Presentación

Introducción y objetivos

La holografía es una técnica que permite capturar y reconstruir el campo de ondas completo de la luz generado en una escena determinada. Se utiliza hoy en día para varios propósitos, entre los que se encuentran los sistemas de pantallas tridimensionales (3D). Las pantallas 3D holográficas permiten reproducir fielmente y sin restricciones todas las señales de profundidad visuales naturales conocidas. Algunas de esas cualidades no se pueden conseguir con otras tecnologías como, por ejemplo, cuando se obtiene una representación 3D estéreo.

Dicho fenómeno que también se puede simular generando el holograma mediante un computador (CGH), actividad análoga pero muchísimo más demandante de cálculo, a la conocida generación de imágenes sintéticas por computador.

Sin embargo, uno de los mayores desafíos de CGH's es obtenerlos en tiempos computacionales aceptables.

En este trabajo de final de grado se aborda la generación de hologramas sintéticos, para ello, se han diseñado e implementado:

1. Un trazador de rayos estándar para generar imágenes sintéticas (2D) por computador.
2. Un sistema de generación de hologramas (3D) por computador que hace uso de la técnica de modelado geométrico basada en nubes de puntos y la técnica de trazado de rayos, pero aplicada a múltiples posiciones del ojo.
3. Por último, con objeto de demostrar la validez de los resultados obtenidos, se visualizan los hologramas sintéticos generados a partir de una escena virtual, en un laboratorio de óptica holográfica.

Estado del arte

La generación de imágenes sintéticas por computador o CGI es un campo de la computación gráfica bien estudiado que se utiliza en una gran variedad de áreas, como el diseño, la industria, el cine y los videojuegos.

Consiste, en capturar lo que vería un ojo al observar una escena virtual formada por objetos y fuentes de iluminación.

Una de las técnicas más utilizadas en CGI es el trazado de rayos. Esta técnica simula la interacción de la luz con una escena para sintetizar escenas virtuales.

Se considera una técnica computacionalmente costosa, por lo que es utilizada principalmente para renderización offline con dispositivos computacionales habituales, o en tiempo real utilizando granjas de potentes procesadores y GPUs.

Un trazador de rayos completo ha de simular al menos 7 objetos y fenómenos:

1. Cámaras: El modelo de una cámara determina cómo y desde dónde la escena se observa, incluyendo como una imagen de la escena es recogida por un sensor.

2. Intersecciones rayo-objeto: Es necesario conocer precisamente cuándo y dónde un rayo intersecta un objeto geométrico, además de determinar algunas propiedades del objeto en el punto de intersección.
3. Fuentes de luz: El trazador de rayos ha de modelar la distribución de la luz en la escena.
4. Visibilidad: Se debe poder conocer si una luz determinada deposita energía en un punto de una superficie.
5. Dispersión de la luz en superficies: Cada objeto ha de proveer información sobre como la luz interactúa con la superficie del objeto.
6. Transporte indirecto de luz: La luz puede llegar a la superficie después de rebotar o atravesar otras superficies.
7. Propagación de rayos: Se necesita conocer el comportamiento de la luz mientras atraviesa un espacio, siendo su velocidad constante en el vacío.

Tal y como se ha mencionado anteriormente, se tiene como objetivo generar hologramas sintéticos simulando la captura del frente de ondas de una escena en un plano siendo este proceso más computacionalmente exigente que el CGI y se puede abordar utilizando distintas técnicas.

La técnica de la nube de puntos consiste en discretizar la escena en puntos luminosos y obtener la suma de todas las funciones de dispersión de los puntos en el plano del holograma

Se puede observar en la figura la nube de puntos generada a partir de un barco y la función de dispersión de un punto evaluada en el plano H.

Las principales limitaciones de esta técnica son la falta de soporte de efectos básicos como la oclusión y el sombreado y el alto coste computacional, aunque el algoritmo es altamente paralelizable.

La técnica de la nube de puntos puede ser utilizada en conjunto con el trazado de rayos, solucionando la falta de soporte de los efectos básicos.

La técnica de trazado de rayos ha de ser adaptada o combinada con otras ya que la holografía se basa en ondas y no en rayos de luz.

Por último, en el caso de la propagación de ondas electromagnéticas entre dos planos paralelos, existe la posibilidad de disminuir drásticamente los tiempos de cálculo. En estas condiciones se pueden utilizar los algoritmos de convolución y de transformada de Fourier.

Esto resulta útil a la hora de propagar un holograma almacenado, como se puede ver en la figura.

Proceso de síntesis ...

El proceso de síntesis de escenas virtuales en 3D mediante hologramas digitales se ha dividido en los siguientes pasos:

1. Definición de la escena virtual
2. Generación de imágenes por computador (CGI)
3. Generación de hologramas por computador (CGH)
4. Reconstrucción de la escena

La implementación se ha realizado en C++ y CUDA para la definición de la escena virtual, CGI y CGH y Python para la reconstrucción de la escena.

Definición de la escena virtual

El primer paso para la síntesis de escenas virtuales consiste en definir la escena que se desea producir.

Veamos el siguiente ejemplo: Esta escena está definida por

1. el color del cielo,
2. la posición, el tamaño y el material de las tres esferas,
3. y la posición y color de una fuente de luz puntual.

La geometría de la escena se definirá mediante primitivas definidas matemáticamente. Estas primitivas son: esferas, definidas mediante su radio y la posición espacial de su centro; y triángulos, definidos mediante la posición espacial de sus vértices.

Utilizando la primitiva del triángulo se pueden representar mallas, definidas por una lista de triángulos. El soporte de mallas resulta muy útil ya que la mayoría de modelos 3D se encuentra en este formato.

Una vez definida la geometría de la escena, es necesario aplicar texturas a las primitivas.

1. Material difuso (lambertiano): Este material dispersa la luz siguiendo una distribución independiente al ángulo de incidencia. Segun la ley coseno de Lambert.

---

2. Material metálico: Al contrario que el material difuso, el material metálico refleja la luz en el mismo ángulo en dirección opuesta respecto al ángulo de incidencia. Este efecto produce un reflejo de la misma manera que un espejo. Este material también cuenta con un parámetro que controla la borrosidad del reflejo.

---

3. Material dieléctrico: Este material representa materiales transparentes como agua y cristal. Cuando la luz incide sobre el material, se divide en luz reflejada (como el material metálico) y en luz refractada. La reflectividad se describe según las ecuaciones de Fresnel y la refracción según la ley de Snell. Este material también cuenta con un parámetro que controla el índice de refracción.

Render demostrando los materiales

La última sección de la definición de la escena virtual es la iluminación. La iluminación que se utiliza se puede dividir en dos tipos de fuente: el cielo y fuentes puntuales de luz.

El cielo ilumina de manera uniforme la escena mientras que las fuentes de luz puntuales siguen el modelo de reflexión de Blinn-Phong que describe la forma en la que una superficie refleja la luz como una combinación de la iluminación difusa y la iluminación especular. No se incluye el término ambiental del modelo ya que se utiliza el cielo.

Generación de imágenes por computador

Como se ha mencionado anteriormente, el algoritmo se ha implementado en el lenguaje de programación C++ debido a su alto rendimiento, control sobre conceptos de bajo nivel (como gestión de la memoria) y compatibilidad con CUDA para acelerar mediante GPUs.

En esta figura se puede ver un render de la escena final del libro en el que se ha basado la implementación.

El primer componente del trazador de rayos es la cámara, encargada de lanzar los rayos ya que la propagación desde la fuente de luz hasta la cámara es equivalente a la propagación desde la cámara hasta la fuente de luz. La cámara se basa en una cámara estenopeica (o cámara oscura) sin lente, aunque también es capaz de simular una lente para obtener el efecto de profundidad de campo.

Cada píxel del viewport se corresponde con el de la imagen de salida por lo que se podría hacer un símil con el sensor de la cámara. Cada píxel indica la dirección de un rayo y una vez trazado el rayo se almacena el color resultante en la imagen de salida.

Se puede obtener mayor calidad perceptual al elegir un punto aleatorio dentro del píxel en vez de su centro y se puede reducir el aliasing y aumentar la calidad al mediar el resultado de varias muestras calculadas para el mismo píxel.

El algoritmo de trazado de rayos tiene la siguiente forma

1. Lanzar el rayo
2. Obtener el objeto intersectado más cercano
    1. Si no existe, atenuar el rayo con el color del cielo y terminar.
3. Obtener el punto de intersección, la normal de la superficie y el material del objeto
4. Calcular atenuación según el material (iluminación global)
5. Calcular iluminación de punto de luz
6. Comprobar si el punto es visible para la fuente de luz
7. Calcular iluminación especular y difusa
8. Repetir desde el paso 2 con el rayo dispersado

También se ha creado una interfaz de usuario para agilizar el proceso de desarrollo que permite modificar en tiempo de ejecución distintos parámetros, como los ajustes de renderizado, la posición y orientación de la cámara, la iluminación y los materiales de los objetos.

Generación de hologramas por computador

Una vez implementado el trazador de rayos para la generación de imágenes, se ha modificado para generar hologramas. Las principales modificaciones realizadas han sido el proceso de lanzar rayos y el cambio del cálculo del color del rayo al cálculo de la amplitud y fase.

Para obtener la amplitud y la fase de cada rayo que llega a cada píxel del holograma se ha de calcular respecto a cada punto de la escena a muestrear, siendo los mismos puntos para cada píxel.

Para determinar los puntos de la escena se ha utilizado la técnica de la nube de puntos, según la cual se construye una lista de puntos en las superficies de los objetos de la escena.

Una vez generado el holograma es almacenado en dos archivos, uno contiene la amplitud y el otro la fase. Como el holograma almacenado es bidimensional, se puede almacenar en formatos de imagen como PNG. Estos archivos serán los utilizados posteriormente para la reconstrucción de la escena. Un ejemplo de la representación visual de los archivos se puede observar en la figura.

Reconstrucción de la escena

Para recuperar una imagen la escena a partir del holograma calculado se debe realizar el proceso inverso al definido en la sección anterior; esto es, se debe propagar una onda electromagnética desde el plano del holograma hasta los diferentes planos que constituyen la escena.

Se han utilizado dos procesos para la reconstrucción de la escena: simulación mediante la propagación de ondas electromagnéticas entre dos planos y propagación en el laboratorio mediante un modulador espacial de luz (SLM), y un láser con longitud de onda de 632.8nm.

Como se ha mencionado anteriormente, se puede utilizar los algoritmos de convolución y de transformada de Fourier para reducir considerablemente el tiempo.

La reconstrucción puede hacer uso de la fase almacenada en el holograma, de la amplitud o de ambas. Ya que el SLM del que se dispone en el laboratorio solamente es capaz de modular la fase, ambas reconstrucciones se han llevado a cabo utilizando solamente la información de la fase con el fin de obtener resultados comparables.

Este es el resultado de reconstruir esta escena, enfocada en la esfera mediana, por simulación. Y este es el resultado del laboratorio.

Con distintos valores de z, se puede apreciar el desenfoque de las distintas esferas.

Técnicas de paralelización

La computación paralela es un tipo de computación en la que muchos cálculos o procesos se pueden llevar a cabo simultáneamente.

Existen varias técnicas de paralelización entre las que se encuentra la que se utilizará en este trabajo, el paralelismo de datos. Esta técnica consiste en dividir los datos entre distintos hilos de procesamiento.

Por ejemplo, se podría dividir una imagen en píxeles, y operar a nivel de píxel o en bloques de píxeles.

CPU

Las unidades centrales de procesamiento (CPUs) actuales cuentan con múltiples núcleos, lo que permite la ejecución paralela de múltiples hilos de procesamiento.

Es necesario dividir el trabajo para los diferentes hilos. En el caso de un trazador de rayos, el método más sencillo de distribuir el trabajo es dividiendo la imagen en píxeles y asignando a cada hilo un rango de píxeles sobre los que operar. Este método cuenta con la limitación de que puede darse el caso de que un hilo acabe antes que otro.

Para solucionar este problema se puede introducir el uso de un grupo de hilos (o thread pool, en inglés), al cual se le puede asignar una serie de tareas que los hilos que lo forman ejecutan. Al dividir el trabajo en más tareas que hilos, se soluciona la limitación anterior. Se ha de tener en cuenta que la gestión de los hilos y las tareas tiene un coste computacional.

GPU

Las unidades de procesamiento gráfico son componentes hardware diseñados originalmente para la renderización de gráficos. Sin embargo, debido a su arquitectura masivamente paralela, también son altamente eficaces en la realización de cálculos matemáticos complejos y tareas que pueden beneficiarse del paralelismo masivo.

Las GPUs están compuestas de miles de núcleos más simples y especializados que los de las CPUs, como se puede apreciar en la Figura. Esto permite la ejecución de miles de hilos simultáneamente, comparado con las CPUs que no suelen superar los 256 hilos.

Para poder utilizar una GPU, es necesario el uso de bibliotecas o herramientas especiales que faciliten la programación en GPUs, como CUDA, exclusivo de Nvidia, o OpenCL. Estas herramientas permiten escribir código el cual puede ser ejecutado en la GPU.

En el caso de CUDA, el lenguaje es similar a C++. Por esto y porque la GPU de la que se dispone es Nvidia, se ha escogido CUDA para este trabajo.

CUDA es una plataforma de computación paralela y una API creada por Nvidia que permite a los desarrolladores utilizar la GPU para tareas de computación general.

Proporciona una extensión al lenguaje de programación C++.

En CUDA, el código se divide en tres tipos, definidos al declarar la función que lo contiene. El tipo host (anfitrión) es el código que se ejecuta en la CPU y no se distingue de C++. El tipo device (dispositivo) es el código que se ejecuta en la GPU. Y el tipo kernel es el código que se llama desde la CPU y asigna el trabajo a la GPU.

Implementación

En la Figura se muestra un diagrama que compara el flujo de ejecución del programa en C++ y CUDA, indicando si se ejecuta en la CPU o en la GPU.

La paralelización mediante hilos se ha implementado asignando un rango de líneas a cada hilo, las cuales se calculan y almacenan en memoria compartida.

La paralelización mediante grupos de hilos se ha implementado gracias a una librería. Al grupo se le añade número de inicio, número de fin, el número de bloques en los que dividir el trabajo y una función que, dado un número de línea, la calcula.

Por último, la paralelización en CUDA se ha implementado mediante un kernel. Una vez los datos necesarios están almacenados en memoria compartida o memoria de la GPU, se ejecuta un kernel en bloques suficientes para cubrir el tamaño de la imagen, en el que está definido como calcular un píxel.

Resultados

Para la obtención de los tiempos de este apartado se han utilizado dos ordenadores con las siguientes CPUs y GPUs:

- Ordenador 1:
  - CPU: Intel i7 7700K (8 hilos, 4.4GHz)
  - GPU: Nvidia RTX 3070 (5888 núcleos)
- Ordenador 2:
  - CPU: Intel i7 9900K (16 hilos, 4.7GHz)
  - GPU: Nvidia RTX 2060 (1920 núcleos)

Todos los tiempos se han obtenido al calcular la escena descrita anteriormente con un sensor de 1920 × 1080 píxeles.

El primer resultado obtenido (Figura 26) compara las dos CPUs con distintos números de puntos y el número máximo de hilos menos uno.

Se puede observar una correlación lineal entre tiempo y puntos.

La media de la primera CPU es de 2.85 puntos por segundo, comparado con los 27.95 de la segunda (9.8 veces más rápida).

En la Figura se comparan las dos GPUs de la misma manera.

La media de la primera GPU es de 91.26 puntos por segundo y la de la segunda es de 50.14, siendo la primera 1.82 veces más rápida respecto a la segunda y 3.27 veces respecto a la CPU más rápida.

Finalmente se han obtenido resultados para distintos números de hilos de las CPUs. Como se puede ver en la Figura, el aumento del número de hilos conlleva a un aumento del rendimiento no lineal, esto puede deberse al coste de crear y gestionar hilos, la contención por recursos compartidos y el aumento de los fallos de caché y de los conflictos de acceso a la memoria.

En la Figura se comparan los resultados tanto de CPUs como de GPUs en una misma gráfica con escala logarítmica en ambos ejes. Se incluye un resultado adicional para la GPU 2060 (455 234 puntos en 9006 segundos (o 2.5 horas)), que también sigue la correlación lineal.

El número de puntos determina la calidad visual del holograma generado. En la Figura 30 se pueden observar distintos resultados con distintos números de puntos y se puede ver que, a mayor cantidad de puntos, mayor calidad visual hasta que se llega a un punto en el que no se aprecia el aumento de calidad.

Conclusiones

Las de la presentación.