# **Estructura de algoritmo detector de posibles colisiones para un grupo de abejas**

| Jairo Alonso Carvajal Ochoa  Universidad Eafit  Colombia  jacarvajao@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |  |
| --- | --- | --- |

# RESUMEN

La detención de colisiones es un problema esencial en muchas aplicaciones de la computación gráfica, realidad virtual y de la robótica. Su principal reto consiste en el diseño de algoritmos rápidos y eficientes para la detención de interferencias entre dos puntos en un plano específico. Los algoritmos más eficientes de este tipo esta diseñados en estructuras jerárquicas incluyendo árboles basados en S-Bounds y representaciones basada en BSP estos métodos llevan a cabo muy bien las pruebas de rechazos cuando los objetos están suficientemente separados, permitiendo un ahorro significativo de tiempo de ejecución de la detección de objetos. Con estos algoritmos se hace la vida más fácil a la hora de ahorrar poder de computo.

# **1. INTRODUCCIÓN**

En este trabajo se pretende dar a conocer problemas algorítmicos que nos llevaran a posibles soluciones sobre la problemática que se enfrenta en el desarrollo de tecnología para la detención de colisiones ya sea en el área de la robótica, para la construcción y la coordinación de poblaciones de abejas, hormigas robóticas o en el área de los videos juegos. En este documento nos referiremos especialmente a la colisión entre abejas robóticas.

Las abejas intervienen en la reproducción de muchas especies vegetales, siendo esta parte fundamental en la vida vegetal. Debido al uso de pesticidas y herbicidas se ha visto reducida su población de manera preocupante, lo que ha llevado a la creación de abejas robóticas que suplan esta tarea en la naturaleza.

# **2. PROBLEMA**

Las abejas intuitivamente mantienen cierta distancia entre ellas para evitar colisiones, con las abejas robóticas se espera que ocurra lo mismo por medio de la implementación de algoritmos. Resolviendo este problema se esperaría que las abejas robóticas puedan polinizar de manera efectiva y sin riesgos los cultivos.

## **3. TRABAJOS RELACIONADOS**

**3.1**“Con el auge de los drones, aumenta el riesgo de colisión con aviones” .

# Los drones cada vez son más utilizados, tanto por civiles como por militares, el riesgo de que causen daños a los aviones se hace cada vez mayor, debido a su mal uso o el error humano. Una posible solución a esto sería que el dron posea un ángulo de 360º y un radio de la distancia que parezca más inofensiva, y que al detectar que esa barrera fue traspasada el dron tome una ruta alternativa alejándose del posible obstáculo.

**3.1“Colisiones con el tren, un problema en crecimiento”.**

# Debido a la imprudencia de los conductores la colisión con trenes se hace cada vez más grande. Con la ayuda de algoritmos que sean implementados en los trenes se podrían evitar futuras colisiones.

**3.2 “El 'coche abeja' que evita colisiones”.**

Este robot permite evitar obstáculos, pero, aún no está terminado del todo. Puesto que esté sólo permite moverse en dos direcciones y en el ángulo que le permitan sus ruedas, no puede moverse en diagonal ni hacía arroba y abajo.

**3.3 “Siguiendo la pista a un avión: qué tecnologías se usan y qué problemas hay”**

Cuando el avión se encuentra en el océano su cobertura se hace mucho menor y su comunicación con el centro de control se vuelve problemática. Esto puede ocasionar accidentes en el vuelo. Por tal razón se buscan soluciones que sean económicas y eficientes para resolver este problema.

**4. COLAS COMO PRIMERA IDEA DE ESTRUCTURA DISEÑADA.**

Para esta primera idea de desarrollo, se propuso y se implemento unas pruebas con las estructuras de colas.

Consiste en comparar dato por dato, ingresando todas las coordenadas de las abejas en una cola y en otra segunda se creara para generar un mecanismos de recursión donde hay un método que recibe 2 parámetros cola 1, cola 2, cuando se saque la primera abeja, y se compare con cada una, se van ingresando a la segunda cola para tener el proceso de comparación recursivo sin dejar de perder alguna coordenada.

**4.1 Operación de la estructura de datos.**

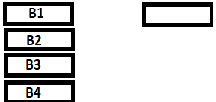


Ilustración 1

Mecanismo recursivo donde se saca la primera abeja y se compara con el resto.

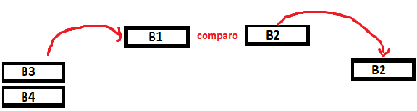


Ilustración 2

Las que ya se compraron con la primera se ingresan a la cola 2.

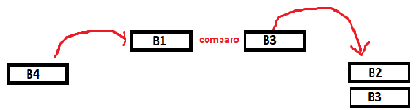


Ilustración 3

En este caso se completó sin perderse ningún dato.

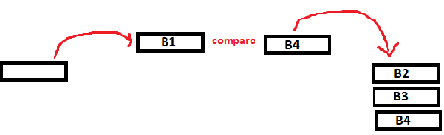


Ilustración 4

Después de que la primera cola no contenga ninguna abeja se inicia proceso recursivo y es pasar la segunda como primer parámetro ya el nuevo dato que se compara es la segunda abeja y así sucesivamente.

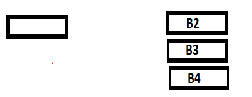


Ilustración 5

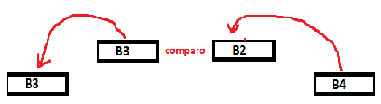


Ilustración 6

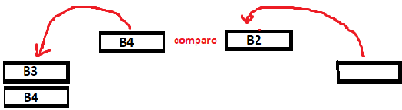


Ilustración 7

Este proceso se seguirá ejecutando hasta que cada una de las abejas se compare con el resto.

**4.2 Criterios de diseño de la estructura.**

Para esta primera estructura se tenían dos secciones importantes, insertar los datos y sacarlos, empezamos a indagar y leer sobre que estructura era mas optima para que se pudieran cumplir las dos secciones anteriores en el menor tiempo posible, de todo el filtro que se hizo se eligieron 3 estructuras, los arreglo, las pilas y las colas.

* En un arreglo se van insertando los datos, posición por posición, este proceso tiene una complejidad O(n), y para sacar los datos, simplemente es ir a cada posición y obtener lo que se encontrara allí, la cosas es que cuando se va y averigua por un dato que se encuentra en una posición x, solamente se muestra el dato, pero el espacio queda ahí. este proceso también tiene una complejidad O(n).
* En una pila se va insertando los datos apilados uno encima del otro, este proceso tiene una complejidad asintótica O(1) por lo cual el algoritmo mejorara con respecto a los arreglos, para sacar los datos simplemente no se necesita ir a una posición en especifica si no que simplemente va sacando el dato que ingreso de ultimas, y se va eliminado cada vez que se saca, este proceso tiene una complejidad O(1)
* Este tiene la misma complejidad y funcionalidad que la pila lo que cambia es la forma de como sale los datos, el primero que ingresa es el primero que sale.

Después de haber mirado ciertas diferencias entre algunas de las estructuras se planteo que para la necesidad inicial se usaría las colas, ya que primero tiene una complejidad de inserción y de búsqueda O(1), dos se facilita el proceso de recursión permitiendo utilizar 2 colas y no tener que crear cada vez que se termine de comparar dato por dato y por otro lados los datos salen en orden teniendo un control mejor de ellos.

**4.3 Análisis de la complejidad del algoritmo.**

|  |  |
| --- | --- |
| **Método** | **Complejidad** |
| Sacar los datos del archivo | O(n) |
| Insertar los datos a la pila | O(1) |
| Comparar datos | O() |

La complejidad mayor el del algoritmo es de O()

**4.4 Prueba del algoritmo con diferentes datos procesados para calcular el tiempo de ejecución.**

|  |  |
| --- | --- |
| **Cantidad de abejas** | **Tiempo en milisegundos** |
|  | 10 |
| 10 | 12 |
| 15 | 13 |
| 100 | 30 |
| 150 | 48 |
| 1000 | 169 |
| 1500 | 288 |
| Solo es capaz de procesar hasta 1500 abejas |  |

**4.5 Prueba del algoritmo con diferentes datos procesados para calcular el consumo de memoria.**

|  |  |
| --- | --- |
| **Cantidad de abejas** | **Memoria en MB** |
| 4 | 0.576 |
| 10 | 0.576 |
| 15 | 0.576 |
| 100 | 1.2087 |
| 150 | 1.271 |
| 1000 | 13.264 |
| 1500 | 0.395 |
| Solo es capaz de procesar hasta 150 abejas |  |

**4.6. Análisis.**

Se puede observar que cada vez que aumente el la cantidad de abejas o datos, el tiempo de ejecución será mucho mayor, ya que se tiene una complejidad asintótica n\*n o O() y lo peor después de cierta cantidad de datos, en este caso después de 15000 abejas el algoritmo empieza a tener problemas.

**5. OCTREE COMO IDEA FINAL DE DESARROLLO.**

Al hacer el análisis y ejecución de la estructura anterior se encontraron varios problemas y resulto que el algoritmo quedó con una complejidad O(), esto no es nada òptimo si se quiere evaluar los datos y verificar qué abejas van a colisionar y poder resolver esto en el menor tiempo posible.

Uno de los problemas es que el proceso se limita a un máximo de 1500 abejas y la idea es procesar al menos 10.000.

Para llegar a este valor o más, se inició con una nueva investigación de posibles estructuras optimas, encontrándonos con frases como “Dividí y vencerás” (Julio César), una frase que dice mucho para resolver este tipo de problemas y nos da mas claridad de cómo se puede construir un algoritmo más optimo, en la investigación se llegó a la conclusión que la estructura más adecuada es octree.

Un octree o árbol octal es una estructura en "árbol" de datos en la cual cada nodo interno tiene exactamente 8 "hijos". Las estructuras octree se usó para particionar recursivamente el espacio tridimensional creando zonas o cubos con la distancia en la que las abejas podrían estar en colisión, esto nos permite filtrar e ingresar las abejas a la zona que pertenecen dependiendo su coordenada, dando como resultado una cantidad de cubos que agrupan las abejas que están en riesgo colisión y solamente se necesita comparar las abejas que se encuentran en esa área.

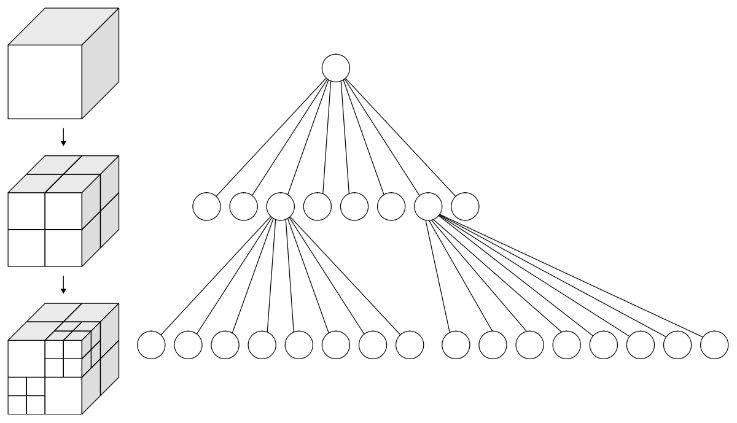


Ilustración 8

**5.1 Operación de la estructura de datos.**

Se crea un nodo que será la raíz, octreel con los valores mínimos y máximos de x,y,z

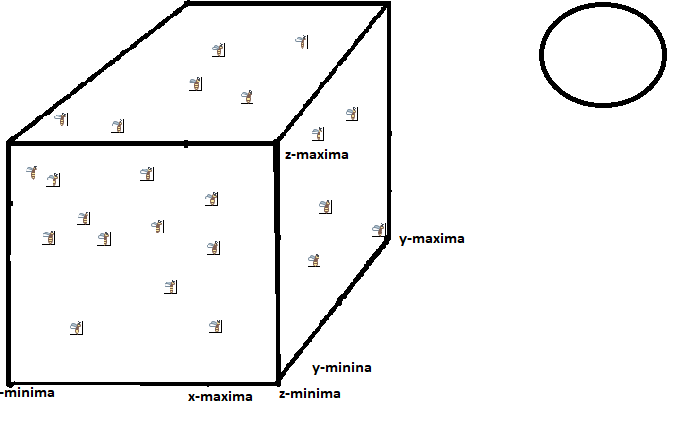


Ilustración 9

El octreel se empieza a subdividir en partes más pequeñas creando 8 hijos a cada nodo, donde los nodos finales estará cada grupo de abejas que se encuentren en el rango de distancia menor o igual a 100 metros.

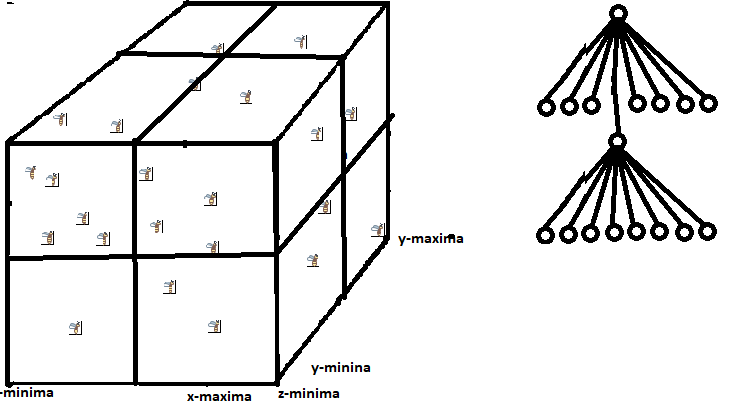


Ilustración 10

Al terminar de dividirse lo suficiente, empieza a ingresar las abejas a los nodos que son de tipo LinkedList para guardar las múltiples abejas que estén en esa área y poder tener un mejor control de ellas.

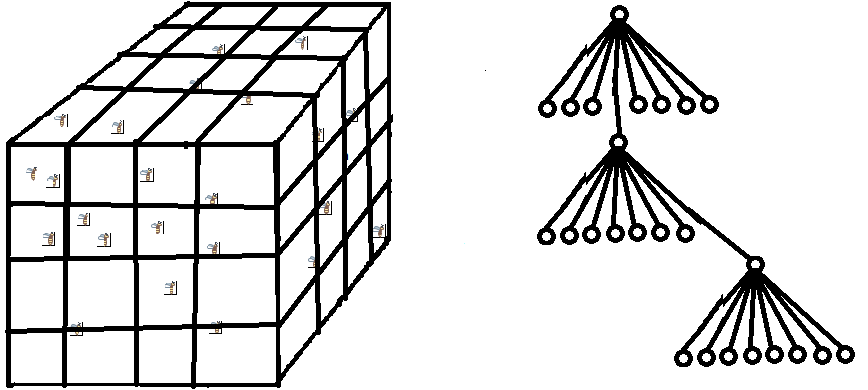


Ilustración 11

**5.3 Criterios del diseño de la estructura final.**

Esta estructura se eligió basándose en el paradigma de diseño de algoritmo, que dice divide y vencerás (DYV), implica resolver un problema difícil, dividiéndolo en partes mas simples tantas veces como sea necesario, hasta que la resolución de la parte se torne obvia.

En el caso del problema de las abejas es dividir el cubo hasta que la distancia en x,y y z sea menor o igual a 100 metros. Permitiendo como se menciono anteriormente filtrar e ingresar las abejas a sus áreas correspondientes.

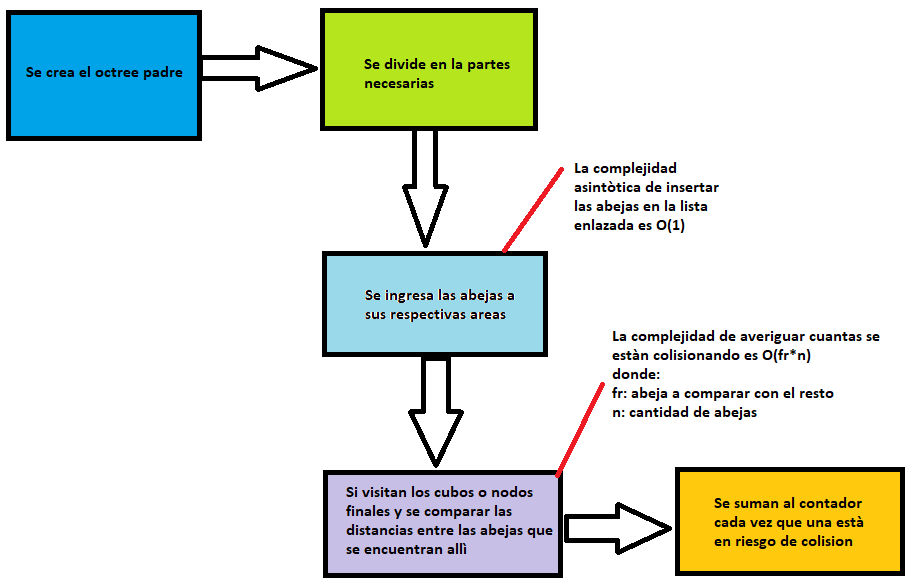


Ilustración 12

Si se hace el análisis generar lo que nos queda es que solo se va a coger y se va a hacer un proceso de comparación de cada una de las abejas con respecto a las que se encuentran en ese lugar o área.

Logrando hacer el proceso mas optimo ya que la complejidad quedaría O(fr\*n)

**fr**: abeja que se va a comparar con el resto **n**: cantidad de abejas que se encuentran en el área.

**5.4 Análisis de complejidad.**

|  |  |
| --- | --- |
| **Método** | **Complejidad** |
| Sacar los datos del archivo | O(n) |
| Insertar los datos a la lista enlazada | O(1) |
| Comparar los datos | O(fr\*n) |

**5.5 Prueba del algoritmo con diferentes datos procesados para calcular el tiempo de ejecución.**

|  |  |
| --- | --- |
| **Cantidad de abejas** | **Tiempo en milisegundos** |
| 4 | 10 |
| 10 | 13 |
| 15 | 14 |
| 100 | 25 |
| 150 | 28 |
| 1000 | 82 |
| 1500 | 76 |
| 10000 | 224 |
| 15000 | 350 |
| 1000000 | 3528 |
| 1500000 | 9526 |

**5.6 Prueba del algoritmo con diferentes datos procesados para calcular el consumo de memoria.**

|  |  |
| --- | --- |
| **Cantidad de abejas** | **Memoria en MB** |
| 4 | 0.576 |
| 10 | 0.576 |
| 15 | 0.576 |
| 100 | 1.2087 |
| 150 | 1.153 |
| 1000 | 1.730 |
| 1500 | 2.30699 |
| 10000 | 10.38 |
| 15000 | 15.572 |
| 1000000 | 155.504 |
| 1500000 | 274.184 |

**5.6 Análisis**

Se puede observar que entre mas se dividan las áreas mas eficientes se vuelve el algoritmo, ya que simplemente es cuestión de comparar las distancias de la abeja por grupos, generando

una complejidad más considerada a la hora de desarrollar algoritmos para este tipo de problemas.

**6. conclusión**

En el mundo día a día nos enfrentamos a femémonos naturales que pueden ser modelados y solucionados por medio de algoritmos véase el caso que abordamos e intentamos buscar una solución óptima para detectar posibles colisiones entre abejas robóticas.

Para esto inicialmente se planteó una estructura basada en colas donde la cantidad x de abejas ingresaban y se acomodaban en una cola para empezar a implementarse la recursión de comparación, este proceso según el análisis y la tabla de datos solo era eficiente para cantidad menor o igual a 1500 abejas, dándonos una visión global del algoritmo que era poco eficiente ya que no necesariamente van a haber esa cantidad limitada de abejas.

Con un nuevo objetivo en mente se procede a buscar he investigar sobre posibles alternativas de desarrollo para convertir el algoritmo lo mas optimo y poder cumplir con una mayor cantidad de datos, dándole vida a la estructura octree, una estructura altamente eficiente que trabaja con el paradigma divide y vencerás, dándole camino un nuevo algoritmo que cumplirá con los requisitos planteados anteriormente.

Al hacer un paralelo de los datos obtenidos sobre estas dos estructuras se concluye que esta ultima estructura es las mas eficiente con respecto a la primera ya que.

1. Permite evaluar la cantidad n de abejas.
2. Resuelve los problemas en un tiempo relativamente corto.
3. Y se puede generalizar para diferentes modelos de algoritmos.

**6.1 Trabajos futuros**

“La curiosidad mato el gato” , Motivado por la construcción y desarrollo de todo tipo de robots se globaliza la idea de cómo sincronizar una cantidad de drones para que trabajen en conjunto sin que estén en riesgo de colisión, por eso a futuros se pretende desarrollar un simulación con gemelos digitales de drones en movimiento ejecutando una tarea en específico, para evaluar de como en la vida real se podría implementar.

**AGRADECIMIENTOS.**

Este proyecto se lo dedico a mi madre y mi padre pilares principales de que yo esté estudiando esta hermosa carrera, agradezco al investigador y científico de datos Diego Cárdenas por sus recomendaciones y sugerencias.

# **REFERENCIAS**

# 1.Alan, L. “Con el auge de los drones, aumenta el riesgo de colisión con aviones. Infobae:

”https://www.infobae.com/america/wapo/2017/12/15/gracias-al-auge-de-los-drones-aumenta-el-riesgo-de-colision-con-aviones/

2.“Colisiones con el tren: problema en crecimiento”https://www.nacion.com/opinion/foros/colisiones-con- el- tren-problema-en-crecimiento/7ILZJCF4NRFDBAM2D57DC6JUNA/story/

3.“El 'coche abeja' que evita colisiones”. Elmundo.es: https://www.elmundo.es/elmundomotor/2008/09/29/tecnica/1222705245.html

4.Juan,G.“Siguiendo la pista a un avión: qué tecnologías se usan y qué problemas hay ”.Xalaka:https://www.xataka.com/otros/siguiendo-la-pista-a-un-avion-que-tecnologias-se-usan-y-que-problemas-hay