Proyecto 2 Análisis de datos

Carlos Badilla Mayorga, Carlos Calderón Salazar
Instituto Tecnológico de Costa Rica, Cartago
Email 1: catobm18200@gmail.com
Email 2: carlosdaniel.cdcs@estudiantec.cr

***Abstract*—In this document we are going to carry out the solution of project number 2 of the algorithm analysis course, the problem that was posed to us was the development of a sonar based on the physics of echolocation. Echolocation is the science of locating objects using sound waves that some animals such as bats and dolphins use to be able to locate themselves within the space that surrounds them or to know where they can find food. For this project obstacles drawn in the tool called "paint" are used, geometric figures and lines were placed so that the waves can produce rebounds and the same logic can be followed. Another way of looking at this project is to see it as if it were a simulation of an ultrasound, having the image to be calculated, rays are reproduced to simulate the sound waves and depending on the time with which they return, after colliding with an obstacle a intensity and the pixel that was calculated is drawn. All the code that was carried out for the project was developed in pyhton and can be found in the github repository at the following link: url https://github.com/cabadilla/analisis.git/**

I. Introducción

En este documento vamos a llevar a cabo la solución del proyecto número 2 del curso de análisis de algoritmos, el problema que se nos planteó fue la elaboración de un sonar basado en la física de la ecolocalización. La ecolocalización es la ciencia de localizar objetos mediante ondas sonoras que algunos animales como los murciélagos y los delfines utilizan para poder ubicarse dentro del espacio que los rodean o saber dónde pueden encontrar alimento. Para este proyecto se utilizan obstáculos dibujados en la herramienta llamada "paint", se colocaron figuras geométricas y líneas para que las ondas puedan producir rebotes y se pueda seguir la misma lógica. Otra forma de ver este proyecto es verlo como si fuera una simulación de un ultrasonido, teniendo la imagen a calcular, se reproducen rayos para simular las ondas de sonido y dependiendo del tiempo con el que vuelvan, luego de chocar con algún obstáculo se calcula una intensidad y se dibuja el pixel que se calculó. Todo el código que se llevo a cabo para el proyecto fue elaborado en pyhton y se encuentra en el repositorio en github del siguiente enlace: https://github.com/cabadilla/analisis.git/

II. Trabajo Relacionado

Dentro de los trabajos que utilizamos para poder desarrollar este proyecto podemos basarnos en el libro de "Álgebra, trigonometría y geometría analítica" en específico el Cap 9 que habla sobre todas las propiedades del círculo unitario, de ahí se basó el algoritmo para determinar las distancias y poder mandar los rayos dentro del proyecto. Se utilizan todas las fórmulas en radianes por lo que las fórmulas se calculan usando senos y cosenos. Otro de los trabajos en los que decidimos apoyarnos fue en el artículo llamado "Ecolocalización (una visión a los quirópteros)" este artículo con el fin de interiorizar un poco más sobre lo que esta definición conlleva, visualizar de una forma más amplica cómo funciona y cómo podría ser implementada dentro del proyecto. Por último, utilizamos el capítulo 2 titulado "Fisicas del sonido", en este nos basamos para poder conseguir una mayor precisión en los algoritmos de intensidad y en los reflejos de las ondas sonoras, además de colocar los rebotes lo más parecidos a la realidad.

III. Métodos

El proyecto consiste de varias partes, primero se procederá a explicar las librerias que se importaron para el desarrollo del proyecto, esto con el fin de al llegar a la implementación saber de dónde provienen. Entre las librerias utilizadas estuvieron: pygame que fue nuestro motor gráfico para todo el desarrollo visual e interacción con el usuario, la librería de math que fue la que nos dio todas las bases y funciones trigonométricas, la librería de random que se utilizó para implementar el Monte Carlo dentro del proyecto para el manejo de números aleatorios, la librería de threading que enlaza los hilos y permite el uso de estos mismos dentro del proyecto y por último Pillow que nos da todo el manejo de imagenes ya sea para realizar una nueva imagen, leer e iterar por pixeles o anexar imagenes. El flujo de control se basa en crear toda la interfaz de python con la imagen que se va a ir formando por los rayos acústicos y a la par tenemos el manejo del sonar, ya sea para rotarlo o re-posicionarlo, seguido de esto el programa va a utilizar un fabric que empezará a crear los hilos de los rayos principales y a la hora de lanzarlos junto a estos van a viajar 5 rayos secundarios que estos mismos tienen una intensidad más baja como se explica dentro de lo que es la ecolocalización, dentro de las funciones de desplazarse en el momento que llegue a un obstáculo se almacenará el tiempo recorrido y luego se crearán los rebotes necesarios y estos viajarán con la misma lógica; el hilo muere en el momento que se realicen los 3 rebotes que estaban planeados que se hicieran o en caso de que el rayo salga del mapeo de la imagen. Al retornar la lista de los tiempos en los que se topó con un obstáculo utilizamos las funciones trigonométricas para calcular la distancia desde el sonar hasta el obstáculo y se pintan dependiendo de lo que retorne la función de intensidad que siempre intenta diferenciar los rebotes, los rayos secundarios y los primarios. Al calcular la intensidad y la distancia en la que se encuentra si el tiempo

es de un rayo principal este se coloca en esa misma dirección con una intensidad mucho mayor para diferenciar entre los primarios y secundarios, en caso de que los tiempos sean de los secundarios, estos se dibujan en la misma línea en la que iba direccionado el rayo principal y esto es lo que produce el efecto llamado como ruido en las imágenes. Todos estos rayos están compuestos por hilos para que se puedan manejar aproximadamente 6 rayos por ciclo de ejecución, siendo como se explicó uno principal y 5 secundarios.

IV. Análisis de resultados

Para el analisis de resultados tomamos el tiempo de ejecución que se demoraba el programa y la cantidad de rayos primarios que se habían creado en ese momento, así como la calidad de la imagen que mostraba hasta dicho tiempo, primero medimos cuanto tiempo se demora el programa al lanzar n rayos primarios, al hacer esto nos dimos cuenta que el rendimiento mostrado siempre es lineal, entre mayor cantidad de rayos haya mayor tiempo de ejecución habrá, sin embargo este crecimiento se mantiene constante, es decir la gráfica no tiene muchos picos ni bajones, siempre se mantiene lineal, la grafica de dicho análisis se puede ver en la figura 1. .

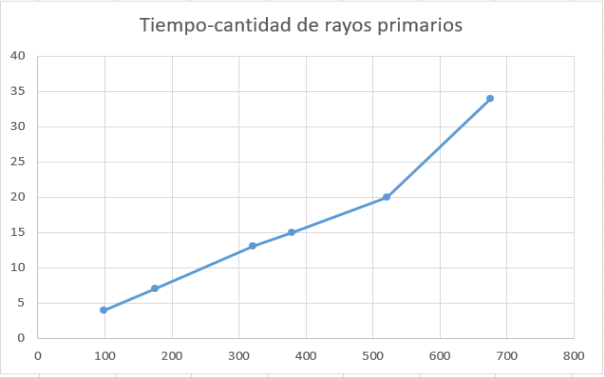

Fig. 1: Grafico tiempo-rayos primarios

Al terminar de analizar como se comportaba el tiempo con respecto a la cantidad de rayos primarios procedimos a analizar la calidad de la imagen mostrada hasta un determinado tiempo de ejecución y una determinada cantidad de rayos tanto primarios como secundarios. Llevamos a cabo varias pruebas y notamos que generalmente a partir de los 13 segundos de exposición a los rayos, las imagenes ya empezaban a tomar forma, es decir se empezaban a distinguir con respecto al resto del ruido, a los 13 segundos de ejecución ya se habían lanzado 338 rayos primarios y 1690 rayos secundarios, la imagen formada a los 13 segundos de ejecución se puede apreciar en la figura 2. .

Al analizar la imagen podemos notar que para este punto a pesar de que no es la imagen que buscamos, ya se pueden distinguir correctamente las formas de la imagen original, por lo que en 13 segundos ya podemos apreciar el correcto funcionamiento del programa sin embargo antes de los 13 segundos no se puede apreciar más que ruido en la imagen, dicho ruido lo podemos ver en la figura 3 la cual nos muestra

Fig. 2: Imagen 13 segundos de ejecución

la imagen formado a los 8 segundos de ejecución con 231 rayos primarios y 1155 rayos secundarios lanzados. .

Fig. 3: Imagen 8 segundos de ejecución

Como notamos a los 8 segundos las formas son practicamente indistinguibles, sin embargo la cantidad de rayos principales enviados no es tan grande y aun así ya nos muestra una cantidad suficiente de pixeles pintados, si nos vamos a un caso totalmente contrario y dejamos las formas por mas de un minuto en exposición a los rayos y además a esto le agregamos la rotación para que dibuje completamente la escena, los resultados dados son muy prometedores ya que se puede apreciar completamente la escena además de las diferencias entre la intensidad de los pixeles dibujados, es decir la imagen luce completamente como un ultrasonido. El tiempo de esta imagen fue de 120 segundos y la cantidad de rayos principales fue de 2888 y de secundarios 14440 Dicha imagen la podemos apreciar en la figura 4. . A pesar de que la cantidad de rayos utilizados para formar la imagen buscada fue más grande de lo que se esperaba, ya que si contamos los secundarios así como los primarios ambos superan los 15 000 rayos, aún así el tiempo que se demoró en mostrar dicha imagen no fue tan grande como se esperaba. Como resultado se obtuvo la imagen deseada ya que el objetivo era mostrar la imagen lo más parecido a un ultrasonido y así se consiguió. Para este punto podríamos dejar el programa corriendo el tiempo que quisieramos, pero el resultado va a ser el mismo, lo que cambiaría es que a medida que pase el tiempo la imagen

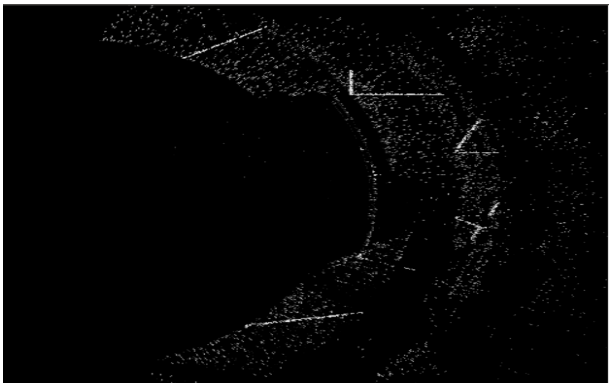

Fig. 4: Imagen mas de un minuto de ejecución

va a ir aumentando en ruido y los pixeles que se pintan debido a los rayos secundarios posiblemente van a cubrir gran parte de la escena pero con una intensidad menor que los rayos secundarios.

V. Conclusiones

Al aplicar conceptos como lo son el uso del método de Monte Carlo, se entiende que a veces la mejor manera de llegar a una solución es a traves de métodos aleatorios, por ejemplo con el uso de algoritmos probabilísticos se pueden llegar a soluciones de manera más rápida, como por ejemplo en la manera que se van lanzando los rayos, la cual es totalmente aleatoria, e incluso así se consigue formar de manera correcta la imagen buscada. Además se comprende de una manera correcta como funcionan los pixeles en un computador y como podemos trabajar con ellos, como en nuestro caso que lo hicimos a través de la biblioteca pillow con python. También se comprende el como usar un algoritmo hecho en un lenguaje de programación, para simular procesos físicos como lo son la propagación de ondas de sonido o la ecolocación que es utilizada por murcielagos o ballenas, esto a través de algoritmos como el raytracer o el pathtracer. Así como la utilización de métodos matemáticos como base para la elaboración de dichos algoritmos como en nuestro caso lo fue el uso de la trigonómetria para llevar a cabo los cálculos de los ángulos hacia donde se dirigían los rayos.

References

[1] . G. Zill and J. M. Dewar, Álgebra, trigonometría y geometría analítica, 3ra ed., McGrawHill, Ed. Mexico: DF, 2012

[2] herya. (2013) Echolocation (a vision to bats) on SciELO [Online]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-33642013000100002

[3] 2012) Fisicas del sonido on InfoMed. [Online]. Available: http://www.sld.cu/galerias/pdf/sitios/rehabilitacion-logo/fisicas_del_sonido.pdf