcta posible la apariencia con la que percibimos los objeto. Para ello primero hay que entender cual es el proceso óptico que produce las imágenes en una cámara. Los fotones que se generan en las fuentes de luz atraviesan la escena interaccionando con los objetos y "tiñendose"del color de estos, tras multitud de rebotes estos acaban llegando al sensor de la cámara, generando la fotografía. De forma análoga, en el proceso de generar una imagen sintética se simula el camino y la interacción de la luz a través de la escena para llegar al sensor. Sin embargo, es computacionalmente imposible simular completamente la luz que interacciona por una escena, dado que se mueve en un espacio continuo que tenemos que discretizar de alguna manera..

En la actualidad se han desarrollado multitud de algoritmos para resolver este paradigma, como norma general basados en el algoritmo de Trazador de Rayos. En

él, en vez de seguir los rayos de luz desde las fuentes de luz hasta el sensor se sigue el camino contrario. De esta forma se trazan rayos desde la cámara hacia la escena, que interaccionan con los objetos generando recursivamente nuevos rayos que siguen explorando el espacio contínuo del transporte de luz hasta que llegan a un emisor.

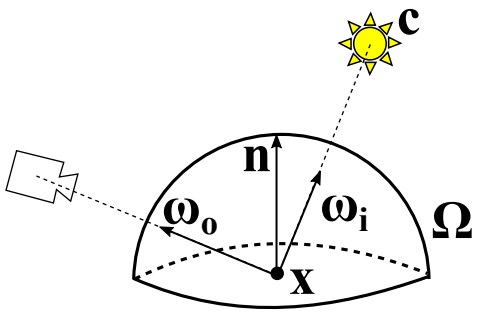


Figura 1.1: La ecuación de render define la radiancia saliente L_o en dirección ω_o en base a toda la iluminación incidente en la hemiesfera Ω considerando la radiancia incidente L_i en cada dirección diferencial ω_i así como la función de apariencia del material.

La interacción entre luz y un punto diferencial de una superficie se modela matemáticamente mediante la *ecuación de render* [Kaj86], que tiene la siguiente forma:

$$L_o(x, \omega_o) = \int_\Omega L_i(x, \omega_i) f_r(x, \omega_i, \omega_o)(n \cdot \omega_i) d\omega_i \tag{1.1}$$

donde:

- x es el punto diferencial en el que ocurre la interacción.
- ω_o es la dirección saliente, desde donde se observa la interacción entre la luz y la materia.
- ω_i es la dirección de entrada, la variable de integración.
- n es la dirección normal a la superficie en el punto x.
- Ω es el rango de integración, que consiste en todo el espacio de direcciones ω_i en la hemiesfera orientada con respecto a la dirección normal n.
- $L_o(x, \omega_o)$ es la radiancia (energía) saliente de la interacción, el "color" observado.
- $L_i(x, \omega_i)$ es la radiancia incidente sobre la interacción, que se explora en toda la hemiesfera y puede provenir de fuentes de luz o de una interacción anterior.
- $f_r(x, \omega_i, \omega_o)$ es la función de apariencia, propiedad específica del material del objeto. En este caso se trata de una *Función de Distribución de Reflectancia Bidireccional* (en inglés, *Bidirectional Reflectance Distribucion Function*, habitualmente mencionada mediante sus siglas *BRDF*).

La ecuación 1.1 define la radiancia saliente de una interacción en base a la integral (suma diferencial) de la luz incidente sobre el punto correspondiente ponderada por las propiedades del material. Los diferentes símbolos así como su relación geométrica pueden verse en la Figura 1.1. El modelo de apariencia en este caso está representado mediante la *BRDF* ($f_r(x, \omega_i, \omega_o)$) para la que puede haber una expresión analítica o puede corresponderse con datos empíricos capturados de la realidad u obtenidos mediante simulación. Estos modelos tabulados, como veremos, son de especial interés en este proyecto.

La ecuación de render (1.1) depende de la iluminación incidente, que puede provenir de fuentes de luz ideales o bien de una interacción anterior. Para ello, como veremos, se generan una serie de rayos secundarios que estiman dicha iluminación incidente, calculando recursivamente cómo se ve afectada la energía tras varias interacciónes (ver figura 1.2).

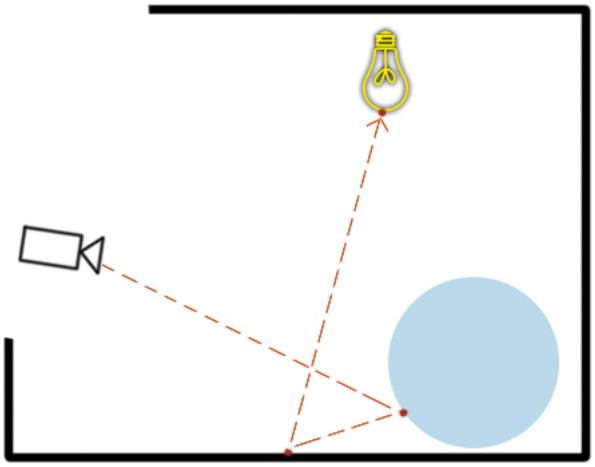


Figura 1.2: Imagen de un caso canónico de un único camino cámara hasta alcanzar un emisor de luz luz. En cada vértice intermedio del camino se calcula la ecuación de render para su dirección de salida y de entrada

1.2. Representación de Apariencia

Utilizar funciones de apariencia adecuadas es fundamental para generar imágenes fotorrealistas. Aparte de tener un algoritmo que calcule correctamente el transporte de luz, son necesarios modelos que definan el comportamiento de la luz al interaccionar con los materiales de la escena. Estos suelen ser funciones que mediante las propiedades del objeto y la dirección incidente de la luz indican cuanta radiancia sale hacia una dirección concreta. Existen funciones analíticas que resuelven el problema mediante

modelos matemáticos, a diferencia de los modelos empíricos, que capturan los datos del material real para obtener una función tabulada que se ajuste a su comportamiento.

Hay multitud de modelos para representar la apariencia de los materiales, como es el caso de la *BRDF* descrita en el apartado anterior.

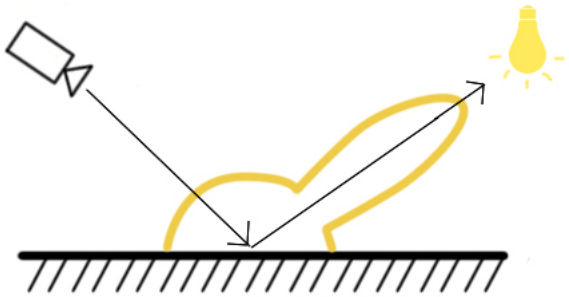


Figura 1.3: Representación de una BRDF en coordenadas angulares. Las direcciones se definen con el ángulo respecto a la normal de la superficie del objeto.

Este modelo se puede generalizar para casos en los que el transporte de la luz al interaccionar con el objeto crea situaciones más complejas, como por ejemplo las refracciones internas que se producen en materiales semitransparentes. Normalmente se usan heuristicas que han demostrado representar multitud de materiales de forma realista. Sin embargo, hay situaciones en las que utilizar modelos analíticos no es suficiente para representar una apariencia compleja y suponen una simplificación demasiado evidente e insalvable.

Los modelos empíricos utilizan datos tabulados para modelar la apariencia masiva de materiales complejos. Los datos tabulados son extraídos a partir de datos del material real o de una simulación, permitiendo obtener todos los detalles de alta frecuencia. Un ejemplo de esto son las fibras que forman los tejidos, su complejidad ha producido diversas propuestas de modelos para representar su apariencia. Uno de ellos es la *Función de Distribución de Dispersion Curva Bidireccional* (*BCSDF* por sus siglas en ingles) [ZW07], aproxima la fibra como un cilindro en el que la luz puede reflejarse, refractarse y transmitirse por el interior. A diferencia de una *BRDF* que considera que la luz que llega al material solo puede ser reflejada hacia alguna otra dirección. Debido a que la luz puede viajar por dentro del cilindro y por tanto incidir desde cualquier dirección el rango de integración Ω en la ecuación de render pasa a ser en toda la esfera cuando se utiliza *BCSDF*:

$$L_o(x, \omega_o) = \int_{\Omega} L_i(x, \omega_i) BCSDF(x, \omega_i, \omega_o) d\omega_i \tag{1.2}$$

Las fibras tienen estructuras muy diversas y complejas imperceptibles a simple

vista. Las microscópicas diferencias de las fibras sintéticas como el poliester (1.4) o las naturales como el algodón o la lana, son las que acaban produciendo la apariencia final en las telas.

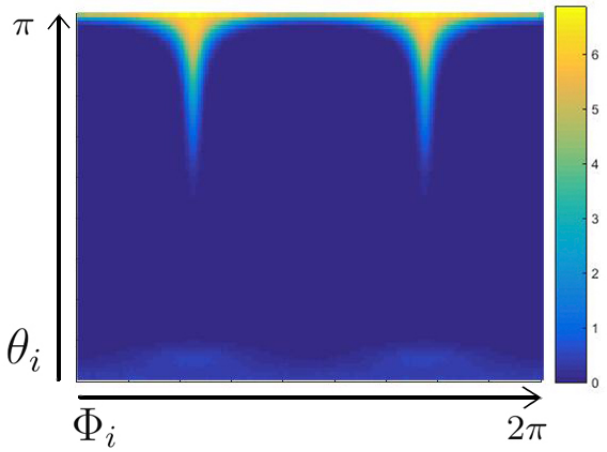


Figura 1.4: Representación de un extracto de la función masiva de apariencia de una fibra de poliester. Cada pixel indica el valor de la función con un ω_o fijo y ω_i definido por los ejes de la imagen. En este caso la dirección fija es $\Theta_o = 0$ y $\Phi_o = 0$, perpendicular a la fibra.

1.3. Muestreo por importancia

Calcular el transporte de luz es una tarea computacionalmente costosa, y el principal culpable de ello es la ecuación de render 1.1. En ella se calcula la radianza que se dirige hacia ω_o de un punto al que le llega luz desde potencialmente todas las direcciones comprendidas en la hemiesfera. Lo que supone un problema de integración numérica que debe ser resuelto.

Es imposible calcular analíticamente la integral para la luz que alcanza la superficie después de haber rebotado por el resto de la escena. Se necesita discretizar el espacio de la integral para poder obtener su resultado. Para ello se utiliza el algoritmo de *Monte Carlo* que propone estimar la iluminación indirecta mediante muestras aleatorias en el rango de integración. Este método garantiza que el resultado de la aproximación converge a la solución de la integral conforme aumenta el número de muestras.

El factor clave para utilizar *Monte Carlo* de forma eficiente es la distribución que siguen las muestras aleatorias. Si estas no se adaptan a la ecuación, como por ejemplo generando muestras de forma uniforme, el ratio de convergencia es muy lento, cercano a $O(\frac{1}{\sqrt{n}})$. Produciendo que para reducir el error a la mitad hay que cuadruplicar el número de muestras. El resultado es que para generar una imagen con poco ruido es necesario utilizar una número excesivo de muestras e invertir una gran cantidad

de tiempo. Mediante el principio conocido como *Muestreo por Importancia* se puede mejorar sustancialmente el ratio de convergencia a la solución de la integral. Este consiste en elegir una función de densidad que dirija las muestras hacia las regiones más relevantes de la integral, como se representa en la figura 1.5.

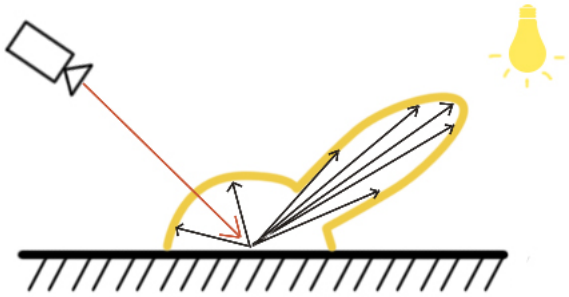


Figura 1.5: Ejemplo de las direcciones que se explorarían al resolver la ecuación de render guiando las muestras según la BRDF del material.

Debido a que la integral de render cuenta con variables que desconocemos es imposible ajustar las muestras a las regiones más importantes de la ecuación. Uno de esos términos desconocidas es la iluminación indirecta L_i que está determinada por la geometría de la escena. Aunque se tenga esta limitación aun hay partes de la ecuación de render que si son conocidas, como la BRDF. Y utilizar una técnica de muestreo que ajuste las muestras a una parte de la integral ya supone una mejoría importante en la convergencia del algoritmo. Que al final se traduce a una mejora en el tiempo necesario para generar una imagen con poco ruido.

Sin embargo, es complicado diseñar una estrategia de muestreo que funcione bien en todas las situaciones. Para mitigar este problema existe la técnica de Muestreo por Importancia Múltiple [Vea98] que mediante la combinación de distintas estrategias de muestreo obtiene un método más robusto. Como se puede comprobar en la imagen 1.6 eficazmente consigue enmascarar las debilidades de cada una de las estrategias de muestreo con las fortaleza del resto.

Esta es una técnica clave que se va a utilizar en este trabajo para combinar un conjunto de funciones de densidad y obtener una buena técnica de muestreo para funciones de apariencia masivas.

1.4. Contribución

Nuestro trabajo tiene como objetivo evitar unas de las principales taras que tiene utilizar datos tabulados para la representación masiva de la apariencia. El alto coste

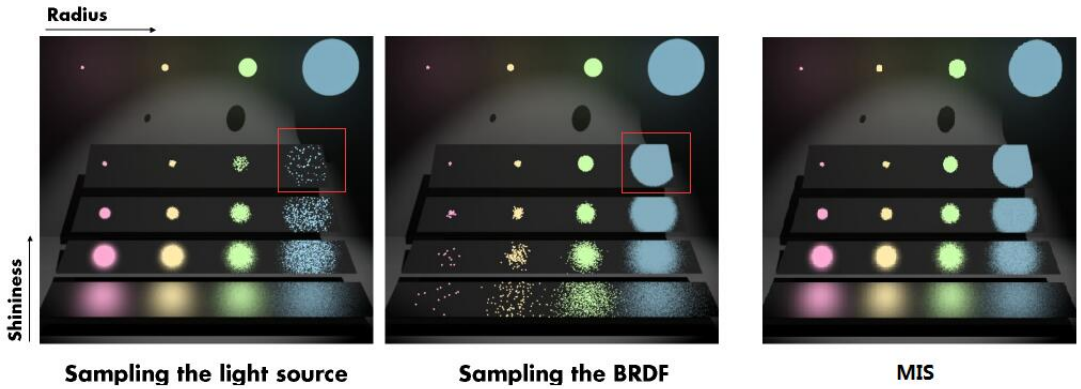


Figura 1.6: Comparativa entre dos técnicas de muestreo para superficies brillantes y la combinación de ellas mediante *Muestreo por Importancia Múltiple*. La escena cuenta con cuatro fuentes de luz esféricas de diferentes radios en la pare superior y con cuatro planos rectangulares con distinto nivel de rugosidad. Imagen extraída de la tesis doctoral de Eric Veach [Vea98].

temporal y el espacio en memoria requerido para almacenarlos suponen un problema a la hora de trabajar con ellos. El coste temporal es debido a la dificultad de realizar correctamente *Muestreo por Importancia*. Para ello se propone un método genérico para aproximar la distribución de los datos tabulados mediante *Gaussian Mixture Model*, junto a otro método que nos permite utilizar este conjunto de Gaussianas para implementar Muestreo por Importancia. Así se evita explorar un gran número de caminos de luz que tengan poca contribución al resultado final de la imagen.

De forma separada, la aproximación con *GMM* nos da, a cambio de introducir cierto error, un método compacto de almacenar la información de los datos tabulados de la función de apariencia. Reduciendo en varios ordenes de magnitud el coste en memoria de utilizar modelos empíricos.

1.5. Trabajo relacionado

Actualmente la generación de imágenes a través de escenas 3D lleva un gran recorrido a sus espaldas. Uno de los principales libros que recopila las bases de esta es *Physically Based Rendering*[1]. En él se hace un completo repaso a la teoría general, junto a los algoritmos más usuales y a su implementación.

Uno de los trabajos que ha motivado la realización de este TFG ha sido *An Appearance Model for Textile Fibers* [ACG+17]. Presentando un modelo empírico para definir la apariencia masiva de diversas fibras. En este ademas, se comparan con anteriores métodos analíticos confirmando que las simplificaciones de estos impedían

[1] http://www.pbrt.org/

representar toda la diversidad de estos materiales. Demostrando la necesidad de funciones de apariencia masivas.

Como se ha explicado en secciones anteriores, para estimar el "color" de un material es necesario estimar el resultado de la ecuación de render 1.1 mediante el método de *Monte Carlo*. Para obtener una imagen libre de ruido en un tiempo razonable es necesario utilizar técnicas como Muestreo por Importancia o Muestreo por Importancia Múltiple [Vea98] basadas en el principio de elegir una función de densidad o varias de ellas que adapten parte del integrando de la ecuación de render.

Multitud de trabajos se han centrado en obtener funciones de densidad adecuadas para generar muestras que garanticen mejoras en la calidad de las imágenes respecto al tiempo requerido para generarlas. Cuanto más conocimiento se tiene de la densidad que van a tomar los valores en la ecuación de render mejor es la estimación de su resultado. Para ello varios trabajos han abordado el problema intentando obtener la pdf analíticamente de los elementos conocidos de la ecuación de render, como por ejemplo la BRDF del material. En *Efficient BRDF Importance Sampling Using A Factored Representation* [LRR04] presentan un modelo de BRDF junto a una estrategia de muestreo basada en la función de densidad acumulada (Explicada en la sección 2.1) de su BRDF. Otros sin embargo, intentan adaptar la función de densidad a factores imposibles de analizar analíticamente. Cómo estimar la iluminancia que alcanza al punto x_i que depende directamente de la geometría de la escena y de como la luz se distribuye por ella. En esta idea trabaja *On-line Learning of Parametric Mixture Models for Light Transport Simulation* [VKv+14], que utiliza un aprendizaje progresivo de la distribución de la luz en la escena para generar una función de densidad que se adapte a $L_i(x_i, w_i)$ para cada posición x_i.

Múltiples trabajos han presentado a lo largo de los años varios modelos paramétricos para aproximar distintos aspectos de la generación de imágenes sintéticas, en busca de optimizar el proceso. En *On-line Learning of Parametric Mixture Models for Light Transport Simulation* [VKv+14], comentado anteriormente, aproximan la distribución de la luz incidente para cada punto de la escena mediante un conjunto de *GMMs* de 2 dimensiones. En otros trabajos como *An Efficient Representation for Irradiance Environment Maps* [RH01] se han utilizado Armónicos Esféricos para modelar los mapas de iluminación de la escena 3D. Con el mismo propósito *Real-Time Distant Light Filtering using Gaussian Mixture Model* [TEO] utiliza GMM para obtener una aproximación eficiente de los mapas de iluminación que permita utilizarlos en imágenes en tiempo real. En *Position-Normal Distributions for Efficient Rendering of Specular Microstructure* [YHMR16] también mediante un conjunto de Gaussianas aproximan los detalles de alta frecuencia que otorgan al metal bruñido su característica apariencia.

Como se puede observar, numerosos trabajos han optado por utilizar Gaussian Mixture Model, su adaptabilidad le permite modelar un distribuciones complejas y ser útil en un gran número de situaciones.

1.6. Distribución de la memoria

En la siguiente sección, «**Fundamentos Teóricos**», se va a hablar y profundizar en los aspectos importantes de nuestra propuesta. Se profundizara en *Importance Sampling* y *Multiple Importance Sampling*. Que juegan un papel fundamental en la solución de nuestro problema. Se introducirá el modelo parametrico *Gaussian Mixture Model* con el que se aproximaran los datos y el algoritmo *Expectatión-Maximization* con el que se ajustara *GMM* a la función de apariencia tabulada.

En la sección «**Aproximaciones y muestreo por importancia mediante *GMM***» se describe todo el proceso que se lleva a cabo y las decisiones tomadas para transformar una función de apariencia empírica en un *GMM*. Ademas de describir el método para generar muestras dado un *GMM* que nos permite implementar *Muestreo por Importancia Múltiple*.

En «**Resultados**» se exponen las comparaciones, gráficas y *renders* generados para validar tanto la etapa entrenamiento que ajusta *GMM* a los datos tabulados, como la técnica de muestreo que presentamos. Y acabando en el último apartado con las conclusiones obtenidas en la realización de este Trabajo Fin de Grado.

Capítulo 2

Fundamentos Teóricos

2.1. Muestreo por Importancia

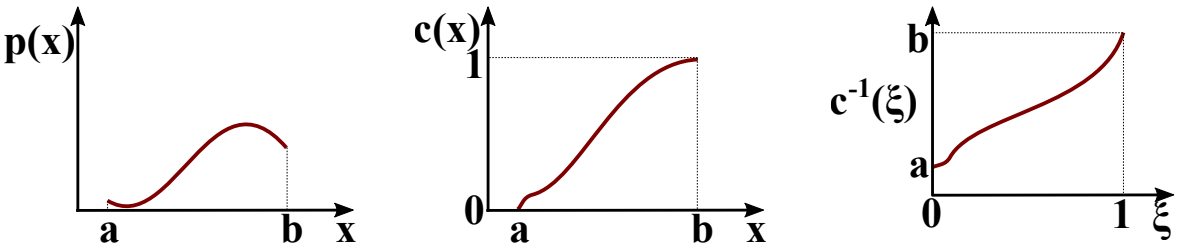
Muestreo por Importancia, como ya se ha comentado, nos permite realizar una integración eficiente mediante *Monte Carlo*. Al discretizar la integral de una función $f()$, se transforma en un sumatorio del integrando para un número de muestras dividido por la probabilidad de elegir esa muestra (*Probabilistic Density Function* abreviada como *pdf*).

$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{pdf(x_i)} \tag{2.1}$$

El algoritmo requiere que se puedan generar muestras según la *pdf* de forma eficiente. Si se tarda demasiado tiempo es posible que utilizando muestreo uniforme y más muestras pueda conseguir el mismo resultado en menos tiempo. Quitándole todo el sentido sentido a utilizar Muestreo por Importancia. Un método habitual para generar muestras instantáneas de una función de densidad es utilizar la inversa de la función de probabilidad acumulada (*cdf* por sus siglas en ingles). Lo primero que requiere es obtener la *cdf* definida por la ecuación 2.2 y después, si es, posible obtener la función inversa. Al evaluar la cdf^{-1} con un número aleatoria se obtendrá la muestra x_i siguiendo la función de densidad.

$$cdf(x) = \int_a^x pdf(t)dt \tag{2.2}$$

La idea detrás de este sistema, es que al crear la función acumulada las regiones con altos valores de la pdf se van a traducir en pronunciadas pendientes crecientes en la cdf (figura 2.1b). Al invertir la función (figura 2.1c) estas pendientes ocupan una zona mayor con lo que hay más probabilidad de que la variable aleatoria coincida con ellas.



(a) PDF (b) CDF (c) CDF^{-1}

Figura 2.1: Ejemplo gráfico de una función de densidad junto a su función acumulad y la inversa de la cdf

Lo ideal para resolver una integral por Muestreo por Importancia es obtener una función de densidad que cumpla $pdf(x_i) = cf(x_i)$ donde c es la constante de proporcionalidad, lo que significa que se ajusta perfectamente a los valores de la integral y convergería inmediatamente a la solución. Pero como se ha comentado en la sección 1.3 hay términos de la ecuación que se desconocen e imposibilitan producir esta *pdf*. Sin embargo, a lo que si se puede aspirar es a cumplir la condición anterior para partes del integrando que son conocidas.

2.2. Muestreo por Importancia Múltiple

La mayoría de problemas de integración numérica en informática gráfica son lo suficiente complejos, debido a por ejemplo discontinuidades o singularidades en el espacio del integrando, para que no sea suficiente con una técnica de muestreo para realizar *Monte Carlo* de forma eficiente. Por ello Eric Veach desarrollo Muestreo por Importancia Múltiple [Vea98], un método que siendo sorprendentemente sencillo mejora notablemente los resultados en este tipo de integrales. Su funcionamiento es simple, combinar varias técnicas de muestreo que funcionan bien en distintos casos, para que compensen los errores en las situaciones desfavorables. La estimación por *Monte Carlo* 2.1 se transforma en una combinación lineal de las técnicas de muestreo que componen el sistema ponderadas por un peso.

$$\int_a^b f(x)dx \approx \sum_{j=1}^{L} \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{w_j(x_i) f(x_i)}{pdf_j(x_i)} \tag{2.3}$$

Donde L es el número de las distintas técnicas de muestreo y n_j es el número de muestras generadas por la función de densidad pdf_j. Y donde $w_j(x_i)$ es el peso que

tiene la técnica de muestreo, que no tiene porque ser constante y pueden variar como una función dependiente de la muestra.

Veach propone una heurística balanceada para definir los pesos de las técnicas de muestreo. Sin embargo, cualquier otra política para el valor de los pesos es valida mientras cumpla que:

$$\sum_{j=1}^{L} w_j(x_i) f(x_i) = f(x_i) \tag{2.4}$$

En nuestro proyecto nos interesa generar muestras siguiendo la distribución que forma un GMM. Como se explica en la sección 3.3, teniendo un método para generar muestras siguiendo una distribución Gaussiana, podemos implementar Muestreo por Importancia Múltiple para producir muestras de todo el conjunto de Gaussianas que forman el GMM.

2.3. Gaussian Mixture Model

Es un modelo parametrico formado por un conjunto de distribuciones Gaussianas. Cada Gaussiana tiene su media μ y covarianza Σ independientes ademas de su prior ϕ, que representa el peso de la Gaussiana en el conjunto.

GMM define una distribución en la que los datos están generados por un conjunto Gaussianas. La probabilidad de un punto x en un GMM esta definida por la suma de todas las Gaussianas en ese punto multiplicadas por su peso.

$$p(x) = \sum_{i=0}^{k} \phi_i G(x|\mu_i, \Sigma_i) \tag{2.5}$$

La distribución Gaussiana es ideal para representar probabilidades desconocidas, ya que es una de las distribuciones que menos suposiciones hace sobre la densidad de los datos. Por ello *GMM* nos otorga un modelo paramétrico lo suficientemente flexible como para que se pueda adaptar a un amplio conjunto de situaciones. Ademas, existe una algoritmo de aprendizaje conocido como Expectation-Maximization con una implementación especifica para aproximar la densidad de los datos mediante *Gaussian Mixture Model*.

También hereda una propiedad muy interesante y útil de las Gaussianas, la convolución de un *GMM* sobre otro da como resultado un nuevo *GMM*. Lo que nos puede permitir combinar *GMM* que modelen distintas partes de la ecuación de render para obtener una técnica de muestreo más eficiente. Por ejemplo, combinando nuestro *GMM*, que modela la función de apariencia, con el de *Vorba et al.* [VKv+14], que represente la luz incidente.

Capítulo 3

Aproximación y muestreo por importancia mediante GMM

Nuestro trabajo expuesto a continuación a consistido en el desarrollo de dos métodos, que aunque siendo independientes, en conjunto nos ha dado la posibilidad de proponer una solución ante los problemas de utilizar modelos de apariencia masivos.

3.1. Descripción general del proyecto

En la introducción se ha expuesto la necesidad y la existencia de funciones de apariencia representadas mediante datos tabulados. Dichos datos tabulados pueden provenir de capturas o de simulaciones, y en general son una representación fiel y exacta de la apariencia de un material. Sin embargo, el incluir dichas funciones dentro de las ecuaciones del transporte de luz dificulta enormemente el muestreo por importancia óptimo de dichas ecuaciones, lo que supone un incremento en el (ya de por sí grande) tiempo requerido para generar una imagen con poco ruido.

Para poder realizar muestreo por importancia necesitamos una función de densidad de probabilidad (en inglés, *probability density function* o *pdf*) que adapte la distribución de las muestras lo mejor posible a la ecuación que está intentando muestrear por la técnica explicada en la sección 2.1. En el peor de los casos, siempre es posible un muestreo uniforme que ignore completamente la estructura de dicha función, pero en general esto es subóptimo y genera muestras con mucho ruido. Habitualmente se suelen buscar fórmulas analíticas que aproximen lo mejor posible (incluso exactamente según ciertos modelos de material) la función que describe la apariencia del material. Sin embargo, las funciones empíricas (tabuladas) son complejas, espacialmente grandes y con detalles de alta frecuencia; por ello también se las conoce como modelos masivos de apariencia.

Debido a la dificultad de aproximar dichas funciones de forma óptima mediante una función de distribución de probabilidad, en este trabajo se opta por un

modelo general estadístico: un *Gaussian Mixture Model* (*GMM*) que modela la distribución de probabilidad como una combinación lineal de distribuciones Guassianas n-dimensionales. Esto nos abre un abanico de posibilidades gracias a las propiedades de la distribución Gaussiana.

Específicamente, los modelos de apariencia tabulados en los que se basa este proyecto son de fibras textiles, que han sido generados mediante simulación [ACG+17]. Estos modelos permiten generar imágenes sintéticas de tela con una apariencia mucho más realista que otros trabajos anteriores. Sin embargo, los datos tabulados de la función de apariencia de una sola fibra pueden llegar a ocupar hasta 526 megabytes, que son difíciles de manejar y de muestrear óptimamente y por tanto presentan un desafío que será abordado en este trabajo. Nótese que cualquier otro modelo masivo de apariencia con datos tabulados podria haberse usado en su lugar y que este trabajo no es dependiente del modelo específico de datos tabulados elegido.

Además de unos datos de los que partir, necesitamos aproximar la función de distribución de probabilidad de dichos datos mediante un *GMM*. Aunque existen algoritmos de optimización generales, en este trabajo se ha elegido el algoritmo de *Expectation-Maximization* (expuesto en anexo A.1) que además de estar diseñado específicamente para aproximar un *GMM* a un conjunto de muestras, existe código disponible optimizado para dicha aproximación.

Debido a decisiones en el diseño que se comentan en la Sección 3.4, en vez de aproximar la matriz de cuatro dimensiones que forma la función de apariencia (donde dos dimensiones definen la dirección de salida de la luz y otras dos la dirección de entrada) se divide en varias rodajas de dos dimensiones fijando las direcciones de salida de la luz. Así para cada dirección de salida tenemos una matriz de dos dimensiones que define la solución para las distintas direcciones de entrada.

Finalmente, se ha implementado en la *API* de *Mitsuba* los métodos pertinentes para poder generar imágenes mediante esta contribución. *Mitsuba*[1] es un software de generación de imágenes sintéticas de código abierto orientado a la investigación. Cuenta con multitud de algoritmos experimentales que nos permitirán comparar nuestra solución con el estado del arte actual.

3.2. Aproximación mediante Gaussian Mixture Models

Un *Gaussian Mixture Model* (*GMM*) es un modelo parametrico formado por una combinación lineal de Gaussianas (descrito en la sección 2.3). Debido a la conocida

[1] https://www.mitsuba-renderer.org/

propiedad de las Gaussianas de ser una distribución probabilista que realiza muy pocas suposiciones sobre los datos, es un modelo perfecto cuando desconocemos completamente la función de distribución de los datos, adaptándose muy bien a nuestro caso si consideramos que las funciones de apariencia masivas esta formada por un conjunto de funciones de distribución desconocidas.

Existen diversos algoritmos de optimización que nos permiten aproximar unos datos tabulados mediante una combinación lineal de un *GMM*. Los métodos de optimización por descenso de gradiente buscan el mínimo de una función (de error en este caso) modificando los parámetros de las funciones en base al gradiente hacia dicho mínimo. Los algoritmos genéticos también permiten explorar el espacio de soluciones buscando aquellas que minimicen el error. Sin embargo son algoritmos por lo general lentos, y aproximar funciones multidimensionales con grandes irregularidades en sus datos tiene una convergencia incierta con una probabilidad alta de caer en mínimos locales.

La decisión principal de haber elegido *Expectation-Maximization* es el modelo de representación por el que hemos optado para aproximar las funciones de apariencia (descritas en la sección 1.2). Al contrario que los métodos genéricos de optimización, Expectation-Maximization es un algoritmo de aprendizaje ajustado específicamente para utilizarlo con GMM. El cual nos garantiza, al menos, alcanzar un máximo local de forma rápida.

Tal y como se detalla en el anexo A.1 el algoritmo, requiere de un conjunto de puntos distribuidos no uniformemente en el espacio del problema. Mediante el conjunto de puntos y un GMM inicial estima la función de probabilidad con la que se han distribuidos los puntos mediante el *Gaussian Mixture Model*. Sin embargo, como se puede observar el formato de los datos de entrada que requiere *Expectation-Maximization* es completamente distinto a nuestra situación. Nosotros contamos con la función de apariencia discreta de un material y queremos obtener una función de probabilidad que se ajuste lo máximo posible a esta función de apariencia.

Para poder utilizar EM y conseguir adaptar los datos tabulados mediante GMM se barajaron dos opciones. La primera, generar un conjunto de puntos siguiendo la probabilidad marcada por la función tabulada. Así adaptamos nuestros datos al algoritmo original. Pero acarreaba un problema, habría que generar un gran número de puntos para poder representar la proporcionalidad de los valores de la función. Por ejemplo, si el resultado de la función de apariencia en el punto r_1 es 5.5 y en el punto r_2 es 1 para que *Expectation-Maximization* ajustase la probabilidad de que en r_1 hay 5.5 más probabilidades de que haya una muestra que en r_2 habría que generar 11 y 2 muestras para esos puntos respectivamente. Y a no ser que se generase una cantidad desproporcional de puntos se va a introducir un error por intentar representar

los valores decimales de la función con un número entero de muestras.

La segunda opción, por la que nos hemos decantamos, consiste en adaptar tanto los datos tabulados de nuestro problema como el algoritmo de *Expectation-Maximization*. Como se ilustra con la figura 3.1, por cada celda de la matriz $n-$dimensional que contiene los datos tabulados se construye un punto r_i en el que su dirección es la coordenada de la celda en la matriz y que ademas contendrá un peso $p(r_i)$ con el valor de la función a aproximar en ese punto. De esta forma en vez de discretizar el valor de la función con un número entero de muestras se representa con un peso real en una sola muestra. El algoritmo de EM se adapta como en el trabajo *Multiple Subunit Fitting into a Low-Resolution Density Map of a Macromolecular Complex Using a Gaussian Mixture Model* [Kaw08] para tener en cuenta puntos con pesos en las ecuaciones del paso de Maximization (ecuaciones en anexo A.1).

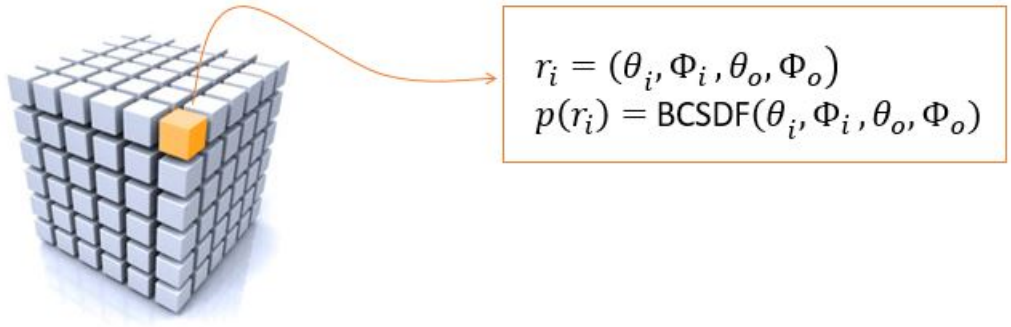


Figura 3.1: Representación de la transformación que se realizaría para una función de apariencia tabulada de una fibra, conocidas como *BCSDF*. La función como las comentadas anteriormente es almacenada como una matriz de 4 dimensiones $\theta_i, \Phi_i, \theta_i, \Phi_i$. Que al ser transformada en un conjunto de puntos, los indices de la celda definirán la posición de cada punto r_i y el valor de la celda el peso $p(r_i)$

Sin embargo, esto claramente contrasta con nuestros datos, que están distribuidos uniformemente pero cada uno de ellos tiene un valor distinto. La función de apariencia tabulada matematicamente es una matriz de n dimensiones, como se ilustra en la figura 3.1, pero el algoritmo de *Expectation-Maximization* busca obtener la distrubición de los puntos en el espacio $n-$dimensional. Para ello, inspirados en el trabajo Takeshi Kawabata [Kaw08], modificamos la fase de *Maximization* del algoritmo para que considere no sólo la distribución de los puntos en el espacio (que en nuestro caso es uniforme y no influye por si solo en la solución) sino también un peso asociado a cada uno de los puntos, que indica su importancia.

Las modificaciones que requieren las ecuaciones de la fase de Maximization consisten en multiplicar la probabilidad de cada punto de pertenecer a una Gaussiana $h(r_t)$ por el peso del punto $p(r_t)$. De esta forma si un punto tiene una alta probabilidad de

pertenecer a una Gaussiana pero el peso de este punto es pequeño, la repercusión sera inferior a otros puntos con mayores pesos. La ecuación del algoritmo original de Expectation-Maximization expuestas en el anexo A.1 se convertirían en las siguientes.

$$\pi_i = \frac{\Sigma_{t=1}^{L} p(r_t) h_i(r_t)}{\Sigma_{t=1}^{L} p(r_t)} \tag{3.1}$$

$$\mu_i = \frac{\Sigma_{t=1}^{L} p(r_t) h_i(r_t) r_t}{\Sigma_{t=1}^{L} p(r_t) h_i(r_t)} \tag{3.2}$$

$$\Sigma_i = \frac{\Sigma_{t=1}^{L} p(r_t) h_i(r_t) \times (r_t - \mu_i)(r_t - \mu_i)^T}{\Sigma_{t=1}^{L} p(r_t) h_i(r_t)} \tag{3.3}$$

De este modo, el algoritmo de *Expectation-Maximization*, que originalmente es para aproximar la distribución probabilista de los puntos, se puede aplicar también para aproximar un *GMM* a una función.

3.2.1. Optimización del número de Gaussianas

Como el algoritmo de *Expectation-Maximization* adaptado aproxima un número fijo de Gaussianas debemos de averiguar cual es el mejor número de componentes del *GMM* para adaptar la función. Para ello vamos a ejecutar el algoritmo con distinto número de Gaussianas y comparar los resultados. Por norma general el error de la aproximación se reduce casi siempre que se aumenta el número, aunque llega un momento en que la mejora es suficientemente despreciable como para que no sea rentable incluir una Gaussiana más. Para representar esto, se ha desarrollado un método iterativo ad hoc que penaliza los resultados conforme aumenta el número de Gaussianas.

Con el número de elementos definido del *GMM* se realizara de nuevo entrenamiento para obtener la aproximación final de la función tabulada. Aunque este con el objetivo de asegurar un mejor resultado se realizara de forma más exhaustiva que en el proceso anterior.

3.2.2. Implementación

Todo el proceso de entrenamiento explicado en esta sección con el objetivo final de obtener un *GMM* que aproxime la función de densidad de una función empírica se ha desarrollado en *Matlab*. Se ha realizado con *Netlab*[2], una *API* escrita para *Matlab* que implementa un conjunto de algoritmos de aprendizaje. Entre los algoritmos se encuentra el algoritmo original de *Expectation-Maximization* que hemos tenido que modificar para poder entrenar con puntos con pesos.

[2]http://www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/

3.3. Muestreo de GMM

En nuestro problema nos interesa obtener las direcciones de entrada de la luz que más información nos puedan aportar para resolver la ecuación de render mediante *Monte Carlo* y obtener el color de ese. Para ello vamos a generar muestras siguiendo la distribución formada por nuestra función de apariencia constituida por un conjunto de *GMM* de 2 dimensiones. Estas muestras se obtienen con la técnica de Muestreo por Importancia Múltiple. Recordando lo explicado en secciones anteriores, consiste en elegir una de las funciones de distribuciones del conjunto y generar la muestra con ella. Pero para ello hay que poder generar muestras de Gaussianas independientes.

Para muestrar por importancia con respecto a una única distribución Gaussiana de N dimensiones se utiliza el método expuesto en el libro Computational Statistics[3]: Se descompone la covarianza mediante el método de *Cholesky* y se obtiene la matriz triangular inferior A que cumple la ecuación 3.4.

$$A \cdot A^t = \Sigma \tag{3.4}$$

Se crea un vector $z = (z_1, z_2, ..., z_N)^t$ de N dimensiones con elementos independientes generados por una distribución normal estándar. Finalmente obtenemos el vector $\omega_i = (\theta_i, \Phi_i)$ con la suma de la media y el producto de la matriz triangular inferior con el vector aleatorio.

$$\omega_i = \mu + A \cdot z \tag{3.5}$$

Ahora para aplicar Muestreo por Importancia Múltiple (expuesto en la sección 2.2) tenemos que definir un método para elegir una Gaussiana entre todas las que forman la función de apariencia. La idea principal es escoger las Gaussianas de forma proporcional a sus prior, lo que equivale a muestrear con mayor probabilidad las que representen un mayor porcentaje de la probabilidad total del conjunto.

Se va a construir la función acumulada de los priors (*cdf* por sus siglas en ingles) definida en la sección 2.1. Pero al no tratarse de una función continua, la cdf se calcula mediante la función 3.6. Esta formara un vector del mismo tamaño que el de priors. Teóricamente cada celda define la probabilidad de que una variable aleatoria pertenezca o no a un elemento inferior o igual al de esa celda.

[3]Gentle, J.E. (2009). Computational Statistics. New York: Springer. pp. 315–316. doi:10.1007/978-0-387-98144-4.text

$$cdf(i) = \sum_{j=0}^{i} \pi_j \tag{3.6}$$

Ahora para obtener la Gaussiana tenemos que generar un número aleatorio comprendido en el rango de la función acumulada y realizar una búsqueda en el vector para encontrar la celda que contenga al aleatorio. Debido a que es una función estrictamente creciente usando búsqueda dicotómica obteniendo la Gaussiana con coste temporal $O(log(n))$ siendo n el número de Gaussianas del *GMM*. Tras aplicar la búsqueda obtendremos el indice de una Gaussiana de forma proporcional a sus priors.

Así hemos obtenido un método con el que podemos dado un GMM obtener una Gaussiana aleatoria pero siguiendo la probabilidad marcada por su peso en el conjunto. Después, podemos generar una muestra aleatoria siguiendo la función de distribución marcada por la Gaussiana que podremos utilizar para evaluar la ecuación de render 2.1 por *Monte Carlo*.

3.4. 4D vs. 2D

Las funciones de apariencia masiva con las que hemos trabajado son 4−dimensionales, por lo que originalmente se planteo aproximar con un GMM de 4 dimensiones que representase toda la función, y cuando fuese necesario fijar las dos dimensiones que representan la dirección de salida ω_o para poder obtener muestras en las otras dos dimensiones. Sin embargo, cuando avanzamos nos percatamos de que tenia ciertas complicaciones tanto en la fase de entrenamiento como a la hora de utilizarlo al generar imágenes sintéticas.

El algoritmo de Expectation-Maximization en cada iteración necesita realizar una gran cantidad de operaciones que requieren de todo el conjunto de datos. El resultado es que para la cantidad de datos que contiene una función de apariencia requería de un día entero para completarse. Por poner un ejemplo, las dimensiones de las funciones de apariencia de las fibras [ACG+17] son $180 \times 90 \times 180 \times 180$ lo que produce mas de 524 millones de puntos. Si ademas necesitásemos realizar el proceso de optimización del número de Gaussianas. Obtener el Gaussian Mixture Model que aproxima la función tardaría semanas.

Para generar muestras aleatorias es necesario fijar las dos dimensiones que forman la dirección de salida ω_o y obtener así una muestra aleatoria de las otras dos. Sin embargo, esto requería recalcular los pesos de las Gaussians en función del valor de su integral con la dirección de salida fija. Lo cual supone una importante penalización en el

rendimiento del algoritmo, que ademas al realizarse durante la generación de imágenes implicaría un incremento en el tiempo requerido que inutilizaría nuestra solución.

Tal y como se ha realizado en otros trabajos [VKv+14] se tomó la decisión de utilizar un conjunto de GMMs de 2D. Esto es posible porque en la mayoría de algoritmos de generación de imágenes sintéticas solo vamos a necesitar generar muestras para la dirección de salida. Con esta decisión, se ha podido realizar el aprendizaje de una función masiva de apariencia en un tiempo razonable y realizar el muestreo para cualquier dirección de salida fija mediante la interpolación bilineal de los 4 GMM más cercanos.

Los priors de cada *GMM* se ponderan, al igual que en interpolación bilineal (figura 3.2), por la distancia inversa entre la dirección fija del GMM y la dirección de entrada. Así si una dirección de entrada esta muy próxima a una que contenga un *GMM* habrá mucha más probabilidad de que se elija una Gaussiana de ese conjunto y no de los otros tres.

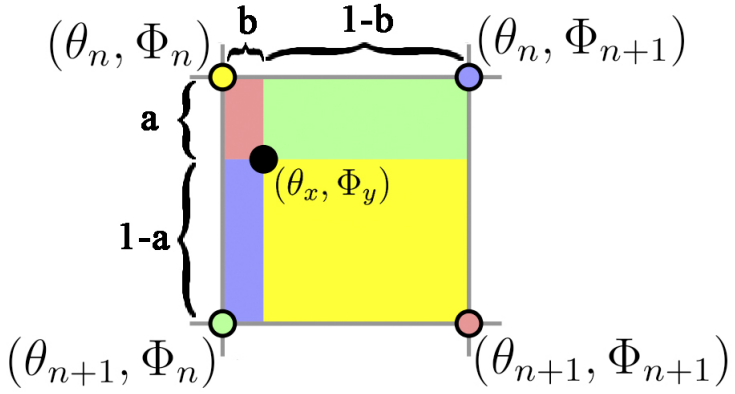


Figura 3.2: Para calcular el valor de una muestra 2D intermedia entre una función discreta se realiza interpolación bilineal. El resultado es aproximado mediante el valor de las cuatro muestras más cercanas definidas en la función ponderadas por la distancia inversa. Por ejemplo, el valor de (θ_n, Φ_n) se multiplicaría por $(1-b)(1-a)$, que sumado a los otros tres valores se calcularía (θ_x, Φ_y).

Finalmente, para realizar *Monte Carlo* (ecuación 2.1) se necesita dividir el resultado de la función $f(x_i)$ por la probabilidad de ser elegida la muestra x_i. Para calcular esta probabilidad se realiza interpolación bilineal (figura 3.2) con los resultados de las 4 *GMMs* para la ecuación 2.5 donde x es la dirección de entrada de la luz ω_i obtenida mediante el método descrito.

$$\begin{aligned} pdf(r_i) =& (1-a)(1-b)p_{\theta_i,\Phi_i}(\omega_o) + (a)(1-b)p_{\theta_i+1,\Phi_i}(\omega_o) \\ &+ (1-a)(b)p_{\theta_i,\Phi_i+1}(\omega_o) + (a)(b)p_{\theta_i+1,\Phi_i+1}(\omega_o) \end{aligned} \tag{3.7}$$

3.5. Utilización de la aproximación con GMM como función de apariencia

Tras el proceso de aprendizaje con Expectation-Maximization se obtiene un conjunto de GMM que aproxima la función de densidad de probabilidad que se adapta a la función de apariencia masiva de distintos tipos de fibras. Mediante esta pdf tenemos la oportunidad de generar muestras que se adapten a la función de apariencia y realizar Muestreo por Importancia Múltiple.

Debido al tamaño que requieren las funciones de apariencia masivas, consideramos como una ida muy interesante tener la opción de usar la aproximación mediante GMM como sustituto a la función de apariencia. Ya que por ejemplo, en el caso de las fibras se pasaría de necesitar más de 500 *MB* del fichero que contiene la matriz 4D que la defina a no superar los 200 *KB* con *GMM*. Gracias a la alta calidad de las aproximaciones que se han realizado, se abordó el problema de la siguiente forma. Como se ha explicado en la sección 2.1 podemos considerar que nuestra aproximación cumple $pdf(x_i) \approx cf(x_i)$ y ya que tenemos acceso a la función f, que en el caso de las fibras es la $BCSDF$ podemos calcular el factor de proporcionalidad c que nos permite hacer la siguiente derivación del método de *Monte Carlo* para la ecuación de render 1.2:

$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{L_i(x,\omega_i)BCSDF(x,\omega_i,\omega_o)}{pdf(\omega_i)} \tag{3.8}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\frac{L_i(x,\omega_i)BCSDF(x,\omega_i,\omega_o)}{cBCSDF(x,\omega_i,\omega_o)} \tag{3.9}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\frac{L_i(x,\omega_i)}{c} \tag{3.10}$$

$$\approx \frac{1}{cN}\sum_{i=1}^{N}L_i(x,\omega_i) \tag{3.11}$$

A pesar de que con este método a la ecuación de render se le añade error debido a la aproximación de la *BCSDF*, se evita la necesidad de utilizar una de las pesadas funciones de apariencia de las fibras. Ademas de solo necesitar calcular la luz incidente para un conjunto de muestras aleatorias generadas mediante nuestro GMM. En la figura 4.8b se muestra que el error introducido casi no se percibe en la imagen final.

Capítulo 4

Resultados

En esta sección se van a presentar los resultados y validaciones obtenidos en los distintos métodos desarrollados en este proyecto, ademas de mostrar los resultados obtenidos con imágenes sintéticas generadas mediante nuestra contribución.

4.1. Aproximación GMM

El factor clave para obtener buenos resultados es el número de Gaussianas. Hay matrices 2D que son significativamente más sencillas de aproximar y un número elevado de Gaussianas produciría sobreajuste. En el caso contrario un número reducido produce demasiado error para resultar una aproximación útil. Por ello en la etapa de elección del número de Gaussians se penalizan las soluciones conforme aumenta el número, evitando el sobreajuste. En el anexo A.2.3 se adjunta una imagen comparativa del resultado de aproximar una función de apariencia con distinto número de Gaussianas.

Para comprobar el error de las aproximaciones obtenidas se ha calculado la diferencia entre las matrices 2D originales y los valores del Gaussian Mixture Model para las mismas direcciones de entrada de la luz. En la figura 4.1 se observa que el algoritmo no es capaz de aproximar correctamente los casos con valores altos en los limites de la matriz. A parte de este ejemplo, en el anexo A.2 se incluyen más comparaciones de la aproximación de la fibra de seda para distintas direcciones de salida de la luz. De forma general, exceptuando el caso descrito anteriormente, GMM consigue ajustar con mucha eficacia los datos originales.

Uno de los problemas que tienen los métodos de simulación con los que se obtienen algunas funciones empíricas es el ruido. Estos métodos de simulación siguen un proceso similar al de generar imágenes sintéticas mediante la simulación del transporte de luz. Pero a diferencia de que los rayos de luz se trazan desde las fuentes de luz hasta los sensores, en vez de desde la cámara hacia la luz como se realiza para generar imágenes. Sin embargo gracias a la capacidad del algoritmo de *Expectation-Maximization* para

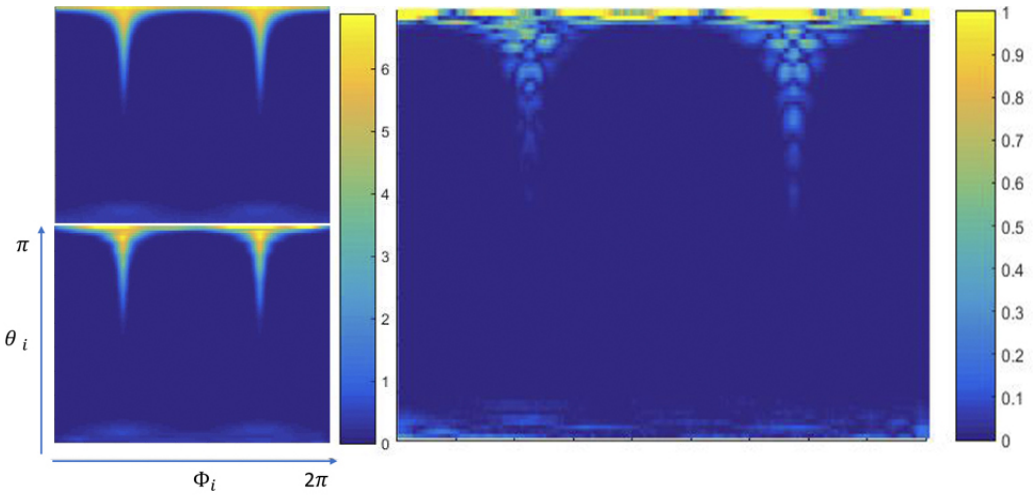


Figura 4.1: Comparación entre la función de apariencia de una fibra de poliester para una dirección de salida de la luz $\theta_o = 0$ y $\Phi_o = 0$. La imagen superior izquierda es la función de apariencia original, y la imagen inferior la aproximación mediante un conjunto de 50 Gaussianas. A la derecha se muestrar el error para cada pixel de la aproximación respecto a la original. Destacar que la escalara de colores en la imagen de error representa número muy inferiores a los de las funciones de apariencia de la izquierda.

generalizar y deducir variables ocultas se consigue mitigar el error en los resultados de las aproximaciones. Un ejemplo de esto es la función de apariencia de la fibra de algodón (figura4.2), lo que produce más variabilidad en los resultados de la ecuación de render 1.2 que acaba resultando en más ruido.

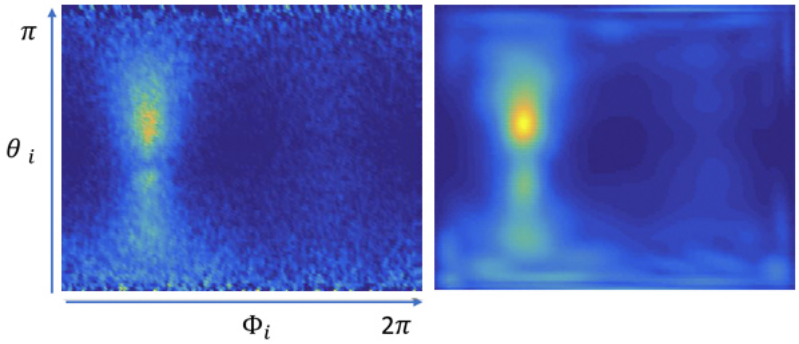


Figura 4.2: Comparación entre la función de apariencia de una fibra de algodón. A la izquierda, la función original en la que se puede observar el ruido generado por la simulación, y a la derecha la aproximación con 50 Gaussianas.

4.2. Muestreo

La base de utilizar Muestreo por Importancia de forma eficiente radica en poder generar un conjunto de muestras siguiendo una distribución de probabilidad que se ajuste a la ecuación de render. Sin embargo en ella intervienen factores que se salen

fuera de nuestro control como la distribución de la luz que llega al material que depende de la geometría de la escena. Por ello se intenta conseguir que la probabilidad de distribuir las muestras se adapte a todas las partes posibles de la ecuación.

En nuestro caso se ha ajustado la distribución de probabilidad a la función de apariencia. Y como se ha demostrado en la sección anterior se han obtenido muy buenas aproximaciones de la distribución de probabilidad. De forma visual en la figura 4.3 se puede comprobar que el método descrito en la sección 3.3 es capaz de generar muestras en las regiones mas importantes de la función de apariencia. Lo que nos permite aplicar *Monte Carlo* para resolver la ecuación de render 1.2 de forma eficiente. En el anexo A.3 se adjuntan mas ejemplos de muestras generadas desde otros GMM.

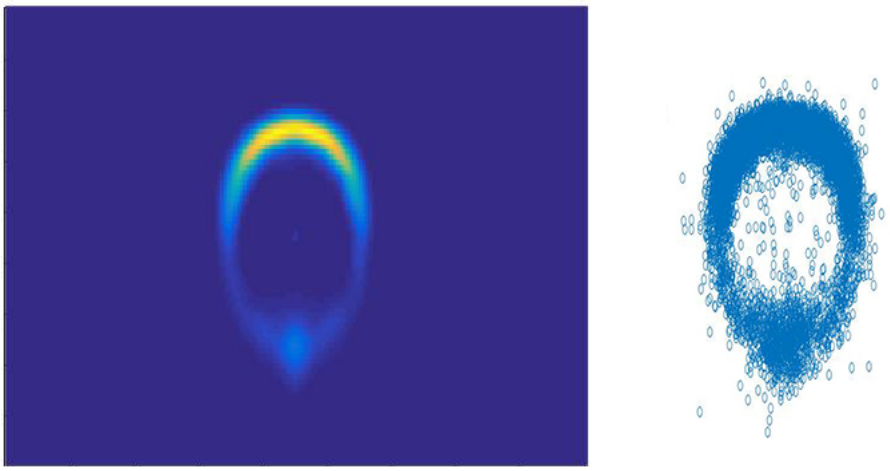


Figura 4.3: *Gaussian Mixture Model* de la función de apariencia de una fibra de poliester para la dirección $\theta_o = \pi/4$ y $\Phi_o = 0$ junto a 10000 muestras generadas a partir de ese *GMM*

Sin embargo, para validar que realmente se han generado bien las muestras se ha comparado la convergencia de aplicar *Monte Carlo* mediante Muestreo Uniforme y por *Muestreo por Importancia Multiple* con *GMM* respecto al valor real de la ecuación (calculado con muestreo uniforme y un millón de muestras). En la gráfica 4.4 se presenta el resultado de la ecuación 1.2 para distintos números de muestras (eje x). Ademas también se incluye en la gráfica los resultados obtenido de utilizar *GMM* en sustitución de la *BCSDF* (sección 3.5).

- $US - Tab$ son los valores de aplicar Muestreo Uniforme junto a la BCSDF.
- $US - GMM$ son los valores de aplicar Muestreo Uniforme junto a GMM como función de apariencia,
- $IS - Tab$ son los valores de aplicar Muestreo por Importancia Múltiple junto a la BCSDF.

– $IS - GMM$ son los valores de aplicar Muestreo por Importancia Multiple junto a GMM como función de apariencia.

El ratio de convergencia de utilizar Muestreo por Importancia Múltiple junto a Gaussian Mixture Model en sustitución de la función de apariencia es constante debido a que, como se ha demostrado en la sección 3.5, acabaría simplificada en la constante de proporcionalidad c.

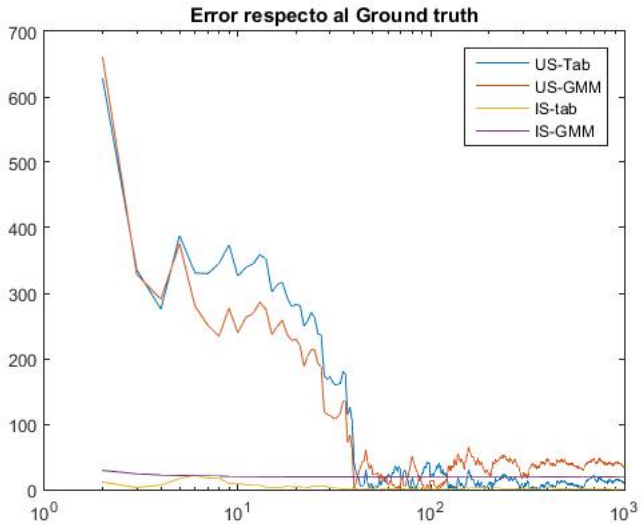


Figura 4.4: Comparación en la convergencia de aproximar la integral de la ecuación de render 1.1, obviando de la ecuación los términos $L_i()$ *cos* y *sin* para simplificarla. La comparación se realiza para la tecnica de muestreo: Muestreo por Importancia Multiple y muestreo uniforme. Y para la utilización de la función de apariencia original (tab) o la función de apariencia aproximada con GMM.

4.3. Imágenes sintéticas

A continuación se van a mostrar una serie de imágenes para comparar el resultado final que hemos obtenido gracias a nuestro trabajo. En la figura 4.5 se puede observar, sobre todo en las regiones oscuras de la tela, que resolver *Monte Carlo* mediante Muestreo por Importancia Múltiple con nuestros *Gaussian Mixture Model* genera menos ruido que resolviéndolo con muestras uniformes.

En la figura 4.6 se puede observar claramente el ruido que produce utilizar Muestreo Uniforme, para obtener resultados aceptables requiere utilizar una gran cantidad de muestras. Mientras que con Muestreo por Importancia Múltiple con tan solo 128 muestras es capaz de estimar adecuadamente la solución de la ecuación de render y generar una imagen prácticamente libre de ruido.

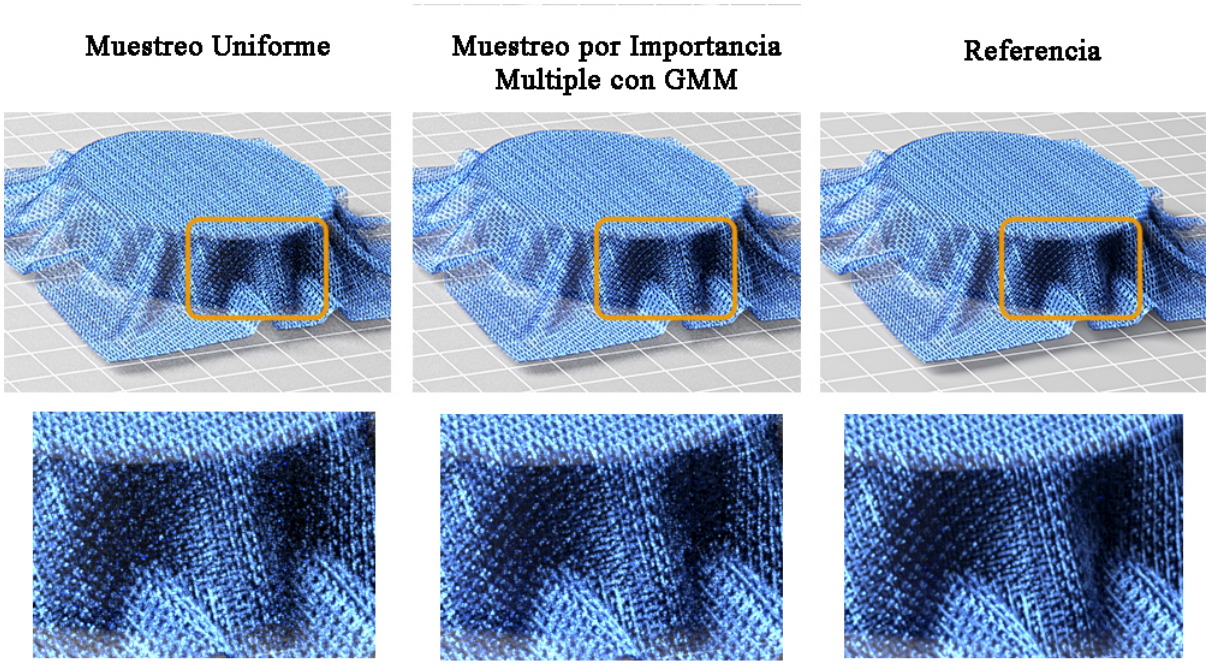


Figura 4.5: Comparación entre dos imágenes sintéticas de una tela de seda junto a una imagen de referencia. con distintas técnicas de muestreo, ambas con 32 muestras por pixel, junto a una imagen de referencia obtenida a partir de 2000 muestras por pixel.

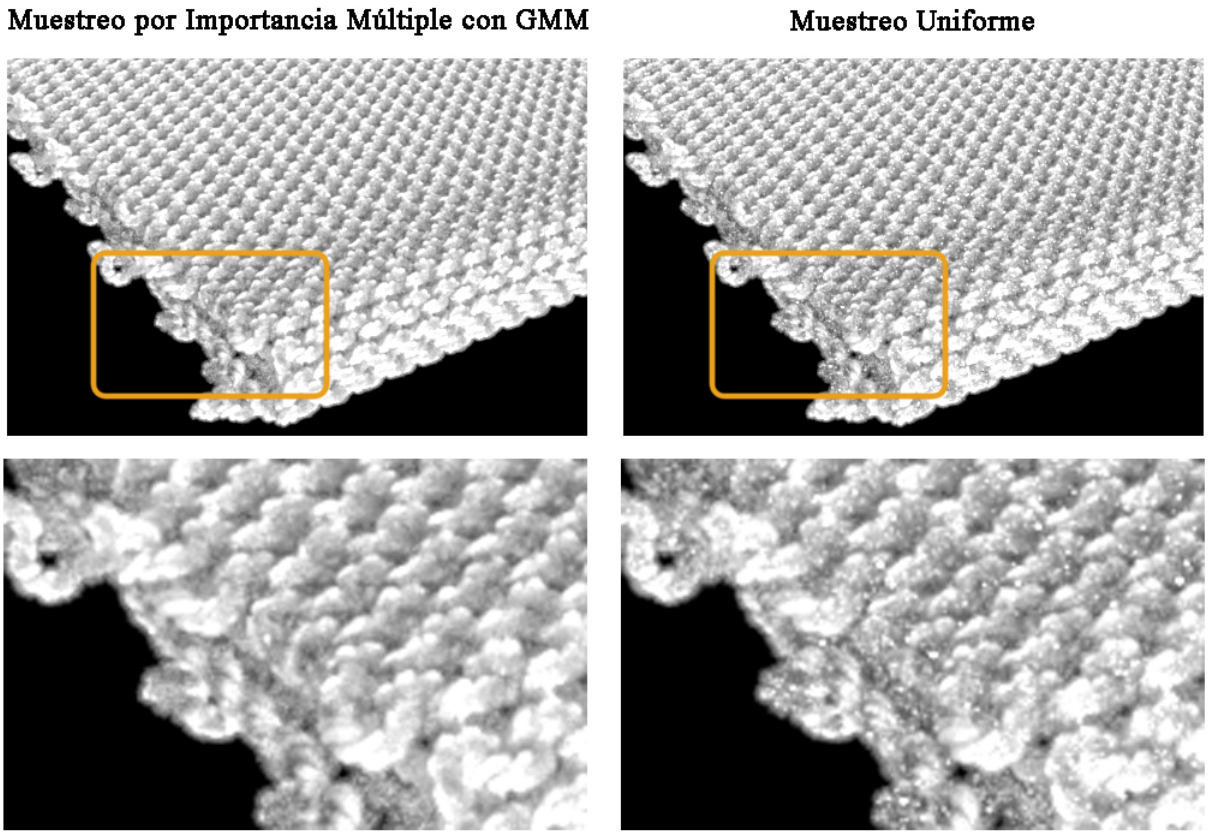


Figura 4.6: Dos imagenes de una tela de algodón en el que en ambas se han utilizado 128 muestras por pixel.

En los casos anteriores, el tiempo requerido para obtener la imagen con nuestro *GMM* ha costado aproximadamente 3 veces más que utilizando muestreo uniforme. Generar una muestra aleatoria uniformemente es sustancialmente más rápido que realizar el proceso descrito en la sección 3.3 para generar muestras de un *Gaussian Mixture Model*. Sin embargo la ventaja de nuestro algoritmo aparece cuando comparamos el tiempo requerido para obtener una imagen similar. En la figura 4.7 obtenemos un resultado muy parecido a la imagen de referencia, que ha utilizado 2000 muestras y requerido de 3 horas para generar la imagen. Mientras que nuestra imagen ha utilizado 128 muestra y ha tardado 52 minutos.

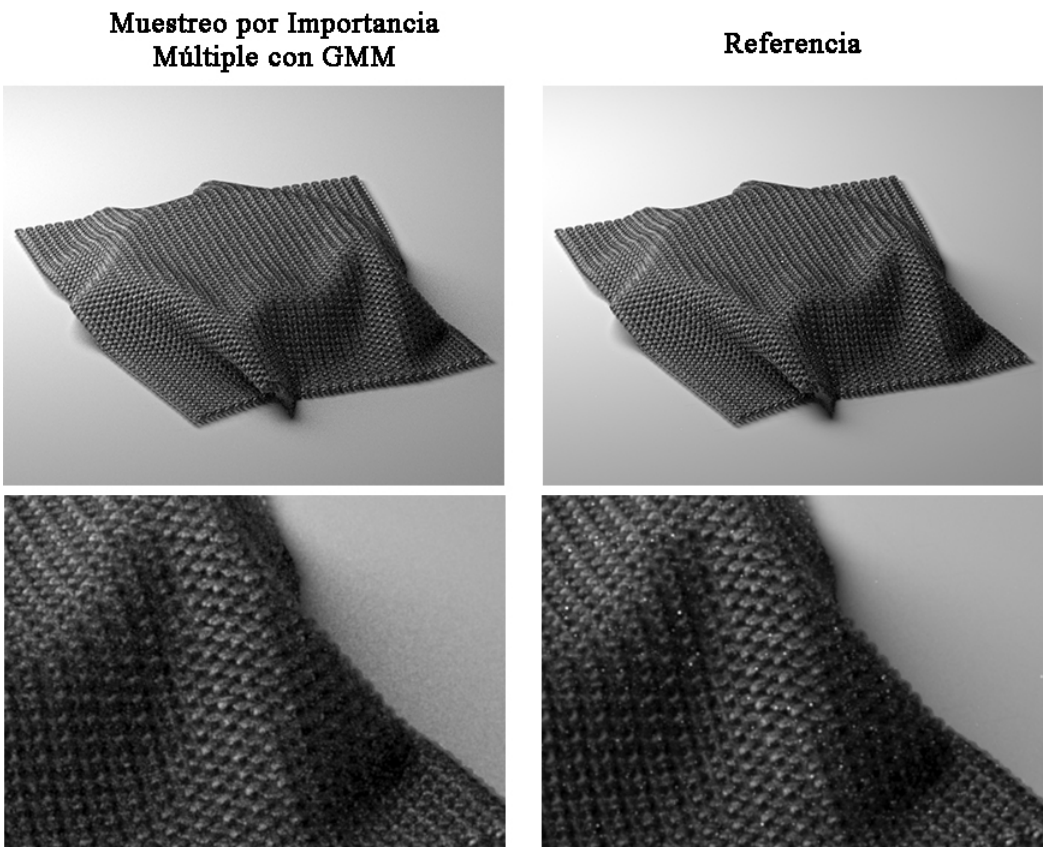
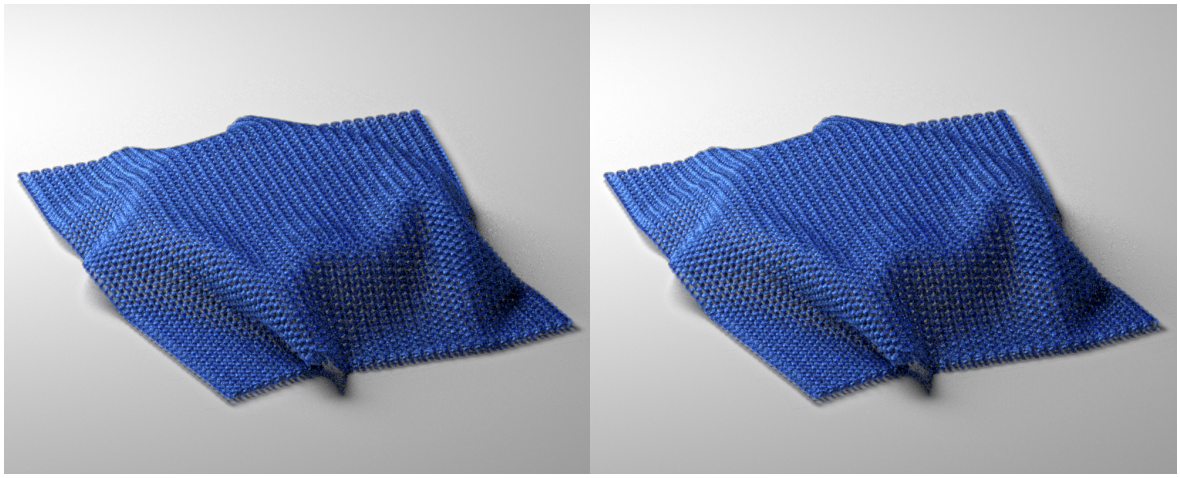


Figura 4.7: Comparación entre dos telas de lana en la que nuestra solución (imagen de la izquierda) ha requerido de 52 minutos para completarse, mientras que la imagen de referencia ha necesita más de 3 horas para generarse.

Las funciones de apariencia masivas son un requisito fundamental si se quiere alcanzar imágenes fotorrealistas con materiales complejos. Pero debido al tamaño que ocupan generar una imagen con diversas funciones de este estilo va a requerir de un ordenador especializado con memoria RAM suficiente como para alojar los datos tabulados que forman la función de apariencia. Mediante la aproximación de las funciones con *GMM* se ha permitido reducir ese requerimiento sustancialmente y obtener resultados, como muestra la figura 4.8 prácticamente irreconocibles.



(a) *BCSDF.* (b) Aproximación de *BCSDF* con *GMM.*

Figura 4.8: La figura de la izquierda ha sido generada con Muestreo por Importancia Múltiple para muestrear la *BCSDF* de una fibra de lana. mientras que la figura de la derecha se ha utilizado Muestreo por Importancia Múltiple pero utilizando la aproximación con *GMM* de la *BCSDF*. Los resultados son prácticamente similares y la segunda imagen no ha requerido de los 500 MB extra de la función de apariencia tabulada.

Capítulo 5

Conclusiones

En este trabajo hemos desarrollado de forma satisfactoria dos métodos que nos han permitido mitigar los inconvenientes de las funciones masivas de apariencia. Hemos implementado una técnica de aprendizaje capaz de aproximar cualquier tipo de función tabulada mediante *Gaussian Mixture Model*, y un método de muestreo que nos permite resolver la ecuación de render para funciones de apariencia masiva de forma eficiente. También para evitar el problema del gran espacio necesario para utilizar estas funciones se ha propuesto la utilización de nuestra aproximación como sustituto a la función de apariencia. Consiguiendo reducir hasta más de 4000 veces el tamaño requerido para generar una imagen que utilice una función de apariencia masiva.

5.1. Discusión y trabajo futuro

Como ya se comentó en la sección 3.4 debido a ciertos problemas y desventajas se decidió abandonar la idea de utilizar una aproximación de 4 dimensiones de las funciones de apariencia a favor de dividirlas en rodajas de 2 dimensiones. Aunque dividir la dimensión tiene sus ventajas a la hora del tiempo requerido para aproximar los datos, también tiene ciertos inconvenientes respecto a la otra opción. Es altamente probable que se requiera un menor número de Gaussianas totales para aproximar los datos, aunque las Gaussianas 4D ocupan más espacio que las bidimensionales. Ademas al dividir en rodajas es necesario realizar interpolación bilineal para obtener la información que no ha sido aproximada, introduciendo error en los valores recuperados. La complejidad computacional de obtener la probabilidad de Gaussiana 4D para una dirección de entrada es lo que hizo decantarse la balanza hacia la división de los datos.

Aunque en balance general las aproximaciones de las rodajas 2D son bastante buenas, ocurren ciertas singularidades en los bordes. Debido a las características del propio algoritmo de *Expectation-Maximization* le es imposible aproximar bien la información que se encuentra en los límites de la matriz. Para solucionar esto se han

planteado varias ideas, una de ellas fue duplicar la matriz en los bordes y hacer un efecto espejo para eliminar los límites durante el entrenamiento. Otra solución y quizá la más interesante para desarrollar en el futuro es la de sustituir las Gaussianas por Gaussianas Esféricas. Aunque estamos tratando la matriz de dos dimensiones como un plano, realmente no es así. Se trata de una proyección de la esfera que define en coordenadas angulares las direcciones de entrada y de salida. Al estar las Gaussianas Esféricas definidas en la esfera no existen para ellas los limites del plano 2D, por ejemplo cuando Φ pasa de 2ϕ vuelve a la coordenada 0. Requeriría adaptar tanto el método de aprendizaje como el de Muestreo por Importancia Múltiple.

Como se ha comentado al principio del trabajo, las Gaussianas cuentan con propiedades muy interesantes que podrían ser útiles para trabajos futuros. Combinar nuestro método con otros que utilicen GMM es una opción muy prometedora que puede mejorar la técnica de muestreo y permitir generar imágenes en menos tiempo.

5.2. Conclusiones personales

A pesar de haber sido el trabajo más laborioso que he realizado, todo lo que he aprendido, la experiencia que he conseguido y los resultados obtenidos han hecho que sea una experiencia más que gratificante. He tenido la oportunidad de participar y aprender en un equipo de investigación fantástico que me ha descubierto la faceta investigadora de la universidad.

Capítulo 6

Bibliografía

[ACG+17] Carlos Aliaga, Carlos Castillo, Diego Gutierrez, Miguel A. Otaduy, Jorge Lopez-Moreno, and Adrian Jarabo. An appearance model for textile fibers. *Computer Graphics Forum*, 36(4):35–45, 2017.

[Kaj86] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986.

[Kaw08] Takeshi Kawabata. Multiple subunit fitting into a low-resolution density map of a macromolecular complex using a gaussian mixture model. *Biophysical Journal*, 2008.

[LRR04] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient brdf importance sampling using a factored representation. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 496–505, New York, NY, USA, 2004. ACM.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 497–500, New York, NY, USA, 2001. ACM.

[TEO] Ozkan Anıl Töral, Serkan Ergun, and Aydın Oztürk. Real-time distant light filtering using gaussian mixture model.

[Vea98] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[VKv+14] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)*, 33(4), aug 2014.

[YHMR16] Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)*, 35(4), 2016.

[ZW07] Arno Zinke and Andreas Weber. Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):342–356, March 2007.