UNIVERSITAT DE BARCELONA

Trabajo Final de Grado.

GRADO DE INFORMÁTICA.

Facultad de Matemáticas e Informática.

Universidad de Barcelona.

Generación de Imágenes Fotorrealistas de Alta Resolución con la Técnica de *Upsampling*.

Autor: Ian Phoenix Carlos Casillas.

Director: Dr. Ricardo Marques.

Realizado a: Departamento de Matemáticas e Informática.

Barcelona, a 13 de junio de 2022.

Abstract

The rendering of a 3D virtual scene to a photorealistic image is a procedure with a high computational complexity, it depends on various factors, one of the most important being illumination, simulating light rays, the number of bounces they can have on the surfaces of 3D models, how this will affect the scene, among other things, requires many calculations, to arrive at a correct approximation, this is directly reflected in high execution times for the generation of quality images.

One of the current solutions to generate quality photorealistic images is the path tracing rendering algorithm, which, being based on the Montecarlo method, will denote that the quality of its results is directly proportional to the number of samples (the simulated light rays per pixel), requiring an extremely high number of samples to achieve a quality photorealistic image.

Given this problem, during the development of this Final Degree Project, various upsampling techniques have been developed, with the basic procedure that working with an image generated at lower resolution by path tracing, using different metrics to obtain the best possible interpolation, to have as a final result, a high-resolution photorealistic image with equal or superior quality in less time than the path tracing could take.

The algorithms developed here expose different theories about why that metric and/or interpolation supposes an improvement, seeking that each new proposed algorithm complements or covers aspects not contemplated by the previous one.

The results obtained with the virtual 3D scenes created, compared to path tracing, allow us to understand the operation of the algorithm while demonstrating that with the appropriate parameters depending on the algorithm used, the quality photorealistic image is achieved in less execution time.

Resum

La renderització d'una escena virtual 3D a una imatge fotorealista és un procediment amb una complexitat computacional elevada, depèn de diversos factors, sent la il·luminació un dels més importants, simular els raigs llum, el nombre de rebots que poden tenir sobre les superfícies de els models 3D, com afectarà això a l'escena, entre altres coses, requereix una gran quantitat de càlculs per arribar a una aproximació encertada, això es veu reflectit directament en temps d'execució elevats per a la generació d'imatges de qualitat.

Una de les solucions actuals per generar imatges fotorealistes de qualitat, és l'algorisme de renderitzat path tracing, el qual basat en el mètode de Montecarlo, denotarà que la qualitat dels seus resultats és directament proporcional al nombre de mostres (els raigs de llum simulats per píxel), i cal una quantitat de mostres molt alta per arribar a una imatge fotorealista de qualitat.

Donada aquesta problemàtica, durant el desenvolupament del present Treball Final de Grau, s'han desenvolupat diverses tècniques d'upsampling, amb el procediment base que treballant amb una imatge generada pel path tracing de menor resolució, utilitzant diferents mètriques per obtenir la millor interpolació possible, obtenir com a resultat final, una imatge fotorealista d'alta resolució amb una qualitat igual o superior en menor temps que amb el mètode de path tracing.

Els algorismes aquí desenvolupats exposen diferents teories sobre el perquè aquesta mètrica i/o interpolació suposa una millora, buscant que cada nou algoritme plantejat complementi o inclogui aspectes no contemplats per algoritme anterior.

Els resultats obtinguts amb les escenes virtuals 3D creades, comparats al path tracing permeten comprendre el funcionament de l'algoritme a l'hora de demostrar que amb els paràmetres adequats segons l'algoritme utilitzat, s'aconsegueix la imatge fotorealista de qualitat en un temps d'execució menor.

Resumen

La renderización de una escena virtual 3D a una imagen fotorrealista es un procedimiento con una complejidad computacional elevada, depende de diversos factores, siendo la iluminación uno de los más importantes, simular los rayos luz, el número de rebotes que pueden tener sobre las superficies de los modelos 3D, el cómo afectará esto a la escena, entre otras cosas, requiere de una gran cantidad de cálculos para llegar a una aproximación acertada, esto se ve reflejado directamente en tiempos de ejecución elevados para la generación de imágenes de calidad.

Una de las soluciones actuales para generar imágenes fotorrealistas de calidad, es el algoritmo de renderizado *path tracing*, el cuál siendo basado en el método de Montecarlo, denotará que la calidad de sus resultados es directamente proporcional al número de muestras (los rayos de luz simulados por píxel), siendo necesario una cantidad de muestras muy alta para llegar a una imagen fotorrealista de calidad.

Dada esta problemática, durante el desarrollo del presente Trabajo Final de Grado, se han desarrollado diversas técnicas de *upsampling*, con el procedimiento base de que trabajando con una imagen generada por el *path tracing* de menor resolución, utilizando diferentes métricas para obtener la mejor interpolación posible, tener como resultado final, una imagen fotorrealista de alta resolución con una calidad igual o superior en menor tiempo al que tomaría el *path tracing*.

Los algoritmos aquí desarrollados exponen diferentes teorías sobre el porqué esa métrica y/o interpolación supone una mejora, buscando que cada nuevo algoritmo planteado complemente o abarque aspectos no contemplados por el anterior.

Los resultados obtenidos con las escenas virtuales 3D creadas, comparados al *path tracing* permiten comprender el funcionamiento del algoritmo al mismo tiempo de demostrar que con los parámetros adecuados según sea el algoritmo utilizado, se consigue la imagen fotorrealista de calidad en un menor tiempo de ejecución.

Agradecimientos

Esta ha sido una de las secciones, que más he pensado, y re – pensado a lo largo de la redacción de la presente memoria, pues, simboliza mucho para mí, el final de una etapa de mi vida, de un largo viaje, que por momentos parecía no tener fin, dónde experimente muchas emociones, positivas y negativas (aunque siendo notablemente marcado por éstas últimas), al igual que ello, he podido compartirlo con diferentes personas, será una lista de agradecimientos larga, pues, para suerte mía, hay mucha gente con la que me siento agradecido.

A mi asesor el Dr. Ricardo Marques, por su gran paciencia y esmero, que, a pesar de mis obvias dificultades, nunca se rindió conmigo, me explicó las cosas las veces que hizo falta, que siempre fue amable al comentarme todo y me dio siempre ánimos de seguir adelante, no tienes idea de cuánto te agradezco.

A mi tía Claudia, que ha sido un apoyo en mi vida más tiempo de lo que yo he sido consciente, que siempre ha estado presente, viendo por nosotros de forma genuina, muchas gracias por permanecer en mi vida, le quiero.

A mi mamá Jéssica y a mi papá Félix, que se han esforzado por estar por mí, resolviendo situaciones, luchando y tratando de hacer lo que consideran mejor para mí, les extraño, y espero sepan que mi amor por ustedes es sincero.

Abuela, no sé cuándo leerás esto, espero que pronto, eres muy importante para mí, y aunque soy consciente que mi presencia aquí es un cúmulo de apoyos, sería una mentira decir que tú no eres la principal razón de que sea así, te amo abuela, y aunque en momentos hemos tenido nuestras diferencias (no se viven tantos años con alguien sin que sea así), no hay un solo día en que no agradezca que hayas decidido ver por mí.

A mi mamá Ileana, por quién siento un profundo orgullo y admiración, he visto su viaje, creo fervientemente que tiene un gran futuro por descubrir, me alegra ser testigo y compañero en esta travesía, le quiero mucho, le agradezco todo lo que ha dado por mí, yo creo en usted, gracias por quererme de la forma en que me quiere, le quiero y no imagino un mejor hogar que el que tenemos ahora.

Marisol, mi novia, gracias por haberme apoyado este transcurso, con tu tiempo, paciencia y cariño, que tanto me han ayudado en los momentos difíciles, y a la gorda, nuestra bonita Boston Terrier, por siempre lograr sacarme una sonrisa, pero podamos vivir más logros juntos, agradezco mucho, que sean parte de mi vida, las quiero.

Este penúltimo párrafo de agradecimientos, se lo dedico a mis amigos: Omar, Dante, César, Mario, Damaris, muchas gracias por existir y estar a mi lado de forma sincera, en serio gracias.

Y finalmente, al Atlas, por mostrarme que no hay nada que con suficiente esfuerzo y tiempo no se pueda conseguir.

Índice

Índice de ilustraciones

Índice de algoritmos

Índice de ecuaciones

Índice de gráficas

Índice de tablas

1 Introducción

Imágenes fotorrealistas

Una imagen fotorrealista es aquella generada por computadora que busca imitar a las imágenes generadas por las cámaras fotográficas, mediante algoritmos con diversos cálculos para simular los efectos de la luz, sombras, texturas y cualquier otro componente que forme parte de la escena.

Un ejemplo de la capacidad actual para la generación de imágenes fotorrealistas se puede observar en la Ilustración 1, dónde podemos ver una comparativa entre una fotografía (a) de la antigua ciudad maya de Uxmal, en México y una imagen fotorrealista (b) del videojuego *Forza Horizon 5* lanzado en 2021, desarrollado por *Playground Games* y distribuido por *Xbox Game Studios*.

a) Fotografía de Uxmal, México.

b) Imagen fotorrealista de Uxmal, México parte del videojuego *Forza Horizon 5.*

Ilustración 1. *Fotografía e imagen fotorrealista de Uxmal, México.*
Fuentes: (Fotografía) Canal de YouTube ElAnalistaDeBits y (Fotografía fotorrealista) Videojuego Forza Horizon 5.

La generación de una imagen fotorrealista es un proceso complicado, el cual, dependiendo de la técnica utilizada, puede tener un coste computacional alto, llegando a tomar un tiempo considerable el poder crear una sola imagen.

Aplicaciones de las imágenes fotorrealistas

Las imágenes fotorrealistas son útiles en multitud de disciplinas, y conforme avanza la tecnología, sus usos se van diversificando, cómo lo es ahora el auge de las tecnologías de realidad virtual o aumentada, dónde las renderizaciones fotorrealistas influyen en generar una experiencia más inmersiva para el usuario.

Los usos más conocidos de las imágenes fotorrealistas se dan en programas específicos para la arquitectura, diseño o ingeniería (siendo estas las más populares, pero no exclusivas), dónde una visualización precisa de la escena 3D resulta de suma importancia para la operación realizada.

Otro uso popular sería en las películas de animación 3D, que con el pasar de los años han ido mejorando sus tecnologías para tener animaciones más fluidas y de mejor calidad al contar con cada vez mayor potencia para su renderizado, un claro ejemplo es *Disney Pixar*, dónde se puede ver como películas que han tenido continuaciones de más de una década de la inicial, han mejorado notoriamente desde las texturas de los modelos 3D hasta la iluminación presente en la escena.

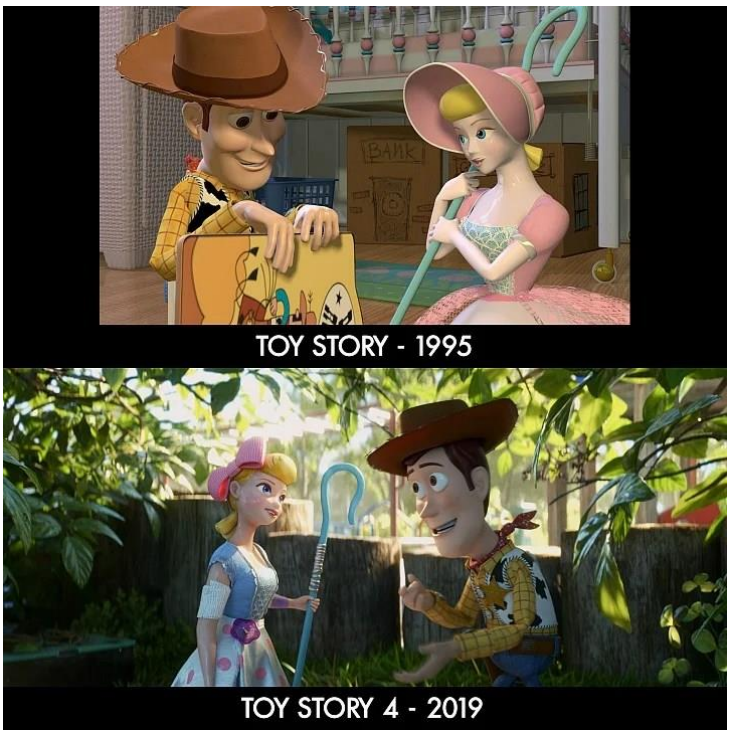


Ilustración 2. *Diferencia de calidad de animación en Toy Story y Toy Story 4 de Disney Pixar. Fuente: https://www.thesuburbanmom.com/*

En la Ilustración 2, se puede observar el notable cambio que existe entre la animación de Toy Story de 1995, y Toy Story 4 de 2019, podemos apreciar una definición más profunda de las texturas (como se puede ver, por ejemplo, en las hojas, la madera) y una simulación de la iluminación global fotorrealista.

Lamentablemente el que se tenga más potencia para generar imágenes de mayor calidad, aún radica en una alta demanda al software y hardware, para la ejecución de los algoritmos de renderizado, siendo aún una problemática actual, los amplios tiempos de ejecución para la renderización de imágenes fotorrealistas.

Renderización

Verma (Verma et al, 2010) describe que, en el mundo real, las fuentes de luz emiten fotones que en su travesía pueden interactuar con diferentes objetos.

Al momento en que uno de estos fotones se encuentra con una superficie, este puede ser absorbido, reflejado, o transmitido; finalmente al pasar por la retina del

espectador o el sensor de una cámara, es cuando se genera imagen (representación 2D) del entorno.

Estos entornos pueden ser simulados por ordenador, dónde el entorno es reemplazado por una escena virtual 3D y la interacción de la luz puede ser simulada por diferentes algoritmos (Verma et al, 2010).

La renderización, es la técnica que busca generar imágenes a partir de una escena 3D creada por ordenador a través de un algoritmo definido para esto (Watt, 1999).

La escena virtual es compuesta por diversos elementos como lo son:

- **Cámara**: es el punto de vista desde el cuál la escena es observada, y por ende el cuál será la imagen generada.
- **Objetos en escena**: los diversos modelos 3D que conforman la escena, pueden ser representaciones de objetos del mundo real u objetos ficticios generados por el ordenador, pudiendo tener color y textura.
- **Fuentes de luz**: es el componente de la escena que iluminará la misma para poder visualizarla, al igual que una habitación sin luz, una escena carente de la misma no permite visualizarla propiamente.

Cabe comentar que, la escena 3D puede ser completa o parcialmente observada por la cámara, pero solo aquella parte de esta, que es observada es que será apreciada en la imagen final generada.

Las técnicas de renderización comenzaron desarrolladas a partir de software, pero durante los últimos años, compañías como Nvidia han desarrollado técnicas de renderizado en GPU, como lo es OptiX (Marques et al, 2010).

Downsampling y *upsampling*

Dumitrescu explica los conceptos de *downsampling* como la técnica empleada para reducir la resolución de una imagen de entrada, lo cual es muy útil para reducir el tamaño de almacenamiento de imágenes mientras se preserva tanta de su información como es posible; *upsampling* es el proceso inverso, consiste en obtener una imagen de salida de mayor resolución que la imagen de entrada, en la práctica, se enfoca en obtener una imagen de alta confianza, que es aquella que no presenta elementos no deseados, como lo puede ser el ruido, y que mantiene altos niveles de detalle (D. Dumitrescu et al, 2019).

En este trabajo final de grado, tiene como interés principal, aprovechar diferentes técnicas de *upsampling* para generar imágenes de alta resolución sin perder calidad en un menor tiempo de ejecución.

1.1 Motivación del trabajo

Soy un estudiante extranjero que llegó a este país con un visado de estudios.

A lo largo del grado pasé por muchas materias que me resultaron difíciles, cuya curva de aprendizaje resultó un reto para mí, pero que, al conseguir afrontarlas, terminaba con una sensación de aprendizaje, satisfacción personal e inclusive curiosidad más allá de lo visto en el temario.

Una de esas materias fue la de Gráficos y Visualización de Datos, la cual, aunque me costó mucho poder superarla, me dejó un interés peculiar sobre este campo de la informática, misma materia dónde tuve mi primer contacto con mi asesor, el cual demostró su habilidad como guía docente durante el curso.

Así que, al momento de elegir un trabajo, esta resultó ser una propuesta más que interesante para mí, que una vez charlada con mi asesor, pude optar a ella.

1.2 Objetivos

Los objetivos planteados para el presente TFG son los siguientes:

- Encontrar un enfoque inicial que permita proveer buenos resultados con cambios al proceso original de la generación de la imagen fotorrealista.
- Reducir el tiempo de obtención de una imagen fotorrealista.
- Utilizar diferentes técnicas para mejorar los resultados obtenidos sobre la generación de la imagen fotorrealista.

Todos los objetivos anteriores son abordados con la técnica de *upsampling* (detallada en el tema de *downsampling* y *upsampling* que se encuentra previamente redactado en la Sección 1).

2 Estado del arte

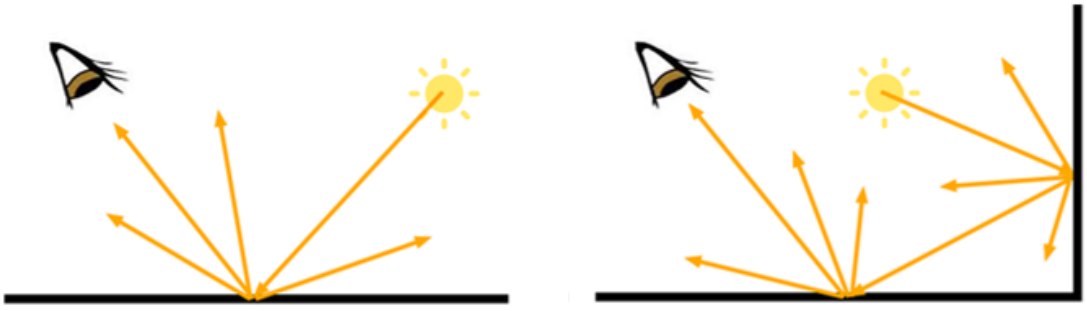
Los seres humanos podemos observar nuestro entorno porqué los fotones emitidos por las fuentes de luz (como lo puede ser una lámpara, o el sol), rebotan sobre las superficies de los objetos, Ferreruela explica que el estímulo luminoso que llega al ojo es enfocado por la córnea y el cristalino en la retina, que lo transforma en estímulo eléctrico. Las neuronas de la retina se juntan para formar el nervio óptico, que conduce estos impulsos hasta las áreas occipitales de la visión (cerebro) para la interpretación (Ferreruela, 2007).

2.1 Iluminación

La iluminación juega un papel crucial en nuestra percepción visual del mundo, y podemos catalogarla en los siguientes tipos:

- **Iluminación directa:** se dice de la iluminación que se da cuando los rayos de luz rebotan una sola vez desde una superficie hasta el ojo.
- **Iluminación indirecta:** se dice de la iluminación que se da cuando los rayos de luz rebotan entre múltiples superficies antes de llegar al ojo.
- **Iluminación global:** se dice de la iluminación que se da como suma de ambas iluminaciones, directa e indirecta.

En la Ilustración 3 se puede ver una explicación visual de cómo funcionan la iluminación directa (a) y la indirecta (b).

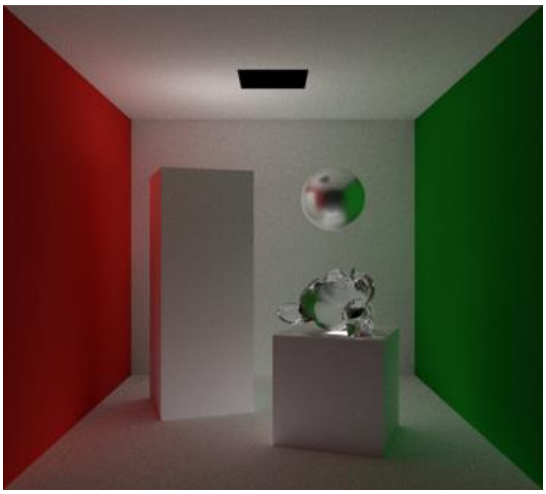


a) Iluminación directa. b) Iluminación indirecta.

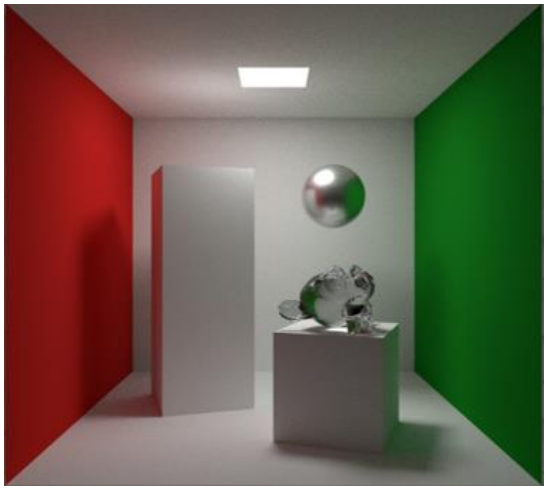
***Ilustración 3**. Tipos de iluminación: directa e indirecta.*
Fuente: https://www.scratchapixel.com/

El poder simular la iluminación global es importante para poder obtener imágenes fotorrealistas, en la Ilustración 4 se puede apreciar esto, teniendo como ejemplo, una imagen en tres definiciones, renderizando únicamente la iluminación directa (a), únicamente la iluminación indirecta (b) y finalmente, usando ambas, la iluminación global (c).

a) Imagen renderizando únicamente la iluminación directa.



b) Imagen renderizando únicamente la iluminación indirecta.



c) Imagen renderizada con iluminación global, es decir, usando ambas, directa e indirecta

Ilustración 4*. Imagen renderizada con iluminación, directa, indirecta y global.*
Fuente: https://sinmantyx.wordpress.com/

2.2 Trazado de rayos de luz

Simular la iluminación dentro de una escena 3D se realiza simulando rayos de luz y sus impactos sobre las diferentes superficies de la escena; en la vida real, los rayos de luz viajan de la fuente de origen, impactan sobre las superficies y llegan al ojo del espectador.

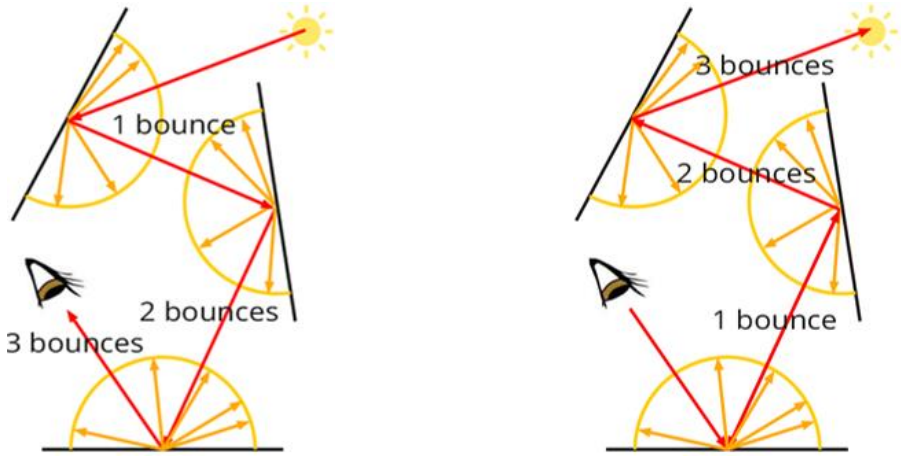
Las matemáticas nos permiten entender el comportamiento de la luz contra las diferentes superficies según su material, definido, por ejemplo, con las leyes de refracción y reflexión; por lo cual, simular por ordenador, el camino que puede tomar un rayo de luz es plenamente asequible.

Las formas de simular estos caminos del rayo de luz por ordenador son dos, explicadas por Farrell, de la siguiente manera:

- ***Forward tracing*****:** se trata de recrear el trazado de los rayos de luz desde la fuente, considerando los múltiples rebotes, hasta llegar el observador.

- ***Backward tracing*:** se trata de recrear el trazado de los rayos de luz desde el observado, considerando, usualmente dos o tres rebotes, a lo largo del camino hasta llegar a la fuente (Farrel et al, 2004).

En la Ilustración 5 se puede ver una explicación visual de cómo funcionan las técnicas de *forward tracing* (a) y *backward tracing* (b).



a) *forward tracing.* b) *backward tracing.*

Ilustración 5. *Tipos de trazado de rayos de luz: forward and backward tracing. Fuente: https://www.scratchapixel.com/*

Forward tracing es la simulación acorde a la realidad, pero no resulta ser una aproximación eficiente, dado a que no todos los rayos de luz recreados siempre llegarán al observador, por lo que se perdería eficiencia generando rayos de luz cuya información al final no podría ser aprovechada.

En este proyecto, se utiliza la aproximación de *backward tracing* para la generación de las imágenes fotorrealistas.

2.3 *Path Tracing*

Para generar imágenes fotorrealistas, es necesario proveer una solución que permita renderizar una imagen con iluminación global, dentro de este proyecto, se trabaja con el algoritmo de renderizado *path tracing.*

Path tracing es un algoritmo basado en el método de Montecarlo, ya que se basa en la obtención de muestras por probabilidad, lo que quiere decir que el algoritmo necesitará un número amplio de muestras para que la imagen resultante sea de calidad, ya que, al ser un muestreo aleatorio, un número bajo de muestras puede causar la aparición de ruido visual en la imagen final.

La forma en que este algoritmo funciona es, realizando un número definido de muestras (que serán rayos de luz que pasarán por la escena y rebotarán en las superficies) por píxel, para poder calcular el color promedio de obtenido de las muestras.

El Algoritmo 1 describe brevemente el algoritmo de *path tracing* utilizado en el proyecto.

Datos de entrada: *scene (Scene object)* **Datos de salida:** *none*
1. *por cada píxel en la imagen* 2. *por cada muestra* 3. *se genera un rayo para calcular el color de la muestra actual* 4. *se suma el color de la muestra actual para mejorar la definición del color final* 5. *fin-por* 6. *se asigna el color promedio de todas las muestras al píxel actual* 7. *fin-por*

***Algoritmo 1**. Algoritmo de path tracing para generar imágenes fotorrealistas.*

El algoritmo recibe como datos de entrada una escena 3D, se ejecuta el muestreo aleatorio de todos los píxeles para obtener el color final de cada píxel y finalmente se genera una imagen fotorrealista con extensión bmp.

En la Ilustración 6 se puede observar como el número de muestras utilizadas (el número blanco a la izquierda, sobre la división de la imagen) afecta a la cantidad de ruido presente en la imagen final, en este caso, usando una escena 3D del videojuego Minecraft (desarrollado por Mojang Studios) siguiendo la definición de la Caja de Cornell (Goral et al, 1984).

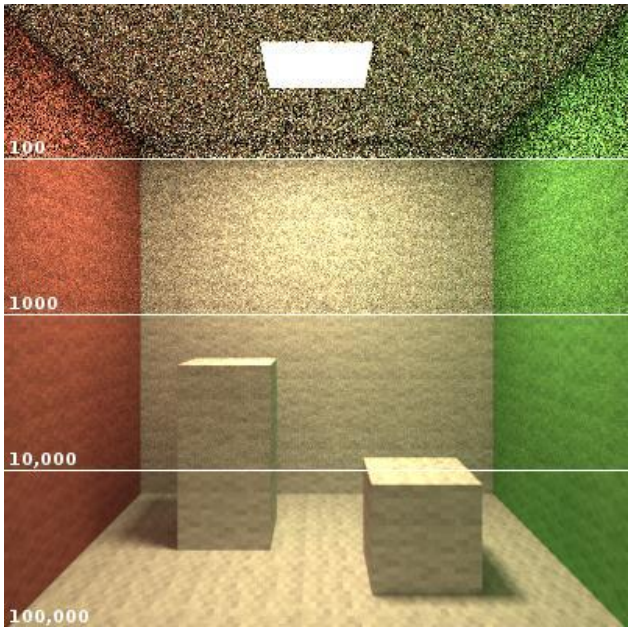


***Ilustración 6.** Imagen generada por algoritmo path tracing con diversas cantidades de muestras.*
Fuente: https://programmerclick.com/

Dado al gran número de muestras aleatorias requeridas para generar una imagen de calidad el tiempo de ejecución requerido por el *path tracing* incrementa conjunto al número de muestras utilizadas, es decir, a mayor número de muestras, mayor tiempo de ejecución.

3 Técnicas de *upsampling* para la iluminación global

3.1 Vista general

El objetivo principal de este proyecto es explorar e implementar técnicas de *upsampling* para acelerar la síntesis de imágenes fotorrealistas. En esta sección se aborda a detalle la planificación del proyecto, definiendo las tareas asignadas y los tiempos dados para estas, en semanas a lo largo del semestre.

Así mismo se describen las diferentes técnicas de *upsampling* desarrolladas a lo largo del proyecto para aproximarse a la problemática de reducción de tiempo de ejecución para la generación de una imagen fotorrealista.

3.2 Planificación

Se listan las tareas realizadas a lo largo del proyecto en la Tabla 1, así como el tiempo designado a cada una de ellas, además, en un diagrama de Gantt, en busca de facilitar la comprensión con un apoyo visual.

Tarea	**Descripción**
Tarea 1 – Familiarización	Familiarizarse con la problemática del renderizado fotorrealista, comprender cómo la técnica de *upsampling* puede optimizar esta situación.
Tarea 2 – Comprensión del software dado.	Comprender el código propuesto para el desarrollo del proyecto.
Tarea 3 – Implementación base	Configurar e implementar el código dado en el entorno dónde el estudiante realizaría el proyecto.
Tarea 4 – Planificación	Terminar de definir la planificación y alcance del proyecto.
Tarea 5 – Comprensión del algoritmo base: *Path tracing.*	Comprensión del p*ath tracing* y pruebas con el mismo.
Tarea 6 – Algoritmo: *Upsampling* basado en el vecino más cercano en el espacio imagen (2D).	Desarrollo del algoritmo *upsampling* basado en el vecino más cercano en el espacio imagen (2D) como aproximación a la optimización de la problemática.
Tarea 7 – Algoritmo: *Upsampling* basado en el promedio de los vecinos en el espacio imagen (2D).	Desarrollo del algoritmo *upsampling* basado en el promedio de los vecinos en el espacio imagen (2D) como aproximación a la optimización de la problemática.
Tarea 8 – Algoritmo: *Upsampling* basado en el promedio ponderado de	Desarrollo del algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio imagen (2D)

los vecinos en el espacio imagen (2D).	como aproximación a la optimización de la problemática.
Tarea 9 – Algoritmo: *Upsampling* basado en el promedio ponderado de los vecinos en el espacio escena (3D).	Desarrollo del algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena (3D) como aproximación a la optimización de la problemática.
Tarea 10 – Algoritmo: *Upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).	Desarrollo del algoritmo *u psampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) como aproximación a la optimización de la problemática.
Tarea 11 – Algoritmo: *Upsampling* intercalado basado en el promedio ponderado de los vecinos calculados en el espacio escena utilizando normales (3D).	Desarrollo del algoritmo *upsampling* intercalado basado en el promedio ponderado de los vecinos calculados en el espacio escena utilizando normales (3D) como paso previo de comprensión al algoritmo de muestreo adaptativo.
Tarea 12 – Algoritmo: *Upsampling* adaptativo basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).	Desarrollo del algoritmo *upsampling* adaptativo basado en el promedio ponderado de los vecinos en el espacio imagen escena normales (3D) como aproximación a la optimización de la problemática.
Tarea 13 – Pruebas y comparativas	Ejecución de pruebas y comparativas de los diversos algoritmos con los mismos parámetros.
Tarea 14 – Desarrollo de la memoria	Redacción de la memoria entregable.

***Tabla 1**. Planificación de las tareas del proyecto.*

Las tareas tenían un tiempo definido de una a tres semanas, el tiempo acordado para realizar las reuniones de seguimiento era de una semana, en las cuáles se discutía el trabajo realizado, complicaciones sobre el mismo y se definía los puntos a realizar para la siguiente semana, aunque dependiendo de las dificultades encontradas, la variación de las tareas podía variar.

El tema electo para realizar fue elegido el día 17 de septiembre del 2021, y el trabajo comenzó el día 03 de enero del 2022, dónde el estudiante comenzó a familiarizarse con los recursos bibliográficos dados por el tutor, y finalmente, las reuniones comenzaron el día 11 de febrero del 2022.

La Ilustración 7 es el diagrama de Gantt con la visualización gráfica de la planificación del proyecto.

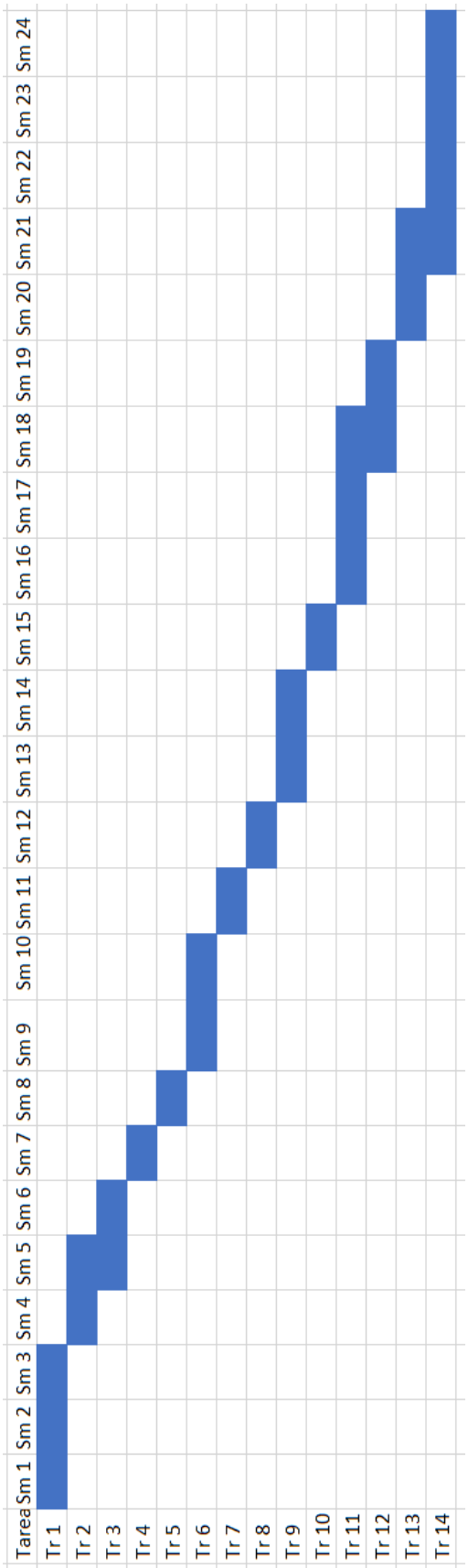


Ilustración 7. *Diagrama de Gantt detallando la duración en semanas de las tareas planeadas.*

3.3 Aproximación base previa a la técnica de *upsampling*

Para aproximarse a la problemática de los altos tiempos de ejecución requeridos por el *path tracing* se realizó el siguiente proceso de dos fases:

1. Utilizando el *path tracing* se genera una imagen de menor resolución en base a un factor de reducción (que afecta al largo y ancho de la imagen por igual).
2. Utilizando una técnica de *upsampling* desarrollada para el proyecto, generar una imagen de alta resolución basada en la imagen de baja resolución.

Para el paso dos, según sea el caso, la generación de la imagen de alta resolución en base a la imagen de baja resolución, se realiza también utilizando información geométrica de la imagen a baja resolución obtenida durante la generación de esta.

El Algoritmo 2 detalla brevemente el código utilizado dentro del paso 1 de la aproximación a la problemática.

Datos de entrada: *scene (Scene object)* **Datos de salida:** *none*
1. *se reduce la resolución en el largo y ancho según el factor de reducción para generar la imagen de baja resolución* 2. *por cada píxel en la imagen de baja resolución* 3. *por cada muestra* 4. *se genera un rayo para calcular el color de la muestra actual* 5. *se suma el color de la muestra actual para mejorar la definición del color final* 6. *fin-por* 7. *se asigna el color promedio de todas las muestras al píxel actual* *8. fin-por*

Algoritmo 2. *Algoritmo de preprocess para generar imágenes fotorrealistas de baja resolución.*

El algoritmo recibe como datos de entrada una escena 3D, se reduce la resolución deseada en base al factor de reducción dado (para el largo y ancho por igual), se ejecuta el muestreo aleatorio de todos los píxeles para obtener el color final de cada píxel y finalmente se genera una imagen fotorrealista con extensión bmp.

En la Ilustración 8 se puede observar de forma visual la aproximación utilizada para abordar la problemática de reducir el tiempo de ejecución requerido para la generación de una imagen fotorrealista.

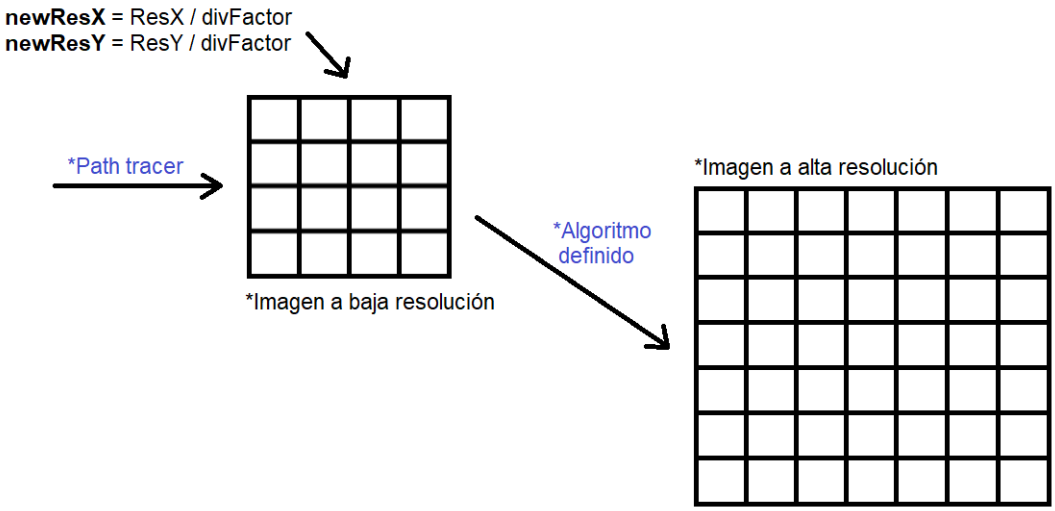


***Ilustración 8**. Representación visual de la aproximación utilizada para la problemática.*

La forma en que se opera con las imágenes es utilizando coordenadas normalizadas UV, para que, sin importar la diferencia de resolución entre estas, se puedan mapear de forma equivalente, utilizando su equivalencia a coordenadas XY para asignar el color a los píxeles

3.4 *Upsampling* basado en el vecino más cercano en el espacio imagen (2D)

Teoría detrás del algoritmo

Este algoritmo se basa en la teoría de que la cercanía de los píxeles en una imagen podría determinar una similitud del valor de color entre ellos, cómo, por ejemplo, en la ilustración 9, cuyos valores de color son dados por las coordenadas XY del píxel.

La Ecuación 1 explica la fórmula utilizada para la asignación de color RGB (*Red, Green, Blue*, en español, Rojo, Verde y Azul), utilizando como parámetros para R, la división entre la coordenada X del píxel a asignar el color y la resolución en tal dimensión de la imagen, para G se tiene el caso equivalente utilizando la coordenada Y, con la resolución en esa dimensión, dejando el valor B en 0.0, ya que se trabaja solamente las dimensiones XY al utilizar esta ecuación.

No es una imagen basada en una escena 3D, por ello es por lo que no se utiliza una coordenada Z, es meramente para representar la teoría propuesta.

$$colorRGB = (\frac{coordX}{resX}, \frac{coordY}{resY}, 0.0)$$

***Ecuación 1**. Fórmula de asignación de color usando las coordenadas XY del píxel.*

La Ilustración 9, busca explicar lo previamente teorizado, siguiendo, como se puede observar, un cambio de color paulatino, prácticamente contiguo entre los píxeles de la imagen.

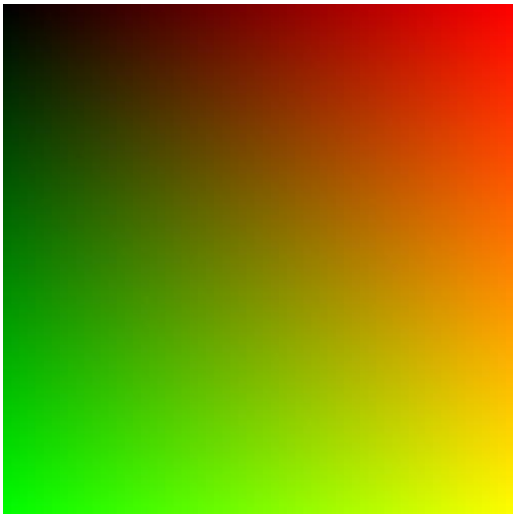


Ilustración 9. *Imagen cuyo valor asignado a los píxeles se basa en sus coordenadas XY.*

Algoritmo

Este algoritmo se basa en utilizar como color del píxel de la imagen en alta resolución, el color del píxel correspondiente de la imagen en baja resolución (es decir, el píxel vecino más cercano, al que corresponde, según la diferencia de resolución).

El Algoritmo 3 detalla brevemente el código utilizado dentro de este algoritmo.

Datos de entrada: *scene (Scene object)* **Datos de salida:** none
1. *por cada píxel en la imagen* 2. *se transforman las coordenadas de la imagen en alta resolución a las de la imagen en baja resolución, y se asigna el color del píxel correspondiente* 3. *fin-por*

Algoritmo 3. *Algoritmo de upsampling basado en el vecino más cercano en el espacio imagen (2D).*

El algoritmo recibe como datos de entrada una escena 3D, y generará una imagen fotorrealista con extensión bmp.

Para generar la imagen final, de alta resolución, el algoritmo irá píxel a píxel, realizando una conversión entre coordenadas UV y XY, se pasará del píxel actual (imagen en alta resolución) al píxel correspondiente en la imagen a baja resolución, para obtener su color y asignarlo al píxel actual.

Una vez se termina de ejecutar este algoritmo, se ha conseguido pasar de una imagen en baja resolución, a una imagen en alta resolución.

La Ilustración 10 describe de forma visual, la idea detrás de este algoritmo, dónde el píxel en la imagen de alta resolución obtiene su color en base al píxel de la imagen en baja resolución.

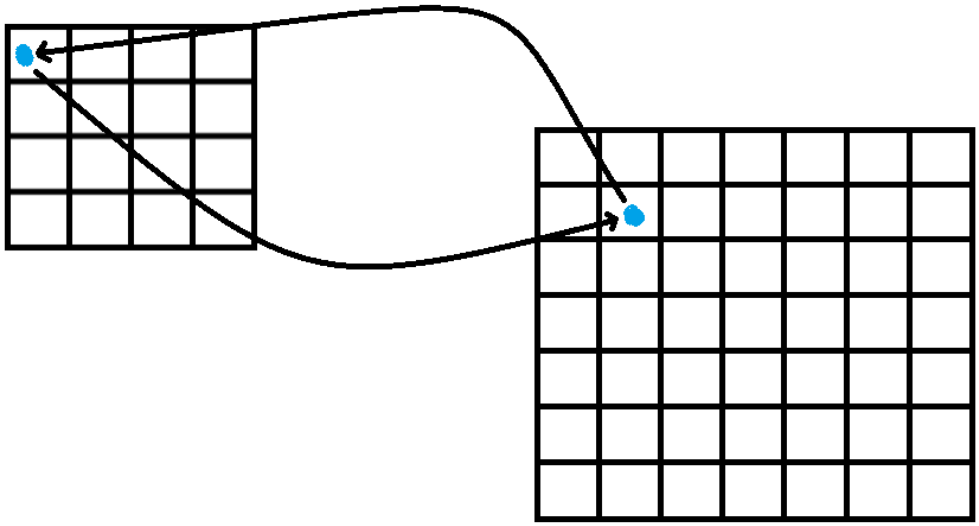


***Ilustración 10**. Representación visual del algoritmo upsampling basado en el vecino más cercano en el espacio imagen (2D).*

3.5 *Upsampling* basado en el promedio de los vecinos en el espacio imagen (2D)

Teoría detrás del algoritmo

Este algoritmo sigue la teoría presentada en el algoritmo anterior, *upsampling* basado en el vecino más cercano en el espacio imagen (2D), dónde, la cercanía entre píxeles puede determinar el color de estos.

Cuenta con la diferencia de tener el añadido de que, en lugar de utilizar únicamente el valor del color del píxel correspondiente en la imagen a baja resolución, se utilizará el color promedio de ese píxel y los píxeles vecinos de un filtro definido por parámetro (pudiendo ser una matriz de 3x3, 5x5, etc.) para obtener un valor más acertado.

Algoritmo

Para este algoritmo se utiliza la Ecuación 2, la cual explica la fórmula utilizada para la asignación de color RGB, el cuál será el promedio de los colores de los píxeles considerados dentro del filtro pasado por parámetro, pudiendo haber *n* píxeles contemplados (claramente esto delimitado por la resolución de la imagen).

$$promedioRGB = \frac{(colorRGB_1 + \cdots + colorRGB_n)}{n}$$

***Ecuación 2**. Fórmula de asignación del color promedio en base a los colores de los pixeles de la imagen en baja resolución en el filtro de pixeles vecinos definidas por el tamaño del filtro dado por parámetro.*

El Algoritmo 4 detalla brevemente el código utilizado dentro de este algoritmo.

Datos de entrada: *scene (Scene object)* **Datos de salida:** *none*
1. *por cada píxel en la imagen* 2. *por cada píxel vecino según el filtro dado* 3. *se transforman las coordenadas de la imagen en alta resolución a las de la imagen en baja resolución, y se suma el color del píxel actual* 4. *fin-por* 5. *se asigna como color del píxel actual, el color promedio de los píxeles vecinos visitados* 6. *fin-por*

***Algoritmo 4**. Algoritmo de upsampling basado en el promedio de los vecinos en el espacio imagen (2D).*

El algoritmo recibe como datos de entrada una escena 3D, y generará una imagen fotorrealista con extensión bmp.

Para generar la imagen final, de alta resolución, el algoritmo irá píxel a píxel, realizando una conversión entre coordenadas UV y XY, se pasará del píxel actual (imagen en alta resolución) al píxel correspondiente en la imagen a baja resolución, para, según un filtro dado (que puede ser una matriz de 3x3, 5x5, etc.) por parámetro obtener el color promedio de los píxeles vecinos visitados y asignarlo al píxel actual.

Una vez se termina de ejecutar este algoritmo, se ha conseguido pasar de una imagen en baja resolución, a una imagen en alta resolución.

La Ilustración 11 describe de forma visual, la idea detrás de este algoritmo, dónde el píxel en la imagen de alta resolución obtiene su color en base al promedio de color del píxel correspondiente de la imagen en baja resolución y los vecinos contemplados según el tamaño del filtro pasado por parámetro.

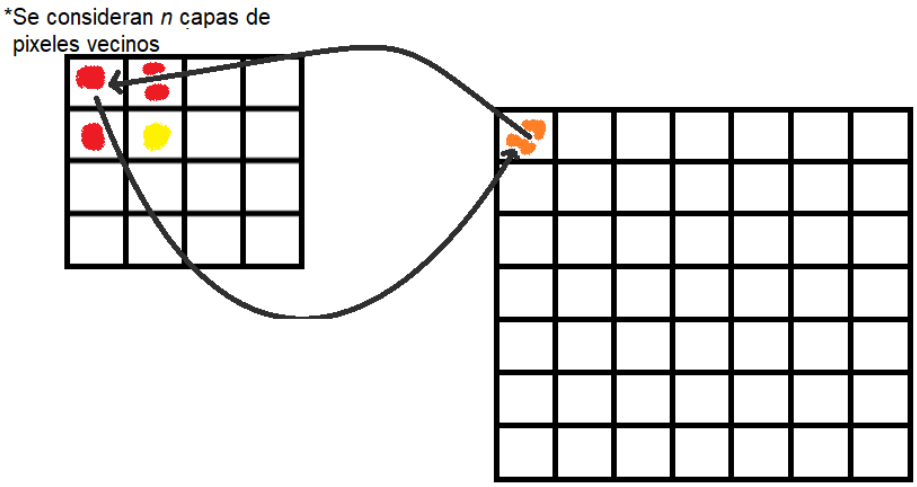


Ilustración 11. *Representación visual del algoritmo upsampling basado en el promedio de los vecinos en el espacio imagen (2D).*

3.6 *Upsampling* basado en el promedio ponderado de los vecinos en el espacio imagen (2D)

Teoría detrás del algoritmo

Este algoritmo sigue la teoría presentada en el algoritmo anterior, *upsampling* basado en el promedio de los vecinos en el espacio imagen (2D), dónde, la cercanía entre píxeles puede determinar el color de estos, con el añadido de que el promedio del color de *n* vecinos en el filtro contemplado podría dar un resultado más acertado.

Cuenta con la diferencia de tener el añadido de que, en lugar de utilizar simplemente el promedio del color del píxel correspondiente y sus vecinos en la imagen a baja resolución, se dará una ponderación al valor del color del píxel contemplado según el filtro dado (pudiendo ser una matriz de 3x3, 5x5, etc.), siendo el factor de la ponderación, la distancia entre el píxel vecino contemplado y el píxel correspondiente en la imagen a baja resolución, para así tener un valor más preciso.

Algoritmo

Se ha explicado que para la realización del promedio ponderado de los colores de los píxeles vecinos contemplados según el filtro dado por parámetro, se utilizará la distancia entre el píxel vecino contemplado y el correspondiente en la imagen a baja resolución.

Para calcular la distancia entre ambos, se utilizará el cálculo de la distancia euclidiana usando las coordenadas UV de los píxeles (*u* como *x*, *v* como *y*). En la Ecuación 3, se utiliza como $pixel_1$ el píxel correspondiente de la imagen en baja resolución, y como $pixel_2$, el píxel vecino considerado en ese momento.

$$distEuc2D(pixel_1, pixel_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

***Ecuación 3**. Fórmula para realizar el cálculo de la distancia euclidiana en 2D.*

La Ecuación 4, explica la fórmula utilizada para la asignación de color RGB, el cuál será el promedio ponderado de los colores de los píxeles considerados dentro del filtro pasado por parámetro, pudiendo haber n píxeles contemplados (claramente esto delimitado por la resolución de la imagen), siendo el factor de ponderación, la distancia euclidiana entre las coordenadas UV del píxel correspondiente y el píxel considerado en ese momento.

$$promPondRGB = \left(colorRGB_1 \times \frac{distPixel_1}{sumDists}\right) + \cdots + \left(colorRGB_n \times \frac{distPixel_n}{sumDists}\right)$$

***Ecuación 4**. Fórmula de asignación del color promedio ponderado en base a los colores de los pixeles de la imagen en baja resolución en el filtro de pixeles vecinos definido dado por parámetro, ponderado según la distancia euclidiana entre los píxeles.*

El Algoritmo 5 detalla brevemente el código utilizado dentro de este algoritmo.

Datos de entrada: *scene (Scene object)*
Datos de salida: *none*

1. *por cada píxel en la imagen*
2. *por cada píxel vecino según el filtro dado*
3. *se transforman las coordenadas de la imagen en alta resolución a las de la imagen en baja resolución, y se suma el color del píxel actual*
4. *se calcula la distancia euclidiana 2D entre el píxel correspondiente y el píxel vecino actual*
5. *fin-por*
6. *se asigna como color del píxel actual, el color promedio mesurado de los píxeles vecinos visitados usando como mesura, la distancia euclidiana*
7. *fin-por*

***Algoritmo 5**. Algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D).*

El algoritmo recibe como datos de entrada una escena 3D, y generará una imagen fotorrealista con extensión bmp.

Para generar la imagen final, de alta resolución, el algoritmo irá píxel a píxel, realizando una conversión entre coordenadas UV y XY, se pasará del píxel actual (imagen en alta resolución) al píxel correspondiente en la imagen a baja resolución, para, según un filtro dado (que puede ser una matriz de 3x3, 5x5, etc.) por parámetro obtener el color promedio ponderado de los píxeles vecinos visitados y asignarlo al píxel actual, utilizando como factor de ponderación la distancia euclidiana entre las coordenadas UV del píxel correspondiente y el píxel considerado en ese momento .

Una vez se termina de ejecutar este algoritmo, se ha conseguido pasar de una imagen en baja resolución, a una imagen en alta resolución.

La Ilustración 12 describe de forma visual, la idea detrás de este algoritmo, dónde el píxel en la imagen de alta resolución obtiene su color en base al promedio mesurado de color del píxel correspondiente de la imagen en baja resolución y los vecinos contemplados según el tamaño del filtro pasado por parámetro, con factor de mesura la distancia euclidiana entre los píxeles.

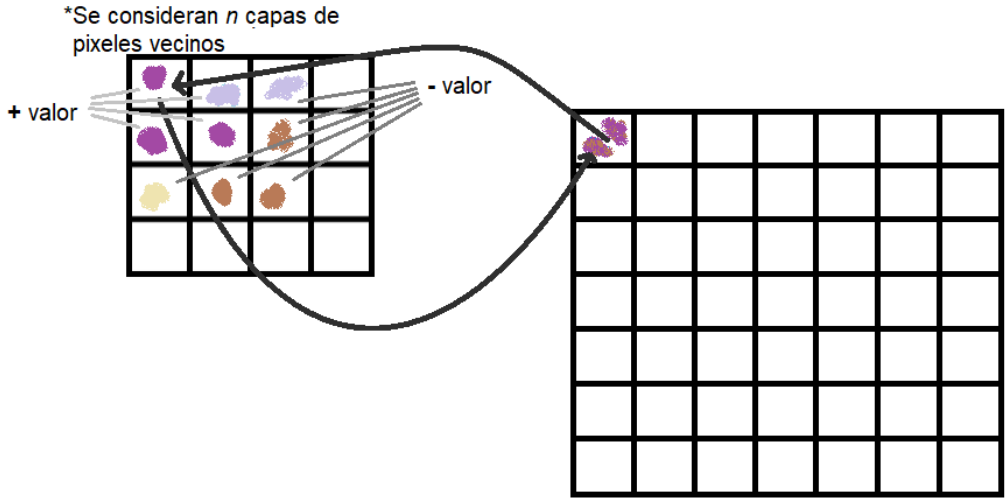


***Ilustración 12**. Representación visual del algortimo upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D).*

3.7 *Upsampling* basado en el promedio ponderado de los vecinos en el espacio escena (3D)

Teoría detrás del algoritmo

Este algoritmo resulta ser similar al algoritmo anterior, *upsampling* basado en el promedio ponderado de los vecinos en el espacio imagen (2D), ya que también utiliza un promedio ponderado, pero para este algoritmo se utilizará la escena 3D.

Este algoritmo supone una mejora considerable con respecto al algoritmo previo, ya que considerar el eje Z de la imagen, ya que el tener información geométrica de la escena permitirá obtener un resultado más acertado.

La idea detrás de esta teoría se define como que, la continuidad de una imagen en 2D, no implica una continuidad en la escena 3D.

Para ejemplificar esto, se utilizará una escena sencilla, visualizada en la Ilustración 13, definida por una caja de Cornell, de 3 paredes azules y con un cubo verde en el centro del suelo (aunque no se observa el techo, se asume en esta escena cierta iluminación para poder ser observada).

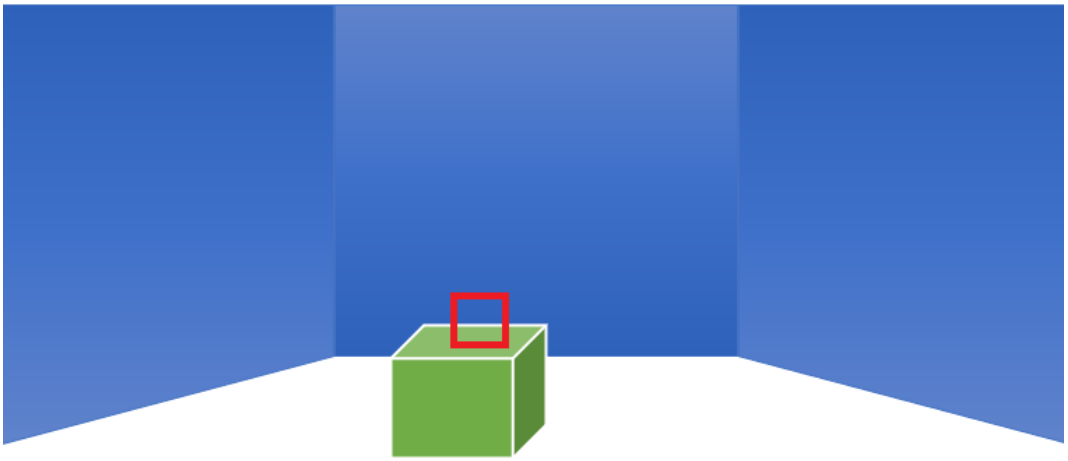


Ilustración 13. *Escena 3D de una caja abierta con un cubo en centro del piso, dónde se puede apreciar que la continuidad de la imagen en 2D no implica una continuidad en la escena 3D.*

En la Ilustración 13, centrándonos en la zona delimitada por el cuadro rojo, si únicamente se utilizará la distancia 2D entre los píxeles para un promedio ponderado, el valor dado a los píxeles de color azul de la pared podría afectar del mismo modo en esa zona, que los píxeles de color verde, pero si se considera la geometría 3D de la escena al momento de generar la imagen, se puede realizar un cálculo más preciso sobre la imagen final.

Utilizando como información geométrica las intersecciones más cercanas de los rayos de luz (siendo estos almacenados en memoria al momento de generar la imagen a baja resolución), se puede usar la distancia euclidiana 3D sobre la intersección del rayo de luz en la escena del pixel correspondiente en la imagen a baja resolución con la intersección del rayo de luz en la escena del píxel vecino considerado en ese momento; de este modo, al momento de dar color al cubo en la escena, se dará un peso menor al color de los píxeles azules, pues la pared se encuentra más lejos, que los píxeles verdes, que son los que componen el cubo.

Algoritmo

Para realizar este algoritmo, se modificará el paso previo a la aproximación de *upsampling*, modificando el código del algoritmo *preprocess*, para que guarde en memoria las intersecciones más cercanas de los rayos de luz (información geométrica) muestreados para generar la imagen en baja resolución.

La modificación mencionada a *preprocess* es descrita en el Algoritmo 6.

Datos de entrada: *scene (Scene object)* **Datos de salida:** *none*
1. *se reduce la resolución en el largo y ancho según el factor de reducción para generar la imagen de baja resolución* 2. *por cada píxel en la imagen de baja resolución* 3. *por cada muestra* 4. *se genera un rayo para calcular el color de la muestra actual* 5. *se suma el color de la muestra actual para mejorar la definición del color final* 6. *si la muestra es la primera* 7. *guardar en memoria la intersección más cercana del rayo de luz generado con la escena 3D* 8. *fin-si* 9. *fin-por* 10. *se asigna el color promedio de todas las muestras al píxel actual* 11. *fin-por*

***Algoritmo 6**. Algoritmo de preprocess para generar imágenes fotorrealistas de baja resolución a la vez que guarda información geométrica mientras se genera la imagen.*

Se ha explicado que para la realización del promedio ponderado de los colores de los píxeles vecinos contemplados según el filtro dado por parámetro, se utilizará la distancia euclidiana en 3D entre las intersecciones del píxel vecino contemplado y el correspondiente en la imagen a baja resolución.

Para calcular la distancia entre ambas, se utilizará el cálculo de la distancia euclidiana 3D usando las coordenadas XYZ de las intersecciones de los píxeles. En la Ecuación 5, se utiliza como $pixel_1$ el píxel correspondiente de la imagen en baja resolución, y como $pixel_2$, el píxel vecino considerado en ese momento.

$$distEuc3D(pixel_1, pixel_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)}$$

***Ecuación 5**. Fórmula para realizar el cálculo de la distancia euclidiana en 3D.*

La fórmula de asignación del color RGB es descrita en la Ecuación 4, explicada en la Sección 3.6, claramente usando las distancias calculadas por la distancia euclidiana en 3D.

El Algoritmo 7 detalla brevemente el código utilizado dentro de este algoritmo.

Datos de entrada: *scene (Scene object)* **Datos de salida:** *none*
1. *por cada píxel en la imagen* 2. *por cada píxel vecino según el filtro dado* 3. *se transforman las coordenadas de la imagen en alta resolución a las de la imagen en baja resolución, y se suma el color del píxel actual* 4. *se calcula la distancia euclidiana 3D entre las intersecciones del píxel correspondiente y el píxel vecino actual* 5. *fin-por* 6. *se asigna como color del píxel actual, el color promedio mesurado de los píxeles vecinos visitados usando como mesura, la distancia euclidiana* 7. *fin-por*

***Algoritmo 7**. Algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D).*

El algoritmo recibe como datos de entrada una escena 3D, y generará una imagen fotorrealista con extensión bmp.

Para generar la imagen final, de alta resolución, el algoritmo irá píxel a píxel, realizando una conversión entre coordenadas UV y XY, se pasará del píxel actual (imagen en alta resolución) al píxel correspondiente en la imagen a baja resolución, para, según un filtro dado (que puede ser una matriz de 3x3, 5x5, etc.) por parámetro obtener el color promedio ponderado de los píxeles vecinos visitados y asignarlo al píxel actual, utilizando como factor de ponderación la distancia euclidiana entre las intersecciones de los rayos de luz con la escena del píxel correspondiente y el píxel considerado en ese momento.

Una vez se termina de ejecutar este algoritmo, se ha conseguido pasar de una imagen en baja resolución, a una imagen en alta resolución.

La Ilustración 14 describe de forma visual, la idea detrás de este algoritmo, dónde el píxel en la imagen de alta resolución obtiene su color en base al promedio mesurado de color del píxel correspondiente de la imagen en baja resolución y los vecinos contemplados según el tamaño del filtro pasado por parámetro, con factor de mesura la distancia euclidiana entre las intersecciones de los píxeles.

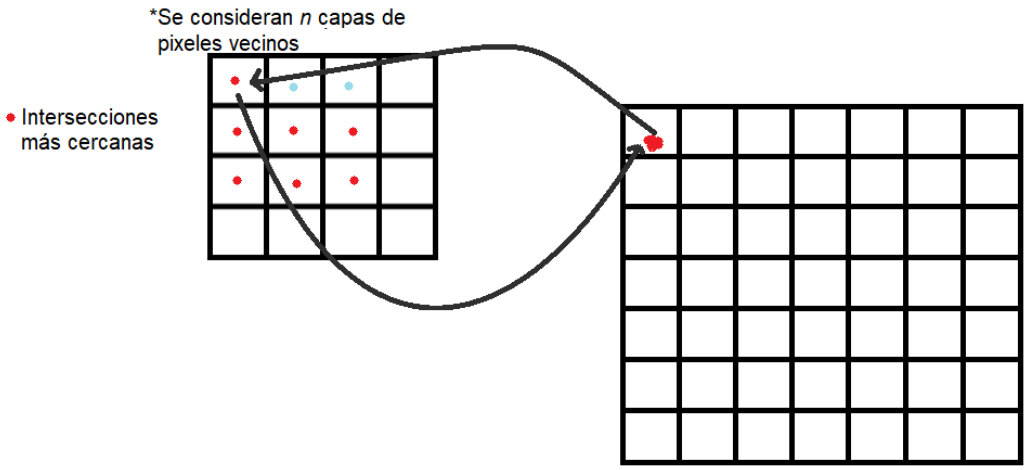


Ilustración 14. *Representación visual del algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D).*

3.8 *Upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D)

Teoría detrás del algoritmo

Este algoritmo sigue la teoría presentada en el algoritmo anterior, *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena (3D), dónde, las intersecciones de los rayos de luz con la escena 3D influyen en la ponderación de los colores de los píxeles considerados para la asignación del color final.

Cuenta con la diferencia de tener el añadido de que, en lugar de utilizar únicamente las intersecciones para realizar la ponderación entre los colores de los píxeles considerados por el cálculo, también se utiliza la normal de luz presente en la intersección.

Este algoritmo supone una mejora con respecto al algoritmo previo, ya que considerar la normal de la luz en la intersección, es posible delimitar aún más los objetos presentes en la escena, lo que permitirá obtener un resultado más acertado.

La idea detrás de esta teoría se define como que, aunque dos objetos se encuentren cerca en la escena 3D, no implica que estos sean similares entre ellos.

Para ejemplificar esto, se utilizará una escena sencilla, visualizada en la Ilustración 15, definida por una caja de Cornell, de 3 paredes (izquierda salmón, central azul y derecha gris) y con un cubo verde en el centro del suelo (aunque no se observa el techo, se asume en esta escena cierta iluminación para poder ser observada).

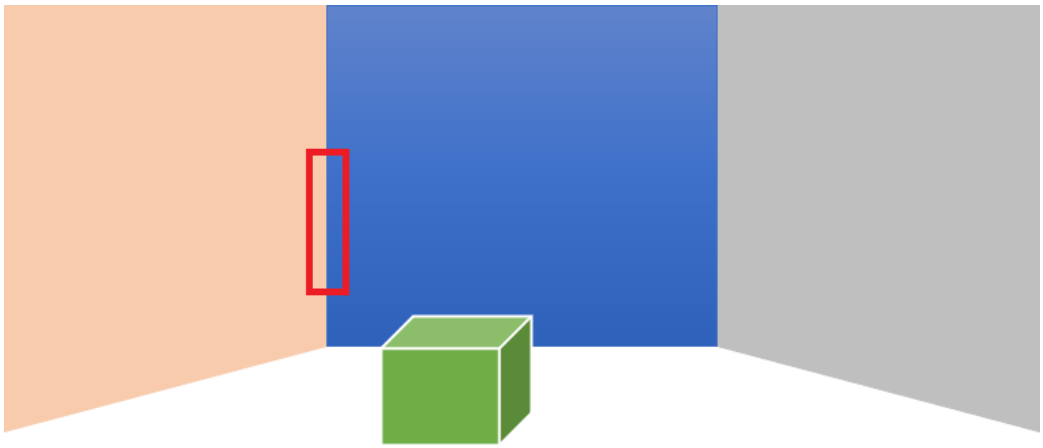


***Ilustración 15**. Escena 3D de una caja abierta con un cubo en centro del piso, dónde se puede apreciar que la proximidad entre dos objetos en la escena 3D no implica una similitud entre ellos.*

En la Ilustración 15, centrándonos en la zona delimitada por el rectángulo rojo, si únicamente se utilizará la distancia entre las intersecciones de los rayos de luz con la escena 3D entre los píxeles para un promedio ponderado, posiblemente el peso dado al color de los píxeles de color azul de la pared, podría afectar del mismo modo en esa zona que los píxeles de color salmón de la otra pared, pero si se considera también la normal de luz en la intersección como un factor delimitante para considerar las distancias podríamos realizar un cálculo más preciso sobre la imagen final.

Se puede utilizar las normales de las intersecciones, de modo que si, el producto escalar de estas es menor a un factor dado, entonces despreciar la distancia calculada.

Esto es considerado así ya que un cambio brusco en la dirección de las normales, como se puede apreciar en la ilustración 16, suele indicar cambios en las superficies, (pudiendo darse a entender también cómo objetos diferentes si es el caso).

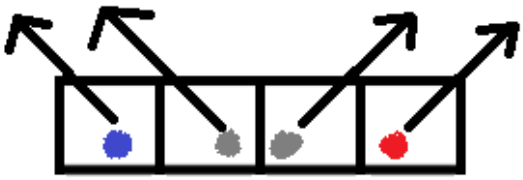


***Ilustración 16.** Ejemplo visual de como grandes cambios en las normales suelen indicar un cambio en la escena 3D, ya sea superficie u objeto.*

Algoritmo

Se ha explicado que para la realización del promedio ponderado de los colores de los píxeles vecinos contemplados según el filtro dado por parámetro, se utilizará la distancia euclidiana en 3D entre las intersecciones del píxel vecino

contemplado y el correspondiente en la imagen a baja resolución como factor de mesura, considerando las normales de las intersecciones de los pixeles como delimitadores para definir si se utiliza o desprecia tal distancia.

Para calcular la distancia entre las intersecciones se utilizará la fórmula descrita en la Ecuación 5, explicada en la Sección 3.7.

La Ecuación 6 describe la fórmula utilizada para definir si la distancia 3D entre las intersecciones del píxel correspondiente ($pixel_1$) y el píxel vecino ($pixel_2$) considerado en ese momento, de la imagen a baja resolución, será mantenida o despreciada si el producto escalar de las normales no pasa un factor dado por parámetro.

$$distFinal = distEuclidian3D(pixel_1, pixel_2) \quad if\ prodEsc((intersection_1, intersection_2) > factorNormal \quad else\ 0.0$$

***Ecuación 6**. Fórmula para definir si la distancia 3D entre las intersecciones de los pixeles será mantenida o despreciada dado a si el producto escalar de las normales pasa un factor definido.*

La fórmula de asignación del color RGB es descrita en la ecuación 4, explicada en la Sección 3.6, claramente usando las distancias ajustadas por la Ecuación 6.

El Algoritmo 8 detalla brevemente el código utilizado dentro del algoritmo.

Datos de entrada: *scene (Scene object)*
Datos de salida: *none*

```
1. por cada píxel en la imagen
2.     por cada píxel vecino según el filtro dado
3.         se transforman las coordenadas de la imagen en alta resolución a
           las de la imagen en baja resolución, y se suma el color del píxel
           actual
4.         se calcula la distancia euclidiana 3D entre las intersecciones del píxel
           correspondiente y el píxel vecino actual
5.         si la distancia entre las intersecciones pasa el factor sobre la normal
6.             mantener la distancia
7.         sino
8.             despreciar la distancia
9.         fin-si
10.    fin-por
11.    se asigna como color del píxel actual, el color promedio mesurado de
       los píxeles vecinos visitados usando como mesura, la distancia
       euclidiana
12. fin-por
```

***Algoritmo 8.** Algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).*

El algoritmo recibe como datos de entrada una escena 3D, y generará una imagen fotorrealista con extensión bmp.

Para generar la imagen final, de alta resolución, el algoritmo irá píxel a píxel, realizando una conversión entre coordenadas UV y XY, se pasará del píxel actual (imagen en alta resolución) al píxel correspondiente en la imagen a baja resolución, para, según un filtro dado (que puede ser una matriz de 3x3, 5x5, etc.) por parámetro obtener el color promedio ponderado de los píxeles vecinos visitados y asignarlo al píxel actual, utilizando como factor de ponderación la distancia euclidiana entre las intersecciones de los rayos de luz con la escena del píxel correspondiente y el píxel considerado en ese momento, si es que el producto escalar de las normales de las intersecciones pasa un factor dado por parámetro.

Una vez se termina de ejecutar este algoritmo, se ha conseguido pasar de una imagen en baja resolución, a una imagen en alta resolución.

La Ilustración 17 describe de forma visual, la idea detrás de este algoritmo, dónde el píxel en la imagen de alta resolución obtiene su color en base al promedio mesurado de color del píxel correspondiente de la imagen en baja resolución y los vecinos contemplados según el tamaño del filtro pasado por parámetro, con factor de mesura la distancia euclidiana entre las intersecciones de los píxeles si el producto escalar de las normales de estas pasa un factor.

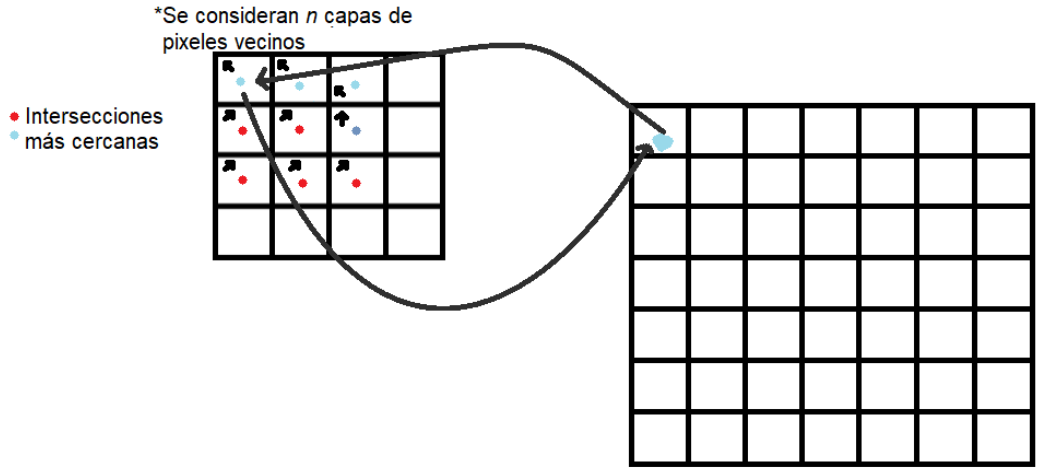


Ilustración 17. *Representación visual del algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).*

4 Implementación

4.1 Descripción del software utilizado

Diagrama de clases del proyecto

En la Ilustración 18 se observa el diagrama con las clases más relevantes dentro del proyecto, dando especial atención a aquellas que se han visto más afectadas a lo largo del mismo.

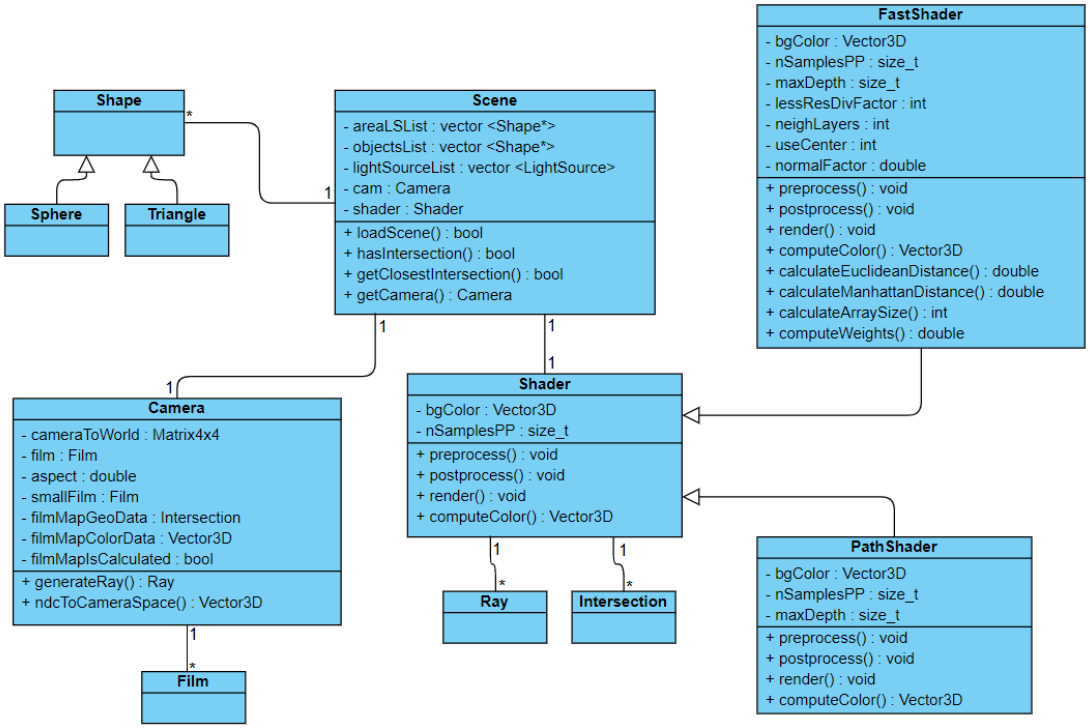


Ilustración 18. *Breve diagrama de clases del proyecto.*

Descripción de las clases

Una breve descripción de las clases relevantes durante el desarrollo de este proyecto:

Main: es la clase principal del proyecto, la que define el flujo de este, y se utiliza para las métricas de comparativa de algoritmos.

Scene: clase encarga de la lectura de la escena, elección del *shader* a utilizar, obtención de las intersecciones y acceso a la cámara que observa la escena.

Camera: clase que visualiza la escena 3D, se encarga de la generación de los rayos utilizados por los *shader*, contiene diferentes atributos para guardar información relevante durante los algoritmos utilizados por el *FastShader*.

Sphere: clase que hereda de *Shape*, permite generar objetos de tipo esfera en la escena 3D.

Triangle: clase que hereda de *Shape*, permite generar objetos de tipo triángulo en la escena 3D.

Shader: *shader* base, contiene la lógica básica para el renderizado de la imagen final.

PathShader: *shader* que utiliza el algoritmo de *path tracing* para la generación de una imagen final, es el algoritmo que tenemos de base para la generación de imágenes de alta calidad.

FastShader: *shader* desarrollado a lo largo de todo el proyecto, implementado diferentes algoritmos para comparar resultados contra el *PathShader*, es la clase central del trabajo de fin de grado.

Ray: clase que se encarga de definir los rayos utilizados por los algoritmos de renderizado para el procesamiento de la escena 3D.

Intersection: clase que se encarga de almacenar la información geométrica de los puntos de intersección encontrados en la escena, como lo es su posición 3D, y su normal.

Diagrama de flujo

La Ilustración 19 muestra el diagrama de flujo con la secuencia realizada por el proyecto contemplando la clase *FastShader*.

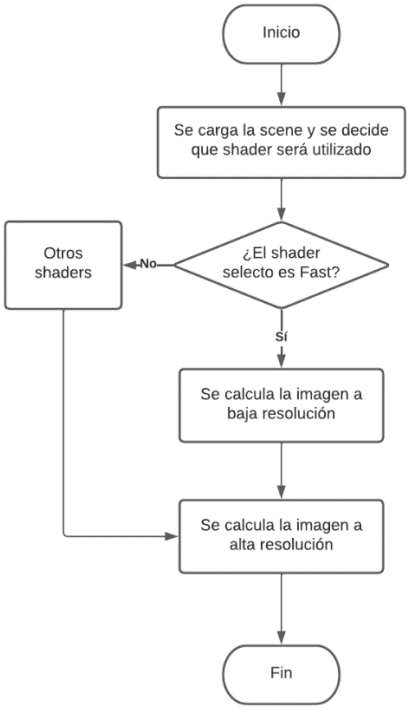


***Ilustración 19**. Diagrama de flujo del proyecto.*

4.2 Ampliaciones al software utilizado

A lo largo del proyecto, se realizaron múltiples modificaciones al código inicial provisto por el asesor, se enlistan aquellas modificaciones realizadas por clase.

*Main***:** Los cambios dentro de la clase *main* son mínimos, se añadieron las librerías

- *FastShader***:** para poder hacer uso de la clase creada para la aproximación a la optimización de la problemática con los diversos algoritmos de *upsampling*.
- *chrono***:** para poder medir el tiempo de ejecución y realizar una comparativa entre algoritmos.
- *fstream***:** para poder escribir en un fichero automáticamente los resultados de los tiempos de ejecución según los parámetros dados

Se midió el tiempo de ejecución de los algoritmos según sus parámetros, guardando la información en un fichero definido.

Finalmente, es esta clase quién libera la memoria asignada a las matrices de la clase camera.

Camera: Se añadieron como atributos

- *smallFilm***:** una referencia a un objeto *film* que será el que se guarde como imagen a baja resolución.
- *filmMapGeoData***:** una matriz bidimensional de punteros que guarda las intersecciones obtenidas al generar la imagen a baja resolución.
- *filmMapColorData:* una matriz bidimensional de punteros que guarda el valor de los colores obtenidos al generar la imagen a baja resolución.
- *filmMapIsCalculated:* una matriz bidimensional que guarda si el píxel en esa posición de la matriz (línea, columna equivalente a Y, X), ha sido procesado o no, y por ende la misma posición en *filmMapColorData* tendría un valor utilizable.
- *filmMapWasExplicitCalculated:* una matriz bidimensional que guarda si el píxel en esa posición de la matríz (línea, columna, equivalente a Y, X), ha sido calculado explícitamente o no.

Scene: se añadió la opción para que según el fichero de escena dado se pudiera utilizar la clase *FastShader*, leer los parámetros dados para la misma, y generar su instancia.

Al igual que es la encargada de la instanciación de las matrices definidas en la clase camera.

*Fastshader***:** la clase *FastShader* fue la utilizada a lo largo del proyecto para codificar y probar los diferentes algoritmos como aproximaciones a la optimización de la problemática.

Se basa en la clase *PathShader*, teniendo los siguientes cambios.

Atributos:

- *lessResDivFactor*: factor por el cuál la resolución dada para la imagen final será dividida en largo y ancho, para generar una imagen con esta calculada nueva baja resolución.
- *neighLayers*: factor que definirá el tamaño del filtro de vecinos serán contempladas al momento de realizar cálculos que los requieran.
- *useCenter*: factor que define si el pixel correspondiente de la imagen en baja resolución será considerado para los cálculos, o solo los vecinos de este.
- *normalFactor*: factor utilizado para definir que, si el cálculo del producto escalar de dos normales es mayor, entonces la distancia euclidiana 3D entre dos intersecciones se tendrá en cuenta, sino será despreciada.

Métodos:

- *preprocess(scene)*
  Utilizando el algoritmo del *PathTracer* se genera una imagen, pero a baja resolución según el atributo lessResDivFactor y en algunos casos se guarda información extra en memoria para futuros cálculos.
- *render(scene)*
  Dependiendo del código contenido en este método (que cambia según que algoritmo se esté utilizando para la aproximación a la optimización de la problemática) se generará una imagen en alta resolución basada en la imagen en baja resolución con el algoritmo definido en el código.
- *calculateEuclideanDistance(firstPoint, secondPoint)*
  Calcula la distancia euclidiana entre dos puntos, ya sea 2D o 3D.
- *calculateManhattanDistance(firstPoint, secondPoint)*
  Calcula distancia Manhattan entre dos puntos, en 2D.
- *calculateArraySize(xPixelCoord, yPixelCoord, resX, resY)*
  Algunos algoritmos requieren almacenar valores en arreglos para posteriormente procesarlos, este método calcula el tamaño exacto que requerirá el mismo para sólo hacer uso de la memoria exacta requerida.
- *computeWeights(firstNormal, secondNormal, distance)*
  Decide si la distancia dada como parámetro será retornada, si el producto escalar entre las dos normales dadas es mayor al atributo normalFactor, en caso contrario, se retornará el valor 0.0

5 Resultados

5.1 Descripción del entorno de ejecución

Hardware utilizado

Para el desarrollo de este proyecto se utilizó un ordenador portátil con las siguientes características:

Procesador: Intel (R) Core (TM) i7-7700HQ CPU 2.80GHz

RAM: 16.0 GB (15.9 GB utilizable)

Sistema operativo: Windows 10 Home de 64 bits.

Para generar los resultados que se muestran a continuación, buscando el mejor rendimiento posible, el ordenador ejecutó únicamente el programa en primer plano en *release mode*, durante todo el tiempo de ejecución, en modo avión.

Escenas utilizadas

Durante el desarrollo del proyecto se utilizaron dos escenas diferentes.

1. **Caja de Cornell.**

   Consiste en una caja abierta, conformada por los siguientes elementos:

   - Una fuente de luz en el centro de un techo blanco.
   - Una pared izquierda roja.
   - Una pared derecha verde.
   - Una pared trasera blanca.
   - Un piso blanco.

   Todos los elementos previamente mencionados, se han generado a partir del uso de dos triángulos.

2. **Caja de Cornell con esfera.**

   Consta de la misma definición geométrica que la Caja de Cornell, añadiendo en el centro inferior de la escena una esfera de material gris difuso.

5.2 Resultados del algoritmo base de *path tracing*

Se tiene como referencia para el presente proyecto, la imagen de alta calidad observada en la Ilustración 20 con el algoritmo de *path tracing* propio del proyecto.

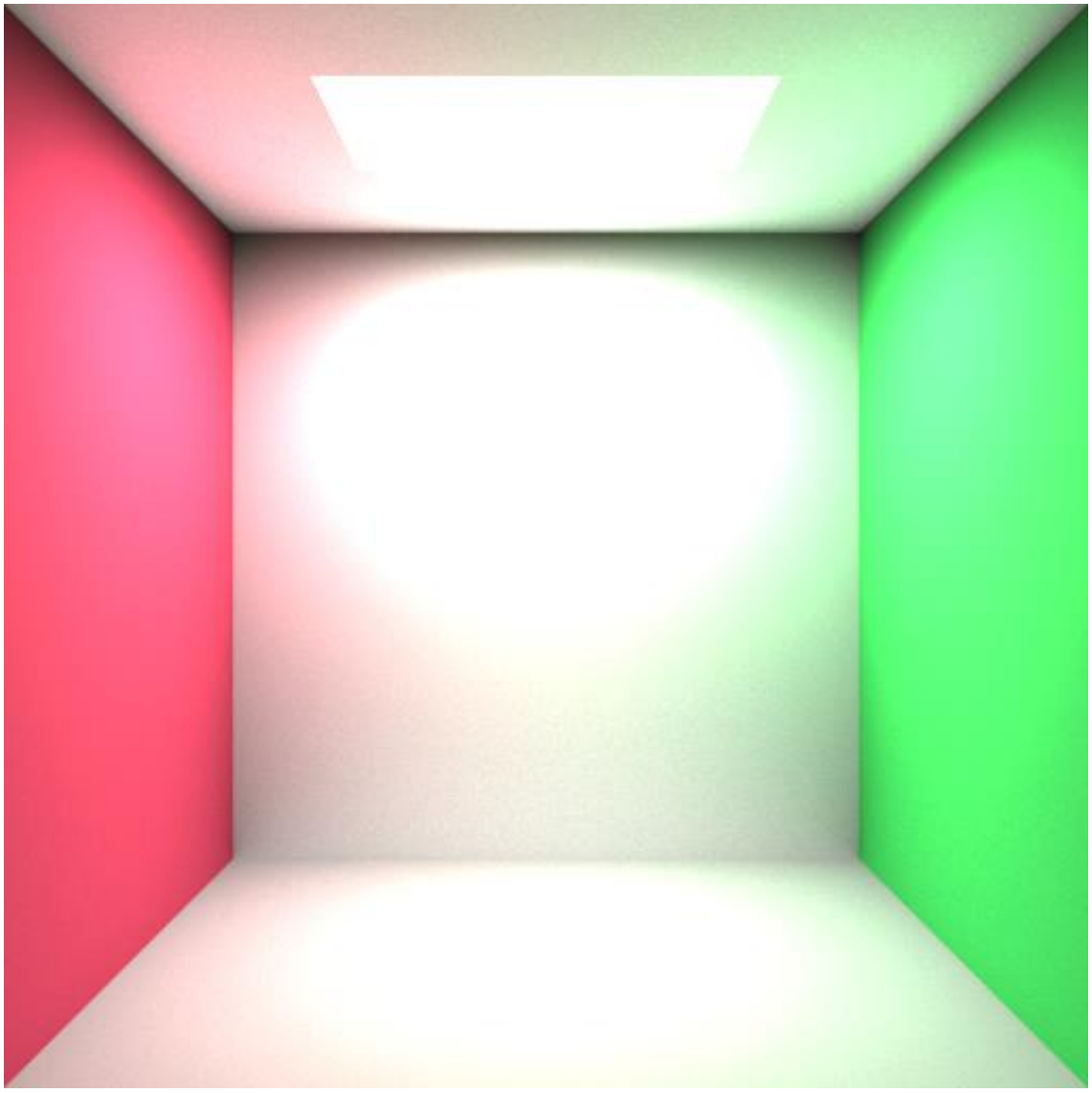


Ilustración 20. *Imagen generada por el path tracing utilizando 400,000 muestras por píxel.*

Para generar la imagen de la ilustración 20 se utilizaron los siguientes parámetros el algoritmo:

nSPP: 400,000 (número de muestras utilizadas por cada píxel calculado).

***maxDepth*:** 2 (número máximo de rebote dado a los rayos generados).

Siendo generada en un tiempo de ejecución de: **78,47 horas.**

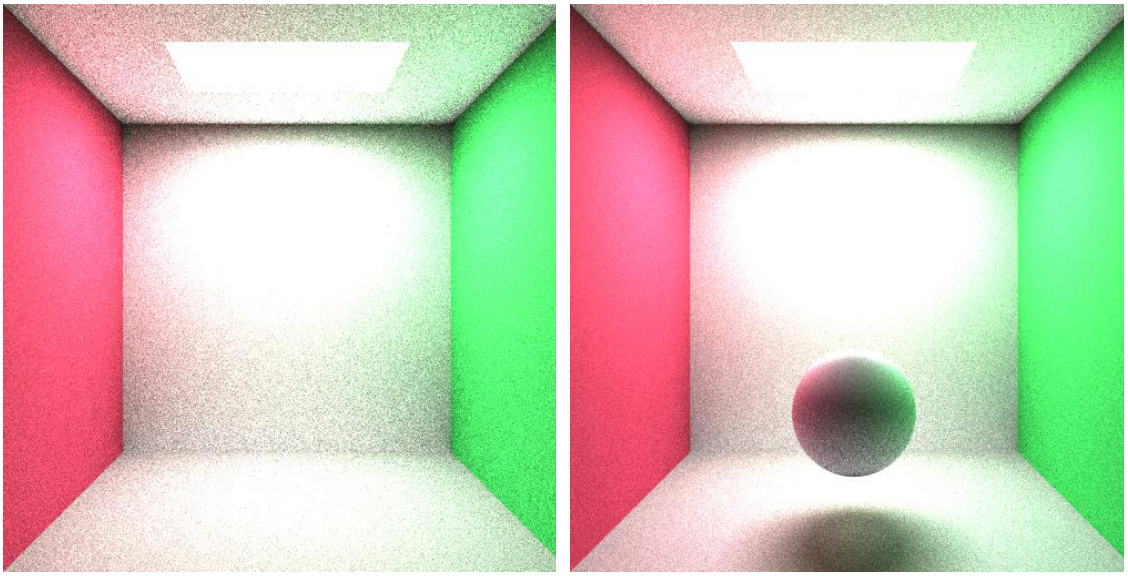
Se ha explicado a detalle dentro de la Sección 2, el *path tracing* y por qué este a mayor número de muestras aumenta su tiempo de ejecución.

Dado esto, y el tiempo que consumiría la generación de imágenes fotorrealistas con un gran número de muestras, para el resto de los resultados generados, sacrificando calidad por tiempo, se utilizaron los siguientes parámetros:

***nSPP*:** 10,000 (número de muestras utilizadas por cada píxel calculado).

***maxDepth*:** 2 (número máximo de rebote dado a los rayos generados).

La Ilustración 21, muestra los resultados obtenidos de las escenas utilizadas con los parámetros previamente mencionados.



a) Escena de caja de Cornell sencilla utilizada a lo largo del proyecto.

b) Escena de caja de Cornell con una esfera utilizada a lo largo del proyecto.

Ilustración 21. *Escenas utilizadas a lo largo del proyecto.*

El tiempo de ejecución obtenido para ambas imágenes fue de:

- Ilustración 21 (a): **48,53 minutos**.
- Ilustración 21 (b): **51,44 minutos.**

5.3 Resultados de las técnicas de *upsampling* para la iluminación global propuestas

A continuación, se describen los resultados de los diversos algoritmos desarrollados durante el proyecto comparándolos con los resultados obtenidos por el *path tracing*.

La información detallada de cómo se abordó la problemática de reducir el tiempo de ejecución para la generación de una imagen fotorrealista y los algoritmos aquí comentados se puede encontrar en la Sección 3.

Se utilizaron los siguientes parámetros:

***nSPP*:** 10,000 (número de muestras utilizadas por cada píxel calculado).

***maxDepth*:** 2 (número máximo de rebote dado a los rayos generados).

***lessResDivFactor*:** 2 o 3 (factor de reducción de la resolución para generar la imagen con el *path tracing)*

***neighLayers*:** 1, 2 o 3 (parámetro que define el tamaño del filtro utilizado en los algoritmos que utilicen promedios, siendo el valor el número de capas alrededor al píxel correspondiente, siendo 1, un filtro de 3x3).

***useCenter*:** 1 (parámetro que define si se considerará también el píxel correspondiente dentro del filtro previamente definido, 1 es que será utilizado).

***normalFactor*:** 0.95 (parámetro que define el valor el cuál el producto escalar de dos normales debe superar, para que la distancia calculada no sea despreciada).

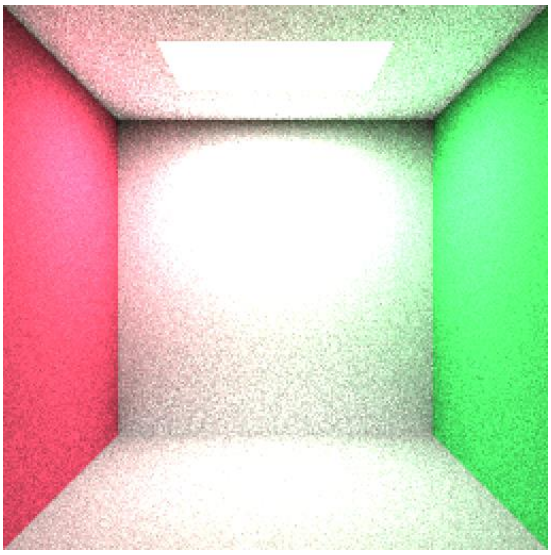
5.3.1 Upsampling basado en el vecino más cercano en el espacio imagen (2D)

Este algoritmo se basa en la asignación del color del píxel de la imagen de alta resolución según el color del píxel correspondiente en la imagen en baja resolución.

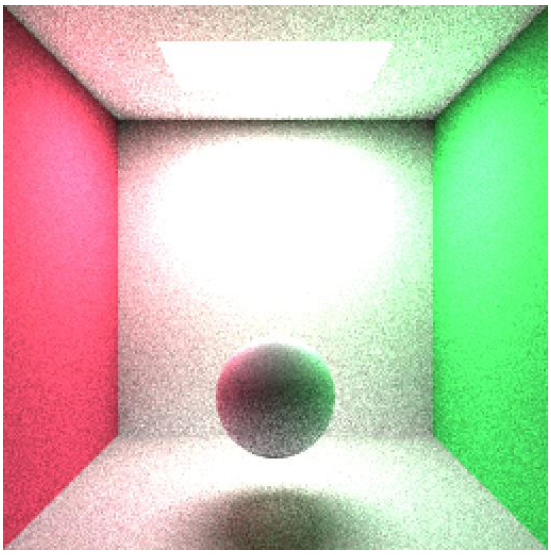
La Ilustración 22, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

Las ilustraciones a) y b) utilizaron factor de reducción de 2.

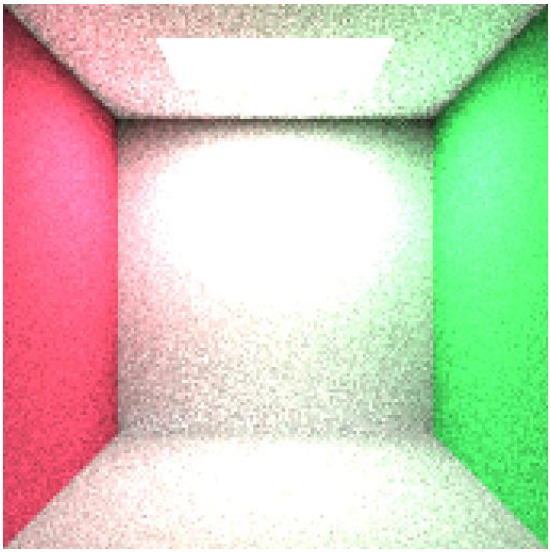
Las ilustraciones c) y d) utilizaron factor de reducción de 3.



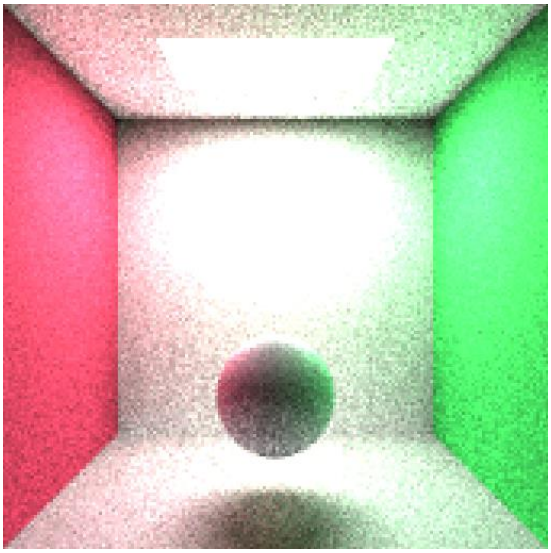
a) Imagen de caja de Cornell generada con factor de reducción 2



b) Imagen de caja de Cornell con esfera generada con factor de reducción 2



c) Imagen de caja de Cornell generada con factor de reducción 3



d) Imagen de caja de Cornell con esfera generada con factor de reducción 3

Ilustración 22. *Imágenes generadas por el algoritmo upsampling basado en el vecino más cercano en el espacio imagen (2D).*

En este método podemos observar en la Ilustración 22 una cantidad de ruido, que aumenta según el factor de división, lo cual es lógico a que ocurra, siendo un método de asignación directa según el píxel correspondiente, a menor sea la cantidad de pixeles de los cuáles se pueda recrear la imagen a alta resolución.

Igualmente se puede observar que, con un factor de reducción de 2, la cantidad de ruido visual presente sobre las imágenes es ligeramente superior a las de las imágenes de referencia de la Sección 5.2 generadas con los mismos parámetros.

A mayor cantidad de píxeles disponibles en la imagen de baja resolución, mejor será la imagen resultante de este algoritmo.

Los tiempos de ejecución obtenidos por las imágenes observadas se presentan en el Gráfico 1:

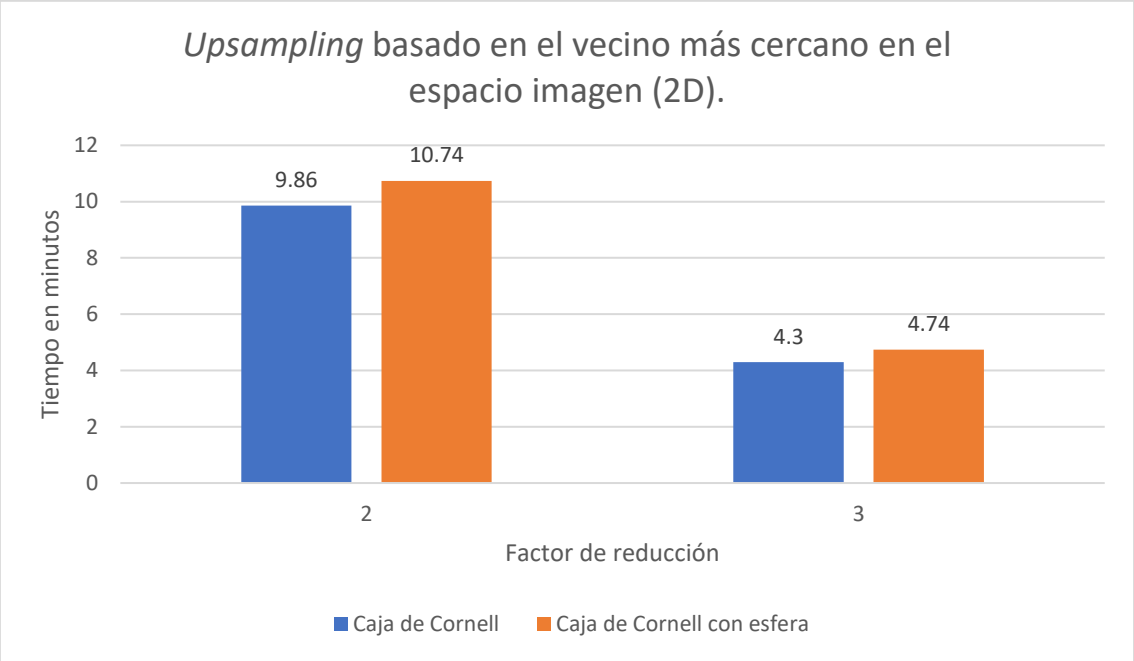


***Gráfico 1**. Tiempos de ejecución obtenidos por el algoritmo de upsampling basado en el vecino más cercano en el espacio imagen (2D).*

Utilizando este algoritmo, como se puede observar en el Gráfico 1, considerando el mayor coste de ejecución obtenido, en comparativa al mayor coste de ejecución obtenido por el *path tracing*, se tiene una mejora dónde el algoritmo se ejecuta en el **20,87%** del tiempo que tomaría hacerlo con *path tracing*.

5.3.2 Upsampling basado en el promedio de los vecinos en el espacio imagen (2D)

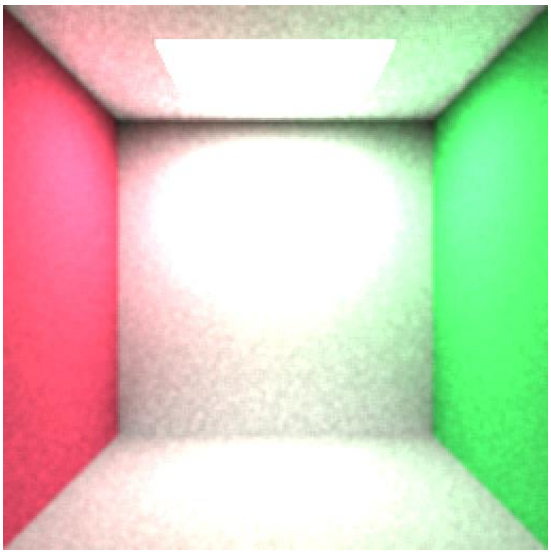
Este algoritmo se basa en la asignación del color del píxel de la imagen de alta resolución según el color promedio del píxel correspondiente y sus vecinos según el tamaño de un filtro dado en la imagen en baja resolución.

La Ilustración 23, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

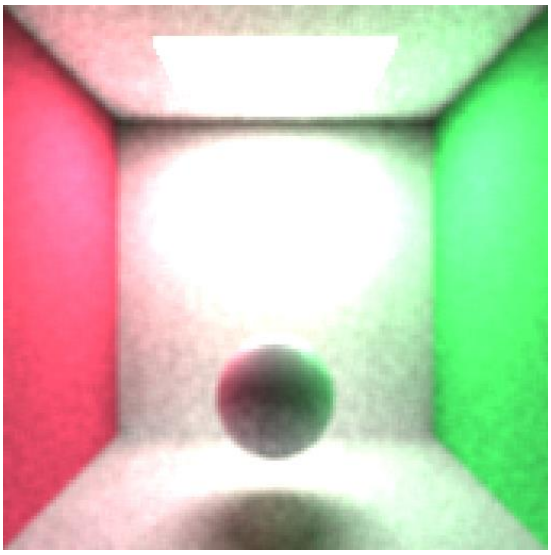
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 2.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 2, pueden ser apreciadas en la Sección 9.1.1

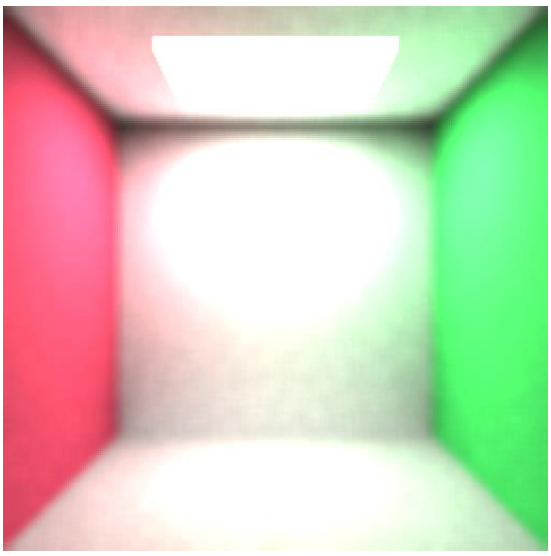
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 2.



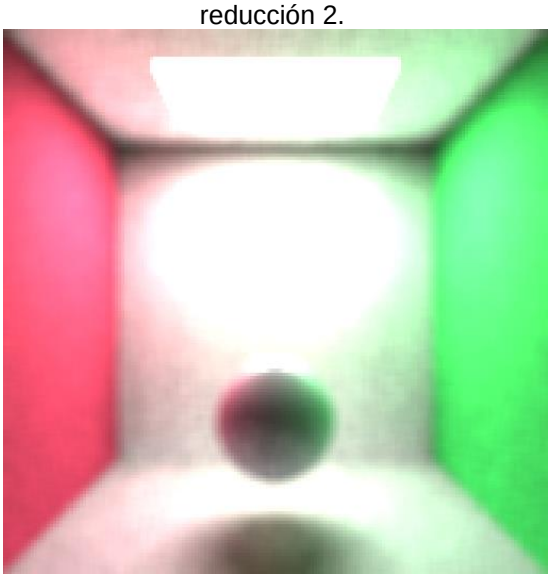
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 2.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 2.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 2.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 2.

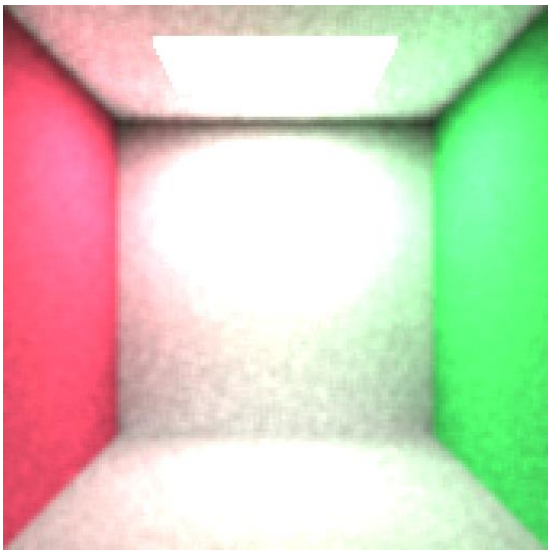
Ilustración 23*. Imágenes generadas por el algoritmo upsampling basado en el promedio de los vecinos en el espacio imagen (2D) con factor de reducción 2.*

La Ilustración 24, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

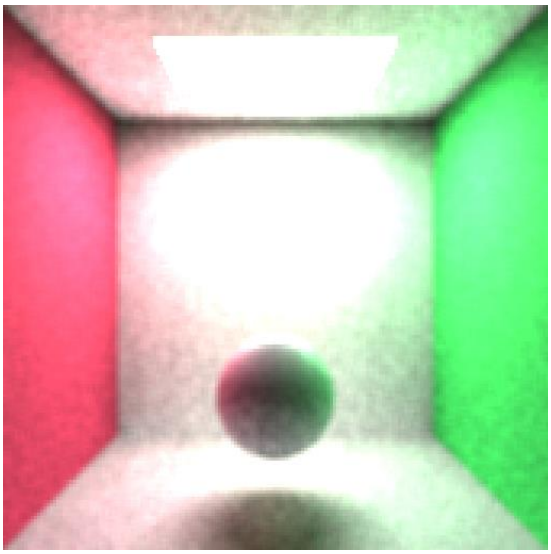
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 3.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 3, pueden ser apreciadas en la Sección 9.1.1

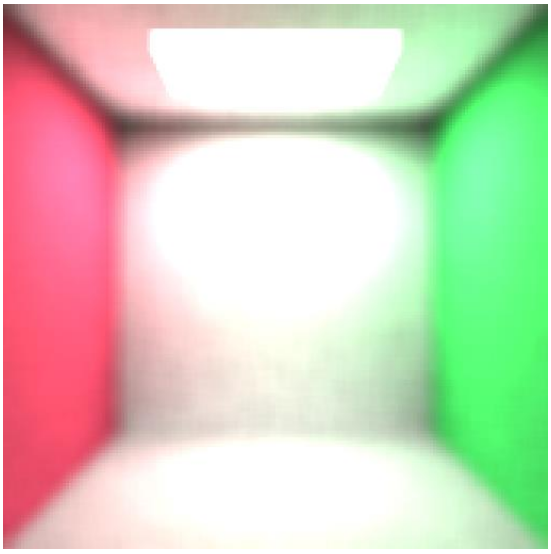
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 3.



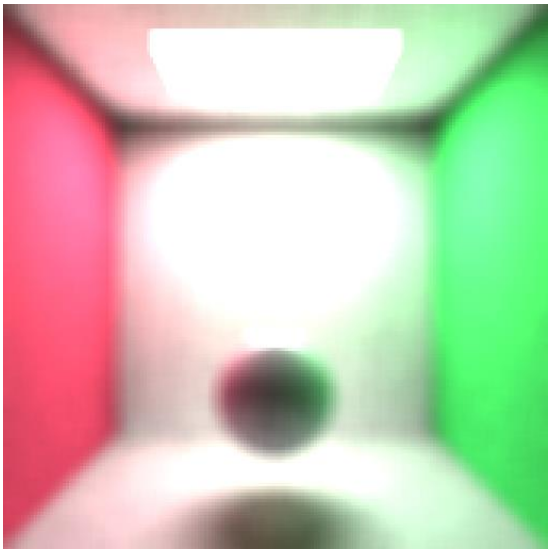
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 3.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 3.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 3.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 3.

Ilustración 24. *Imágenes generadas por el algoritmo upsampling basado en el promedio de los vecinos en el espacio imagen (2D) con factor de reducción 3.*

Este algoritmo realiza un promedio simple, sobre un filtro de píxeles vecinos contemplados, al ser un promedio, simple, a mayor sea el filtro, si bien el ruido visual irá desapareciendo, la definición de la imagen también, ya que será aplicado un suavizado sobre los píxeles.

Si se compara con las imágenes de referencia generadas por los mismos parámetros de la Sección 5.2, se tiene un ligero aumento del ruido con un filtro bajo, pero este disminuye al suavizar ligeramente la imagen.

En este caso, los mejores parámetros a optar sería que según el factor de reducción se pudiese escoger un tamaño de filtro dónde el nivel de suavizado sea lo suficientemente bueno para disminuir el ruido, pero mantener un equilibrio

con la calidad de la imagen, siendo entonces filtros pequeños aquellos a los que se debería optar.

Los tiempos de ejecución obtenidos por las imágenes observadas se presentan en el Gráfico 2:

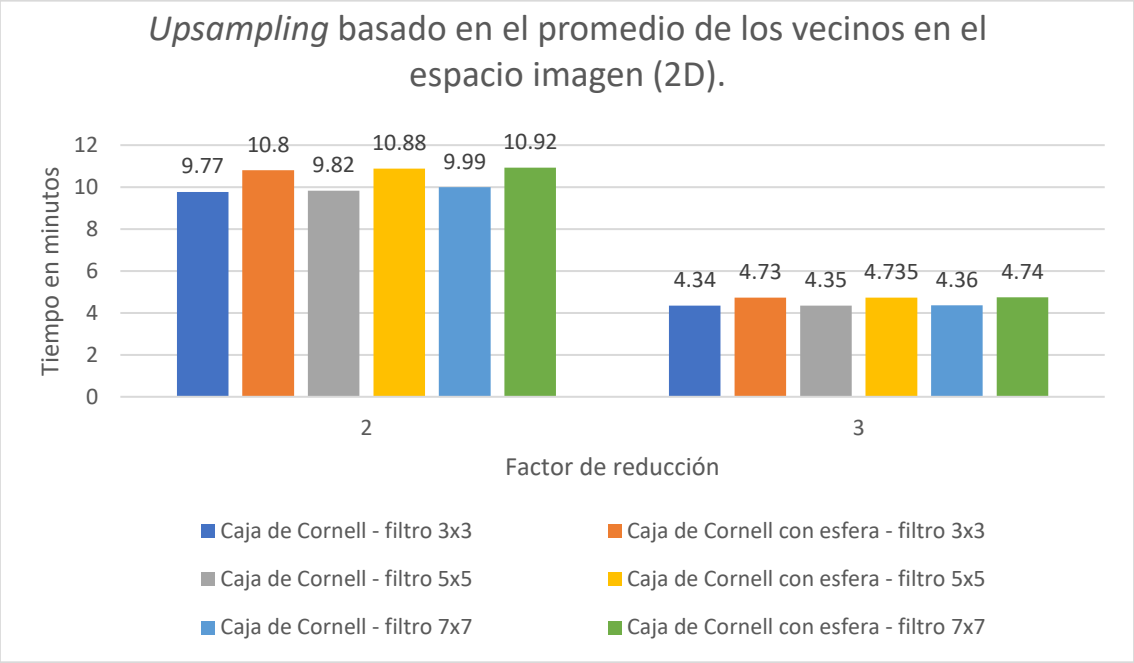


***Gráfico 2**. Tiempos de ejecución obtenidos por el algoritmo upsampling basado en el promedio de los vecinos en el espacio imagen (2D).*

Utilizando este algoritmo, como se puede observar en el Gráfico 2, considerando el mayor coste de ejecución obtenido, en comparativa al mayor coste de ejecución obtenido por el *path tracing*, se tiene una mejora dónde el algoritmo se ejecuta en el **21,22%** del tiempo que tomaría hacerlo con *path tracing*.

5.3.3 Upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D)

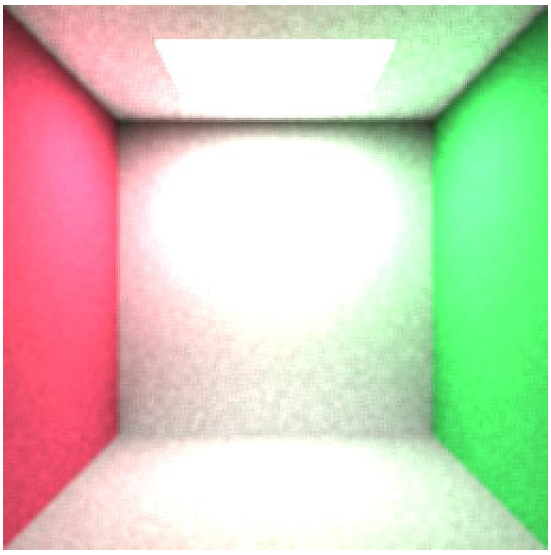
Este algoritmo se basa en la asignación del color del píxel de la imagen de alta resolución según el color promedio ponderado (usando la distancia euclidiana como mesura) del píxel correspondiente y sus vecinos según el tamaño de un filtro dado en la imagen en baja resolución.

La Ilustración 25, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

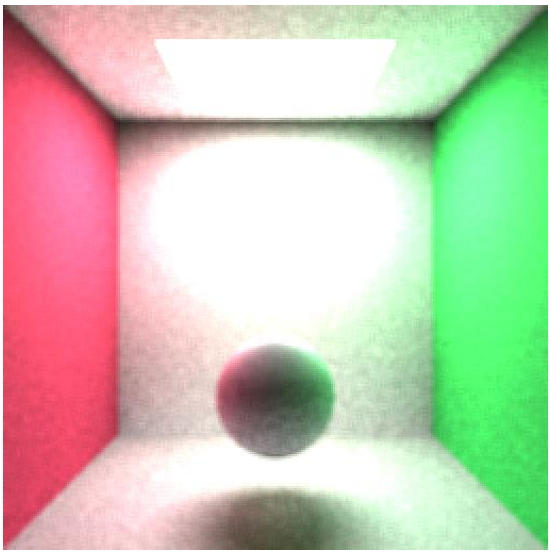
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 2.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 2, pueden ser apreciadas en la Sección 9.1.2

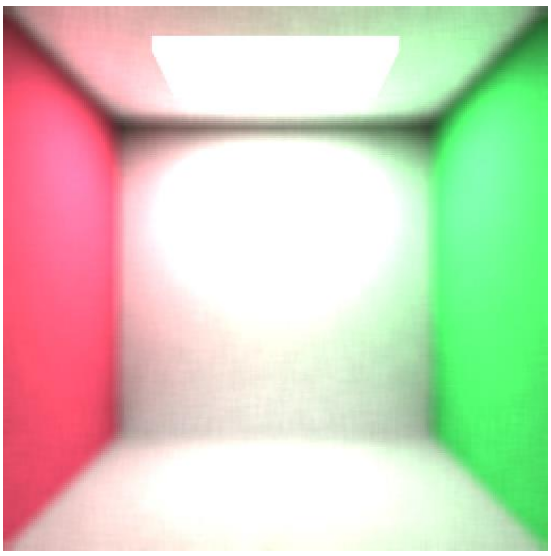
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 2.



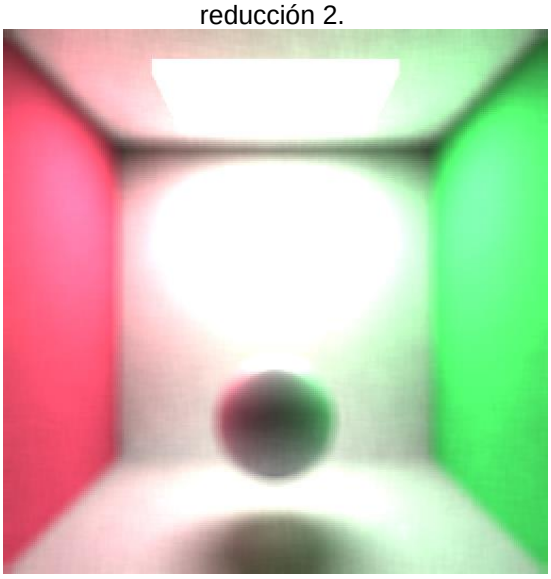
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 2.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 2.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 2.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 2.

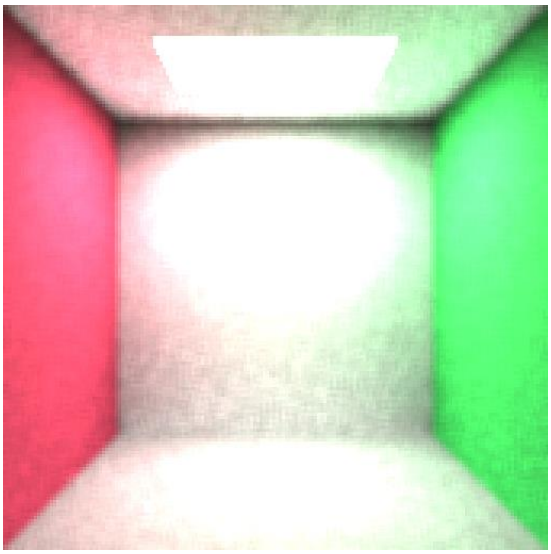
***Ilustración 25**. Imágenes generadas por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D) con factor de reducción 2.*

La Ilustración 26, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

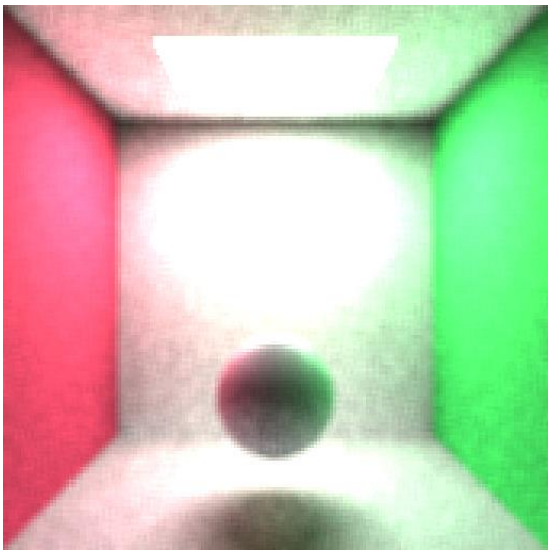
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 3.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 3, pueden ser apreciadas en la Sección 9.1.2

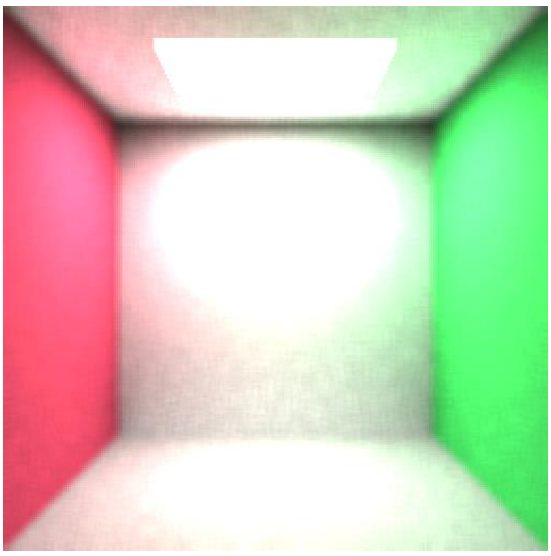
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 3.



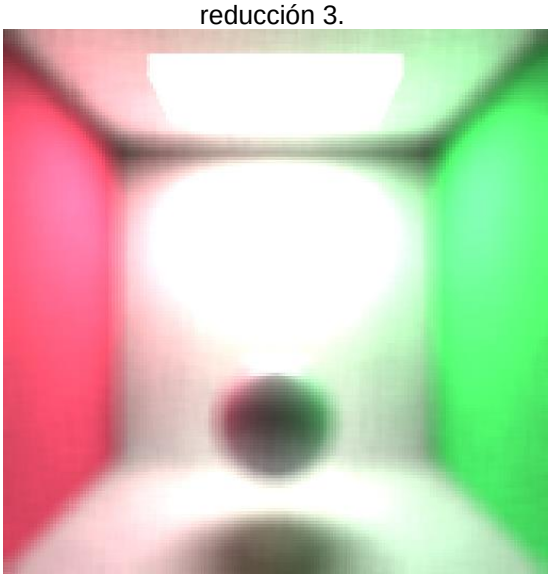
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 3.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 3.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 3.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 3.

Ilustración 26. *Imágenes generadas por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D) con factor de reducción 3.*

Este algoritmo, al igual que el previo, sigue manejando un promedio, pero ahora ponderado, al manejar un promedio, el ruido de la imagen es disminuido gracias al suavizado de esta, pero la ponderación en base a la distancia permite que, con filtros de un tamaño moderado, como se observa en la Ilustración 25 (d), la definición entre los diferentes objetos, sin ser perfecta, mejora, a comparación del algoritmo previo que era un promedio no ponderado.

Si se comparan las imágenes de la Ilustración 25, con las imágenes de referencia de los mismos parámetros de la Sección 5.2, se puede observar que no hay una diferencia notoria entre el ruido visual de estas.

Por lo qué, un filtro óptimo según los píxeles disponibles podrá decantar en un suavizado adecuado con una relativa definición entre los componentes de la imagen.

Los tiempos de ejecución obtenidos por las imágenes observadas se presentan en el Gráfico 3:

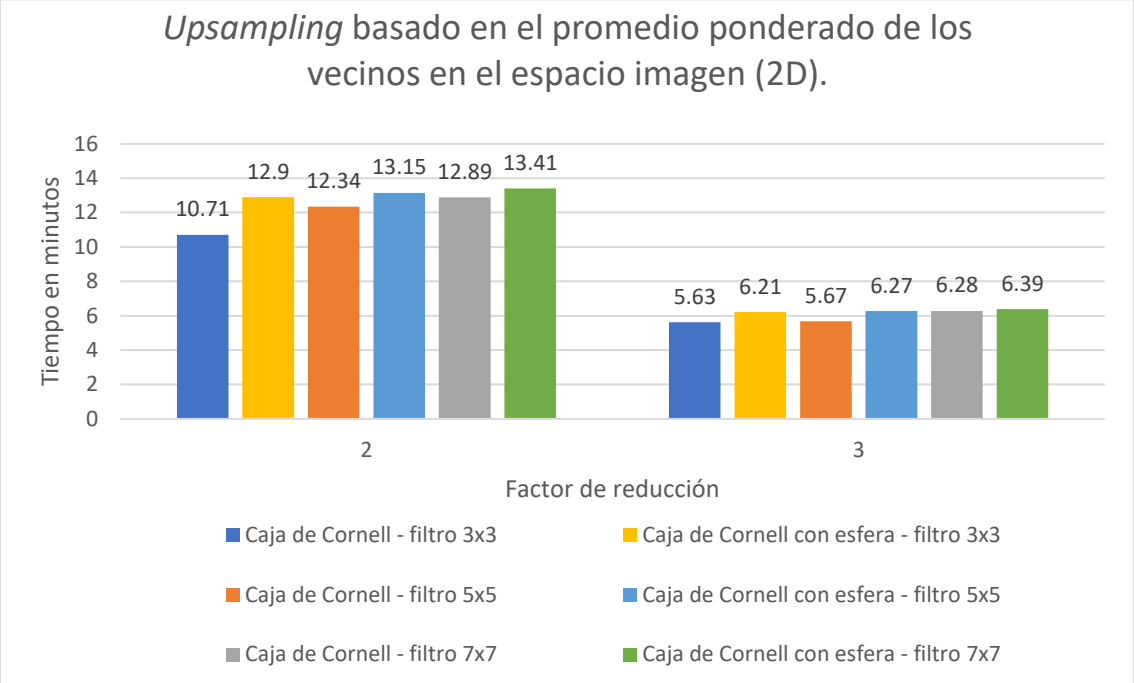


Gráfico 3. *Tiempos de ejecución obtenidos por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D)*

Utilizando este algoritmo, como se puede observar en el Gráfico 3, considerando el mayor coste de ejecución obtenido, en comparativa al mayor coste de ejecución obtenido por el *path tracing*, se tiene una mejora dónde el algoritmo se ejecuta en el **26,06%** del tiempo que tomaría hacerlo con *path tracing*.

5.3.4 Upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D)

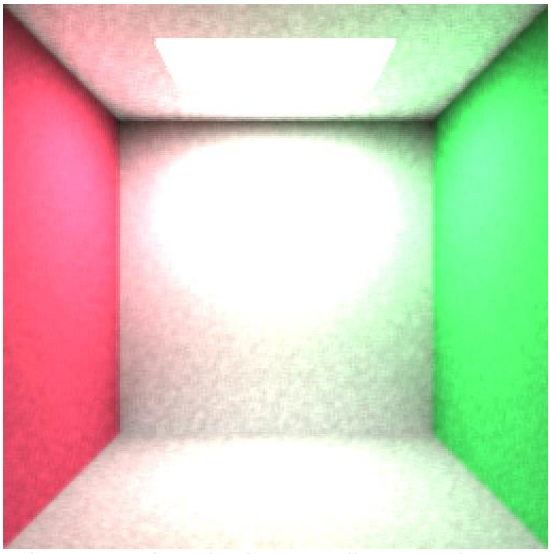
Este algoritmo se basa en la asignación del color del píxel de la imagen de alta resolución según el color promedio ponderado (usando la distancia euclidiana entre las intersecciones como mesura) del píxel correspondiente y sus vecinos según el tamaño de un filtro dado en la imagen en baja resolución.

La Ilustración 27, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

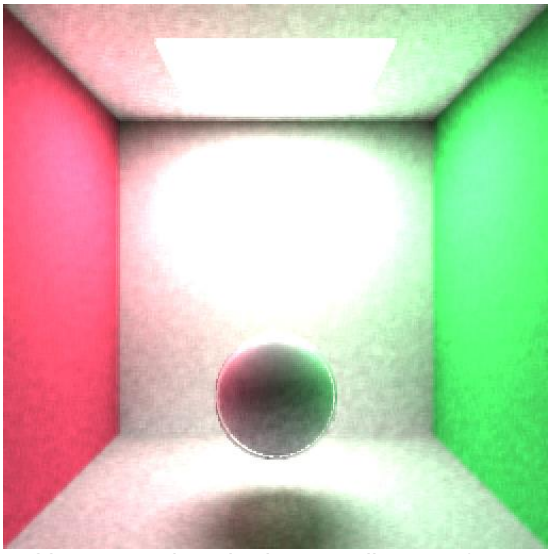
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 2.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 2, pueden ser apreciadas en la Sección 9.1.3

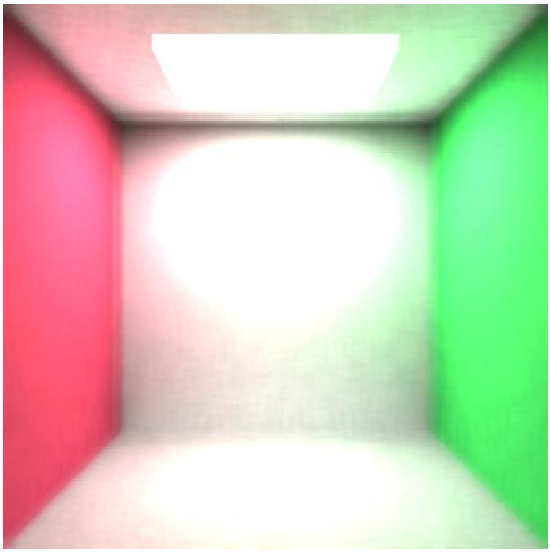
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 2.



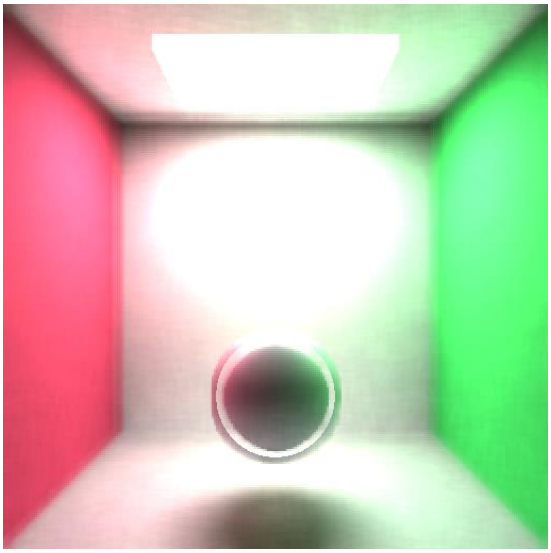
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 2.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 2.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 2.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 2.

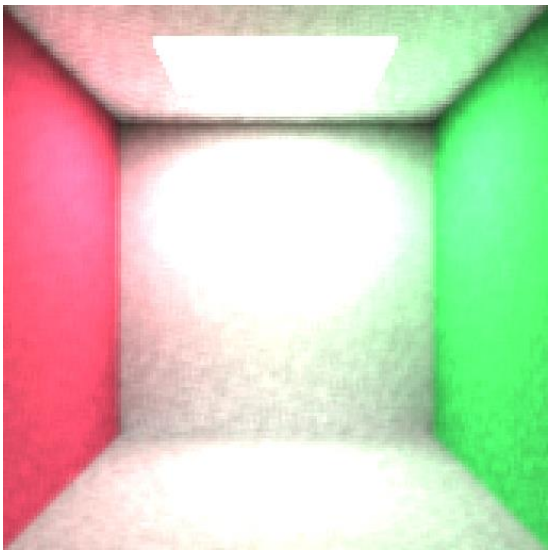
Ilustración 27. *Imágenes generadas por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D) con factor de reducción 2.*

La Ilustración 28, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

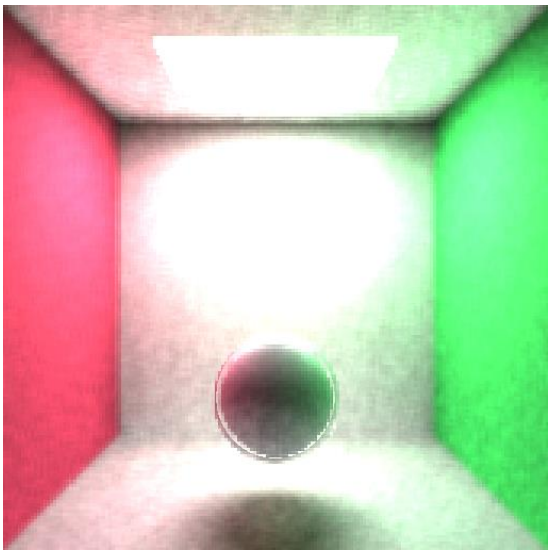
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 3.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 3, pueden ser apreciadas en la Sección 9.1.3

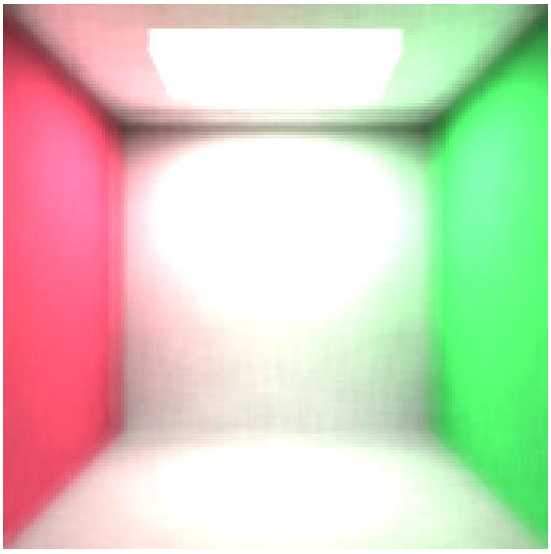
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 3.



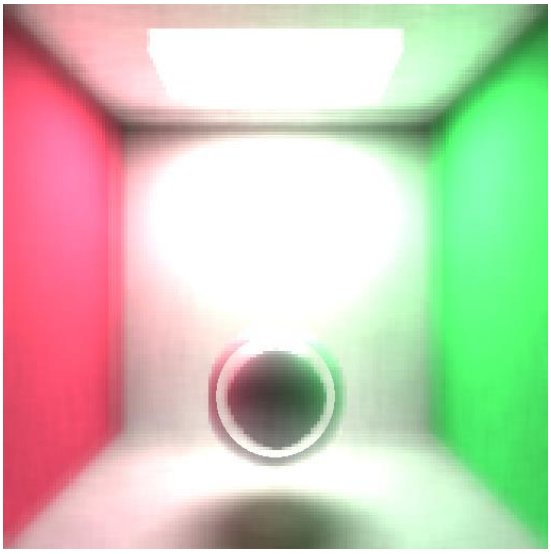
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 3.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 3.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 3.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 3.

Ilustración 28. *Imágenes generadas por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D) con factor de reducción 3.*

Tanto en la Ilustración 27, como la Ilustración 28 podemos observar que la escena de caja de Cornell con esfera, el trabajar con un promedio ponderado usando la distancia euclidiana en las intersecciones 3D, puede generar errores en la definición de los objetos, dado a que es un promedio ponderado, sin ningún delimitante de que se debe ponderar, todos los pixeles en el filtro serán ponderados, y afectarán al resultado final, como se puede ver en el anillo formado alrededor de la esfera, a mayor sea el filtro, más pixeles serán contemplados, y mayor será la probabilidad de errores no deseados en la definición de los diferentes objetos en la imagen.

Dicho lo anterior, la mejor forma de aproximarse a un buen resultado es según la cantidad de píxeles disponibles para interpolar definir un filtro pequeño que pueda aprovechar bien la información sin comprometer la calidad de la imagen.

Los tiempos de ejecución obtenidos por las imágenes observadas se presentan en el Gráfico 4:

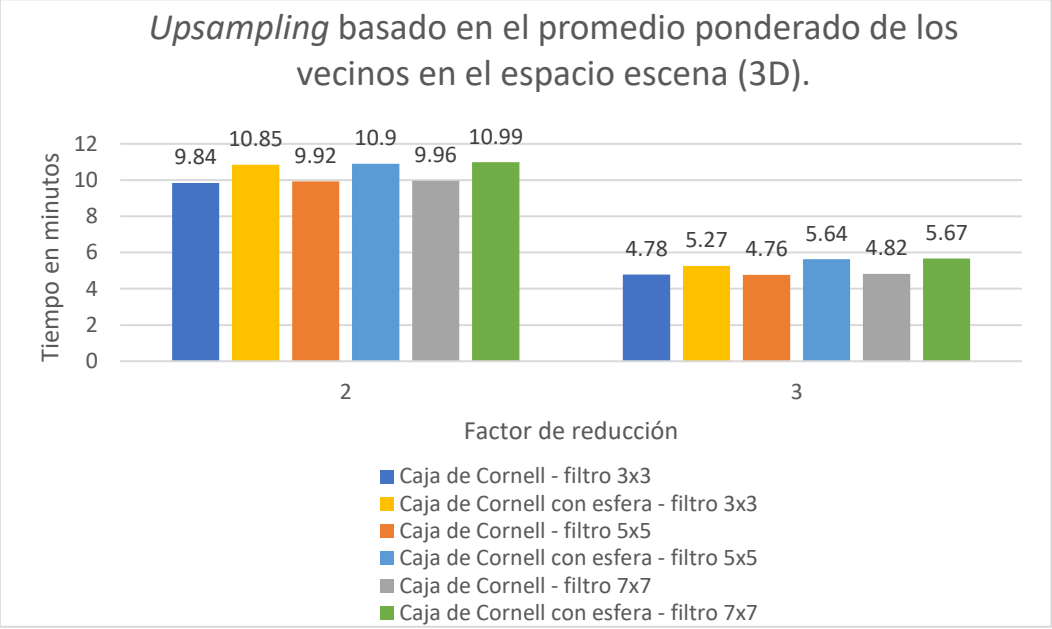


***Gráfico 4**. Tiempos de ejecución obtenidos por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D)*

Utilizando este algoritmo, como se puede observar en el Gráfico 4, considerando el mayor coste de ejecución obtenido, en comparativa al mayor coste de ejecución obtenido por el *path tracing*, se tiene una mejora dónde el algoritmo se ejecuta en el **21,36%** del tiempo que tomaría hacerlo con *path tracing*.

5.3.5 Upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D)

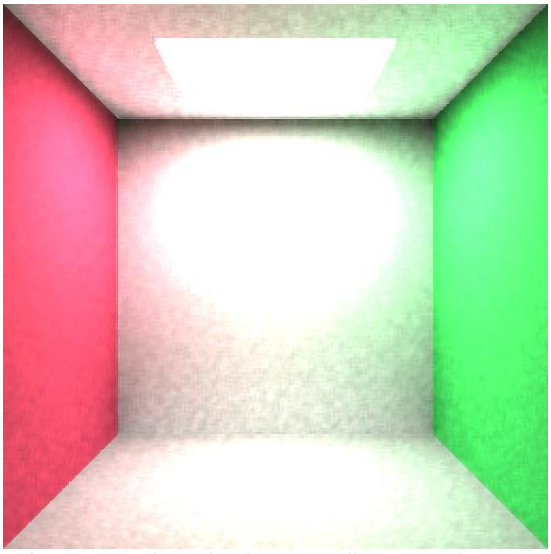
Este algoritmo se basa en la asignación del color del píxel de la imagen de alta resolución según el color promedio ponderado (usando la distancia euclidiana entre las intersecciones como mesura si es que el producto escalar de las normales de las intersecciones es mayor a un parámetro dado) del píxel correspondiente y sus vecinos según el tamaño de un filtro dado en la imagen en baja resolución.

La Ilustración 29, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

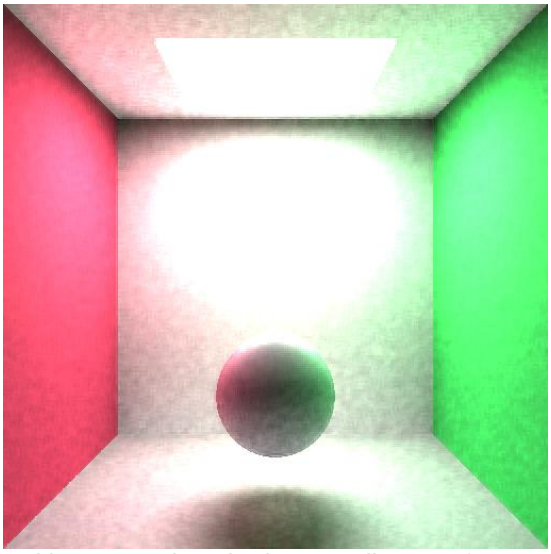
Las ilustraciones a) y b) usaron filtro de 3x3 y factor de reducción 2.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 2, pueden ser apreciadas en la Sección 9.1.4

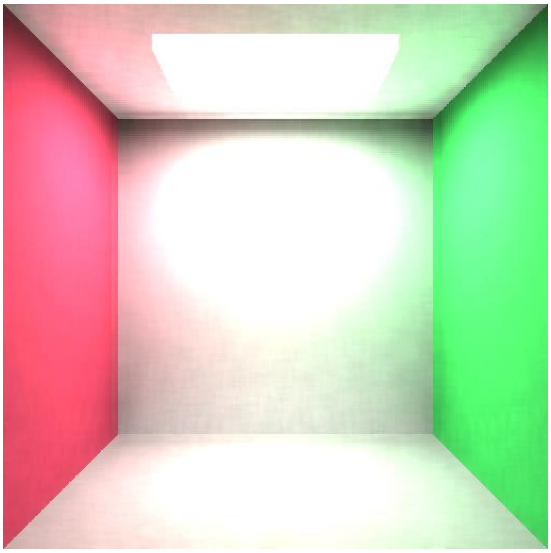
Las ilustraciones c) y d) usaron filtro de 7x7 y factor de reducción 2.



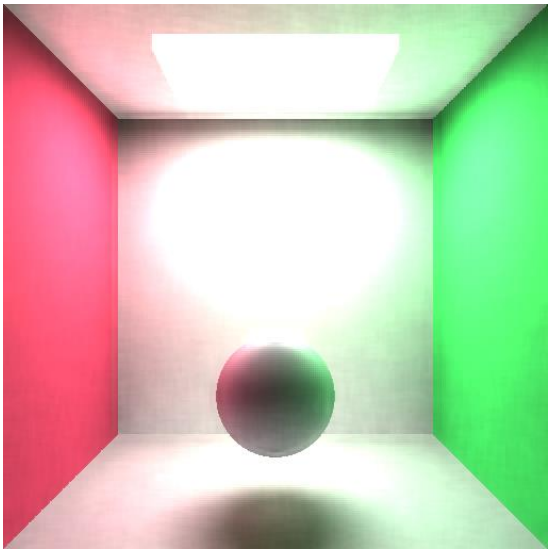
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 2.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 2.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 2.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 2.

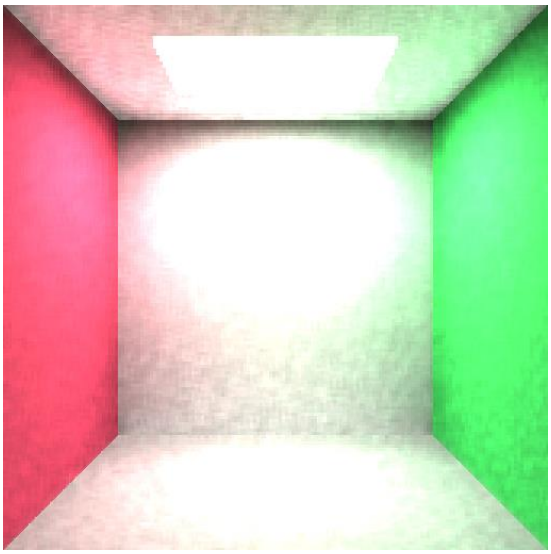
Ilustración 29*.Imágenes generadas por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) con factor de reducción 2.*

La Ilustración 30, muestra las imágenes obtenidas con los parámetros previamente descritos, con la siguiente definición específica:

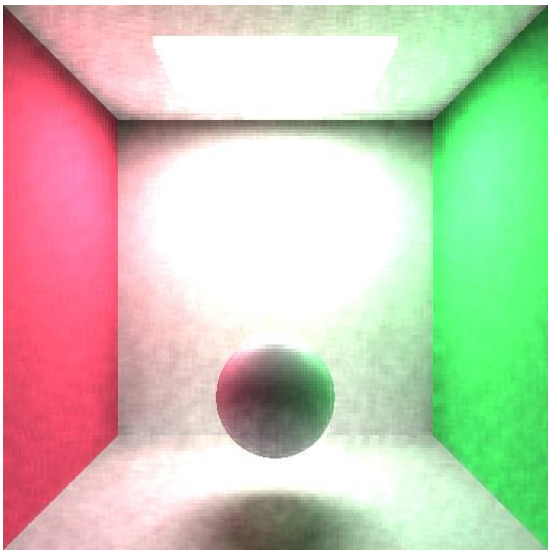
Las ilustraciones a) y d) usaron filtro de 3x3 y factor de reducción 3.

Las ilustraciones que usaron filtro de 5x5 y factor de reducción 3, pueden ser apreciadas en la Sección 9.1.4

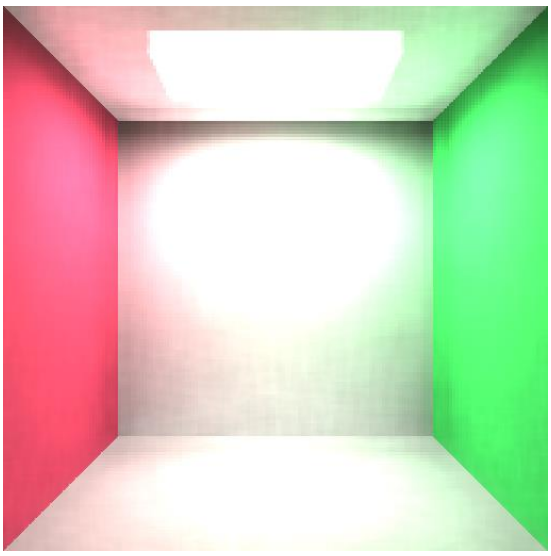
Las ilustraciones c) y f) usaron filtro de 7x7 y factor de reducción 3.



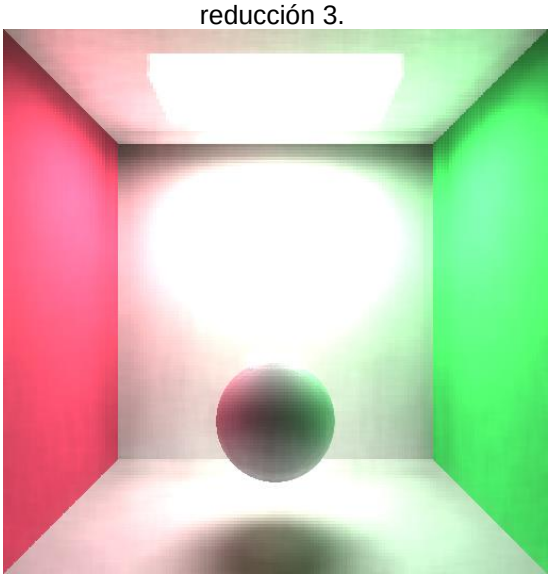
a) Imagen de caja de Cornell generada con filtro de 3x3, factor de reducción 3.



b) Imagen de caja de Cornell con esfera generada con filtro de 3x3, factor de reducción 3.



c) Imagen de caja de Cornell generada con filtro de 7x7, factor de reducción 3.



d) Imagen de caja de Cornell con esfera generada con filtro de 7x7, factor de reducción 3.

***Ilustración 30**. Imágenes generadas por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) con factor de reducción 3.*

En este algoritmo conseguimos observar una notoria mejora en la calidad de imagen, con un ruido visual apenas perceptible en bordes.

Al ser un promedio ponderado, cuyo delimitante es la normal de la luz, podemos observar que la definición entre objetos de la imagen, es mucho mejor que los algoritmos predecesores, al igual que, al aumentar el tamaño del filtro la calidad de la imagen mejora, ya que gracias al delimitante, solo utilizará la información de aquellos píxeles que se consideren útiles, mejorando así conjunto con el filtro, la calidad de imagen, llegando a superar por mucho, en la disminución de ruido visual a las imágenes de referencia generadas por los mismos parámetros de la Sección 5.2

Los tiempos de ejecución obtenidos por las imágenes observadas se presentan en el Gráfico 5:

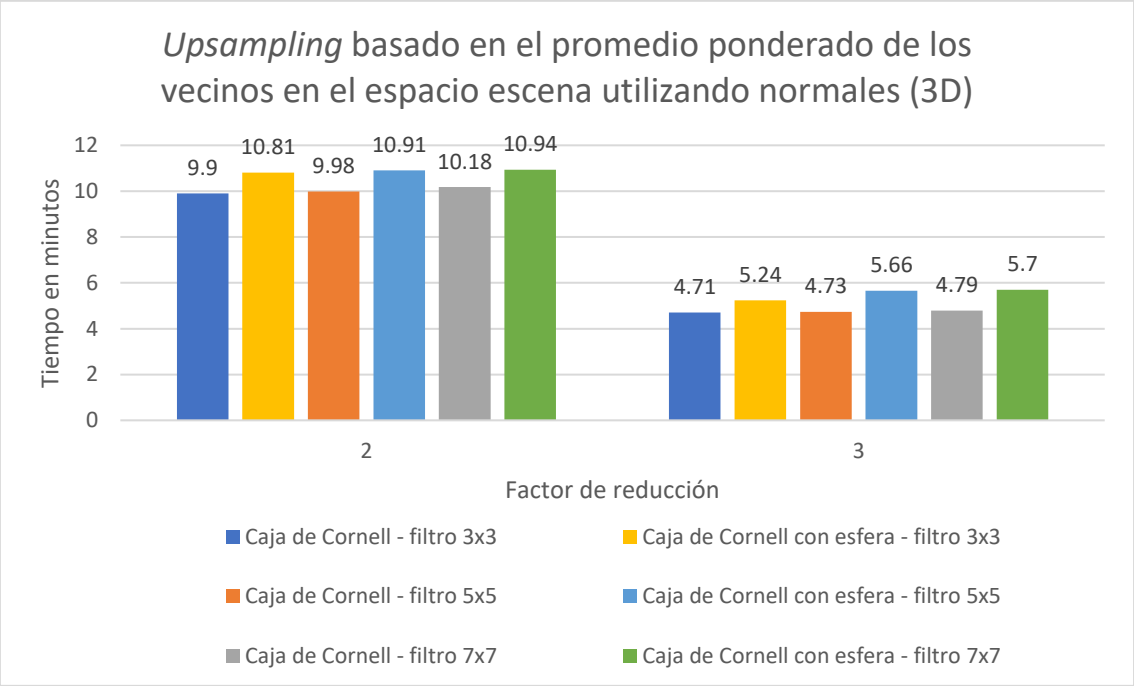


***Gráfico 5**. Tiempos de ejecución obtenidos por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).*

Utilizando este algoritmo, como se puede observar en el Gráfico 5, considerando el mayor coste de ejecución obtenido, en comparativa al mayor coste de ejecución obtenido por el *path tracing*, se tiene una mejora dónde el algoritmo se ejecuta en el **21,26%** del tiempo que tomaría hacerlo con *path tracing*.

5.4 Comparación de resultados del algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) con respecto al *path tracing*

En esta sección se realizarán dos aproximaciones con el algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) dónde se compararán sus resultados generados con parámetros equivalentes a los del algoritmo *path tracing* para deliberar sobre la calidad de estos.

5.4.1 Equiparación del número de muestras totales

Teoría

Dado el factor de reducción que se utilice para generar la imagen en baja resolución, se pierde un número de muestras (el factor 2, dado a que se reduce la resolución tanto en largo como en ancho, se reducen las muestras en 4).

Para probar la capacidad de nuestro algoritmo para crear imágenes fotorrealistas de mayor calidad, se ha optado por equiparar el mismo número de muestras a las que usaría el *path tracing,* al decir equiparar número de muestras, se trata de tener el mismo número de muestras totales en la imagen y no por píxel, es decir:

En el caso del *path tracing*, generar una imagen de 512x512 píxeles, con 10,000 muestras por píxel, existe un total de 2,621,440,000 muestras efectuadas sobre toda la escena.

Entonces, en el caso del algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D), si se tiene un factor de reducción de 2, la imagen de baja resolución generada por el método *preprocess* (para mayor detalle sobre el proceso de *upsampling* referirse a la Sección 3), sería de 256x256 píxeles, una cuarta parte del tamaño de una imagen de 512x512 pixeles, entonces, para equiparar el mismo número de muestras totales sobre la imagen, en lugar de muestrear 10,000 muestras por píxel, se muestrearían 40,000 muestras, así, con la resolución de 256x256, se tendrían la misma cantidad de 2,621,440,000 muestras efectuadas sobre toda la escena.

La teoría es que, siendo el píxel más muestreado, el color asignado será más acertado, y con la interpolación de *upsampling* la imagen final de alta resolución generada, será de una calidad superior a la que da el algoritmo *path tracing.*

Parámetros

La imagen final para generar será de 512x512 píxeles.

En este caso, se usarían los siguientes parámetros para el algoritmo de *path tracing*:

nSPP: 10,000 (número de muestras utilizadas por cada píxel calculado).

maxDepth: 2 (número máximo de rebote dado a los rayos generados).

Para el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) los parámetros serían los siguientes:

nSPP: 40,000 (número de muestras utilizadas por cada píxel calculado).

maxDepth: 2 (número máximo de rebote dado a los rayos generados).

lessResDivFactor: 2 (factor de reducción de la resolución para generar la imagen con el *path tracing)*.

neighLayers: 3 (parámetro que define el tamaño del filtro utilizado en los algoritmos que utilicen promedios, siendo el valor el número de capas alrededor al píxel correspondiente, siendo 1, un filtro de 3x3).

useCenter: 1 (parámetro que define si se considerará también el píxel correspondiente dentro del filtro previamente definido, 1 es que será utilizado).

normalFactor: 0.95 (parámetro que define el valor el cuál el producto escalar de dos normales debe superar, para que la distancia calculada no sea despreciada)

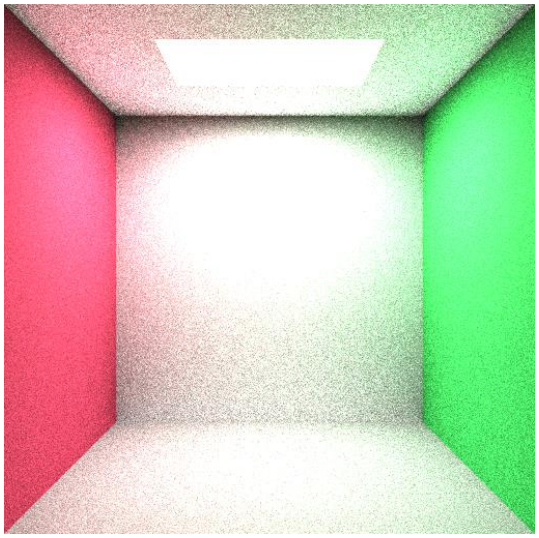
Con estos parámetros, ambos algoritmos utilizan un número total de muestras (es decir, en toda la escena y no por píxel) de 2,621,440,000.

Resultados

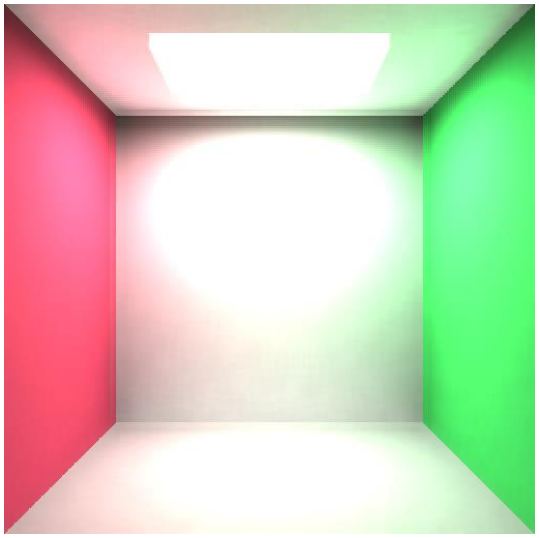
Los resultados obtenidos se pueden apreciar en la Ilustración 31 (escena de caja de Cornell sencilla) y en la Ilustración 32 (escena de caja de Cornell con esfera).

Resulta sencillo observar la amplia mejora de calidad de imagen en las imágenes (b) generadas por el algoritmo de *upsampling* de cualquiera de las ilustraciones, dónde la definición entre los modelos de la escena es alta, y la cantidad de ruido visual es prácticamente nulo.

Escena de caja de Cornell.



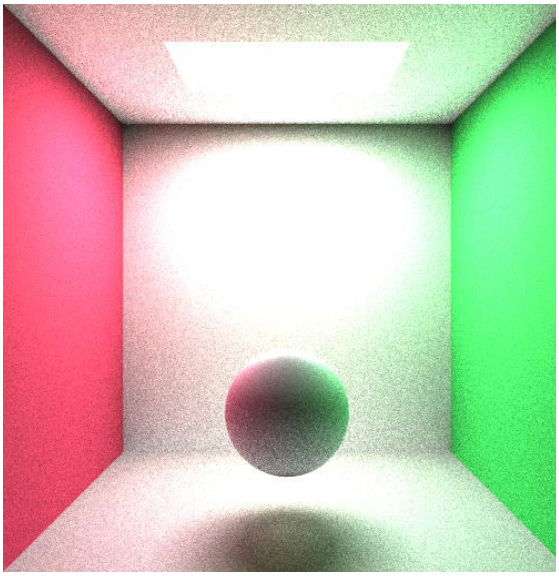
a) Escena de caja de Cornell sencilla utilizada a lo largo del proyecto generada por el *path tracing*.



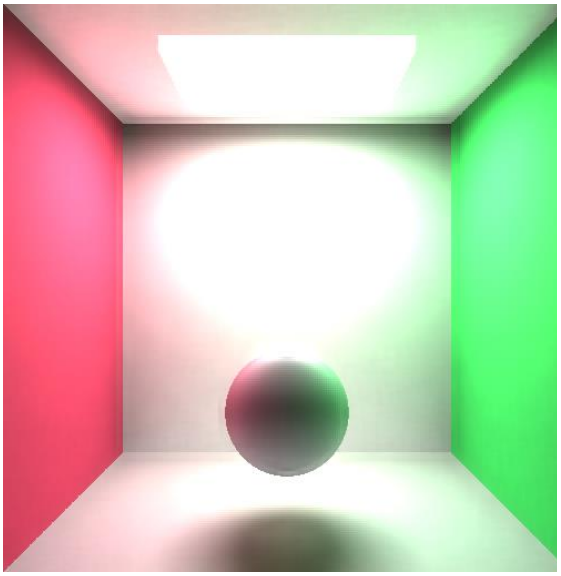
b) Imagen de caja de Cornell generada por el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).

Ilustración 31. *Escena de caja de Cornell generada por el algoritmo de path tracing y el algoritmo de* upsampling *basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) equiparando el número total de muestras.*

Escena de caja de Cornell con esfera.



a) Escena de caja de Cornell con una esfera utilizada a lo largo del proyecto generada por el algoritmo *path tracing.*



b) Imagen de caja de Cornell con esfera generada por el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).

Ilustración 32. *Escena de caja de Cornell con esfera generada por el algoritmo de path tracing y el algoritmo de* upsampling *basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) equiparando el número total de muestras.*

Los tiempos de ejecución obtenidos por las imágenes observadas se presentan en el Gráfico 6:

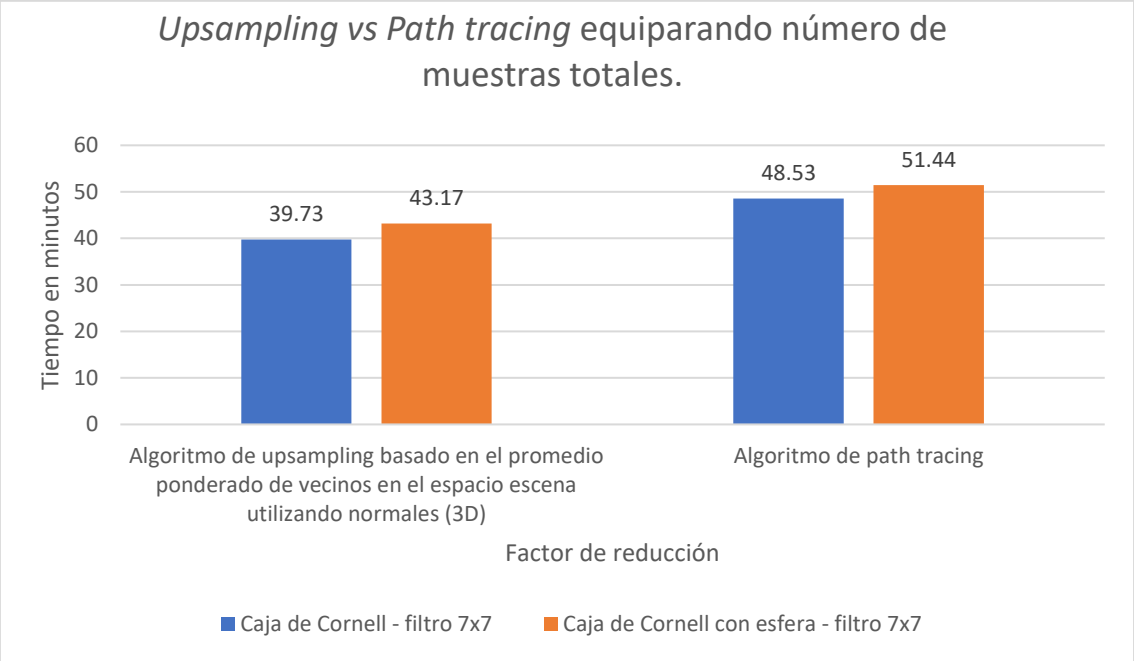


Gráfico 6. *Tiempos de ejecución obtenidos por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales utilizando normales (3D) con factor de reducción 2 equiparando el número de muestras a las del path tracing.*

Utilizando este algoritmo, como se puede observar en el Gráfico 6, considerando el mayor coste de ejecución obtenido, en comparativa al mayor coste de ejecución obtenido por el *path tracing*, se tiene una mejora dónde el algoritmo se ejecuta en el **83,92%** del tiempo que tomaría hacerlo con *path tracing*.

Esta aproximación permite demostrar que, no solamente se ahorra tiempo de ejecución con el algoritmo de *upsampling* a comparación del *path tracing*, sino que es posible, mejorar la calidad de la imagen con el mismo número de muestras totales utilizadas a través de este algoritmo.

5.4.2 Equiparación del tiempo de ejecución

Teoría

Este experimento es similar al de la Sección 5.4.1 dónde hemos podido observar que aun utilizando el mismo número de muestras totales en la imagen que el *path tracing* la técnica de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) toma un tiempo menor en generar un resultado de mayor calidad.

En este experimento, buscaremos incrementar el tiempo de ejecución del algoritmo de *upsampling* aproximadamente al mismo que el tiempo de ejecución del *path tracing.*

La teoría es que, tardando el mismo tiempo de ejecución con el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D), que el algoritmo de *path tracing*, la imagen resultante del algoritmo de *upsampling* tendrá una calidad superior a la generada por el *path tracing*.

Parámetros

La imagen final para generar será de 512x512 píxeles.

Con los siguientes parámetros, para el *path tracing*:

nSPP: 10,000 (número de muestras utilizadas por cada píxel calculado).

maxDepth: 2 (número máximo de rebote dado a los rayos generados).

El tiempo de ejecución utilizado fue:

- Escena de caja de Cornell - Ilustración 33 (a): **48,53 minutos**.
- Escena de caja de Cornell con esfera – Ilustración 34 (a): **51,44 minutos.**

Basándonos en el resultado de tiempo de ejecución obtenido por el algoritmo de *upsampling* en la Sección 5.4.1, y recordando el proceso de la aproximación de las técnicas de *upsampling* (detallado en la Sección 3), que se basa en interpolar una imagen de baja resolución creada por el algoritmo *path tracing*, y este aumenta su tiempo de ejecución según el número de muestras utilizadas, aumentaremos la cantidad de muestras a utilizar por el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) para aproximarnos a los tiempos previamente mencionados del algoritmo *path tracing*.

Los parámetros definidos para el algoritmo de *upsampling* son los siguientes:

nSPP: 48,000 (número de muestras utilizadas por cada píxel calculado).

maxDepth: 2 (número máximo de rebote dado a los rayos generados).

lessResDivFactor: 2 (factor de reducción de la resolución para generar la imagen con el *path tracing)*.

neighLayers: 3 (parámetro que define el tamaño del filtro utilizado en los algoritmos que utilicen promedios, siendo el valor el número de capas alrededor al píxel correspondiente, siendo 1, un filtro de 3x3).

useCenter: 1 (parámetro que define si se considerará también el píxel correspondiente dentro del filtro previamente definido, 1 es que será utilizado).

normalFactor: 0.95 (parámetro que define el valor el cuál el producto escalar de dos normales debe superar, para que la distancia calculada no sea despreciada)

El tiempo de ejecución utilizado fue:

- Escena de caja de Cornell - Ilustración 33 (b): **47,12 minutos**.
- Escena de caja de Cornell con esfera – Ilustración 34 (b): **53,63 minutos.**

Los resultados obtenidos son aproximados al tiempo utilizado por el *path tracing*, no utilizando el tiempo exacto, pero con variaciones de +- 2 minutos, dado a que la aproximación por número de muestras no tiene una relación de tiempo exacto por muestra, lo daremos como una aproximación válida por ser una variación pequeña con respecto al tiempo de ejecución total.

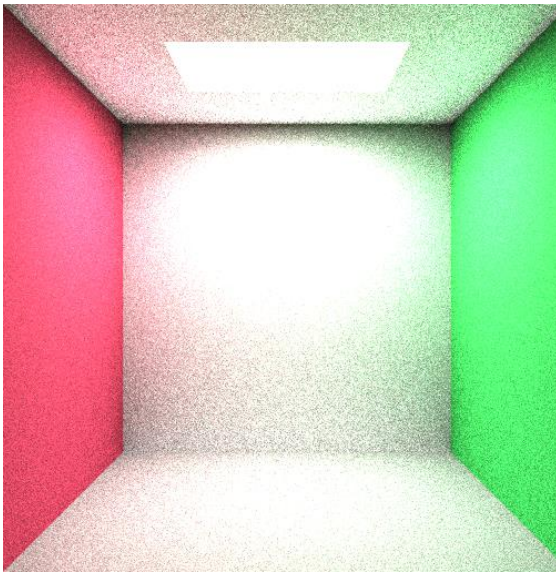
Resultados

Los resultados obtenidos se pueden apreciar en la Ilustración 33 (escena de caja de Cornell sencilla) y en la Ilustración 34 (escena de caja de Cornell con esfera).

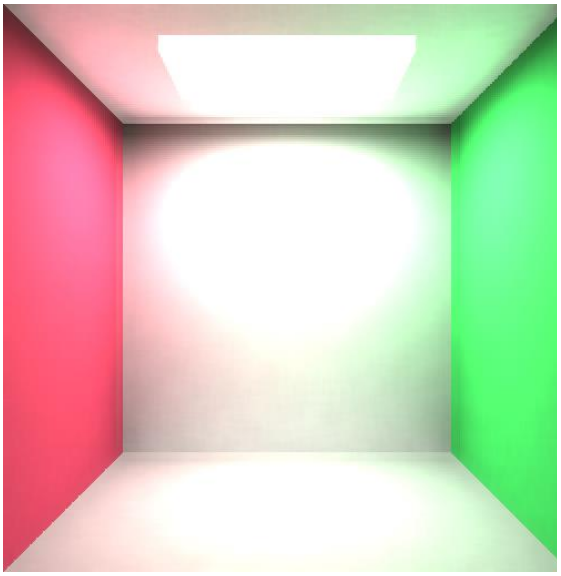
Se puede observar claramente la mejora en la calidad de imagen en las imágenes (b) generadas por el algoritmo de *upsampling* de ambas ilustraciones, el ruido visual es prácticamente nulo y los diferentes modelos en la escena se observan bien definidos.

Si decidimos también comparar estos resultados con los de la Sección 5.4.1, podemos ver que la mejora entre las imágenes generadas por el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D), es poco notoria, prácticamente imperceptible, esto porqué para tener una diferencia de calidad notoria con imágenes generadas por el *path tracing*, a mayor cantidad de muestras utilizadas, mayor debe ser la diferencia, siendo que 8,000 muestras de diferencia (entre 48,000 y 40,000) no supone un cambio lo suficientemente grande, para percibir una mejora a simple vista.

Escena caja de Cornell



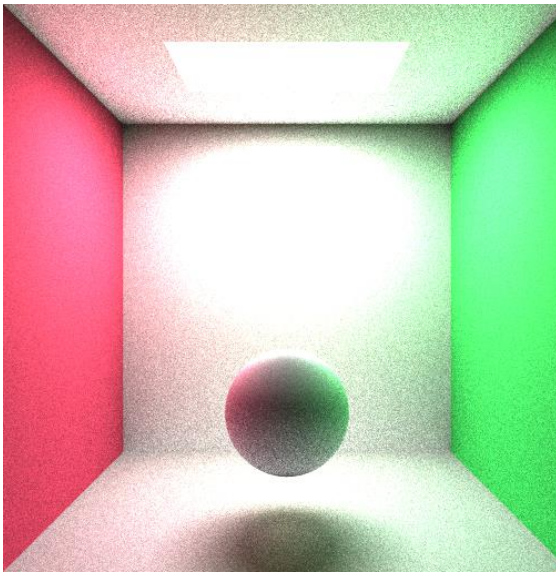
a) Escena de caja de Cornell con una esfera utilizada a lo largo del proyecto generada por el algoritmo *path tracing.*



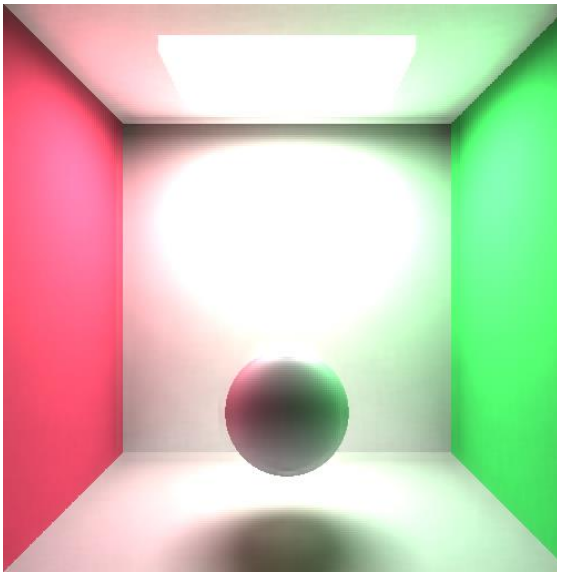
b) Imagen de caja de Cornell generada por el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).

Ilustración 33. *Escena de caja de Cornell generada por el algoritmo de path tracing y el algoritmo de* upsampling *basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) equiparando el tiempo de ejecución.*

Escena caja de Cornell con esfera



a) Escena de caja de Cornell con una esfera utilizada a lo largo del proyecto generada por el algoritmo *path tracing.*



b) Imagen de caja de Cornell con esfera generada por el algoritmo de *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D).

Ilustración 34. *Escena de caja de Cornell con esfera generada por el algoritmo de path tracing y el algoritmo de* upsampling *basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) equiparando el tiempo de ejecución.*

Los tiempos de ejecución obtenidos por las imágenes de la Ilustración 31 se presentan en el Gráfico 7:

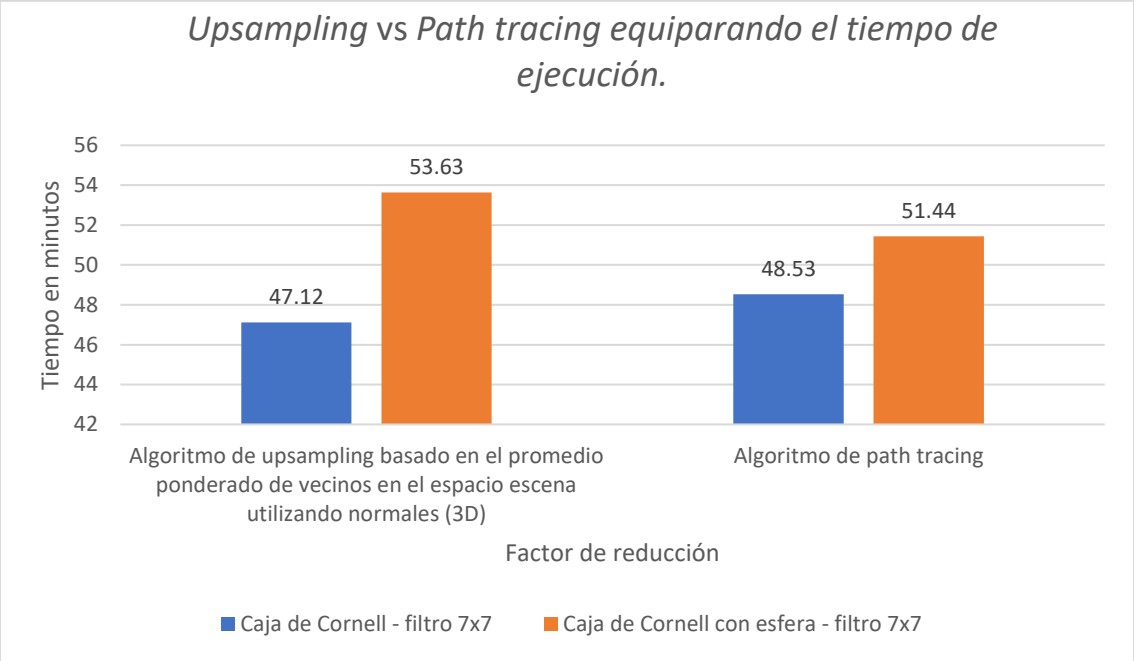


Gráfico 7. *Tiempos de ejecución obtenidos por el algoritmo upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales utilizando normales (3D) con factor de reducción 2 equiparando al tiempo de ejecución del path tracing.*

Esta aproximación permite desmotrar que utilizando el mismo tiempo de ejecución que el *path tracing*, los resultados obtenidos por la aproximación de *upsampling* desarrollada son mejores.

Al aumentar el tiempo de ejecución, se aumenta la información obtenida sobre la cuál el *upsampling* puede interpolar, generando mejores resultados.

6 Discusión de resultados y trabajo futuro

6.1 Discusión

Para discutir los resultados, resulta propio ver cómo las técnicas de *upsampling* desarrolladas, van mejorando de forma contigua, cada vez considerando algún aspecto más que la previamente planteada no utilizó.

Es fácil observar en todas pruebas realizadas que la presencia del ruido es mayor cuándo el factor de reducción es más grande, siendo lógico, ya que, a mayor factor de reducción, se pierde número de muestras utilizables, y si esto no se compensa de alguna manera, la interpolación realizada por la técnica de *upsampling* dará resultados con un notorio ruido visual.

Se puede observar como el algoritmo de *upsampling* basado en el vecino más cercano en el espacio imagen (2D), que se basa en una mera asignación directa entre la correspondencia de los pixeles da imágenes mejores que algoritmos como lo son aquellos que se basan en promedios; un promedio no ponderado, dará una imagen suavizada, que no es lo deseado, en el caso de los algoritmos que utilizan promedios ponderados, podemos ver cierto nivel de mejora en la ausencia de ruido en la imagen, pero si observamos a detalle, los algoritmos ponderados basados únicamente en la distancia empezarán a dar imágenes suavizadas o con elementos no deseados dentro de las mismas a partir de cierto tamaño del filtro.

El algoritmo con el que mejor resultados se han obtenido es el *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena (3D) usando normales, el cual, a diferencia de los algoritmos de promedio ponderado que sólo se basan en la distancia, al aumentar el tamaño del filtro, mejora la calidad de la imagen generada.

Lo cual nos permite ver que, para poder realizar propiamente técnicas de *upsampling* no se basa en tener mucha información, sino, en tener la información correcta que sea útil para la interpolación de la imagen de baja resolución a la grande.

Cada algoritmo según los parámetros dados y la escena 3D en la que se base, dará una imagen de cierta calidad, es por ello que el analizar sus resultados y comprender que rangos sobre sus parámetros son más convenientes para cada algoritmo, es relevante.

El algoritmo más lento tardo en ejecutar apenas poco más del 26% del tiempo de lo que tardaría el *path tracing*; fue posible a su vez, según el algoritmo y los parámetros dados a este, equiparar la calidad de las imágenes de referencia o mejorarla, permitiendo ver así que los resultados de los algoritmos propuestos como aproximación a la problemática de la reducción del tiempo de ejecución para la generación de una imagen fotorrealista, consiguen su cometido.

Las aproximaciones de equiparación de muestras o tiempo de ejecución permiten observar cómo con la interpolación correcta (en este caso el algoritmo selecto *upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales utilizando normales 3D)*, el *upsampling* genera resultados de un nivel de calidad superior al que tomaría el *path tracing* con los mismos parámetros.

6.2 Trabajo futuro

En busca de seguir aproximándonos de forma efectiva a la problemática de reducir los tiempos de ejecución para la generación de imágenes fotorrealistas, se ha contemplado la idea sobre la posibilidad de utilizar diferentes técnicas para reducir los tiempos de ejecución a la vez que se mejora la calidad de imagen.

Como trabajo futuro, se considera que sería factible realizar un muestreo adaptativo, el cual tiene como idea base, reducir el número de muestras necesarias para generar una imagen fotorrealista de calidad.

Para poder generar una imagen fotorrealista y poder interpolarla propiamente después, es necesario utilizar una métrica que permita definir cuáles píxeles es necesario muestrear, y cuántas muestras sobre los mismos son requeridas para que el color calculado sea lo más acertado posible.

Teoría de reutilización

La idea de reutilización de píxeles ya muestreados parece una aproximación aceptable y siguiendo la misma lógica que han seguido los diferentes métodos de *upsampling* que se han desarrollado a lo largo del proyecto, el reutilizar colores de los píxeles en las áreas cuyas intersecciones con los modelos de la escena virtual 3D sean lo suficientemente cercanas, y delimitadas por la normal para mejor entre objetos, sería una aproximación tal vez factible, ya sea que se utilice el valor más cercano, o un promedio ponderado sobre valores debajo de un factor definido.

Teoría de variación media entre muestras

Una aproximación alternativa, para reducir el número de muestras, sería muestrear todos los píxeles, pero, reduciendo en medida de lo posible, tantas muestras como sea posible sobre ese pixel, la idea detrás de esta teoría sería calcular la diferencia media del color obtenido entre todo las muestras según una cantidad de muestras definidas, y si esta variación es menor a un factor dado, dar el color hasta ese momento calculado como correcto, y en caso de no ser así, seguir muestreando el píxel hasta que la variación sea menor al factor previamente dicho.

Estas teorías de trabajo futuro, al igual que los algoritmos desarrollados a lo largo del proyecto se basan en la ubicación de los parámetros correctos según la técnica de *upsampling* utilizada, contando con un mayor tiempo para su desarrollo y pruebas, sería interesante observar a que punto podemos llevar la mejora sobre el *path tracing* con las nuevas aproximaciones planteadas.

7 Conclusiones

El objetivo principal de este TFG era la generación de imágenes fotorrealistas con la técnica de *upsampling*, mejorando a su vez, el tiempo de ejecución que tomaría crear la imagen fotorrealista con el *path tracing*, buscando desarrollar una o varias alternativas de algoritmos más eficientes.

Se realizaron multitud de resultados, con diferentes parámetros, para los diversos algoritmos codificados, pudiendo definir cuáles algoritmos parecían funcionar mejor con qué parámetros y hasta que nivel de calidad se podía llegar con ellos, se intentó realizar aproximaciones más avanzadas, de las cuáles sólo se quedó en bases y teoría.

Imágenes fotorrealistas de buena calidad, con bajo número de muestras, si bien, no perfectas fueron conseguidas, y es posible ver cómo incluso superaron a las imágenes referentes del *path tracing*, los experimentos equiparando al *path tracing* también dejan resultados de mejor calidad con una ejecución más eficiente con posibilidades de considerable mejora a trabajo futuro.

Por lo que, se considera, que si bien, existe la posibilidad de seguir explorando técnicas de *upsampling* y la optimización de estas, los objetivos del TFG han sido resueltos con éxito.

8 Bibliografía

Verma, V., & Walia, E. (2010). 3D Rendering-Techniques and challenges. International Journal of Engineering and Technology, 2 (2), 29 – 33.

Watt, A. H. (1999). 3D Computer Graphics (3 Har/Cdr ed.). Addison-Wesley Pub (Sd).

Marques, R., & Santos, L. P. (2010). Instant Global Illumination on the GPU using OptiX.

ElAnalistaDeBits. (2021, 7 noviembre).

Forza Horizon 5 VS Reality | The Beauty of Mexico | Comparison [Vídeo].

YouTube. https://www.youtube.com/watch?v=r1xK6PRk1DU

Dumitrescu, D., & Boiangiu, C. A. (2019). A study of image upsampling and downsampling filters. Computers, 8 (2), 30.

Ferreruela, R. (2007). La visión y el ojo. Apunts Educación Física y Deportes, 88, 8 – 14.

Scratchapixel. (2015, 24 agosto). Global Illumination and Path Tracing. Recuperado 10 de junio de 2022, de https://www.scratchapixel.com/lessons/3d-basic-rendering/global-illumination-path-tracing

HJHORNBECK. (2015, 18 marzo). The Perfect Clamp, pt. 1: Cycles and Light Paths. SINMANTYX. Recuperado 10 de junio de 2022, de

https://sinmantyx.wordpress.com/2015/03/18/perfect-clamp-1/

Farrel, A. J., Norton, B., & Kennedy, D. (2004). Lightpipe daylight simulation modelling using Radiance backward and forward ray tracing methods: a comparison with monitored data for commercial lightpipes in Ireland.

Programador clic. (2020). Path Tracing. Recuperado 10 de junio de 2022, de https://programmerclick.com/article/8671590661/

Goral, C. M., Torrance, K. E., Greenberg, D. P., & Battaile, B. (1984). Modeling the interaction of light between diffuse surfaces. ACM SIGGRAPH computer graphics, 18 (3), 213 – 222.

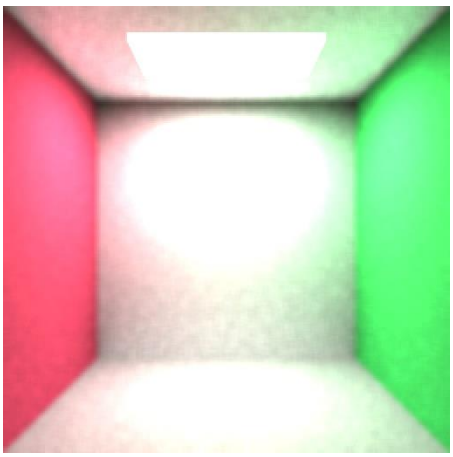
9 Apéndice

En esta sección se agrega material extra del TFG desarrollado a lo largo de lo visto en el documento y que se consideró lo suficientemente importante como para ser mencionado en la memoria, pero que por disposición de información se anexa en esta sección para comodidad del lector.

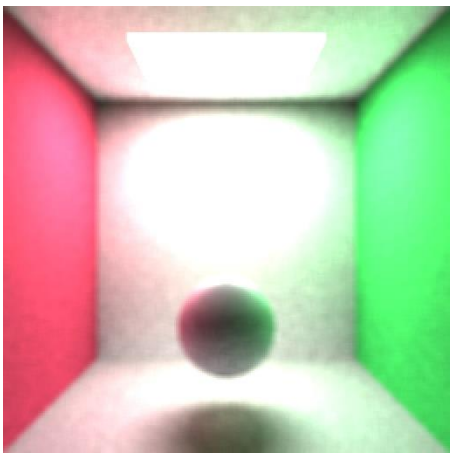
9.1 Imágenes generadas con filtro de 5x5 por las técnicas de *upsampling* para la iluminación global propuestas

9.1.1 Algoritmo *upsampling* basado en el promedio de los vecinos en el espacio imagen (2D)

Factor de reducción 2



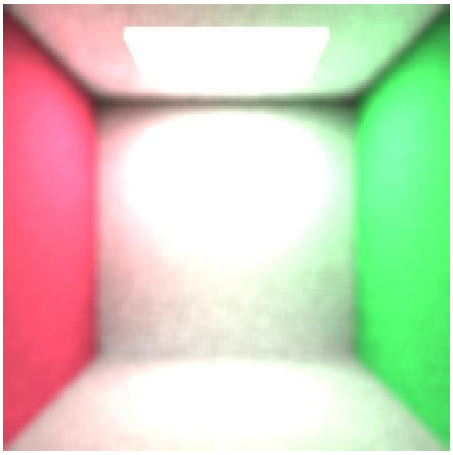
a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 2.



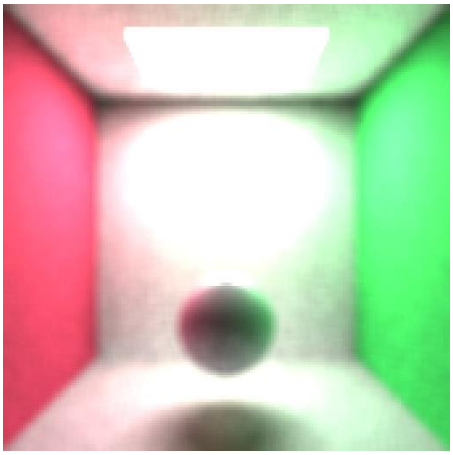
b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 2.

Ilustración 35. *Escenas generadas con el algoritmo de upsampling basado en el promedio de los vecinos en el espacio imagen (2D) con factor de reducción 2.*

Factor de reducción 3



a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 3.

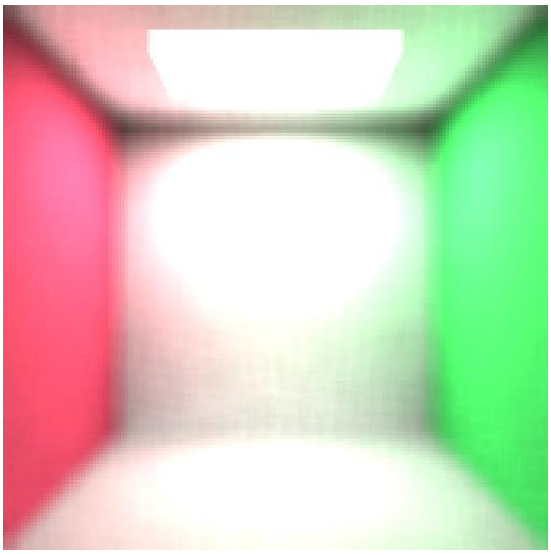


b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 3.

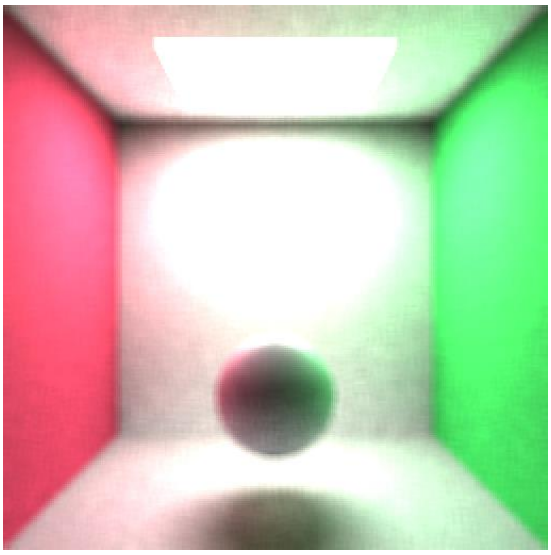
Ilustración 36. *Escenas generadas con el algoritmo de upsampling basado en el promedio de los vecinos en el espacio imagen (2D) con factor de reducción 3.*

9.1.2 Algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio imagen (2D)

Factor de reducción 2



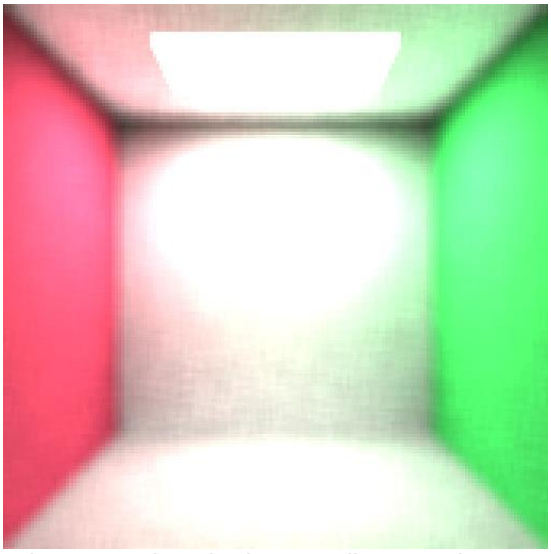
a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 2.



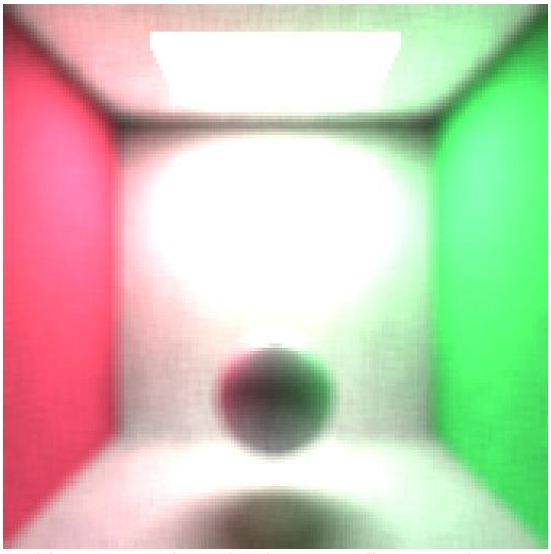
b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 2.

Ilustración 37. *Escenas generadas con el algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D) con factor de reducción 2.*

Factor de reducción 3



a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 3.

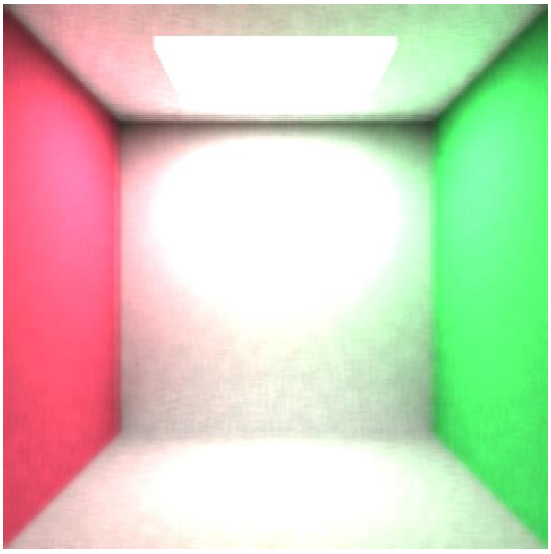


b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 3.

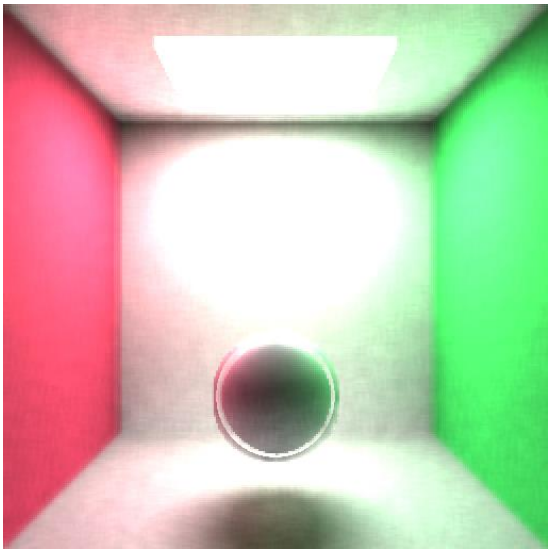
Ilustración 38. *Escenas generadas con el algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio imagen (2D) con factor de reducción 3.*

9.1.3 Algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena (3D)

Factor de reducción 2



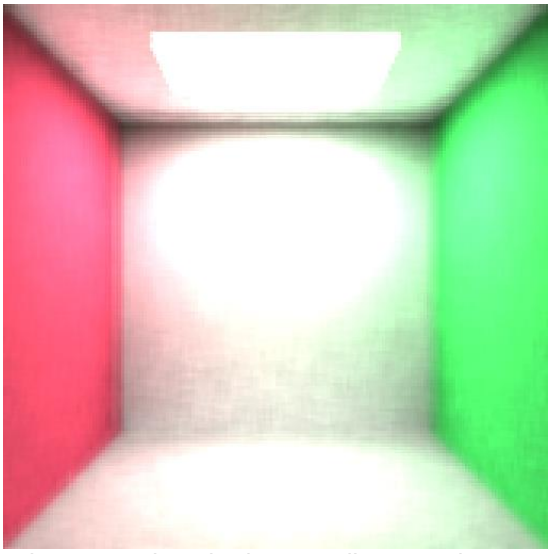
a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 2.



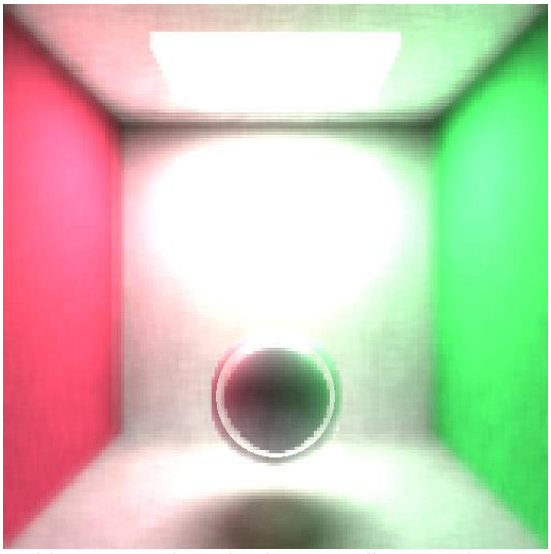
b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 2.

Ilustración 39. *Escenas generadas con el algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D) con factor de reducción 2.*

Factor de reducción 3



a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 3.

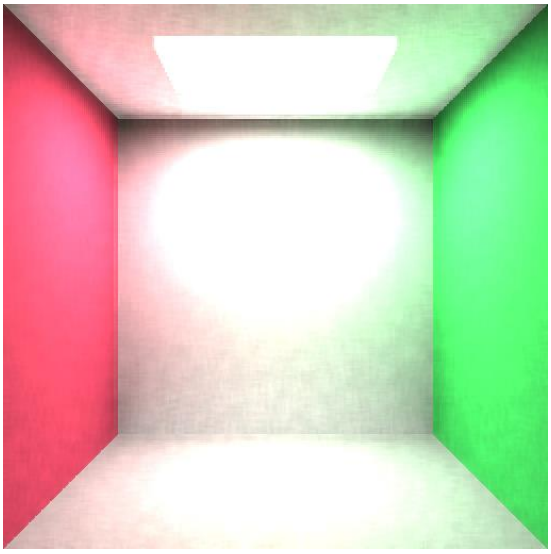


b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 3.

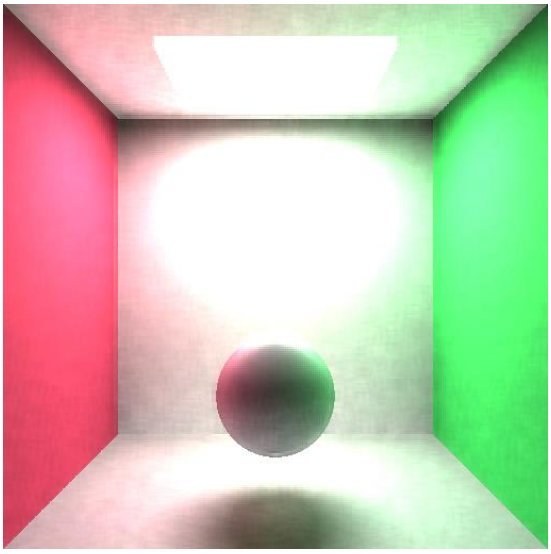
Ilustración 40. *Escenas generadas con el algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio escena (3D) con factor de reducción 3.*

9.1.4 Algoritmo *upsampling* basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D)

Factor de reducción 2



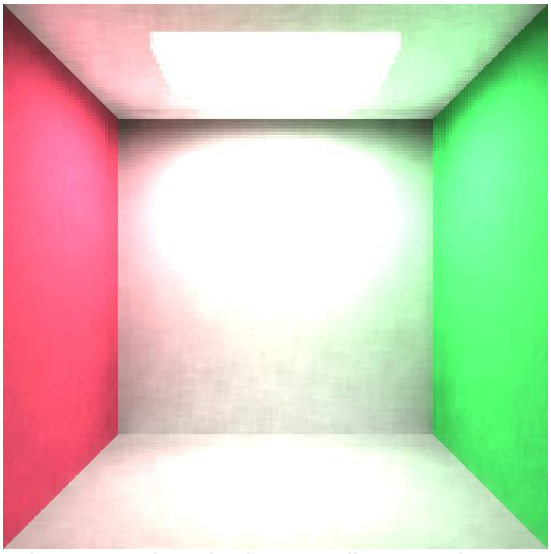
a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 2.



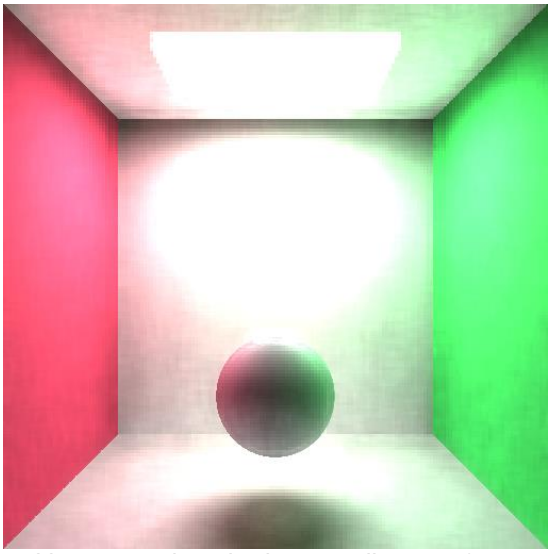
b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 2.

Ilustración 41. *Escenas generadas con el algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) con factor de reducción 2.*

Factor de reducción 3



a) Imagen de caja de Cornell generada con filtro de 5x5, factor de reducción 3.



b) Imagen de caja de Cornell con esfera generada con filtro de 5x5, factor de reducción 3.

Ilustración 42. *Escenas generadas con el algoritmo de upsampling basado en el promedio ponderado de los vecinos en el espacio escena utilizando normales (3D) con factor de reducción 3.*