Universidad Zaragoza
1542

Trabajo Fin de Grado

Técnicas de muestreo para simulación de transporte de luz sin línea de visión

Sampling techniques for non-line-of-sight light transport simulation

Autor

Ismael Tienda Hernando

Directores
Diego Royo Meneses
Jorge García Pueyo

Ponente
Adolfo Muñoz Orbañanos

Titulación
Grado en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2023

Técnicas de muestreo para simulación de transporte de luz sin línea de visión

RESUMEN

El problema de la imagen sin línea de visión o *non-line-of-sight (NLOS) imaging* consiste en la reconstrucción de escenas parcial o totalmente ocultas analizando la luz indirecta capturada por una cámara en una superficie visible secundaria. La imagen *NLOS* presenta múltiples aplicaciones en diversos ámbitos, como en la conducción autónoma, al poder ver peatones o coches ocultos tras esquinas, o en la imagen médica, al reconstruir órganos o tejidos ocultos tras varias capas de la piel. Para ello, se necesitan dispositivos especializados, como cámaras y láseres ultra-rápidos, capaces de trabajar con una resolución temporal del orden de picosegundos. Ya que este hardware tiene un elevado coste de adquisición y operación, aparece la simulación del transporte de la luz transitorio, siendo muy útil para el prototipado de aplicaciones de imagen transitoria. Sin embargo, estas simulaciones pueden tener un alto coste computacional al tratar la luz indirecta en escenas *NLOS*. Es en el trabajo de Royo et al. [1] donde se plantean nuevos métodos de simulación para escenas *NLOS* detrás de *una* esquina. Sin embargo, presentan dificultades a la hora de simular entornos detrás de *dos* esquinas, siendo este el foco de este trabajo.

Para ello, en este trabajo se implementan diversas técnicas para simulación con el objetivo de mejorar los resultados obtenidos en escenas ocultas detrás de dos esquinas, siendo estas *Hidden Geometry Rejection Sampling* (*HGRS*) y *Kernel Density Estimation* (*KDE*). *HGRS* es una técnica basada en el muestreo por rechazo estadístico, con el fin de obtener muestras de la escena visibles desde un punto determinado. *KDE* es una técnica estadística de estimación de densidad, que permite hacer un uso más preciso de las muestras además de aplicar un suavizado a la señal.

Finalmente, se ha realizado una evaluación de dichas técnicas utilizando métricas cualitativas y cuantitativas respecto a convergencia y tiempo de generación de la imagen, utilizando escenas detrás de una o dos esquinas. Se han obteniendo mejoras de un 93 % en el error cuadrático medio y una reducción del tiempo de hasta un 91 % dependiendo de la técnica utilizada con respecto al simulador de Royo et al. [1] en escenas detrás de una esquina, y de hasta un 87 % en convergencia y un 99 % de reducción de tiempo en escenas detrás de dos esquinas. Además, se ha comprobado que algunas de las técnicas implementadas proporcionan también una mejoría general en escenas con línea de visión.

Índice

Capítulo 1

Introducción

1.1. Contexto y problemas abordados

El problema de la imagen sin línea de visión, o *non-line-of-sight (NLOS) imaging* consiste en la reconstrucción de escenas parcial o totalmente ocultas analizando la luz indirecta capturada por un sensor en una superficie visible secundaria [2] [3]. Para esto es necesario un láser ultra-rápido que emite un pulso de luz hacia dicha superficie visible, y un sensor ultra-rápido con una resolución de picosegundos (alrededor de un trillón de fotogramas por segundo) que captura la luz indirecta proveniente de la escena oculta. De esta forma, es posible medir el tiempo de vuelo de la luz desde el láser a la superficie visible hasta la escena oculta y de la escena oculta a la superficie visible hasta el sensor. Este camino de la luz se puede observar en la Figura 1.1. Estas medidas de tiempo son usadas por los algoritmos de reconstrucción para obtener una estimación de los objetos que se encuentran detrás de la esquina.

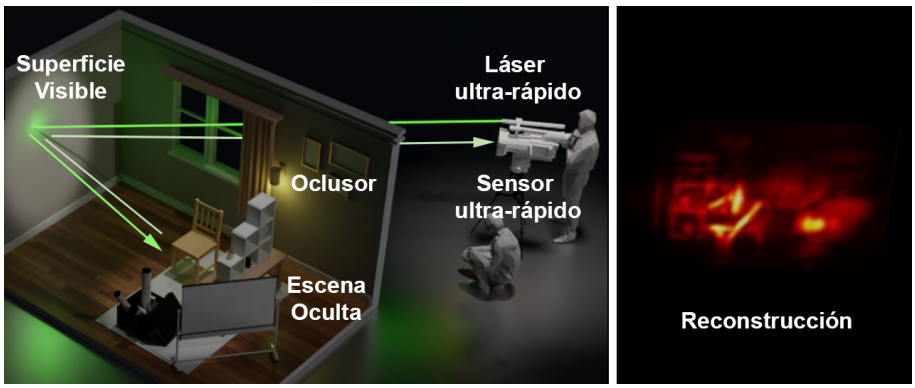

Figura 1.1: **Izquierda:** Ejemplo de configuración para una escena *NLOS* detrás de una esquina y camino de la luz desde el láser emisor de luz hasta el sensor. Se pueden apreciar los diferentes elementos propios de un sistema de imagen sin línea de visión. **Derecha:** Reconstrucción de la escena detrás de la esquina. Adaptado de Liu et al. [4]

Este tipo de reconstrucciones pueden resultar de gran utilidad en distintos escenarios, como podrían ser labores de rescate, imagen médica o conducción autónoma (Figura 1.2). Como ejemplo, un coche autónomo puede ser capaz de ver a un peatón o a otro coche detrás de una esquina y anticipar sus acciones.

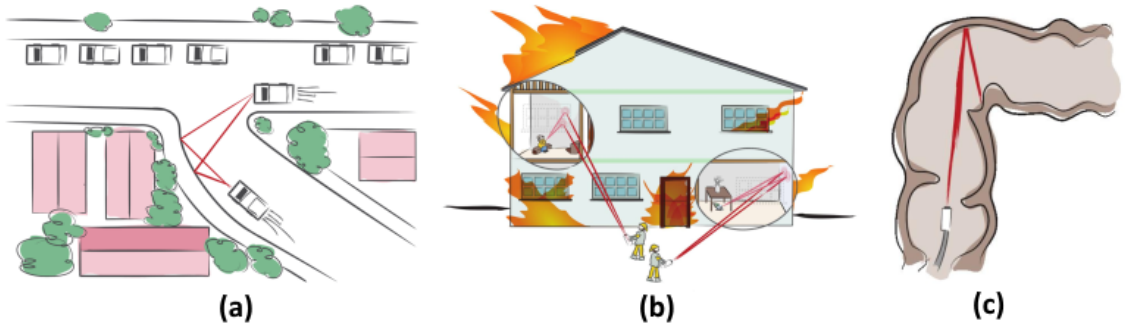

Figura 1.2: Diferentes escenarios de utilidad para la imagen fuera de línea de visión. **(a)** Detección de objetos detrás de esquinas en la conducción autónoma. **(b)** Localización de supervivientes en labores de rescate. **(c)** Visualización de regiones inaccesibles en imágenes médicas. Fuente: Maeda et al. [2].

Los trabajos recientes de imagen *NLOS* se centran en la reconstrucción de escenas ocultas detrás de una esquina. Es en el estudio de Royo et al. [5] donde se busca posibilitar la reconstrucción de escenas sin línea de visión detrás de dos esquinas. Esto resulta de gran utilidad ya que permite añadir un mayor rango de visión a los escenarios de uso mencionados anteriormente. En el caso de labores de rescate o imagen médica, puede permitir a los especialistas una mayor flexibilidad a la hora de realizar determinadas acciones al ser capaz de obtener visión detrás de dos esquinas.

A pesar de su gran utilidad, la imagen fuera de línea de visión es un campo en continuo desarrollo que requiere de hardware muy especializado como el láser y el sensor, de difícil utilización y elevado coste. Para ello, se utiliza simulación por software para ahorrar en costes del prototipado y agilizar el proceso de desarrollo de algoritmos de imagen *NLOS* al simular el láser y el sensor [1].

Principal problema a resolver Los sistemas de simulación por software de transporte de luz transitorio están preparados especialmente para situaciones en las que sea necesario simular escenas ocultas detrás de una esquina, pero **presentan dificultades a la hora de computar el transporte de luz detrás de dos esquinas** (Figura 1.3). De forma general, el cálculo de luz indirecta tiene un gran coste computacional. En el caso de objetos ocultos detrás de dos esquinas, la luz emitida por el láser necesita de cinco rebotes indirectos para llegar a la escena oculta

y volver al sensor. El trabajo de Royo et al. [1] adapta el cálculo de luz indirecta para escenas detrás de una esquina, con tres rebotes, mejorando el tiempo de convergencia. Sin embargo, es necesario modificarlo para escenas detrás de dos esquinas, teniendo en cuenta el el coste computacional.

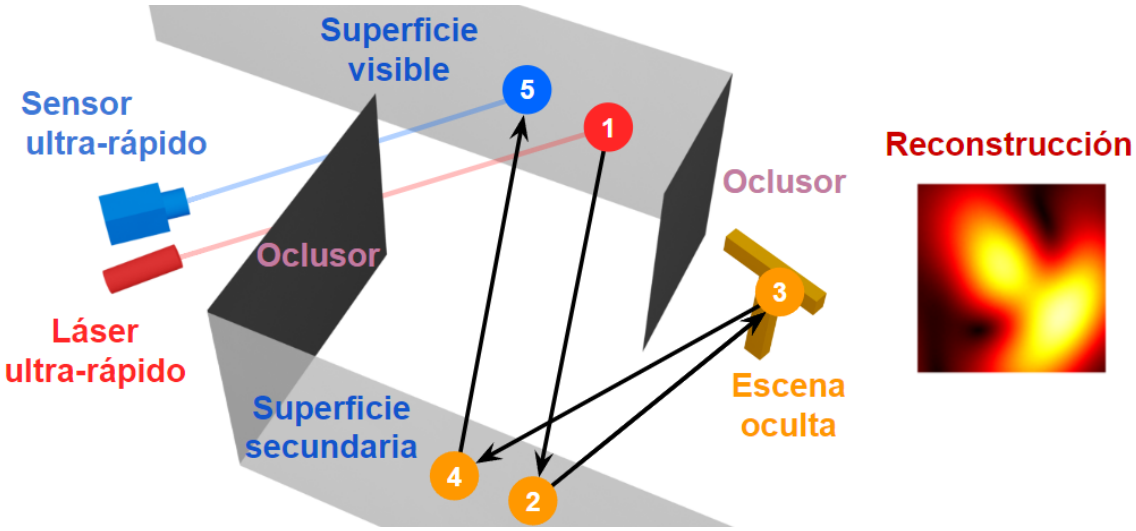

Figura 1.3: Escena básica para representar la reconstrucción de imágenes sin línea de visión detrás de dos esquinas, así como el camino y los rebotes de la luz emitida por el láser. Se pueden observar los elementos propios de una escena *NLOS* detrás de dos esquinas: Sensor ultra-rápido, láser ultra-rápido, oclusores que impiden ver la escena oculta, superficie visible desde el sensor, superficie secundaria y escena oculta que se busca reconstruir. Además, se muestran los rebotes que sufre la luz desde que es emitida por el láser hasta que llega al sensor, así como la reconstrucción obtenida. Adaptado de Royo et al. [5].

El problema de la imagen sin línea de visión es una de las principales utilidades de la llamada imagen transitoria, que busca analizar cómo la luz se propaga a través de una escena, utilizando medidas temporales del orden de picosegundos [6]. De esta forma, se puede analizar cómo la luz interactúa con los elementos de una escena y sus materiales.

El trabajo se enmarca dentro de Mitsuba 3 [7], un *renderizador* escrito en C++17 sobre el compilador en tiempo real Dr. Jit [8], desarrollado por *Realistic Graphics Lab* en *EPFL*. Un renderizador es un software de simulación de transporte de luz que dada una escena conformada por luces, sensores, objetos y materiales simula cómo la luz interactúa con todos ellos produciendo imágenes fotorrealistas.

1.2. Objetivos del trabajo

Los objetivos del trabajo serán implementar **técnicas de muestreo y reconstrucción** para escenas sin línea de visión en el sistema de simulación de propagación de luz Mitsuba 3 [7] y evaluar dichas técnicas en convergencia y tiempo. Las técnicas de muestreo en informática gráfica son métodos utilizados para obtener muestras (o puntos de datos) de una imagen o escena para generar una representación realista, mientras que las de reconstrucción buscan generar representaciones visuales a partir de dichas muestras. Con estas técnicas de muestreo y reconstrucción se busca que la señal de la luz a lo largo del tiempo obtenida como resultado en simulación sea de una mayor calidad, permitiendo abordar los problemas al recrear escenas *NLOS* detrás de dos esquinas como se ha mencionado en el Apartado 1.1.

En nuestro trabajo se ha optado por implementar las técnicas de *Hidden Geometry Rejection Sampling* y *Kernel Density Estimation*. De forma más detallada, el procedimiento llevado a cabo en el trabajo ha sido el siguiente:

1. Contextualización acerca de técnicas ya implementadas previamente.
2. Implementación de **Hidden Geometry Rejection Sampling** y posterior evaluación de los resultados obtenidos.
3. Implementación de **Kernel Density Estimation** y posterior evaluación de los resultados obtenidos.
4. Evaluación y comparación final de cada técnica implementada, mostrando reconstrucciones de las escenas para obtener una mejor idea de las mejoras obtenidas.

1.3. Estructura de la memoria

En este documento se recoge el trabajo realizado durante todo el procedimiento de estudio previo acerca de la imagen sin línea de visión y el transporte de luz transitorio, implementación de nuevas técnicas de muestreo y reconstrucción y evaluación de los resultados, así como las herramientas utilizadas y su función. La estructura seguida es la siguiente: en el **Capítulo 2** se explica de manera breve la teoría de simulación de transporte de luz en estado estacionario y transitorio, y la imagen sin línea de visión necesaria para entender los contenidos de esta memoria. En el **Capítulo 3** se procede a explicar de forma teórica las distintas técnicas de muestreo

y reconstrucción utilizadas en el trabajo, y en el **Capítulo 4** se detalla la implementación en Mitsuba 3 de las mismas. En el **Capítulo 5** se muestran y se evalúan los resultados obtenidos de las diferentes técnicas implementadas. Finalmente, en el **Capítulo 6** se describen las conclusiones finales obtenidas, así como la importancia de este trabajo de cara al futuro.

1.4. Herramientas utilizadas

La implementación de las técnicas de muestreo, así como su posterior evaluación se ha realizado en Python, utilizando *Mitsuba 3 Transient NLOS* [9] como base, una extensión para Mitsuba 3 que permite simular la propagación de la luz en escenas sin línea de visión a lo largo del tiempo. La validación experimental de los resultados obtenidos se ha llevado a cabo utilizando métricas de error cuantitativas y cualitativas respecto a convergencia y tiempo de generación de la imagen, utilizando escenas *NLOS* detrás de una o dos esquinas además de una escena con línea de visión.

Capítulo 2

Trabajo previo relacionado

Este capítulo tiene como propósito proporcionar un marco teórico sobre el que se sustenta el problema de que los sistemas de simulación de transporte de luz transitorio presentan dificultades a la hora de computar el transporte de luz detrás de dos esquinas. Para ello, en el Apartado 2.1 se hablará de la simulación de transporte de luz, tanto en estado estacionario como transitorio. En el Apartado 2.2 se presentará el problema de la imagen sin línea de visión, destacando los motivos por los que los simuladores de transporte de luz transitorio presentan problemas en escenas sin líneas de visión con una o más esquinas.

2.1. Simulación de transporte de luz

Una cámara de fotos captura imágenes al recibir luz a través de un sensor durante cierto período de tiempo, llamado tiempo de exposición. En la fotografía convencional, este tiempo de exposición es lo suficientemente largo como para que la luz tenga tiempo de propagarse por toda la escena. Como resultado, en la imagen final se integra la iluminación recibida en diferentes instantes temporales, y se pierde la información temporal sobre el transporte de la luz desde que se emite hasta que llega al sensor de una cámara, lo que da la impresión de que la luz presenta una velocidad infinita y se propaga instantáneamente por la escena.

La idea de que la luz presenta una velocidad infinita se ha convertido en una de las principales asunciones en el campo de la simulación de transporte de luz por computador, lo que recibe el nombre de **luz en estado estacionario**. Esta argumentación resulta razonable ya que los tiempos de exposición de la mayoría de cámaras estándar que se pueden utilizar resultan muy lentos en comparación con la velocidad de la luz al comparar tiempos de exposición del orden de milisegundos

frente a los cerca de 300 kilómetros por milisegundo de la velocidad de la luz.

Simulación de transporte de luz en estado estacionario En la simulación del transporte de luz basado en física se utilizan principios físicos para calcular el comportamiento de la luz al interactuar con las diferentes geometrías de una escena, sus materiales y otras fuentes de luz. Con estos cálculos se busca determinar el color e intensidad de la luz que llega a cada punto de la escena y a la cámara. Para ello, se trazan rayos de luz desde la cámara hasta la escena y se calcula cómo interactúan dichos rayos de luz con los objetos con los que intersecta por el camino, determinando de esta forma qué objetos son visibles y cómo llega a ellos la luz (Figura 2.1). Esto es conocido como *Ray Tracing*. De forma más avanzada, se puede simular la propagación de forma aleatoria de los rayos de luz en la escena, generando múltiples caminos de luz que se continúan hasta alcanzar una fuente de luz, y que pueden rebotar un número cualquiera de veces. Este proceso recibe el nombre de *Path Tracing*, y se puede resumir en calcular la imagen I utilizando la *Path Integral* [10]

$$I = \int_{\Omega} f(\overline{x})\, d\mu(\overline{x}), \tag{2.1}$$

siendo Ω el conjunto de caminos de transporte de luz de cualquier longitud, $f(\overline{x})$ la contribución de un camino cualquiera $\overline{x} = x_0 \ldots x_k$ de longitud k y $d\mu(\overline{x})$ la integración de área para vértices de la superficie. Por lo tanto, la imagen final se calcula como la suma de las contribuciones de todos los caminos posibles.

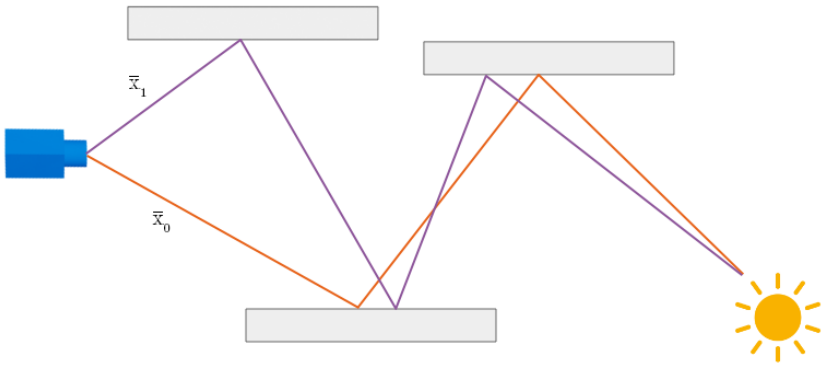

Figura 2.1: Representación de un *path tracer* tradicional, en el que se resaltan dos caminos cualesquiera $\overline{x_0}$ y $\overline{x_1}$ desde la cámara hasta la fuente de luz, siguiendo la Ecuación (2.1) presentada anteriormente. En este caso $\Omega = \{\overline{x_0}, \overline{x_1}\}$.

El problema con esta forma de calcular la imagen final es que no se tiene en

cuenta la dimensión temporal, asumiendo la velocidad de la luz infinita.

Simulación del transporte de luz en estado transitorio El transporte de luz en estado transitorio busca analizar cómo la luz se propaga a lo largo de una escena utilizando una resolución temporal extremadamente alta, del orden de picosegundos, para conseguir así eliminar la asunción de una velocidad infinita de la luz en contraposición con la luz en estado estacionario, considerando una velocidad de 299.792,458 kilómetros por segundo en el vacío [11]. De esta forma, la luz tarda un instante de tiempo en viajar de un punto x_a de la escena a otro punto x_b. De forma más concreta, conociendo la velocidad de la luz, se puede computar dicho tiempo de viaje utilizando la ecuación (2.2):

$$\mathrm{t}(x_a \leftrightarrow x_b) = \frac{|x_b - x_a|}{c} \tag{2.2}$$

siendo t el tiempo que tardará en viajar la luz, x_a y x_b dos puntos cualesquiera de la escena y c la velocidad de la luz en el vacío, 299.792,458 km/s.

Para este trabajo se utiliza el marco de trabajo de Jarabo et al. [12], que establece una base teórica para la **simulación del transporte de luz en estado transitorio** utilizando también el dominio temporal. En él, se introduce la *transient path integral*, además de distintas técnicas de reconstrucción eficientes y nuevas estrategias de muestreo.

En la formulación de la *path integral* de Veach (2.1) la intensidad de píxel de una imagen I se computa como una integral sobre el conjunto de caminos de transporte de luz Ω. En la *transient path integral*, además de integrar sobre coordenadas espaciales, es necesario integrar sobre el conjunto de retrasos temporales ΔT de todos los caminos:

$$I = \int_{\Omega} \int_{\Delta T} f(\overline{x}, \overline{\Delta t})\, d\mu(\overline{\Delta t})\, d\mu(\overline{x}), \tag{2.3}$$

donde $\overline{x} = x_0 \ldots x_k$ representa las coordenadas espaciales de los vértices de los caminos de longitud k, siendo x_0 un vértice en la fuente de luz y x_k en el sensor de la cámara. $\overline{\Delta t} = \Delta t_0 \ldots \Delta t_k$ define una secuencia de retrasos temporales. El diferencial $d\mu(\overline{x})$ denota, al igual que en la Ecuación (2.1), la integración de área para vértices de la superficie, y $d\mu(\overline{\Delta t})$ la integración temporal en cada vértice del camino. La contribución de cada camino $f(\overline{x}, \overline{\Delta t})$ se define como en la original, pero depende también del tiempo. En la Figura 2.2 se describe un ejemplo de propagación de la

luz en un camino de longitud $k = 2$.

Como no es posible obtener una solución analítica de la ecuación (2.3) debido a la complejidad del espacio de búsqueda de una escena, se opta por utilizar un estimador de Monte Carlo, el cual usa un conjunto de muestras aleatorias (caminos $\overline{x}$, retrasos Δt) para aproximar el resultado de la integral en la Ecuación (2.3).

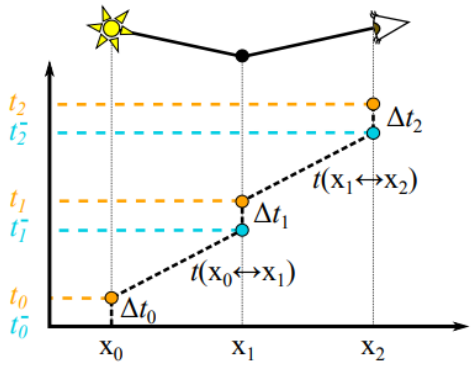

Figura 2.2: Diagrama espacio-temporal de la propagación de la luz en un camino con k = 2. La luz se emite en el instante t_0 y llega a x_1 en $t_0 + t(x_0 \leftrightarrow x_1)$, calculado con la Ecuación (2.2). Después de un pequeño retraso temporal Δt_1, la luz emerge desde x_1 en el instante t_1 y tarda $t(x_1 \leftrightarrow x_2)$ en llegar a x_2 (el sensor o cámara). Además, el sensor puede añadir otro pequeño retraso temporal Δt_2. Fuente: Jarabo et al. [12].

En la Figura 2.3 se puede observar una comparación entre renders estacionarios y transitorios. Para cada escena, se obtienen resultados para diferentes instantes de tiempo. En las figuras (b-d) se ven distintas partes iluminadas debido a que la luz se traslada por la escena con una velocidad finita, por lo que tarda más en llegar a zonas más lejanas respecto al punto emisor de la luz.

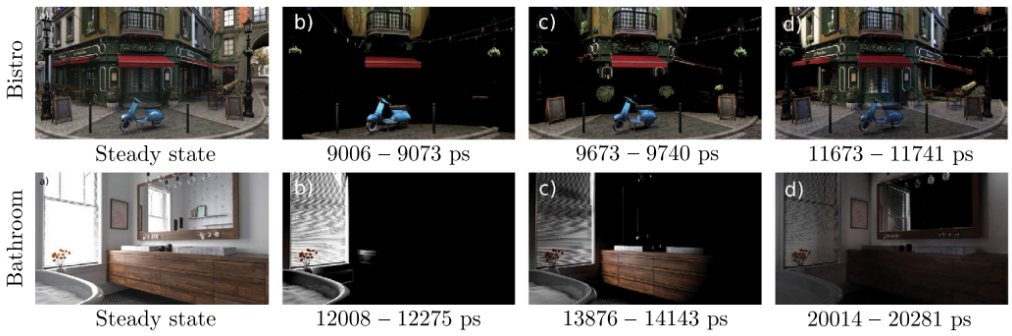

Figura 2.3: Comparaciones de renders estacionarios y transitorios. La escena *Bistro* tiene la fuente de luz localizada en el cielo, mientras que en *Bathroom* se localiza en la ventana. Las imágenes (a) son renders estacionarios, y las (b-d) son transitorios. Fuente: Royo et al. [1].

2.2. Imagen sin línea de visión

La imagen sin línea de visión, *non-line-of-sight imaging* o imagen *NLOS* busca reconstruir una escena total o parcialmente oculta analizando la luz indirecta de la geometría oculta sobre otra superficie visible secundaria.

Existen varios métodos para la reconstrucción de escenas ocultas, todos ellos clasificados en dos grupos dependiendo del tipo de información que utilizan: métodos de transporte de luz estacionario y métodos de transporte de luz transitorio. Los métodos de transporte de luz estacionario utilizan tiempos de exposición muy largos comparados con la velocidad de la luz, lo que provoca que la información que producen sea muy limitada. Los métodos de transporte de luz transitorio utilizan resoluciones temporales comparables a la velocidad de la luz, utilizando sensores y emisores de luz ultra-rápidos. A partir de esta información, se han desarrollado diversos algoritmos para reconstruir la geometría oculta como *backprojection* [13] y *phasor fields* [14]. Este trabajo se enmarca dentro de los métodos de reconstrucción basados en el transporte de luz transitorio.

El análisis de la luz indirecta que interactúa con la escena oculta en la imagen *NLOS* se consigue emitiendo pulsos de luz desde un láser. Estos pulsos de luz viajan hasta ser reflejados en una superficie visible, y rebotan hacia la geometría oculta que se quiere reconstruir. Dependiendo de los rebotes necesarios para que dichos pulsos alcancen a la geometría oculta y luego sean capturados por el sensor, se habla de objetos detrás de una o dos esquinas.

En la Figura 2.4 se presenta una configuración básica de una escena oculta detrás de una esquina. Aparecen todos los elementos mencionados anteriormente: emisor de pulsos de luz (láser ultra-rápido), sensor ultra-rápido, oclusor, pared visible o *relay wall* y escena oculta. El láser emite pulsos de luz hacia la pared visible, iluminando por dispersión la geometría oculta. La luz es reflejada hacia la pared visible y es capturada finalmente por el sensor. Por lo tanto, la luz emitida sufre 3 rebotes hasta llegar al sensor.

En una escena oculta tras dos esquinas, serían 5 los rebotes que sufre la luz emitida. Esto se muestra representado en la Figura 2.5.

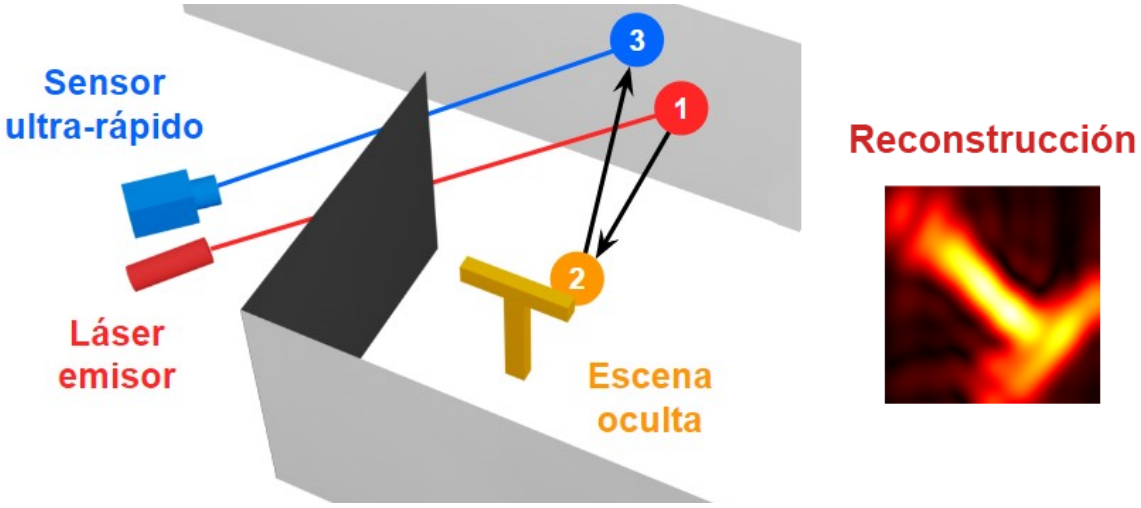

Figura 2.4: Ejemplo de configuración para una escena *NLOS* de tres rebotes (geometría detrás de una esquina). Se remarcan los elementos: sensor, láser emisor y geometría oculta. También se muestran los tres rebotes que sufre el pulso de luz durante su trayectoria, así como la reconstrucción obtenida. Adaptado de: Royo et al. [5].

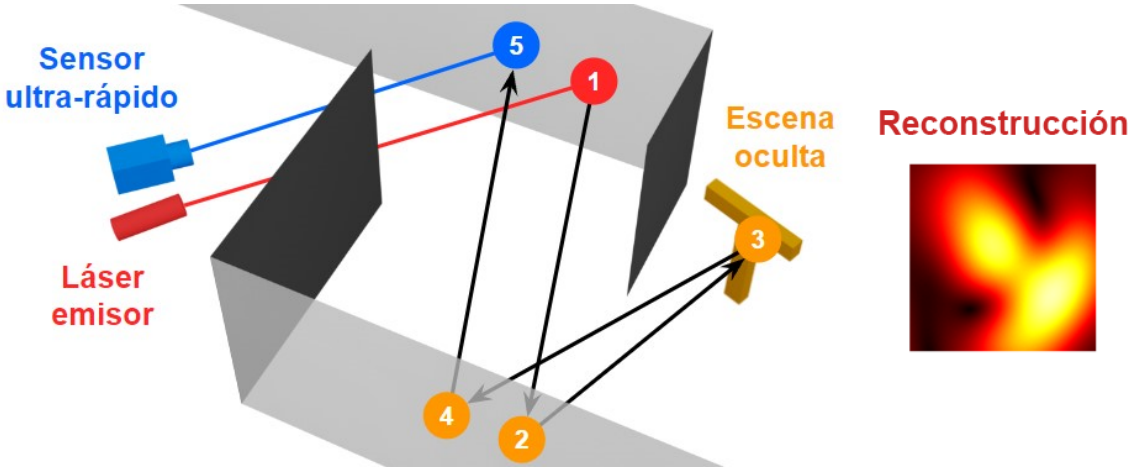

Figura 2.5: Ejemplo de configuración para una escena *NLOS* de cinco rebotes (geometría detrás de dos esquinas). Se puede apreciar el sensor, el láser emisor y la geometría oculta, además de los diferentes rebotes que sufre la luz desde que es emitida por el láser hasta que llega al sensor y la reconstrucción obtenida. Adaptado de: Royo et al. [5].

La imagen sin línea de visión es una de las principales aplicaciones del transporte de luz en estado transitorio, explicado en el Apartado 2.1, el cual permite capturar y analizar como la luz se propaga por una escena con una gran resolución temporal, al suponer la velocidad de la luz finita.

A pesar de sus múltiples posibilidades, la imagen transitoria requiere hardware de elevado coste, como pueden ser dispositivos de iluminación por pulsos y cámaras ultra rápidas, que además son difíciles de utilizar correctamente. Aparece entonces

la simulación del transporte de luz transitoria, que proporciona herramientas para desarrollar, prototipar y testear imágenes *NLOS* a un coste económico infinitamente menor.

Renderizado transitorio para escenas sin línea de visión Aunque la simulación del transporte de luz transitoria permite la realización de experimentos para imágenes *NLOS*, el uso de renderizado transitorio sigue siendo más costoso que el renderizado estacionario, al generar imágenes que utilizan la dimensión temporal además de las dimensiones habituales de la imagen.

Además, para la simulación de transporte de luz detrás de una esquina, es necesario estimar la luz después de tres rebotes indirectos. Teniendo en cuenta que la convergencia de luz indirecta es más lenta por lo general que la de la luz directa, es necesario encontrar métodos para obtener una mejora.

Este es el motivo del trabajo de Royo et al. [1] en el que se presentan nuevas técnicas de muestreo particularmente enfocadas en mejorar la convergencia de las escenas *NLOS* detrás de una esquina, como *Laser Sampling* o *Hidden Geometry Sampling*, explicado en detalle en el Apartado 3.1. Intuitivamente, estos métodos intentan generar muestras aleatorias con una distribución de probabilidad que se ajuste más a la escena simulada, lo cual resulta en una reducción de la varianza de la señal resultante.

En la Figura 2.6 se muestra el resultado del renderizado transitorio en una escena sin línea de visión donde la captura resuelta en tiempo de la derecha muestra la luz que llega a cada punto de la pared visible o *relay wall* (x, y) a lo largo del tiempo t.

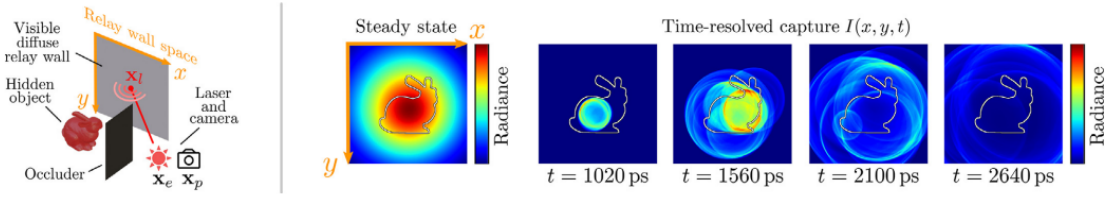

Figura 2.6: **Izquierda:** Configuración básica de simulación de datos en escenas *NLOS*, utilizando un laser emisor de luz, una cámara, un oclusor, una pared visible difusa y la escena o geometría oculta. **Derecha:** Resultado de la simulación *NLOS* en forma transitoria. El pulso de luz se emite hacia la pared visible, iluminando por dispersión la geometría oculta. La luz es reflejada de vuelta a la pared visible y se captura por el sensor. Fuente: Royo et al. [1].

Capítulo 3

Técnicas de muestreo y reconstrucción

En este Capítulo se explican las técnicas estudiadas para solventar las dificultades de los sistemas de simulación transporte de luz transitorio en escenas detrás de una y dos esquinas, mencionados en el Apartado 1.1, sus bases matemáticas y el cómo se van a utilizar en el trabajo. Dichas técnicas diseñadas para algoritmos de simulación de luz son *Hidden Geometry Rejection Sampling*, técnica de muestreo (Apartado 3.1) y *Kernel Density Estimation*, técnica de reconstrucción (Apartado 3.2).

Las técnicas de muestreo en la simulación del transporte de luz son métodos utilizados para obtener muestras de una escena para calcular los valores de los píxeles en la imagen final a partir de muestras discretas. Estas muestras son los puntos x_i de la Ecuación (2.3) que pertenecen al conjunto de caminos Ω. En el caso de este trabajo, se ha decidido implementar la técnica denominada *Hidden Geometry Rejection Sampling*, o muestreo de geometría oculta por rechazo (Apartado 3.1).

Las técnicas de reconstrucción buscan reconstruir la señal a partir de muestras discretas obtenidas por las técnicas de muestreo. Estas técnicas permiten aproximar la integral de la Ecuación (2.3) a partir de una serie finita de muestras aleatorias. Tratan de eliminar ruido y suavizar las transiciones entre varias muestras para mejorar el resultado de la imagen. En este trabajo, se implementarán estimadores de densidad por kernel o *Kernel Density Estimation*, utilizando tanto el kernel Gaussiano como el kernel Epanechnikov (Apartado 3.2).

Es importante resaltar la diferencia entre los términos "técnica de reconstrucción"de una señal, explicada en el párrafo anterior, y la "reconstrucción de la escena oculta", la cual utiliza la señal transitoria generada para producir una representación de la escena gracias a una biblioteca externa no implementada en el trabajo.

3.1. Hidden Geometry Rejection Sampling

Los *path tracers* funcionan generando caminos que unan la luz con la cámara (Figura 2.1). En el caso de escenas sin línea de visión, estos caminos tienen que tener al menos tres rebotes, luz → pared (I) → geometría oculta (II) → pared (III) → cámara. En el caso de escenas sin línea de visión, el primer rebote resulta fácil de calcular, pero el resto depende de la componente aleatoria de la generación de los caminos. Por ello, se buscan métodos de muestreo capaces de generar muestras aleatorias que sigan una distribución favorable a las escenas *NLOS*.

Hidden Geometry Sampling Tradicionalmente, se suelen computar nuevos rebotes de la luz a partir de las propiedades de los materiales. Por ejemplo, para un material difuso que refleja luz en todas las direcciones de forma uniforme, el rayo también podrá rebotar en todas las direcciones de forma uniforme. Esto recibe el nombre de *BRDF Sampling* (Figura 3.1 **(a)**). Royo et al. [1] trata de facilitar el cálculo en escenas *NLOS* directamente muestreando puntos de la geometría oculta, técnica que recibe el nombre de *Hidden Geometry Sampling* (Figura 3.1 **(b)**). De esta forma, es mucho más probable que los rayos aleatorios sigan los caminos propios de escenas *NLOS*. Ya que la simulación depende de un proceso aleatorio, un mejor muestreo provoca una disminución de la variancia y error en la estimación, haciendo que la señal resultante converja más rápidamente al resultado correcto. Esta técnica puede presentar algún problema, especialmente para escenas detrás de dos esquinas, ya que puede generar puntos muestreados de la geometría oculta que no sean visibles desde el vértice actual del camino.

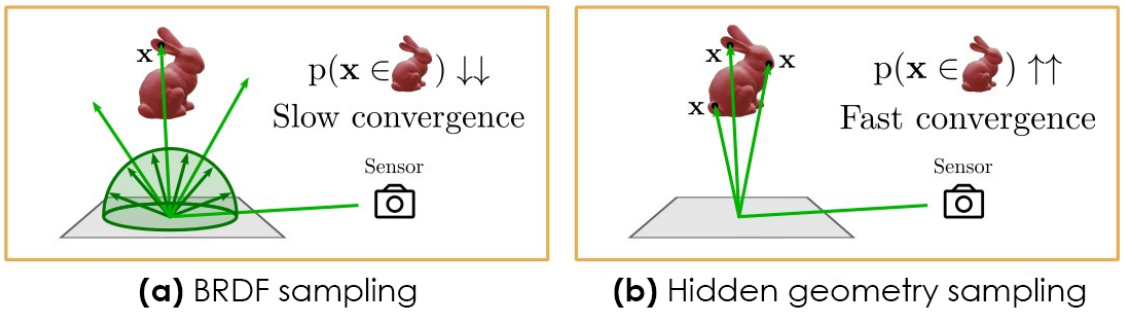

Figura 3.1: Resumen gráfico de las diferencias entre el muestreo de un *path tracer* convencional, también llamado *BRDF Sampling* **(a)** y *Hidden Geometry Sampling* propuesto por Royo et al. [1] **(b)**. En el caso de *BRDF sampling* se puede apreciar que la mayoría de los rayos no son capaces de intersectar con la geometría de forma efectiva, por lo que la convergencia será lenta, mientras que en el caso del nuevo muestreo propuesto genera los vértices directamente sobre la geometría, provocando una rápida convergencia.

Rejection Sampling Se busca por tanto obtener una mejora complementando *Hidden Geometry Sampling* con una técnica estadística conocida como *Rejection Sampling* o muestreo por rechazo. Este es un método utilizado en estadística para la obtención de muestras de una distribución de probabilidad objetivo desconocida. Consiste en la simulación de valores aleatorios que siguen una distribución objetivo no conocida, cuando solo se tiene acceso a una distribución diferente.

De esta manera, si se quieren obtener valores de una distribución $p(x)$ desconocida, se utilizará otra distribución similar ya conocida $q(x)$, ya que es posible verificar que una muestra pertenece a la distribución desconocida. En la Figura 3.2 se puede ver cómo, partiendo de una distribución conocida $q(x)$ (cuadrado que contiene a todos los puntos) y pudiendo verificar si una muestra de esta distribución conocida pertenece a $p(x)$, es posible muestrear valores de la distribución desconocida $p(x)$ (círculo que contiene los puntos azules).

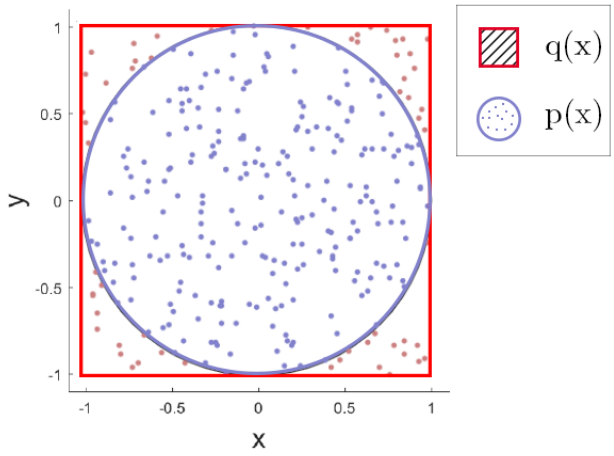

Figura 3.2: Explicación gráfica de la técnica de *Rejection Sampling* en estadística. A partir de una muestra de una distribución conocida $q(x)$ podemos obtener puntos de la distribución $p(x)$ desconocida. Los puntos aceptados como pertenecientes a $p(x)$ se muestran en azul, mientras que los rechazados se encuentran en rojo. Adaptado de: Pharr et al. [15].

En el caso de la imagen sin línea de visión o *NLOS*, se busca poder **muestrear puntos de la escena oculta que son visibles desde el vértice muestreado** en cada intersección de un camino, pero **solo se tiene acceso a la distribución de los vértices** sobre la geometría oculta, visibles o no. El problema de *Hidden Geometry Sampling* es que puede generar puntos no visibles para estos casos, por lo que se puede utilizar la técnica de *Rejection Sampling* para obtener las muestras de la distribución deseada, aquellos puntos visibles de la escena desde el vértice muestreado en la intersección del camino. Utilizando la notación matemática pre-

viamente explicada, la distribución de puntos visibles de la geometría oculta desde cada vértice del camino de luz es $p(x)$ y la distribución de todos los puntos de la geometría oculta del camino de luz es $q(x)$. De esta forma, a la hora de elegir el nuevo vértice del camino de luz, para cada punto en $q(x)$ se comprobará si resulta visible desde el vértice y si, por lo tanto, pertenece a $p(x)$ y se puede utilizar como muestra de $p(x)$.

Hidden Geometry Rejection Sampling En este trabajo, una vez realizada la estimación de vértices de *Hidden Geometry Sampling*, se comprueba si realmente hay visibilidad desde el camino actual. Si no, se vuelve a generar un nuevo vértice utilizando *Hidden Geometry Sampling* y se repite el proceso. Esto se realiza hasta que el punto sea visible, o hasta que se llegue al máximo de iteraciones establecidas (Figura 3.3). Esta nueva técnica será llamada *Hidden Geometry Rejection Sampling* (*HGRS*).

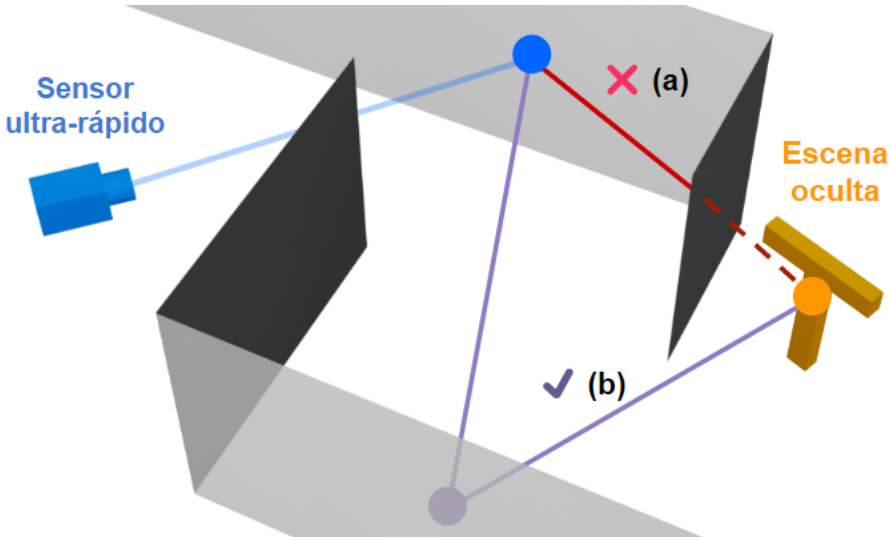

Figura 3.3: Escena en la que se muestran dos caminos posibles. El camino **(a)** se trata de un camino no visible, por lo que sería descartado y se generaría un nuevo vértice. El camino **(b)** sí se trata de un camino visible.

La técnica de *Hidden Geometry Rejection Sampling* busca mejorar la convergencia de las simulaciones, mejorando en gran medida la calidad visual de los resultados y permitiendo la utilización de un menor número de muestras por píxel para obtener resultados similares, ya que un muestreo con probabilidad proporcional a la función que se busca integrar mejora la convergencia.

3.2. Kernel Density Estimation

Kernel Density Estimation (KDE) o estimación de densidad del kernel es una técnica muy utilizada en estadística que permite aproximar una función de densidad de una distribución de datos discretos observados utilizando sumas de funciones (kernels) en cada punto de los datos.

En el caso de este trabajo, la señal a reconstruir es la función de radiancia o luz indirecta que captura la cámara en una superficie visible secundaria a lo largo del tiempo, mientras que la muestra finita de observaciones son las distintas observaciones asociadas a cada camino generado por el algoritmo de *path tracing*.

La técnica de *Kernel Density Estimation* se aplicará a la dimensión temporal t de la señal $I(x, y, t)$ obtenida de la simulación, consiguiendo una reconstrucción de esta para valores del dominio donde no ha intersectado ningún camino con este tiempo de vuelo.

En nuestro sistema de simulación de luz transitoria, la dimensión temporal se representa utilizando *bins* o segmentos temporales discretos, calculados utilizando la distancia que recorre la luz para cada camino, el tiempo de exposición utilizado y la velocidad de la luz (Figura 3.4). De esta forma, las contribuciones de los caminos se asignan a un segmento temporal específico, es decir, cada muestra cae en un único espacio temporal, como si el láser emisor de luz estuviera encendido y al instante apagado. En realidad, los láser ultra-rápidos que se quieren simular suelen ocupar varios espacios temporales dependiendo de su perfil de emisión.

Para cada contribución de un camino, se centra el kernel y se suman todas las funciones de densidad generadas por los kernels previamente colocados, dividiendo finalmente la suma por el ancho de banda o factor de escala (Figura 3.5).

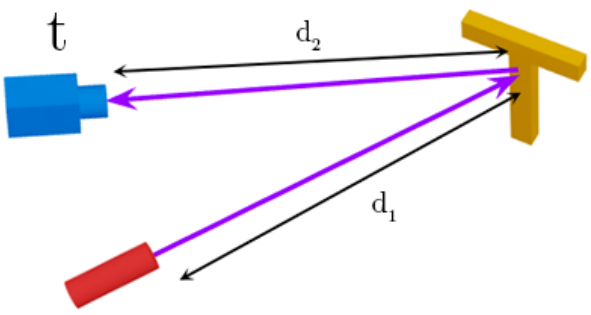

Figura 3.4: Representación del momento en el que llega al sensor un camino cualquiera en el instante de tiempo **t**, obtenido como la suma de las distancias recorridas por un rayo de luz (d_1 y d_2) entre la velocidad de la luz c.

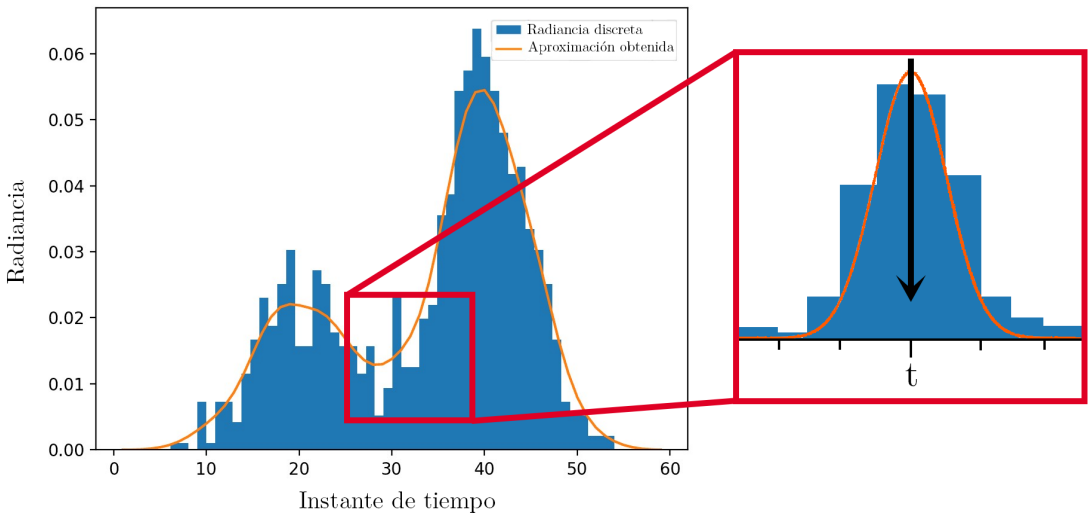

Figura 3.5: Histograma que representa la función de densidad de un conjunto de datos discretos observados, y su aproximación obtenida utilizando estimación de densidad por kernel, obteniendo una función continua. **Ampliado:** Kernel centrado en el instante t utilizado para la estimación de densidad para una de las muestras discretas.

Con la técnica de estimación de densidad se busca representar de la forma más fiable posible aquellos datos que ocupen varios *bins* temporales. Al utilizar esta técnica, estos datos contribuirán al valor de todos los segmentos temporales cercanos (como se ha podido observar en la Figura 3.5).

La estimación de densidad de este trabajo se ha basado en el trabajo ya existente de Jarabo et al. [12] para mejorar los resultados de las imágenes transitorias, tanto en el espacio temporal como entre valores de píxeles.

Kernel Density Estimation y *Hidden Geometry Rejection Sampling* son dos técnicas completamente ortogonales y complementarias, que actúan en aspectos completamente diferentes del proceso de generación de imágenes.

3.2.1. Kernel Gaussiano

El kernel Gaussiano o *Gaussian Kernel* es una función de densidad que tiene su origen en la distribución normal. Se caracteriza por su forma de curva suave y simétrica en forma de campana de Gauss. Se define por la media o valor central de la distribución (μ) y por la desviación estándar que establece el ancho de la campana (σ), y que realizará la función de ancho de banda. La función Gaussiana del kernel Gaussiano se define como:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

siendo x el valor puntual para el cual se calcula el kernel, μ el valor sobre el que se busca centrar el kernel y σ la desviación estándar utilizada como ancho de banda de la estimación de densidad, utilizado como parámetro. En este caso en concreto, el valor de μ será el instante de tiempo en el que el camino $\bar{\mathbf{x}}$ llega a la cámara (Figura 3.4), es decir, $\mu = \Sigma_{i=0}^{k-1} t(x_i \leftrightarrow x_{i+1})$, según la Ecuación (2.2). En la Figura 3.6 se muestra un ejemplo de utilización del kernel Gaussiano para la contribución de un camino cualquiera $\bar{\mathbf{x}}$, centrado en el instante t.

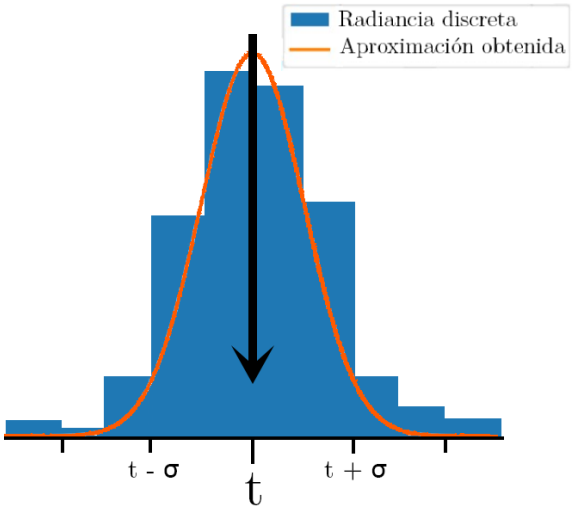

Figura 3.6: Gráfica que representa una función Gaussiana utilizando la Ecuación 3.1, centrada en t (utilizando $\mu = t$), siendo t un *bin* temporal cualquiera de la escena transitoria, y con un valor de σ definido.

Cuanto mayor sea el valor de σ utilizado, mayor serán las dimensiones del kernel, por lo que también aumentará el grado de suavizado realizado ya que para cada instante temporal se tendrán en cuenta un mayor número de contribuciones de los caminos.

La función Gaussiana resulta una buena opción cuando se utilizan datos totalmente desconocidos, ya que influencia a un gran número de datos. Además, su forma

suave y simétrica permite una transición gradual entre los puntos de datos, resultando en una estimación más suave de la distribución. También proporciona una gran flexibilidad a la hora de ajustar dicha suavidad en la estimación utilizando el parámetro σ.

3.2.2. Kernel Epanechnikov

El kernel Epanechnikov o *Epanechnikov Kernel* es una función de densidad simétrica de forma parabólica, que se encuentra definida solamente en el intervalo [-b, b], siendo b el ancho de banda y se anula fuera de dicho rango (Figura 3.7). Su función viene definida por:

$$f(x) = \frac{3}{4}\left(1 - \left(\frac{x-\mu}{b}\right)^2\right) \quad \text{si} \quad \left|\frac{x-\mu}{b}\right| \leq 1, \; f(x) = 0 \quad \text{si} \quad \left|\frac{x-\mu}{b}\right| > 1 \quad (3.2)$$

siendo x el valor puntual para el cual se calcula el peso del *kernel*, b el valor de ancho de banda utilizado y μ el valor sobre el que se busca centrar el kernel. Al igual que para el kernel Gaussiano, $\mu = \Sigma_{i=0}^{k-1} t(x_i \leftrightarrow x_{i+1})$, según la Ecuación (2.2).

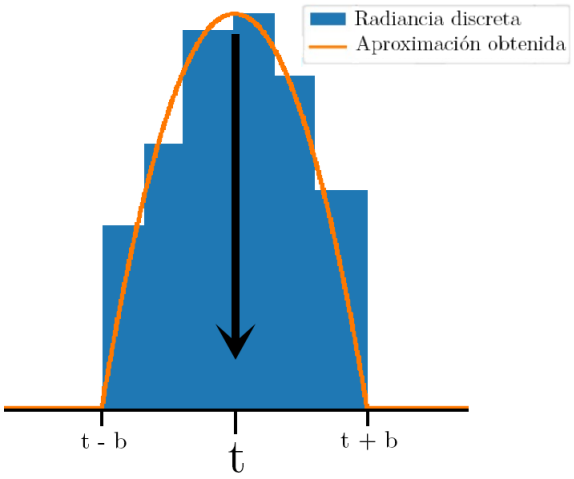

Figura 3.7: Gráfica que representa una función Epanechnikov utilizando la Ecuación 3.2, definida en el intervalo [-b, b] y centrada en t (utilizando $\mu = t$), siendo t un *bin* temporal cualquiera de la escena transitoria.

Se ha optado por utilizar el kernel Epanechnikov ya que se trata de una función más óptima que la Gaussiana. Está demostrado que el kernel Epanechnikov resulta especialmente eficaz computacionalmente para conjuntos de datos amplios [16].

Capítulo 4

Implementación

En este Capítulo se realizará una introducción a la herramienta Mitsuba 3 [7] (Apartado 4.1), se explicará la forma en la que se ha decidido realizar la implementación de las técnicas de muestreo abordadas en el Capítulo 3 y las modificaciones del código realizadas (Apartado 4.2).

4.1. Mitsuba 3

Como se ha mencionado en el Capítulo 1, un renderizador es un software de simulación de transporte de luz que dada una escena conformada por luces, sensores, objetos y materiales simula como la luz interactúa con todos ellos produciendo imágenes fotorrealistas. Es capaz de calcular cómo rebota la luz en los materiales utilizando diferentes algoritmos, llamados **integradores**, los cuales reciben este nombre ya que se encargan de integrar la *Path Integral* (Ecuación 2.1), presentada en el Apartado 2.1.

Mitsuba 3 [7] es renderizador orientado a la investigación para la simulación de transporte de luz escrito en C++17 sobre el compilador *Just-In-Time* Dr.Jit [8], desarrollado por el Realistic Graphics Lab en EPFL.

Se ha elegido Mitsuba 3 por su extensibilidad y facilidad de modificación, además de su interfaz con Python, que favorece la sencillez en la evaluación de resultados y comparaciones.

Este trabajo se implementa sobre la librería *Mitsuba 3 Transient NLOS*, una extensión a su vez de la librería *Mitsuba 3 Transient* [9]. *Mitsuba 3 Transient* permite la utilización del dominio temporal en Mitsuba 3, añadiendo por lo tanto la posibilidad de realizar renders transitorios. *Mitsuba 3 Transient NLOS* añade la posibilidad

de realizar renders *NLOS*, aplicando las técnicas propuestas por por Royo et al. [1]. Ambos son scripts de Python accesibles al usuario.

4.2. Implementación de las técnicas de muestreo y reconstrucción

Para implementar las distintas técnicas de muestreo y reconstrucción se ha modificado el código en Python de la extensión *Mitsuba 3 Transient NLOS*.

La técnica de muestreo de *Hidden Geometry Rejection Sampling* (Apartado 3.1) se ha implementado como un nuevo integrador de escena de Mitsuba 3 llamado «transient_nlos_path_rs».

Las técnicas de estimación de densidad (Apartado 3.2) se han implementado como opciones dentro de la escena. La opción «kernel_sigma» permite utilizar kernel Gaussiano al añadir los bloques transitorios a la imagen final, especificando el valor de σ deseado para la función Gaussiana, mientras que la opción «kernel_bandwidth» activa el uso del kernel Epanechnikov, concretando el valor del ancho de banda a utilizar. Dentro de *Mitsuba 3 Transient NLOS*, se han extendido las clases «transient_hdr_film» y «transient_block», para incorporar en cada clase la opción de utilizar modelos de estimación de densidad. Concretamente, dentro de la clase «transient_block» se ha añadido el código referente a la aplicación de los pesos del kernel a las señales.

En Mitsuba, los **films** convierten los resultados al finalizar el proceso de renderizado y los almacenan en unas estructuras de datos llamadas **image block**, las cuales permiten al renderizador almacenar la radiancia de forma paralela.

El código se ha realizado de forma modular, para poder incluir otros tipos de funciones de forma simple.

Las modificaciones al código realizadas se detallan en el diagrama de clases de la Figura 4.1. Las clases destacadas en color amarillo son aquellas que se han modificado en la implementación del trabajo, mientras que las resaltadas en verde son las creadas.

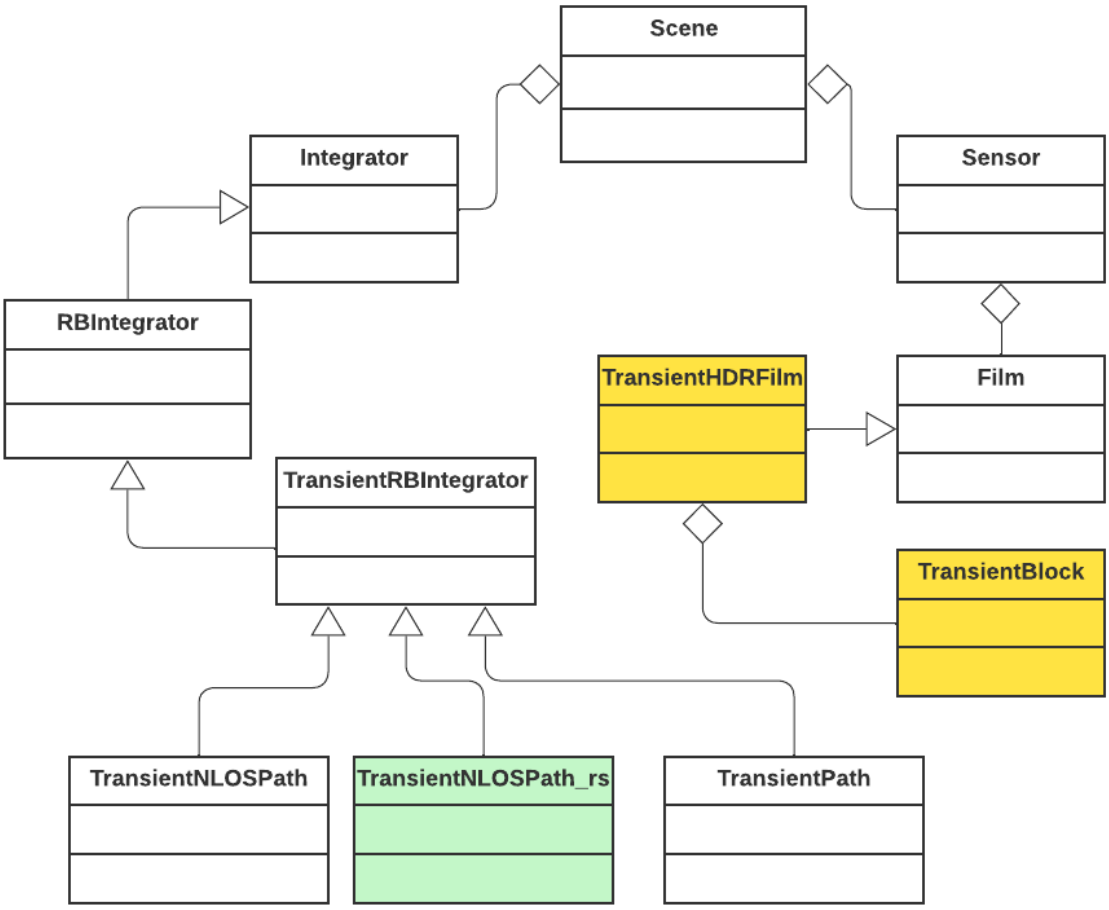

Figura 4.1: Diagrama de Clases del trabajo. Las clases en amarillo representan clases modificadas y las clases en verde las creadas.

Capítulo 5

Resultados obtenidos

En este Capítulo se aportará una explicación acerca de las pruebas realizadas utilizando las diferentes técnicas de muestreo y reconstrucción implementadas (Apartado 5.1), así como las formas de evaluación (Apartado 5.2) y los resultados de dichas pruebas, mostrando también los resultados de las evaluaciones (Apartado 5.3).

5.1. Pruebas realizadas

Las pruebas se han realizado en un subsistema Debian 11 para Windows, utilizando paralelización de CPU. El subsistema en el que se han realizado las pruebas cuenta con 32 GB de memoria RAM, además de un procesador AMD Ryzen 5 5600X, de 6 núcleos y 12 hilos.

Para obtener una variedad de resultados suficiente para realizar valoraciones sobre las nuevas técnicas de muestreo y reconstrucción implementadas, se ha decidido utilizar diferentes escenas con objetos fuera de la línea de visión y una escena con línea de visión. Se han probado las siguientes **cuatro** escenas:

Z_nlos_1corner: Escena Z NLOS detrás de una esquina La primera escena se trata de una letra Z detrás de una esquina en el vacío (Figura 5.1), la cual será denominada en las pruebas escena «*Z_nlos_1corner*». Esta es una de las configuraciones básicas en imagen *NLOS*, por lo que se buscará comprobar la mejoría de las técnicas desarrolladas respecto a la base implementada por Royo et al. [1].

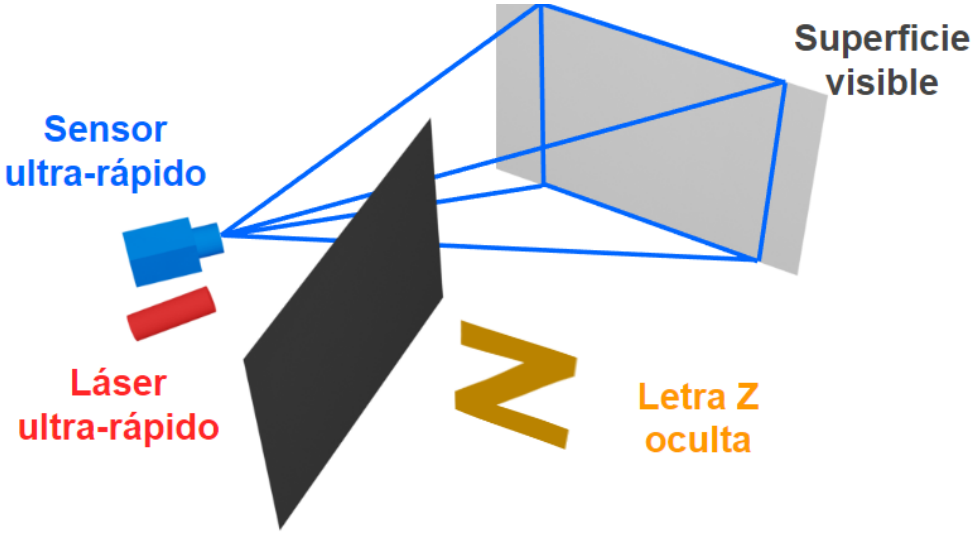

Figura 5.1: Escena «*Z_nlos_1corner*», con el sensor y láser ultra-rápidos, la superficie visible y una geometría oculta (letra Z).

T_nlos_2corner_small: Escena T NLOS detrás de 2 esquinas, superficie secundaria pequeña La segunda escena de prueba que se ha utilizado consiste en una letra T detrás de dos esquinas, utilizando una superficie secundaria con un tamaño reducido (Figura 5.2). Esta escena será denominada en las pruebas «*T_nlos_2corner_small*». En esta escena se buscará comprobar cómo funcionan las nuevas técnicas en escenas detrás de dos esquinas.

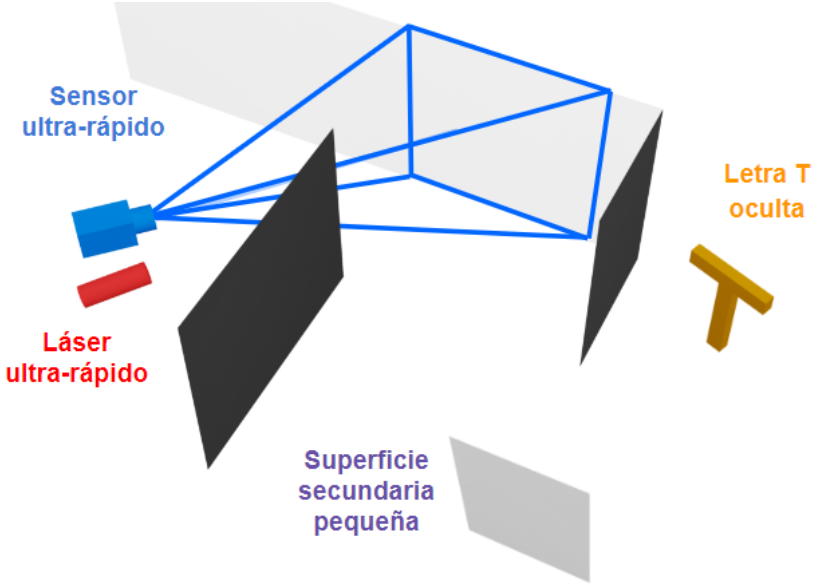

Figura 5.2: Escena «*T_nlos_2corner_small*», con el sensor y láser ultra-rápidos, la superficie secundaria y una geometría oculta detrás de dos esquinas (letra T).

T_nlos_2corner_big: Escena T NLOS detrás de 2 esquinas, superficie secundaria grande La siguiente escena de prueba utilizada consiste de nuevo en una letra T detrás de dos esquinas, utilizando una superficie secundaria de un tamaño más elevado que en la prueba anterior (Figura 5.3). Esta escena recibirá la denominación de «*T_nlos_2corner_big*» en las pruebas, y servirá para volver a comprobar el funcionamiento de las nuevas técnicas implementadas en escenas detrás de dos esquinas y para comparar cómo influye el tamaño de la superficie secundaria en los resultados, ya que al ser una superficie más grande tendrá más posibilidades de ser encontrada al generar caminos aleatorios que una superficie pequeña como la de la escena «*T_nlos_2corner_small*».

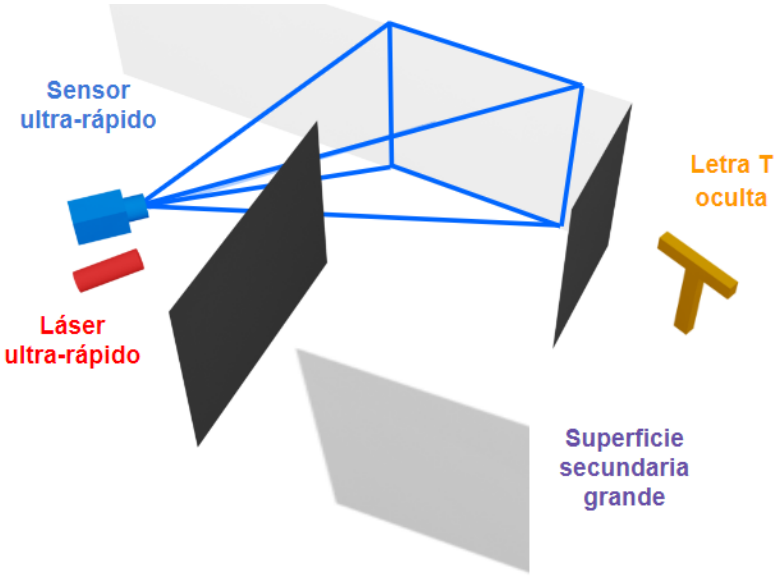

Figura 5.3: Escena «*T_nlos_2corner_big*», con el sensor y láser ultra-rápidos, la superficie secundaria y una geometría oculta detrás de dos esquinas (letra T).

Escena Cornell Box con línea de visión Finalmente, se ha utilizado una escena de tipo *Cornell Box* para probar la estimación de densidad en renders transitorios con línea de visión. La escena consta de la configuración básica de una *Cornell Box*, una caja cerrada completamente excepto por uno de los lados, con una luz de área en la pared superior y dos geometrías básicas en el interior. En la Figura 5.4 se muestra una imagen estacionaria de la escena en cuestión.

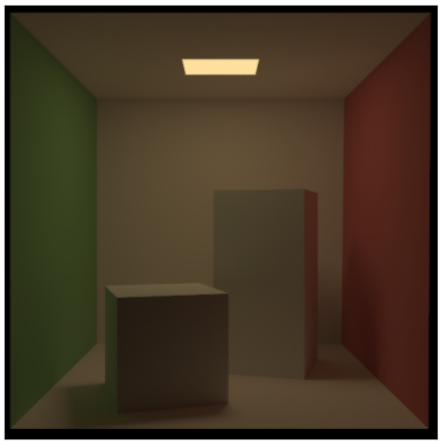

Figura 5.4: Escena «*cornell_box*» con dos geometrías básicas en el interior.

5.2. Evaluación de los resultados

El resultado que se obtiene de las simulaciones utilizando distintas técnicas de muestreo es un vídeo $I(x, y, t)$ de la luz indirecta reflejada hacia una pared (entrada para algoritmos de imagen sin línea de visión), que registra los valores de radiancia de los píxeles (x, y) a lo largo de un tiempo t, siendo la radiancia la luz indirecta que llega a la superficie visible. Para realizar una correcta comparación entre técnicas y los resultados originales, se busca evaluar el tiempo de computación requerido para el vídeo y la convergencia del algoritmo.

La evaluación del **tiempo de renderizado** o de generación de la imagen se realizará mediante las mediciones de tiempos de cada algoritmo.

La **convergencia** se evaluará comparando el vídeo obtenido como resultado de cada técnica utilizando un número reducido de muestras con un *Ground Truth* en el que se utilizará un número de muestras por píxel muy elevado, suponiendo que la señal ha convergido por completo y la señal no cambiará aunque se computen más muestras.

Para realizar la comparación de la convergencia se han utilizado *MSE (Mean Squared Error*, o Error Cuadrático Medio) y *PSNR (Peak Signal-To-Noise Ratio* o Proporción Máxima de Señal a Ruido). Tanto el *MSE* como el *PSNR* será utilizado para medir la discrepancia entre los valores predichos por el modelo (resultados obtenidos por las nuevas técnicas de muestreo) y los valores reales observados (valor *Ground Truth* obtenido utilizando un número muy elevado de muestras). Cuanto

menor sea el valor del *MSE*, menor será la discrepancia entre el modelo y el valor de referencia. Cuanto mayor sea el valor del *PSNR*, menor será la diferencia entre el valor de referencia y el nuevo valor obtenido.

La fórmula utilizada para calcular el *MSE* (Error Cuadrático Medio) es la siguiente:

$$\mathrm{MSE}(I, \hat{I}) = \frac{1}{T} \sum_{j=1}^{T} (I_j - \hat{I}_j)^2, \tag{5.1}$$

siendo I la imagen de referencia *ground truth*, $\hat{I}$ la imagen obtenida por la técnica de muestreo, T el número de componentes (número de instantes de tiempo), I_j el valor *ground truth* y $\hat{I}_j$ el resultado obtenido por nuestra técnica de muestreo para su comparación.

Para la evaluación utilizando *PSNR (Peak Signal-To-Noise Ratio)* se ha utilizado la fórmula que se muestra a continuación:

$$\mathrm{PSNR}(I, \hat{I}) = 20 \log_{10} \mathrm{MAX}(I) - 10 \log_{10} \mathrm{MSE}(I, \hat{I}), \tag{5.2}$$

siendo I la imagen de referencia *ground truth*, $\hat{I}$ la imagen obtenida por la técnica de muestreo, $MAX(I)$ el valor máximo posible para un píxel de la imagen de referencia y $MSE(I, \hat{I})$ el resultado de la evaluación utilizando el Error Cuadrático Medio para los mismos valores.

5.3. Resultados de las pruebas

En este apartado se comentarán los resultados obtenidos para cada escena utilizando diferentes configuraciones y técnicas de muestreo, así como la evaluación final utilizando métricas cualitativas y cuantitativas.

5.3.1. Z_nlos_1corner: Escena Z NLOS detrás de una esquina

Como prueba inicial, se ha utilizado la escena «*Z_nlos_1corner*» mostrada en la Figura 5.1, una escena con una geometría oculta detrás de una esquina.

Primero de todo, se ha renderizado una escena utilizando la base de Royo et al. [1] con 3 millones de muestras por píxel. Este resultado se considerará como el valor de referencia con el que comparar el resto de pruebas. Se ha usado este valor de muestras por píxel ya que se ha comprobado que en la gráfica de radiancia generada el ruido de la señal resulta despreciable. Por lo tanto, y debido a que no se puede utilizar un valor de muestras por píxel infinito (el cual sería el valor ideal), este resulta válido para realizar las comparaciones.

Como se puede observar en la Figura 5.5, el primer impacto en la radiancia se produce en la diagonal de la letra Z (a los 16711 ps), ya que se encuentra centrada en la imagen. Unos instantes después, se produce el mismo efecto en las partes superiores e inferiores del objeto de la letra (16811 ps).

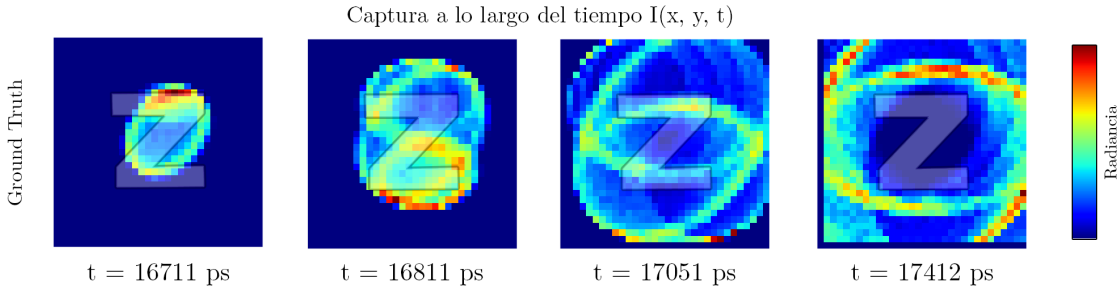

Figura 5.5: Radiancia simulada en la superficie visible para la escena «*Z_nlos_1corner*» en distintos instantes de tiempo t, utilizando la escena de referencia y superponiendo la reconstrucción de la escena oculta.

Los resultados obtenidos para cada una de las modificaciones utilizadas se muestran en la Figura 5.6. Para cada apartado (a-e) se detallan los renders transitorios obtenidos en varios momentos temporales para cada una de las pruebas realizadas con la escena.

Ground Truth En la Figura 5.6 **(a)** se muestran los resultados obtenidos para la escena *Ground Truth* de referencia. Es importante resaltar la utilización de 3 millones de muestras por píxel para la obtención de estos resultados, un número muy elevado comparado con las muestras por píxel utilizadas en el resto de pruebas de la misma escena con distintas modificaciones, lo cual resulta muy ineficiente.

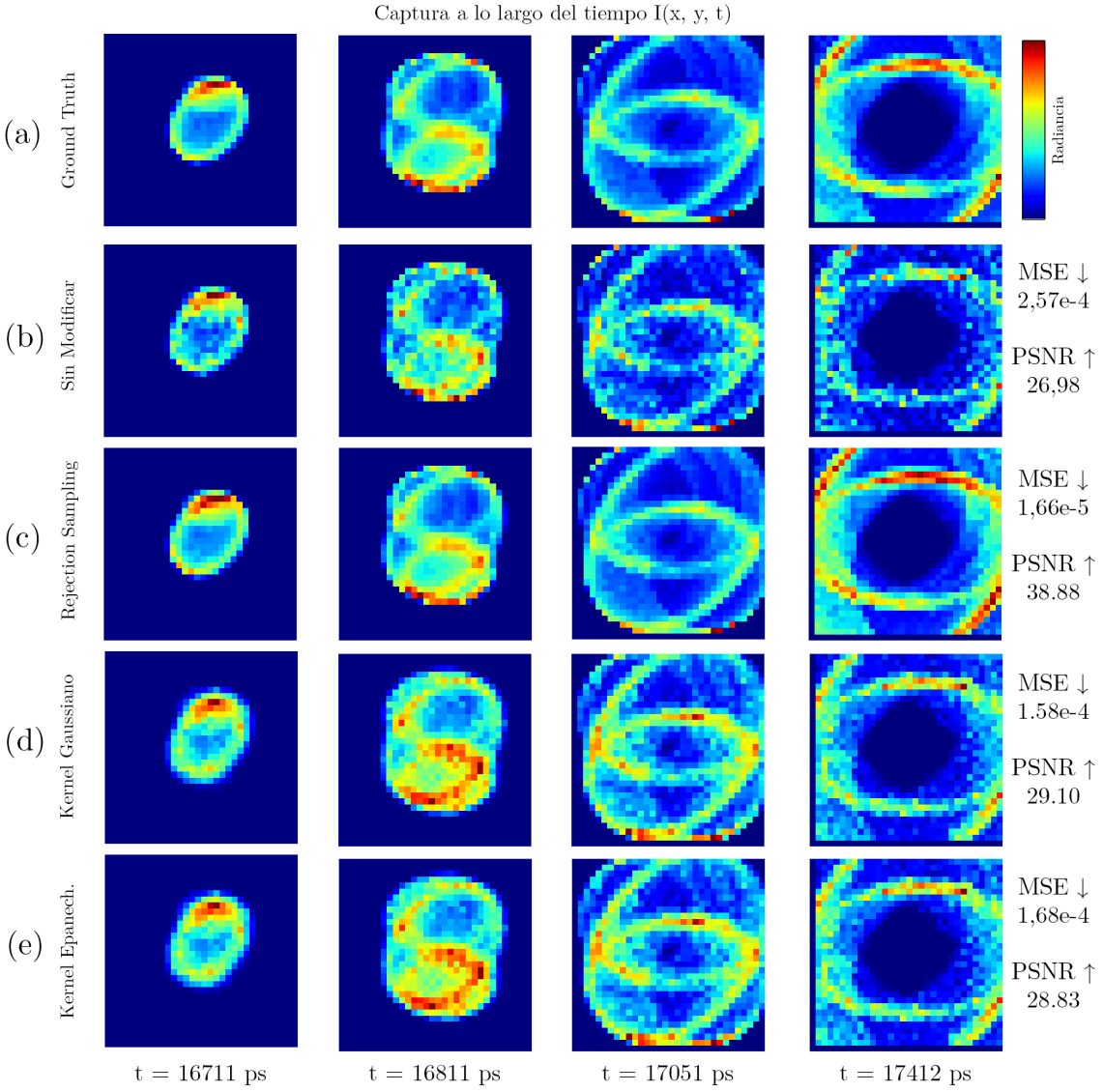

Figura 5.6: Radiancia simulada en la superficie visible para la escena «*Z_nlos_1corner*» en distintos instantes de tiempo t, utilizando la base de Royo et al. [1] para los apartados (a-b). En el apartado (a) se utilizan 3 millones de muestras por píxel, el resto (b-e) usan 30.000. HGRS usa máximo 10 iteraciones, kernel Gaussiano usa $\sigma = 1$ y kernel Epanechnikov usa $b = 2$.

Sin modificaciones En la Figura 5.6 **(b)** se muestran los resultados obtenidos sin utilizar ninguna modificación, con 30.000 muestras por píxel, un número mucho

menor que el del valor de referencia (3 millones). Como se puede observar, las imágenes en los distintos instantes temporales presentan mucho más ruido, como era de esperar al disminuir tan drásticamente el valor de muestras por píxel.

Este ruido que se puede comprobar visualmente también puede apreciarse en la gráfica de radiancia-tiempo para un píxel cualquiera de la imagen (Figura 5.7). En estas escenas se ha utilizado el píxel [0, 0] para realizar las comparaciones. Los valores de radiancia de la escena sin modificaciones adicionales presentan alteraciones muy bruscas al compararse con los valores de referencia. Además, se pueden observar datos espurios que no representan correctamente la escena.

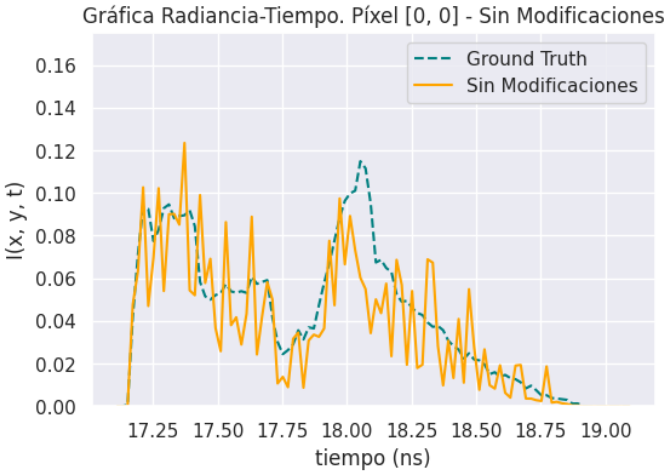

Figura 5.7: Gráfica radiancia-tiempo de los resultados obtenidos en la escena de referencia *Ground Truth* y la escena sin modificaciones adicionales. Cabe destacar que se muestran los resultados en los instantes de tiempo relevantes, obviando valores de tiempo en los que la radiancia en ambos casos era nula.

Hidden Geometry Rejection Sampling En la Figura 5.6 **(c)** se pueden observar los resultados obtenidos utilizando la técnica de *Hidden Geometry Rejection Sampling* (Apartado 3.1). Para ello, se ha utilizado una escena igual que la anterior en la que no se utilizaban modificaciones adicionales, con 30.000 muestras por píxel. Como se puede observar, los resultados presentan mucho menos ruido en comparación con los resultados de la Figura 5.6 **(b)**, siendo incluso muy similares a los obtenidos como *Ground Truth* (Figura 5.6 **(a)**). Esto es conseguido gracias al muestreo por rechazo utilizado, el cual busca puntos visibles de la geometría oculta desde el camino actual.

La mejoría mencionada anteriormente se puede observar de forma muy clara en la gráfica de radiancia-tiempo de la Figura 5.8, en la que se puede apreciar como los resultados son mucho más parecidos a los valores de referencia. De esta forma, utilizando un número de muestras más de diez veces inferior a los de la escena *Ground Truth*, se han obtenido resultados similares.

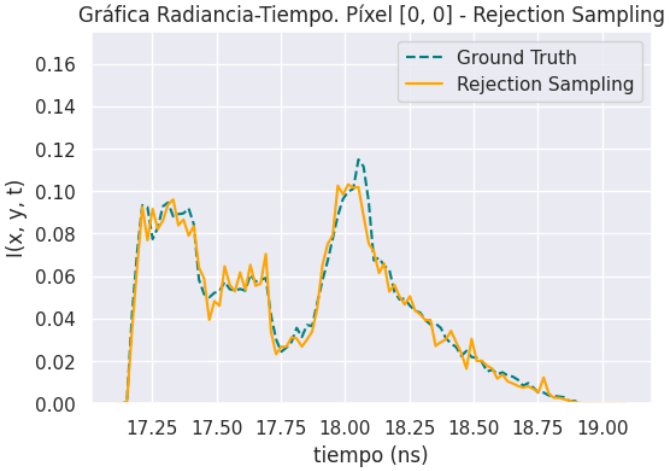

Figura 5.8: Gráfica radiancia-tiempo de los resultados obtenidos en la escena de referencia *ground truth* y en la escena utilizando *Hidden Geometry Rejection Sampling*. Cabe destacar que se muestran los resultados en los instantes de tiempo relevantes, obviando valores de tiempo en los que la radiancia en ambos casos era nula.

Density Estimation: Gaussian Kernel Los resultados obtenidos utilizando kernel Gaussiano con $\sigma = 1$ se muestran en la Figura 5.6 **(d)**. Se puede apreciar como se vuelven a mejorar los resultados obtenidos en la Figura 5.6 **(b)**, aunque se mantiene algo de ruido con respecto a los resultados utilizando *Hidden Geometry Rejection Sampling*.

En la gráfica radiancia-tiempo de la Figura 5.9 se puede observar la mejoría con respecto a la escena con simplemente las modificaciones base. Se puede apreciar ese efecto de suavizado Gaussiano, aunque aparecen picos de radiancia que no son fieles a la escena de referencia, algo totalmente esperable al tratarse de una técnica de reconstrucción que se aplica sobre las muestras existentes.

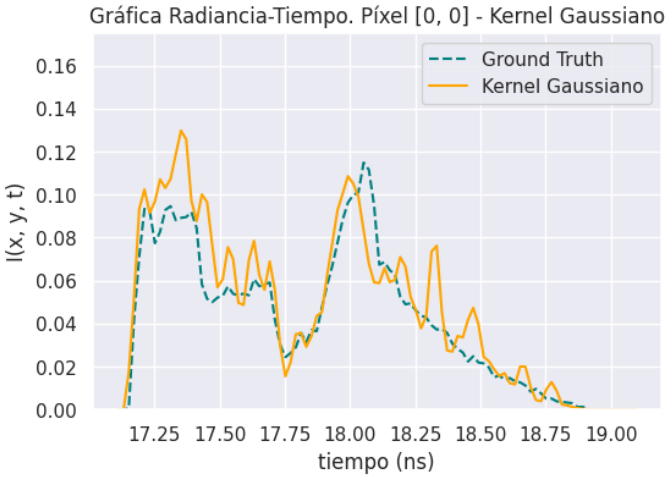

Figura 5.9: Gráfica radiancia-tiempo de los resultados obtenidos en la escena de referencia *Ground Truth* y en la escena utilizando *Kernel Density Estimation* con un kernel Gaussiano. Cabe destacar que se muestran los resultados en los instantes de tiempo relevantes, obviando valores de tiempo en los que la radiancia en ambos casos era nula.

Density Estimation: Epanechnikov Kernel En la Figura 5.6 **(e)** se pueden observar los resultados obtenidos utilizando el kernel Epanechnikov $b = 2$. Las imágenes son muy similares a las obtenidas para el kernel Gaussiano (Figura 5.6 **(d)**), resultando peores que los resultados obtenidos utilizando *Hidden Geometry*

Rejection Sampling (Figura 5.6 **(c)**), aunque mejoran a la escena sin modificaciones adicionales (Figura 5.6 **(b)**).

En lo que respecta a su gráfica radiancia-tiempo (Figura 5.10), se puede observar que es muy similar a la gráfica del kernel Gaussiano (Figura 5.9), aunque en determinados puntos se puede ver una mejor correspondencia respecto a la función de referencia.

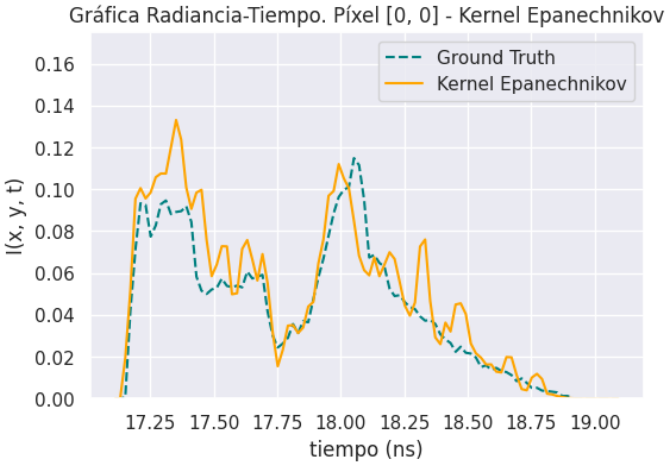

Figura 5.10: Gráfica radiancia-tiempo de los resultados obtenidos en la escena de referencia *Ground Truth* y en la escena utilizando *Kernel Density Estimation* con un kernel Epanechnikov. Cabe destacar que se muestran los resultados en los instantes de tiempo relevantes, obviando valores de tiempo en los que la radiancia en ambos casos era nula.

Evaluación de la convergencia Se han utilizado las métricas explicadas en el Apartado 5.2, siendo estas *MSE* y *PSNR*. Para ambas métricas se han evaluado los resultados obtenidos, utilizando la Ecuación 5.1 para calcular el valor *MSE* de cada técnica, y la Ecuación 5.2 para calcular el *PSNR*.

En la Figura 5.11 se muestran los resultados obtenidos después de realizar la evaluación utilizando el error cuadrático medio o *MSE* para cada técnica. Cabe recordar que el valor de muestras por píxel es el mismo para cada una de las técnicas, así como los parámetros de escena utilizados excepto la técnica de muestreo o reconstrucción utilizada. Cuanto menor sea el dato obtenido, menor es la diferencia de esa escena con respecto a la escena de referencia o *Ground Truth*.

Como era de esperar por los resultados visuales observados en la Figura 5.6 **(c)** y en la gráfica radiancia-tiempo de la Figura 5.8, el menor valor *MSE* es el de la escena que utiliza *Hidden Geometry Rejection Sampling*, por lo que se trata de la escena más similar a la escena de referencia. En términos porcentuales, la mejora con respecto a la escena sin modificaciones es de un 93,51 %, valor esperable ya que genera puntos sobre la geometría hasta que uno de ellos sea visible desde el camino actual.

Los estimadores de densidad Epanechnikov y Gaussiano también han producido

una mejora importante en el error cuadrático medio, aunque menor con respecto a la técnica de muestreo por rechazo, siendo estas mejoras del 34,87 % y 38,63 %, respectivamente. Entre ellas, el kernel Gaussiano ofrece resultados algo mejores, aunque muy similares al kernel Epanechnikov. Estos valores son totalmente lógicos, ya que tanto *HGRS* como los estimadores de densidad son técnicas totalmente diferentes, por lo que tampoco era esperable que tuviesen resultados similares.

Lo que se busca al utilizar *MSE* con los estimadores de densidad es ver cómo el suavizado reduce el efecto de los datos espurios sobre estas medidas, por lo que se produce una reducción del error.

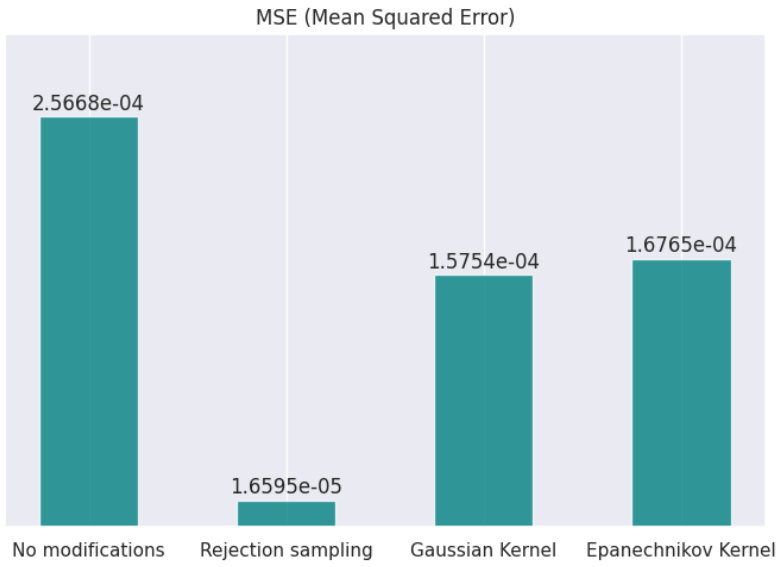

Figura 5.11: Valores *MSE* obtenidos para cada escena, utilizando el mismo número de muestras por píxel y configuración de escena para todas ellas. Cuanto menor sea el valor *MSE* obtenido, menor diferencia con respecto a la escena de referencia.

En la Figura 5.12 se muestran los resultados obtenidos después de realizar la evaluación utilizando la proporción máxima de señal a ruido o *PSNR* para cada técnica. Cuanto mayor sea el dato obtenido, menor es la diferencia de esa escena con respecto a la escena de referencia o *Ground Truth*. Los valores se muestran utilizando escala logarítmica, expresada en decibelios (dB).

Al igual que para el error cuadrático medio, los resultados proporcionan una clara ventaja a la escena que utiliza *Hidden Geometry Rejection Sampling* con respecto a la escena sin modificaciones adicionales, aventajándola en cerca de 12 dB, tratándose de una mejora del 44,6 %. Para las estimaciones de densidad también se obtienen mejoras, aunque menos significativas como ya pasaba en la evaluación anterior, en torno al 8 %.

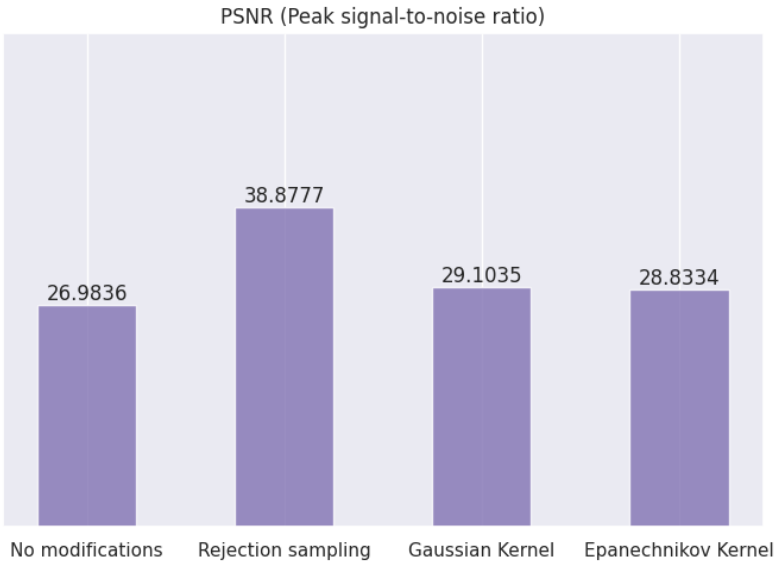

Figura 5.12: Valores *PSNR* obtenidos para cada escena, utilizando el mismo número de muestras por píxel y configuración de escena para todas ellas. Cuanto mayor sea el valor *PSNR* obtenido, mayor similaridad con la escena de referencia. (Valores expresados en decibelios, dB. Escala logarítmica.)

Después de valorar estas métricas se puede concluir que, en lo que respecta a la convergencia de la imagen o similaridad respecto a la escena de referencia, la técnica de *Hidden Geometry Rejection Sampling* mejora de forma muy significativa las métricas de la escena sin modificaciones adicionales, mientras que las técnicas de reconstrucción producen mejoras de una forma menos significativa, pero efectiva.

Evaluación del tiempo de renderizado A continuación se realizará la comparación de los tiempos de renderizado o generación de imagen para cada técnica utilizada. Los resultados obtenidos se muestran en la Tabla 5.1.

Escena	**Muestras por píxel**	**Tiempo**	**MSE ↓**	**PSNR ↑**
Ground Truth	3.000.000	8 min 40 s	-	-
Sin modificaciones	30.000	3,51 s	$2,6 \times 10^{-4}$	26,98
HG Rejection Sampling	30.000	40,1 s	$1,66 \times 10^{-5}$	38,88
Kernel Gaussiano	30.000	4,95 s	$1,58 \times 10^{-4}$	29,10
Kernel Epanechnikov	30.000	3,79 s	$1,68 \times 10^{-4}$	28,83

Cuadro 5.1: Tiempos de renderizado y muestras por píxel de cada técnica evaluada anteriormente, *Ground Truth*, sin modificaciones, *Hidden Geometry Rejection Sampling* y los estimadores de densidad Gaussiano y Epanechnikov.

Cabe recordar la utilización de 3 millones de muestras por píxel para la escena *Ground Truth*, así como 30.000 muestras por píxel para el resto de escenas de la

Tabla 5.1. Es importante remarcar que se está utilizando un simulador en el que el número de muestras utilizado afecta directamente al tiempo de renderizado, cuanto mayor sea el número de muestras, mayor será también el tiempo obtenido.

Como se puede observar, si comparamos la escena de referencia o *Ground Truth* con respecto a la técnica con los mejores resultados de la evaluación anterior (*Hidden Geometry Rejection Sampling*), se puede observar como se pasa de un tiempo de renderizado de 8 minutos y 40 segundos a 40,1 segundos, siendo esta es una mejora temporal del 91,63 %. Por ello, podemos concluir que gracias a esta nueva técnica, es posible generar una imagen con una similaridad máxima con respecto a la imagen de referencia un 91,63 % más rápido.

Reconstrucciones Para finalizar, se ha utilizado la biblioteca *tal* [17] para realizar reconstrucciones del objeto oculto detrás de una esquina a partir de los datos simulados, como los de la Figura 5.6. En la Figura 5.13 se muestra de forma más visual cómo se realiza la reconstrucción, con la señal transitoria generada representada sobre la superficie visible, a partir de la cual se obtiene la reconstrucción de la escena oculta.

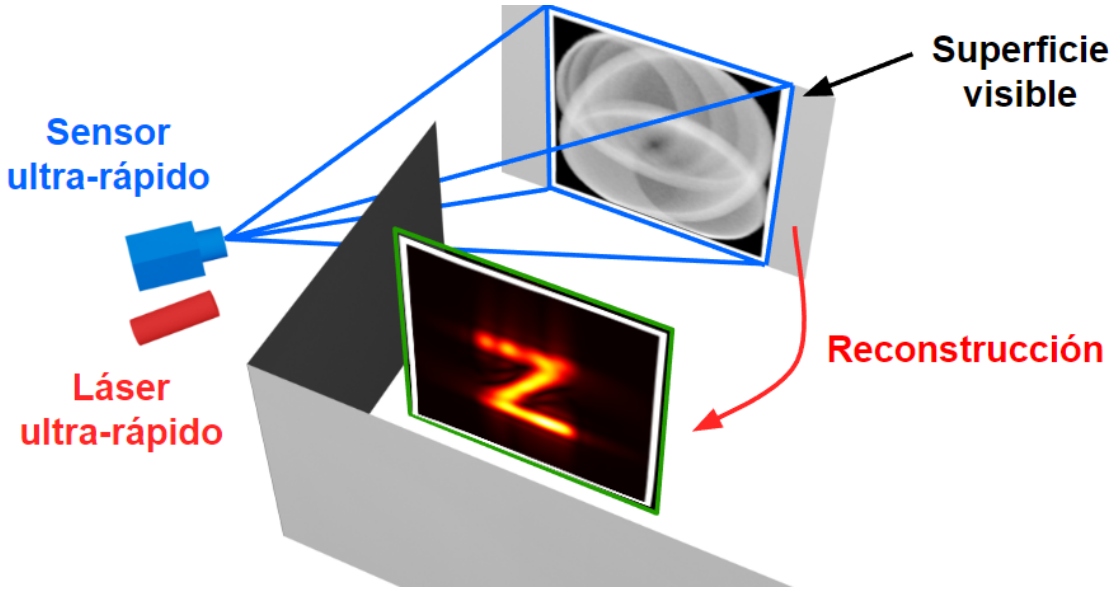

Figura 5.13: Reconstrucción de la escena Z detrás de una esquina mostrada en la Figura 5.1, donde se ve el sensor y láser ultra-rápidos, la superficie visible donde se representa la señal transitoria generada y la escena reconstruida donde estaría la geometría oculta. Adaptado de Royo et al. [5].

En la Figura 5.14 se muestran las reconstrucciones para la escena de referencia, sin modificaciones adicionales y para la que utiliza *Hidden Geometry Rejection*

Sampling. Se han utilizado valores de muestras por píxel menores a las utilizadas para la Figura 5.6, ya que el algoritmo de reconstrucción es bastante resistente al ruido de la señal. Los valores utilizados han sido 500.000 muestras por píxel para la escena de referencia, y 50 para el resto de ellas. Como se puede observar, la letra Z se puede vislumbrar perfectamente en la escena de referencia, prácticamente igual que en la escena utilizando la técnica de muestreo por rechazo. En la escena sin modificaciones se muestra una letra Z con mucho más ruido en comparación con la escena de referencia. De esta forma, se puede apreciar la mejora en términos porcentuales utilizando el *MSE* y *PSNR* de una forma mucho más visual.

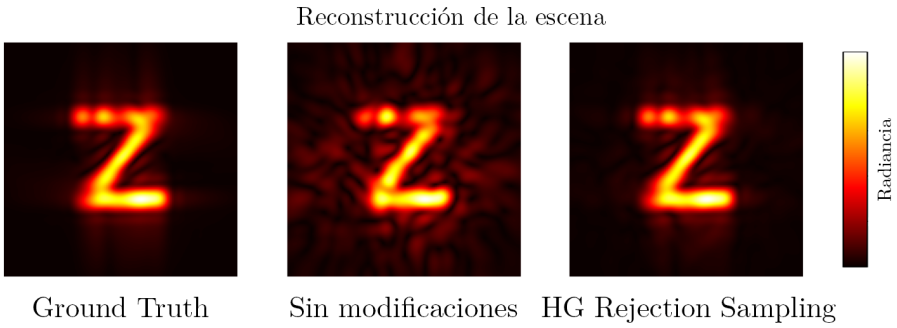

Figura 5.14: Reconstrucciones obtenidas utilizando la escena de referencia *Ground Truth* con 500.000 muestras por píxel, la versión sin modificaciones adicionales y la versión utilizando *Hidden Geometry Rejection Sampling* con 50 muestras por píxel.

5.3.2. T_nlos_2corner_small: Escena T NLOS detrás de dos esquinas, superficie visible pequeña

Como segunda prueba se ha utilizado la escena «*T_nlos_2corner_small*» de la Figura 5.2, una escena con una geometría oculta (letra T) detrás de dos esquinas, en la que se utiliza una superficie secundaria de tamaño reducido.

Al igual que en la prueba anterior, primero de todo se ha renderizado una escena utilizando la base de Royo et al. [1] con 25 millones de muestras por píxel, resultado que será considerado como valor de referencia para comparar el resto de pruebas.

Los resultados obtenidos para cada una de las modificaciones utilizadas se muestran en la Figura 5.15. Se han obtenido resultados para el mismo instante temporal, t = 23616 ps, uno de los instantes en los que la señal llega después de haber viajado a través de las dos esquinas, para cada técnica utilizada. En comparación con la Figura 5.6 del apartado anterior donde solo hay un rebote oculto, en esta la luz es más tenue al haber dos rebotes. Por ello la dificultad para comparar visualmente entre técnicas es mayor.

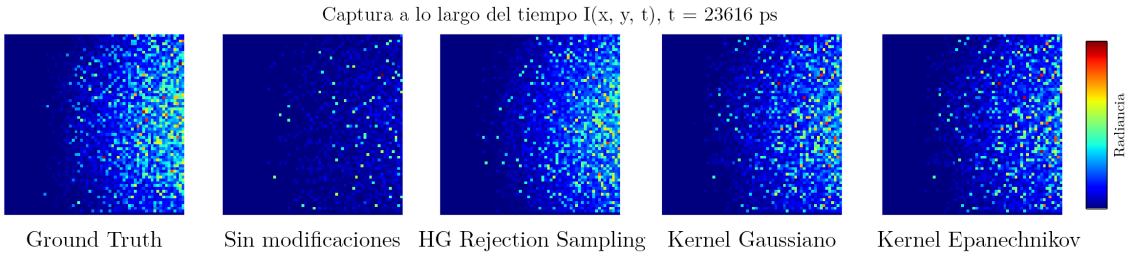

Figura 5.15: Radiancia simulada en la superficie visible para la escena «*T_nlos_2corner_small*» en $t = 23616$ ps. La simulación *Ground Truth* y sin modificaciones usa la base de Royo et al. [1]. La simulación *Ground Truth* usa 25 millones de muestras por píxel, el resto usan 100.000. HGRS usa máximo 10 iteraciones, Kernel Gaussiano usa $\sigma = 2$ y Kernel Epanechnikov usa $b = 3$.

Sin modificaciones Como se puede observar en la Figura 5.15, la señal que ha viajado a través de dos esquinas en la escena sin modificaciones adicionales, es mucho más tenue que la de la escena de referencia. Esto es debido al número de muestras por píxel utilizado, mucho menor que el utilizado para la escena de referencia. Presenta más ruido y se trata de una señal muy débil.

Hidden Geometry Rejection Sampling En la Figura 5.15 se pueden observar también los resultados utilizando la técnica de muestreo de *Hidden Geometry Rejection Sampling* (Apartado 3.1). Se ha utilizado una escena igual que la utilizada

en el apartado anterior, utilizando 100.000 muestras por píxel. En este caso, para el muestreo por rechazo se han utilizado un máximo de 10 iteraciones. Como se puede observar, mejora en gran medida los resultados obtenidos para la escena sin modificaciones, obteniendo una señal mucho más clara después de viajar a través de las dos esquinas, con una intensidad similar a la utilizada como valor de referencia.

Density Estimation: Gaussian Kernel Los resultados obtenidos utilizando estimación de densidad (Apartado 3.2) con kernel Gaussiano también se encuentran en la Figura 5.15. Presenta resultados muy similares a la escena de referencia y a la escena que utiliza *Hidden Geometry Rejection Sampling*, aunque al tratarse de técnicas tan dispares no resultaría correcto compararlas.

Density Estimation: Epanechnikov Kernel Finalmente, los resultados obtenidos utilizando estimación de densidad (Apartado 3.2) con kernel Epanechnikov también se encuentran igual que el resto en la Figura 5.15. Son resultados prácticamente idénticos a los obtenidos utilizando kernel Gaussiano, similares a los de la escena de referencia.

Evaluación de la convergencia Al igual que para la escena anterior, se han utilizado las métricas comentadas en el Apartado 5.2, siendo estas *MSE* y *PSNR*. Para ambas métricas se han evaluado los resultados obtenidos, utilizando la Ecuación 5.1 para calcular el valor *MSE* de cada técnica, y la Ecuación 5.2 para calcular el *PSNR*.

En la Figura 5.16 se muestran los resultados obtenidos después de realizar la evaluación utilizando el error cuadrático medio o *MSE* para cada técnica. Cabe recordar que el valor de muestras por píxel es el mismo para cada una de las técnicas, así como los parámetros de escena utilizados excepto la técnica de muestreo o reconstrucción utilizada. Cuanto menor sea el dato obtenido, menor es la diferencia de esa escena con respecto a la escena de referencia.

La escena con el menor valor *MSE* es la que utiliza *Hidden Geometry Rejection Sampling*, algo esperable después de los resultados visuales obtenidos en la Figura 5.15. En términos porcentuales, la mejora con respecto a la escena sin modificaciones es de un 79,18 %, algo menor de la obtenida utilizando la escena detrás de una esquina, aunque sigue siendo una mejora muy importante.

Los estimadores de densidad Epanechnikov y Gaussiano también han producido una mejora importante en el error cuadrático medio, aunque menor con respecto

a la técnica de muestreo por rechazo, la del kernel Gaussiano de un 70,22 % y la del kernel Epanechnikov de un 66 %. Los resultados del kernel Gaussiano son algo mejores, aunque resulta poco significativo, ya que modificando los valores de σ y b se podrían conseguir otros resultados distintos.

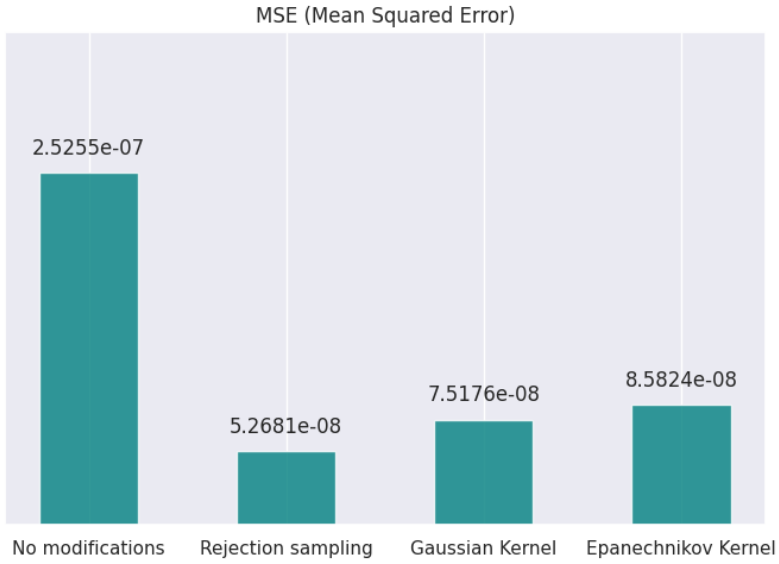

Figura 5.16: Valores *MSE* obtenidos para cada escena, utilizando el mismo número de muestras por píxel y configuración de escena para todas ellas. Cuanto menor sea el valor *MSE* obtenido, menor diferencia con respecto a la escena de referencia.

En la Figura 5.17 se muestran los resultados obtenidos después de realizar la evaluación utilizando la proporción máxima de señal a ruido o *PSNR* para cada técnica. Cuanto mayor sea el dato obtenido, menor es la diferencia de esa escena con respecto a la escena de referencia. Los valores se muestran utilizando escala logarítmica, expresada en decibelios (dB).

Al igual que para el error cuadrático medio, los resultados proporcionan una clara ventaja a la escena utilizando *Hidden Geometry Rejection Sampling* con respecto a la escena sin modificaciones adicionales, aventajándola en cerca de 7 dB, tratándose de una mejora del 35,7 %, similar a la obtenida en la escena detrás de una esquina. Para las estimaciones de densidad, la mejora obtenida al utilizar kernel Gaussiano es del 27,79 %, y para el kernel Epanechnikov del 24,71 %.

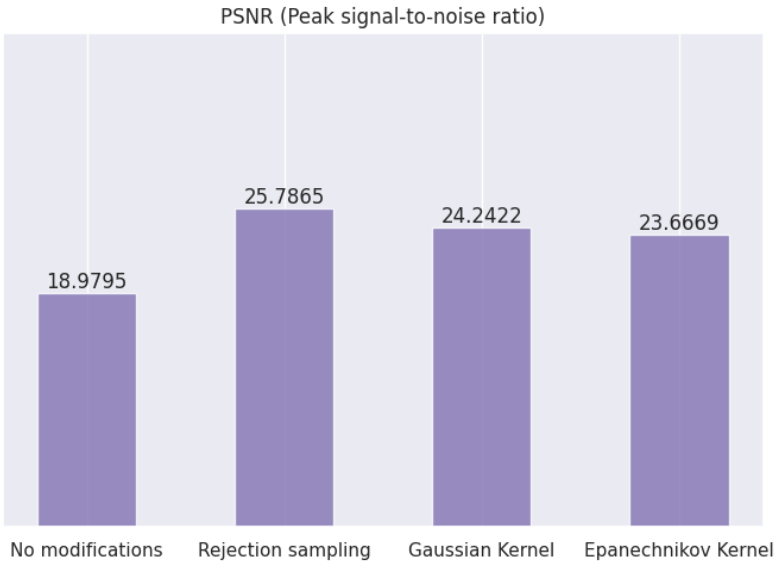

Figura 5.17: Valores *PSNR* obtenidos para cada escena, utilizando el mismo número de muestras por píxel y configuración de escena para todas ellas. Cuanto mayor sea el valor *PSNR* obtenido, mayor similaridad con la escena de referencia. (Valores expresados en decibelios, dB. Escala logarítmica.)

Con todo, se puede observar como todas las métricas indican que las nuevas técnicas de muestreo y reconstrucción mejoran el resultado final obtenido de forma muy significativa, siendo las mejoras producidas por *Hidden Geometry Rejection Sampling* las más destacables.

Evaluación del tiempo de renderizado A continuación se realizará la comparación de los tiempos de renderizado para cada técnica utilizada. Los resultados obtenidos se muestran en la Tabla 5.2.

Escena	**Muestras por píxel**	**Tiempo**	**MSE ↓**	**PSNR ↑**
Ground Truth	25.000.000	11 min 23s	-	-
Sin modificaciones	100.000	1 min 15 s	$2,5 \times 10^{-7}$	18,98
HG Rejection Sampling	100.000	6 min 14 s	$5,3 \times 10^{-8}$	25,78
Kernel Gaussiano	100.000	1 min 33 s	$7,5 \times 10^{-8}$	24,24
Kernel Epanechnikov	100.000	1 min 32 s	$8,6 \times 10^{-8}$	23,67

Cuadro 5.2: Tiempos de renderizado para cada técnica evaluada anteriormente, *Ground Truth*, sin modificaciones, *Hidden Geometry Rejection Sampling* y los estimadores de densidad Gaussiano y Epanechnikov.

Cabe recordar al igual que en el apartado anterior la utilización de 25 millones de muestras por píxel para la escena *Ground Truth*, así como 100.000 muestras por píxel para el resto de escenas de la Tabla 5.2.

Como se puede observar, se ha obtenido una mejora al utilizar la técnica de muestreo por rechazo del 45,06 % respecto al tiempo de renderizado de la escena de referencia. Al utilizar los estimadores de densidad, la mejora obtenida ha sido de un 86,34 %.

Reconstrucciones Al igual que para la escena detrás de una esquina, se ha utilizado la biblioteca *tal* [17] para realizar reconstrucciones del objeto oculto detrás de una esquina a partir de los datos simulados como los de la Figura 5.15. En la Figura 5.18 se muestra de forma más visual cómo se realiza la reconstrucción, con la señal transitoria generada representada sobre la superficie visible, a partir de la cual se obtiene la reconstrucción de la letra T oculta, rotada.

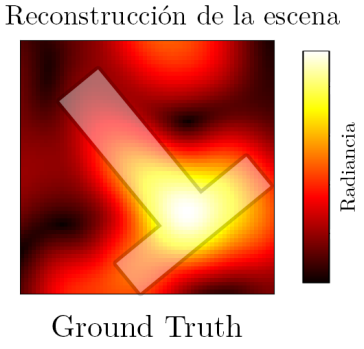

Figura 5.18: Reconstrucción de la escena T detrás de dos esquinas mostrada en la Figura 5.2, donde se superpone la escena oculta a la reconstrucción obtenida como referencia.

Cabe destacar que las reconstrucciones para escenas detrás de dos esquinas resultan mucho más complejas, por lo que no se puede esperar obtener resultados tan visuales como en las escenas detrás de una esquina, en las que se vislumbraba sin ningún problema la geometría oculta. Lo esperable es obtener una ligera intuición de que dicha geometría se encuentra detrás de las dos esquinas, como en el caso de la reconstrucción de la Figura 2.5, obtenida del trabajo de Royo et al. [5].

En la Figura 5.19 se muestran las reconstrucciones para la escena de referencia con 5 millones de muestras por píxel, sin modificaciones adicionales y con *Hidden Geometry Rejection Sampling*, utilizando 500.000 muestras por píxel para estas últimas.

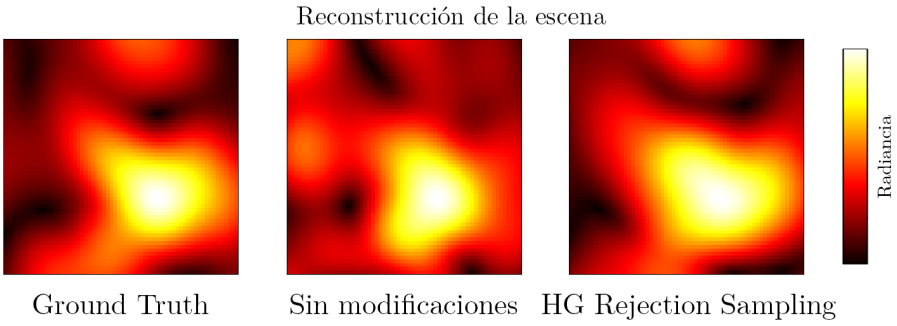

Figura 5.19: Reconstrucciones obtenidas utilizando la escena de referencia *Ground Truth*, la versión sin modificaciones adicionales y la versión utilizando *Hidden Geometry Rejection Sampling*)

Se puede observar como los resultados en la escena de referencia y utilizando *Hidden Geometry Rejection Sampling* son muy parecidos, obteniendo una ligera silueta de la letra T que nos proporciona la información de que detrás de las dos esquinas hay un objeto con una forma similar a la obtenida.

5.3.3. T_nlos_2corner_big: Escena T NLOS detrás de dos esquinas, superficie visible grande

Como tercera prueba se ha utilizado la escena «*T_nlos_2corner_big*» de la Figura 5.3, una escena con una geometría oculta (letra T) detrás de dos esquinas, que se diferencia de la escena anterior en que la superficie secundaria tiene un tamaño mayor.

De igual forma que en todas las pruebas, se ha utilizado una escena de referencia con la base de Royo et al. [1] con 5 millones de muestras por píxel, llamado valor *Ground Truth*. Se ha utilizado dicha base como configuración de una escena sin modificaciones adicionales, utilizando 10.000 muestras por píxel, y una escena para cada una de las técnicas implementadas, *Hidden Geometry Rejection Sampling*, kernel Gaussiano y Epanechnikov, utilizando en todas 10.000 muestras por píxel.

Cabe destacar que la evaluación se ha realizado para los instantes en los que la señal llega después de haber viajado a través de las dos esquinas, no para la señal completa, ya que este es el fragmento que nos resulta más relevante.

El procedimiento seguido ha sido idéntico al de las pruebas anteriores, por lo que se procederá directamente a mostrar la evaluación de la convergencia de las diferentes configuraciones de escena.

Evaluación de la convergencia De forma similar a las escenas anteriores, se han utilizado las métricas comentadas en el Apartado 5.2, *MSE* y *PSNR*. Para ambas métricas se han evaluado los resultados obtenidos, utilizando la Ecuación 5.1 para calcular el valor *MSE* de cada técnica, y la Ecuación 5.2 para calcular el *PSNR*.

En la Figura 5.20 se muestran los resultados obtenidos después de realizar la evaluación utilizando el error cuadrático medio o *MSE* para cada técnica. Cuanto menor sea el dato obtenido, menor es la diferencia de esa escena con respecto a la escena de referencia.

La escena con el menor valor *MSE* es la que utiliza estimación de densidad con kernel Gaussiano, seguido de cerca del kernel Epanechnikov y de *Hidden Geometry Rejection Sampling*. En términos porcentuales, la mejora con respecto a la escena sin modificaciones es de un 65,75 % al utilizar kernel Gaussiano, de un 60,34 % con kernel Epanechnikov y por último de un 49,72 % al utilizar muestreo por rechazo. Respecto a la escena anterior, al utilizar una superficie secundaria de mayor tamaño,

aumentan las posibilidades de que los rayos trazados con un *path tracer* convencional alcancen dicha superficie, por lo que el efecto de *Hidden Geometry Rejection Sampling* será mucho menos significativo. Esto provoca la caída de mejora porcentual del muestreo con rechazo con respecto a las escenas anteriores.

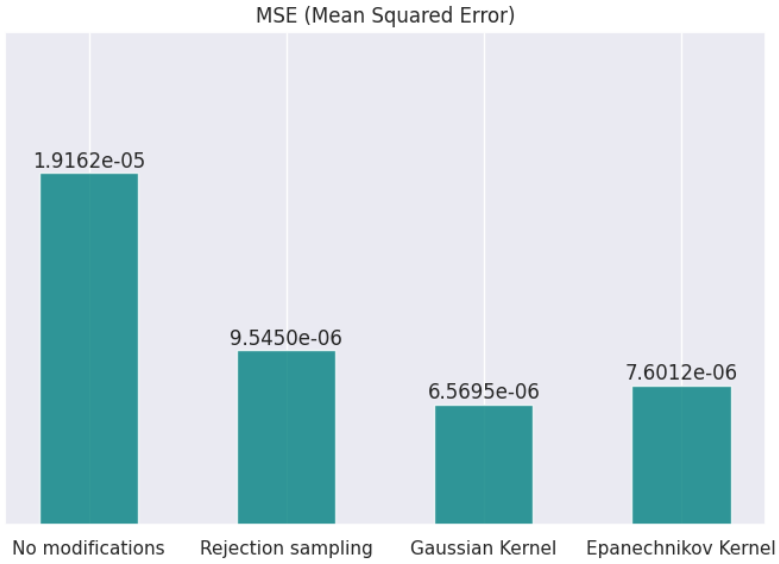

Figura 5.20: Valores *MSE* obtenidos para cada escena, utilizando el mismo número de muestras por píxel y configuración de escena para todas ellas. Cuanto menor sea el valor *MSE* obtenido, menor diferencia con respecto a la escena de referencia.

En la Figura 5.21 se muestran los resultados obtenidos después de realizar la evaluación utilizando la proporción máxima de señal a ruido o *PSNR* para cada técnica. Cuanto mayor sea el dato obtenido, menor es la diferencia de esa escena con respecto a la escena de referencia. Los valores se muestran utilizando escala logarítmica, expresada en decibelios (dB).

De igual forma que el error cuadrático medio, los resultados muestran que la técnica de estimación de densidad utilizando kernel Gaussiano es la que más mejora produce en la escena mejorando el resultado en 5 dB, siendo esta una mejora del 47,36 %. La mejora de la escena sin modificaciones con respecto a la que utiliza kernel Epanechnikov es de un 41,10 % y la que utiliza *Hidden Geometry Rejection Sampling* de un 30,94 %. De nuevo, se puede extrapolar la misma explicación sobre la técnica de muestreo por rechazo que en el error *MSE*.

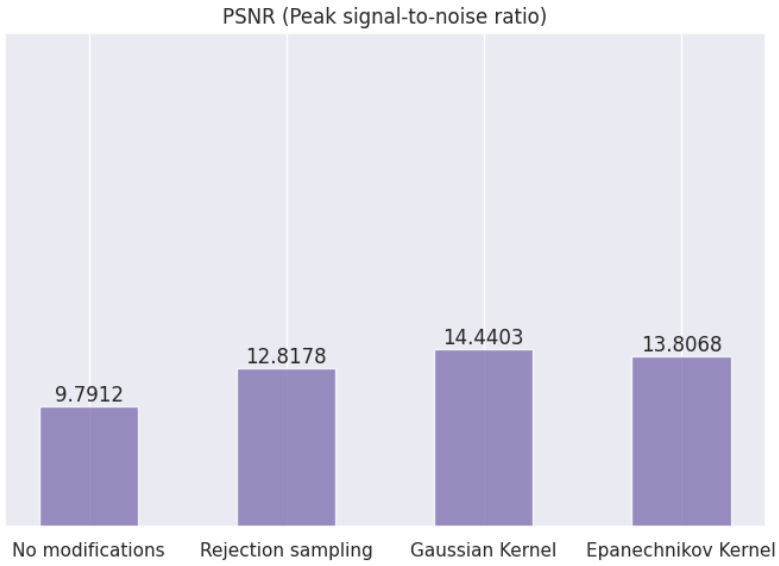

Figura 5.21: Valores *PSNR* obtenidos para cada escena, utilizando el mismo número de muestras por píxel y configuración de escena para todas ellas. Cuanto mayor sea el valor *PSNR* obtenido, mayor similaridad con la escena de referencia. (Valores expresados en decibelios, dB. Escala logarítmica.)

Evaluación del tiempo de renderizado A continuación se realizará la comparación de los tiempos de renderizado para cada técnica utilizada. Los resultados obtenidos se muestran en la Tabla 5.3.

Escena	**Muestras por píxel**	**Tiempo**	**MSE ↓**	**PSNR ↑**
Ground Truth	5.000.000	21 min 23s	-	-
Sin modificaciones	10.000	12,08 s	$1,92 \times 10^{-5}$	9,79
HG Rejection Sampling	10.000	1 min 9 s	$9,55 \times 10^{-6}$	12,82
Kernel Gaussiano	10.000	14,42 s	$6,57 \times 10^{-6}$	14,44
Kernel Epanechnikov	10.000	14,92 s	$7,60 \times 10^{-6}$	13,80

Cuadro 5.3: Tiempos de renderizado para cada técnica evaluada anteriormente, *Ground Truth*, sin modificaciones, *Hidden Geometry Rejection Sampling* y los estimadores de densidad Gaussiano y Epanechnikov.

Se han utilizado 5 millones de muestras por píxel para la escena *Ground Truth*, así como 10.000 muestras por píxel para el resto de escenas de la Tabla 5.3.

Los resultados indican mejoras importantes con respecto al tiempo de renderizado al utilizar cualquiera de las nuevas técnicas con respecto a la escena de referencia, llegando incluso a un 99 % de mejora de los estimadores de densidad con respecto a la escena de referencia.

5.3.4. Escena cornell box con línea de visión

Como prueba final, se ha utilizado una escena con linea de visión para confirmar que los algoritmos de estimación de densidad implementados causan mejoría en el resultado final de la imagen transitoria, como se menciona en Jarabo et al. [12]. Se trata de una escena de tipo *Cornell Box*, la cual consiste en una caja cerrada con uno de sus lados abiertos, con dos elementos geométricos dentro. En la Figura 5.22 se muestran los resultados obtenidos después del renderizado sin utilizar modificaciones (con 15.000 muestras por píxel como referencia y con 1000 para comparaciones), con estimación de densidad mediante kernel Gaussiano y utilizando kernel Epanechnikov, durante varios instantes de tiempo.

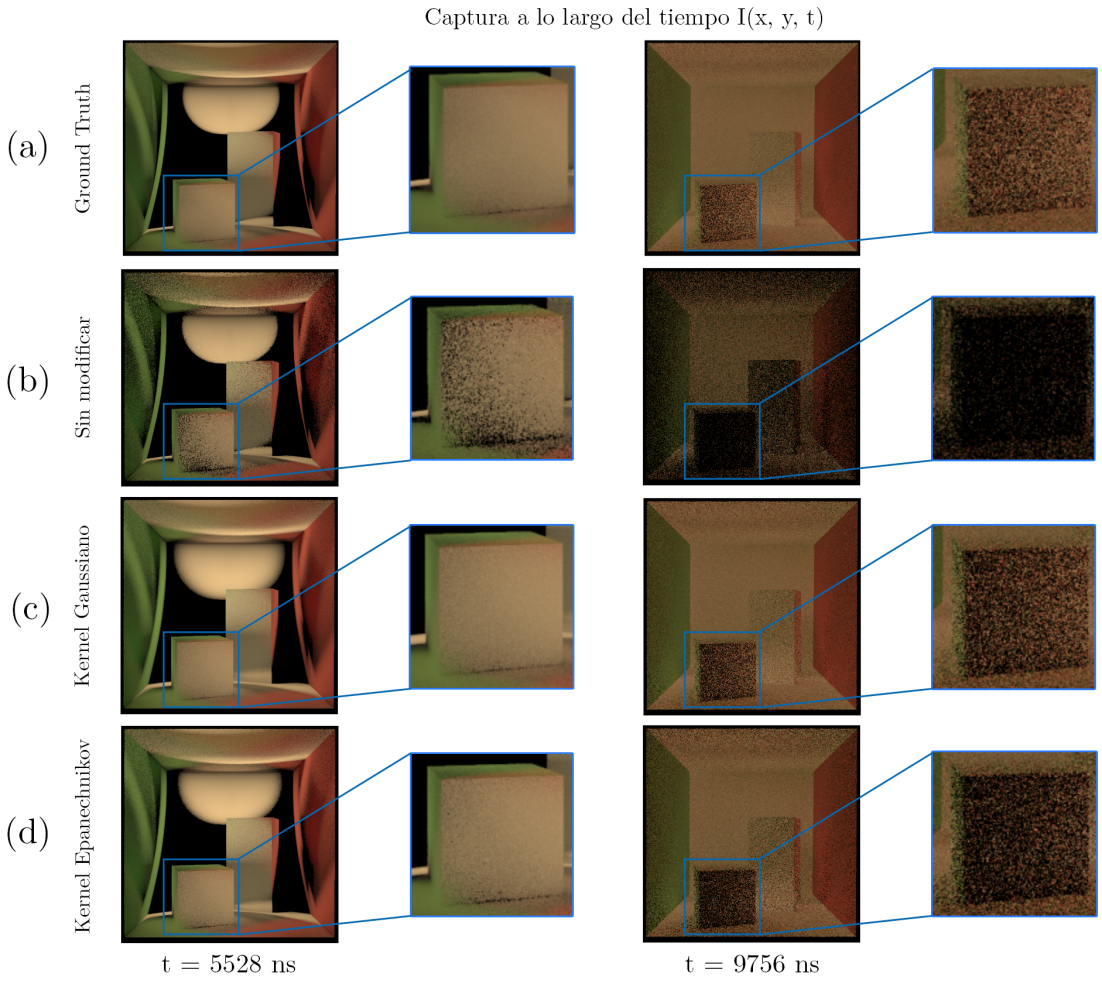

Figura 5.22: Resultados obtenidos para la escena «*cornell_box*». (a) Resultados de referencia utilizando 15.000 muestras por píxel. (b) Resultados sin utilizar modificaciones. (c) Resultados utilizando estimación de densidad con función Gaussiana, $\sigma = 3$. (d) Resultados obtenidos utilizando estimación de densidad con kernel Epanechnikov, $b = 3$. 1000 muestras por píxel para las escenas (b-d).

Se puede observar claramente una mejoría notoria respecto al ruido de la imagen sin modificaciones en ambos casos, obteniendo resultados prácticamente idénticos a los de referencia, tardando menos tiempo en converger. Por lo tanto, se puede concluir que ambos métodos de reconstrucción pueden ser utilizados sin duda para cualquier tipo de escena transitoria.

En contraposición con los métodos de reconstrucción, el método de muestreo de *Hidden Geometry Rejection Sampling* no resulta de utilidad en escenas con línea de visión, ya que al no haber escena oculta no resulta necesario obtener muestras de una geometría específica, si no muestrear toda la geometría.

Capítulo 6

Conclusiones

Este trabajo se ha centrado en la simulación de transporte de luz en escenas sin línea de visión [1]. Se ha propuesto un nuevo método de muestreo (*Hidden Geometry Rejection Sampling*) y se han implementado métodos de reconstrucción ya existentes (*Kernel Density Estimation*), aplicándolos a la simulación del transporte de luz transitorio [12] con geometría detrás de dos esquinas, el cual requería de un gran coste computacional utilizando los métodos ya existentes. Estos métodos mejoran los resultados obtenidos en convergencia y tiempo, tanto de la señal transitoria como de la reconstrucción de la escena oculta respecto a los métodos ya existentes

Para evaluar nuestros métodos se han utilizado métricas cualitativas y cuantitativas. Se ha conseguido ver que, utilizando el mismo tiempo de renderizado que las escenas sin nuevas modificaciones, han mejorado los resultados (gracias a los estimadores de densidad Gaussiano y Epanechnikov). Por otro lado, con algo más de tiempo de renderizado, se han mejorado de forma muy importante los resultados al utilizar muestreo por rechazo. De forma más concreta, se han obtenido mejoras porcentuales sobre la versión sin modificaciones de hasta un 93 % en escenas detrás de una esquina y un 79 % en escenas detrás de dos esquinas en lo que respecta al error cuadrático medio. En términos temporales, se han obtenido reducciones respecto a la escena de referencia de más de un 90 % en escenas detrás de una esquina, e incluso del 99 % en determinadas escenas detrás de dos esquinas al utilizar los estimadores de densidad.

La implementación de las diferentes técnicas de muestreo y reconstrucción ha sido de gran complejidad, ya que ha resultado necesario un estudio previo exhaustivo acerca de todo aquello sobre lo que está sustentado el trabajo, imagen estacionaria,

transitoria y sin línea de visión, además de un proceso de documentación sobre el renderizador Mitsuba 3 [7] y sus extensiones para simulación transitoria y sin línea de visión, las cuales han sido modificadas. También ha sido necesario realizar un estudio previo extenso acerca de las técnicas estadísticas utilizadas como base para *Hidden Geometry Rejection Sampling* y *Kernel Density Estimation*.

Este trabajo será utilizado en futuros proyectos de investigación, sirviendo de ayuda en el renderizado de escenas sin línea de visión especialmente a través de dos esquinas, aunque se ha demostrado su gran aportación a escenas detrás de una esquina e incluso escenas transitorias con línea de visión, utilizando los estimadores de densidad.

Como trabajo futuro a partir de este trabajo realizado, sería posible implementar nuevos kernels para la estimación de densidad mencionados en Jarabo et al. [12], ya que al haber desarrollado el código de forma modular no resultaría demasiado costoso.

También sería posible tratar de mejorar los resultados de *Hidden Geometry Rejection Sampling* combinándolo con otras técnicas utilizando *Multiple Importance Sampling* o *Russian Roulette*, así como dividir la geometría del segundo *relay wall* en estratos para hacer muestreo estratificado (*Stratified Sampling*).

Gracias a este trabajo se espera que se puedan obtener mejoras en las situaciones en las que la imagen sin línea de visión resulta de gran utilidad, y que se pueda seguir mejorando en un campo con mucho futuro y posibilidades.

Personalmente, el desarrollo de este trabajo me ha resultado de gran interés, ya que me ha permitido ver cómo trabajan los grupos de investigación ligados a la universidad, además de poder utilizar los conocimientos adquiridos durante estos años de estudio, especialmente los de la asignatura Informática Gráfica, pero también muchas otras, como Matemáticas I y Estadística en materia relacionada con integrales y conceptos matemáticos y estadísticos o Programación de Sistemas Concurrentes y Distribuidos ya que el renderizado en Mitsuba 3 utiliza programación concurrente al lanzar varios hilos de renderizado al mismo tiempo. También es importante destacar la asignatura de Proyecto Software, la cual me ha permitido tener una mejor visión sobre cómo redactar una memoria relacionada con un proyecto informático, así como todas las asignaturas relacionadas con la programación para desarrollar el código necesario.

Capítulo 7

Bibliografía

[1] Diego Royo, Jorge Garcia, Pablo Luesia-Lahoz, Julio Marco, Diego Gutiérrez, Adolfo Muñoz, and Adrián Jarabo. Non-line-of-sight transient rendering. New York, NY, USA, 2022. Association for Computing Machinery.

[2] Tomohiro Maeda, Guy Satat, Tristan Swedish, Lagnojita Sinha, and Ramesh Raskar. Recent advances in imaging around corners, 2019.

[3] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Moungi Bawendi, and Ramesh Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature communications*, 3:745, 03 2012.

[4] Xiaochun Liu, Sebastian Bauer, and Andreas Velten. Phasor field diffraction based reconstruction for fast non-line-of-sight imaging systems. *Nature Communications*, (11), 2020.

[5] Diego Royo, Talha Sultan, Adolfo Muñoz, Khadijeh Masumnia-Bisheh, Eric Brandt, Diego Gutierrez, Andreas Velten, and Julio Marco. Virtual mirrors: Non-line-of-sight imaging beyond the third bounce. *ACM Transactions on Graphics*, 42(4), 2023.

[6] Adrian Jarabo, Belen Masia, Julio Marco, and Diego Gutierrez. Recent advances in transient imaging: A computer graphics and vision perspective, 2016.

[7] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org.

[8] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022.

[9] Miguel Crespo, Diego Royo, and Jorge Garcia. Mitsuba 3 transient, 2022. https://github.com/mcrescas/mitsuba3-transient.

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

[11] Andreas Velten, Di Wu, Adrian Jarabo, Belen Masia, Christopher Barsi, Chinmaya Joshi, Everett Lawson, Moungi Bawendi, Diego Gutierrez, and Ramesh Raskar. Femto-photography: Capturing and visualizing the propagation of light. *ACM Trans. Graph.*, 32(4), jul 2013.

[12] Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. A framework for transient rendering. *ACM Trans. Graph.*, 33(6), nov 2014.

[13] Victor Arellano, Diego Gutierrez, and Adrian Jarabo. Fast back-projection for non-line of sight reconstruction. *Opt. Express*, 25(10):11574–11583, May 2017.

[14] Xiaochun Liu, Ibón Guillén, Marco La Manna, Ji Hyun Nam, Syed Azer Reza, Toan Huu Le, Adrian Jarabo, Diego Gutierrez, and Andreas Velten. Non-line-of-sight imaging using phasor-field virtual wave optics. *Nature*, pages 1–4, 2019.

[15] Wenzel Jakob Matt Pharr and Greg Humphreys. Physically based rendering: From theory to implementation. 2004-2021.

[16] Matthew P. Wand and William R. Schucany. Gaussian-based kernels. *Canadian Journal of Statistics*, 18(3):197–204, 1990.

[17] Diego Royo and Pablo Luesia-Lahoz. (Your)-Transient Auxiliary Library, 2022. https://github.com/diegoroyo/tal.

Lista de Figuras

Anexos

Apéndice A

Planificación temporal

El trabajo se ha desarrollado en aproximadamente 315 horas, repartidas a lo largo de tres meses. Las distintas fases en las que se ha dividido el proyecto se muestran en el diagrama de Gantt de la Figura A.1, las cuales han sido: estudio previo acerca de la imagen transitoria y NLOS, documentación y pruebas sobre el simulador Mitsuba 3, implementación de la técnica de muestreo *Hidden Geometry Rejection Sampling*, implementación de la técnica de reconstrucción *Kernel Density Estimation*, evaluación de dichas técnicas utilizando métricas de error cuantitativas y cualitativas, pruebas de las técnicas implementadas utilizando reconstrucciones y redacción de la memoria del trabajo realizado.

El código desarrollado se encontrará público en el futuro en GitHub[1]

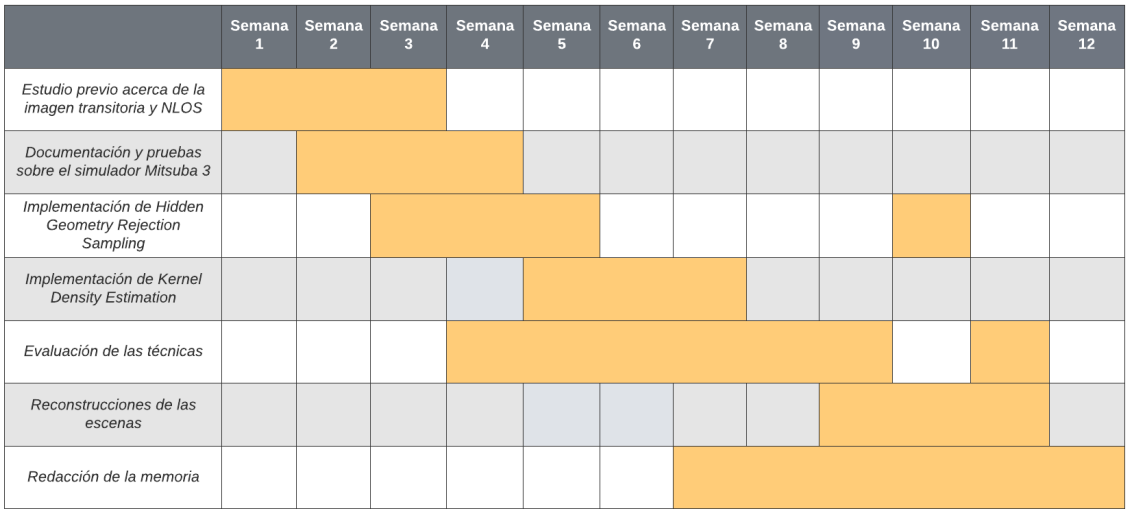

Figura A.1: Diagrama de Gantt correspondiente al trabajo realizado.

[1]Enlace al repositorio: `https://github.com/Ismati5/mitsuba3-transient-nlos.git`