

INFORME SOBRE ALGORITMOS DE ENVOLVENTE CONVEXA VISTOS EN GEOMETRIA COMPUTACIONAL

REPORT ON CONVEXA ENVIRONMENTAL ALGORITHMS SEEN IN GEOMETRY COMPUTATIONAL

Sandra Milena Caicedo, Carlos Andres López

Maestría en ingeniería de sistemas y computación, Universidad Tecnológica de Pereira, Pereira, Colombia

Correo-e: scaicedo@utp.edu.co, asrael@utp.edu.co

Resumen— Existen diferentes metodologías para hallar el envoltorio convexo de un conjunto finito de puntos en un plano que forman un polígono, en este informe evaluaremos el desempeño de los algoritmos Quick hull, Gift wrapping y Graham, los cuales sirven para este fin; estos algoritmos son útiles para la solución de diferentes problemas en el mundo real

Palabras clave— Envoltorio Convexo, Gift Wrapping, Graham Scan, Quick Hull.

Abstract— There are different methodologies to find the convex envolved in a finite set of points in a plane that form a polygon. in this report we will evaluate the performance of the algorithms Quick hull, Gift wrapping, Graham, which serve for this purpose; these algorithms are useful for solving different problems in the real world

Key Word — Convex Envelope, Gift Wrapping, Graham Scan, Quick Hull,

I. INTRODUCCIÓN

El convex hull es una pequeña región convexa que incluye un grupo de puntos específicos, El borde o frontera de los mismos es un polígono cerrado convexo; el convex hull puede ser visto como una banda elástica alrededor de los puntos de la envoltorio, y los otros puntos como si estuvieran dentro de esta banda elástica; La envoltorio convexa juega un papel central en el campo de la geometría computacional. Este concepto geométrico ofrece aplicaciones prácticas en muchas áreas ya que puede ser usada computacionalmente para fines tales como procesamiento de imágenes, reconocimiento de patrones, sistemas de información geográfica, automatización de diseño e investigación de operaciones entre otros. [1][3].

II. CONTENIDO

Vamos a describir cada uno de los Algoritmos

Algoritmo Quick-Hull:

Es un algoritmo rápido que tarda en tiempo de cómputo $O(n \log n)$, para encontrar la envoltorio y en el peor de los casos $O(n^2)$.

Una idea muy efectiva para acelerar el proceso de cálculo de la envoltorio convexa, consiste en desechar la mayor cantidad de puntos interiores, pues éstos no califican para la envoltorio.

Como funciona:

Iniciamos el proceso dividiendo al conjunto S en dos partes casi iguales, mediante una recta horizontal L . Denotamos las partes por S_1 y S_2 . Tanto S_1 como S_2 contienen un gran triángulo cuyos vértices forman parte de la envoltorio convexa de S_1 (resp. de S_2). Entonces este triángulo será una primera aproximación de la envoltorio convexa de S_1 (resp. de S_2). Ver la figura.

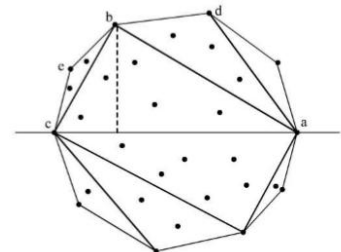


Figura 1. Quick Hull

El punto c (respec. a) es aquel cuya coordenada X es mínima (respec. máxima). En la figura la recta L divide a S en dos partes, una arriba y la otra hacia abajo; S_1 y S_2 . El punto b es el punto de S_1 más alejado de la recta L en la dirección perpendicular a L . El triángulo abc es una primera aproximación de la envoltorio convexa de S_1 . Los puntos interiores los desechamos.

El punto d es el más alejado de la recta ba dentro de S_1
El punto e es el más alejado de la recta bc dentro de S_1

Luego la poligonal $adbce$ es otra aproximación a la envoltorio convexa de S_1 . Como se ve, después de aplicar recursivamente este algoritmo, un número finito de pasos, llegamos a la envoltorio convexa de S_1 y S_2 . El paso final

consiste en concatenar o unir las dos envolventes para obtener la envolvente de S . [3]

Algoritmo Gift Wrapping

También es conocido como Marcha de Jarvis,; tiene costo de $O(nh)$, donde n es el número de puntos y h es el número de puntos en la envolvente convexa. Su rendimiento real en comparación con otros algoritmos de esta clase es favorable cuando n es pequeño o se espera que h sea muy pequeño con respecto a n . En general, puede ser superado por otros algoritmos.

Como funciona:

Supongamos que conocemos un lado e de la envolvente, el cual yace en posición horizontal en la parte de abajo y cuyo extremo derecho es igual al punto x . Sabemos que x debe estar conectado con el siguiente punto de la envolvente, cuando este polígono se recorre en sentido contrario a las agujas del reloj. Llamaremos a este punto hipotético y . La pregunta es: ¿Cómo hacemos para calcular y ?

La idea consiste en trazar una línea L en el plano que tiene un extremo pivote en x , y la cual hacemos girar en sentido anti horario, observando el ángulo que forma con el lado e . Entonces el primer punto de S que toca esta línea es y . Este paso se lleva a cabo considerando todas las rectas del tipo Lx con $s \in S$. Podemos entonces comparar las pendientes de todas ellas y tomamos el punto y en S tal que la pendiente de Lxy sea mínima. (ver la figura)

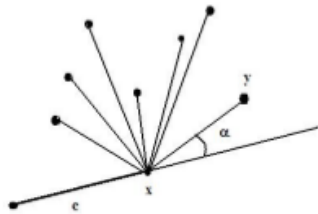


Figura 2. Gift Wrapping

En realidad, en la etapa de implementación del algoritmo, no es necesario calcular en ningún momento las pendientes de estas rectas. Calcular ángulos requiere usar aritmética de números reales, funciones trigonométricas y por lo tanto caer en errores de redondeo y otro tipo de problemas que debilitan a los algoritmos. Una forma bastante delicada de hacer esto, es comparar las pendientes usando nuestra fórmula de área signada para un triángulo. Es fácil ver entonces que la recta xa tiene menor pendiente que la recta xb si y sólo si el área del triángulo $\Delta(xba)$ es positiva.

A medida que vamos avanzando, vamos caminando sobre todos los puntos de la envolvente y por esto se llama también a este algoritmo Marcha de Jarvis. Al llegar a la parte más alta

del recorrido, los ángulos empiezan a ser negativos y los medimos de arriba hacia abajo. Después que la línea de barrido ha hecho un giro completo de 3600 volvemos a la posición inicial y se ha completado el algoritmo. Finalmente, para hallar el primer punto x en este proceso, ordenamos todos los puntos de S de acuerdo a su coordenada X , y tomamos el mínimo. Esto tiene un costo de $O(n \log n)$

Algoritmo Graham Scan

Es uno de los algoritmos utilizados para calcular la envolvente convexa. Requiere en tiempo de computo $O(n)$ y en el peor caso de $O(n \log n)$ (es decir, $n \log_2 n$) debido a un paso de clasificación inicial que ordena los puntos en coordenadas cartesianas]. Graham Scan verifica regularmente la lista ordenada de puntos y elimina la envolvente convexa de puntos extremos. [1]

Como funciona:

Tomamos un punto sobre la envolvente y ordenamos en forma radial todos los puntos de S . Al hacer el recorrido en este orden vamos conectando con lados los puntos adyacentes y de esta manera vamos construyendo un polígono. Cuando un punto sea del tipo reflex, entonces lo eliminamos.

Para iniciar el algoritmo elegimos el primer punto de S , denotado por p_0 , como el punto de mas baja altura. En caso de empate, se toma el del extremo derecho. Luego ordenamos todos los puntos de S en el sentido contrario a las agujas del reloj de acuerdo al ángulo que forma en el rayo que parte de p_0 y termina en el punto. Ver figura: Entonces p_1 es un punto de la envolvente, al igual que p_2 , pues $p_0p_1p_2$ es un giro a la izquierda.

Continuando de esta manera, podemos decir que p_3 es también de la envolvente, pues $p_1p_2p_3$ es un giro a la izquierda. Sin embargo, en el siguiente paso vemos que $p_2p_3p_4$ es un giro a la derecha. Por lo tanto eliminamos el punto p_3 y nos quedamos con los puntos p_0, p_1, p_2, p_4 . Para el paso siguiente vemos que $p_2p_4p_5$ es un giro a la derecha y por lo tanto eliminamos al punto p_4 y nos quedamos con p_5 . Continuando de esta manera, después de dar un recorrido completo anti horario, llegamos al punto de inicio p_0

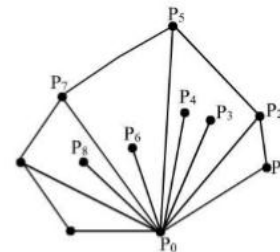


Figura 3. Graham Scan

La pila (stack) es una estructura de datos muy usada que permite guardar objetos como número, puntos, figuras, etc. Comenzamos a almacenar los objetos en el fondo (Bottom) de la pila, colocándolos unos encima de otros. El último objeto almacenado se encuentra en el tope (Top). Con una pila S se manejan dos operaciones básicas

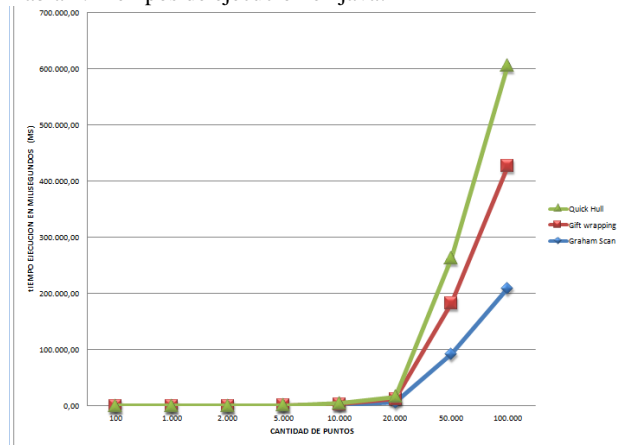
1. PUSH (p,S), la cual inserta un objeto p en el tope de la pila
2. POP (S), mediante la cual se elimina el objeto que se encuentra en el tope de la pila

Implementación Prueba de los algoritmos, los tiempos son en ms

TABLA I Comparativo en Java en ms

PUNTOS	Quick Hull	Gift Wrapping	Graham Scan
100	1,1	18,8	2,0
1000	11,8	27,5	17,7
2000	36,6	116,1	48,2
5000	351,7	394	377
10000	1.590,7	1.484,5	1.541,9
20000	3.772,1	5.689	6.736,3
50000	79.446,7	90.943,1	92.470,8
100000	179.562,4	217.274,3	209.456,9

Tabla 1. Tiempos de ejecución en java.



III. CONCLUSIONES

Los algoritmos se corrieron 50 veces cada uno de ellos con (100, 1.000, 2.000, 5.000, 10.000 y 20.000 puntos), los 50.000 y 100.000 puntos fueron ejecutados 10 veces por cada uno de los algoritmos debido a que la ejecución de estos tiene un alto costo computación.

Se realizaron pruebas en lenguaje de programación Java, las pruebas se ejecutaron en una máquina con las siguientes características (Procesador AMD A10-8700P, RAM 16 GB, Sistema Operativo Windows 10).

Como se puede evidenciar en la grafica y en la tabla de puntos, el algoritmo que resulto más eficiente fue el Graham Scan con menores tiempos de ejecución, seguido del algoritmo Gift Wrapping y por último fue el Quick Hull

RECOMENDACIONES

Los algoritmos se ejecutaron sin la parte grafica y sin el cálculo del área para que esta no afectara el tiempo de ejecución.

Es recomendable, mantener un mínimo de procesos en ejecución en el procesador porque esto afecta los tiempos del algoritmo.

REFERENCIAS

[1] Muhammad Sharif, Safdar Khan, Sadaf Jameel Khan, Mudassar Raza, An Algorithm to Find Convex Hull Based on Binary Tree, December 2009.

[2] Maher M. Atwah , Johnnie W. Baker and Selim Akl, An Associative Implementation Of Graham's Convex Hull Algorithm, January 1995.

[3] Rivero F, Geometría Computacional, Venezuela

[4]Borgwardt K. H, Average Complexity of a Gift-Wrapping Algorithm for Determining the Convex Hull of Randomly Given Points, 1997.