ESTRUCTURAS DE DATOS PARA LA ORGANIZACIÓN Y PREDICCION EFICIENTE DE COLICIONES EN UN SISTEMA DE DRONES.

Braimer Esteban Zapata Espinosa
Universidad Eafit
Colombia
bzapata2@eafit.edu.co

Juan Camilo Restrepo Vargas
Universidad Eafit
Colombia
jcrestrepv@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

RESUMEN

Para muchos está claro que las abejas cumplen un rol muy importante en nuestro ecosistema, se sabe que el rol más importante de la abeja es la polinización de plantas, sin polinización un tercio de nuestra alimentación desaparecería. Pero esto se está viendo afectado por la gran epidemia de muerte que está sucediendo en las colmenas, mes tras mes son millones de abejas que mueren en el mundo, siendo este dato un llamado de alerta a tomar cartas en el asunto. Según el Instituto Humboldt, el 70% de los cultivos alimenticios para los seres humanos incrementa su producción de frutas o semillas gracias a la polinización animal.

1. INTRODUCCIÓN

Dejando claro que el rol de las abejas es esencial en nuestra vida y que el problema de la extinción es cada vez mayor, ya lo que se debe tratar es buscar una solución a esto, lo cercano es el manejo de drones para la polinización, pero la cantidad de drones que se manejarían es una cantidad masiva, y el control de estas se ve un poco complicado, por la cantidad de factores que pueden afectar al vuelo de estos drones.

Un ejemplo del uso de esto es en el manejo de autos con piloto automático, se ve el auto como representación de un dron, el cual es controlado por un software manejando 2 tipos de contexto, el primero es el posicionamiento global por medio de GPS, el cual indica que carreteras tomar y que rutas seguir, el otro contexto ya es del escenario real, en el que el auto se enfrenta a diferente escenarios en el que pueden haber colisiones, que gracias a los sensores y cámaras sumado con el software buscan reacciones a las cuales evitar los choques.

2. PROBLEMA

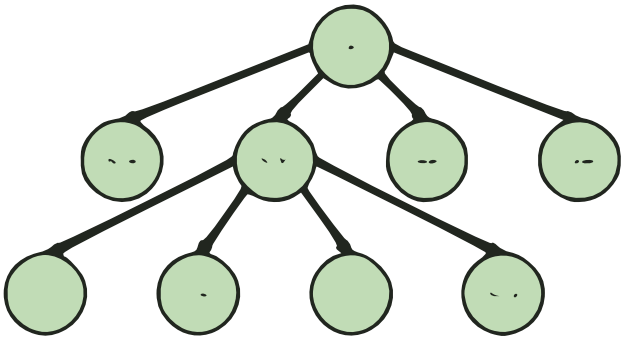
El manejo de de posicionamiento de las abejas en un espacio para predecir y evitar posible soluciones es el problema en este trabajo, ya que la cantidad de datos que se manejarían alrededor del mundo es demasido extenso.

El buen uso de arboles o sistemas que permitan la optima organización de estas abejas resolvería el problema del posicionamiento y control masivo de las mismas.

3. TRABAJOS RELACIONADOS

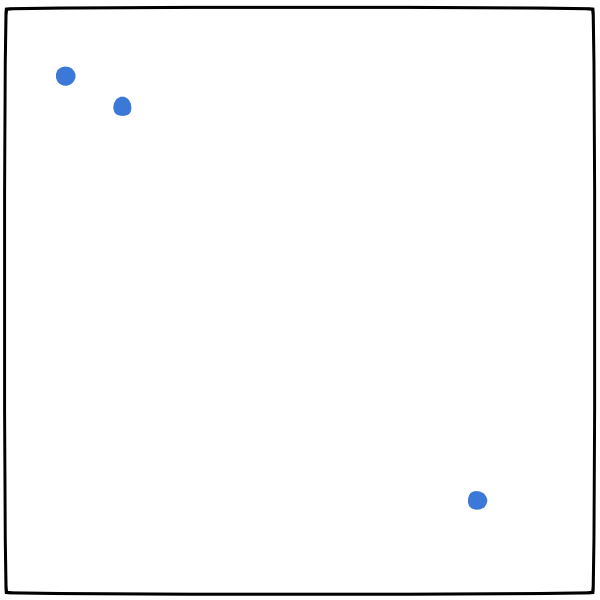
3.1 Arboles QuadTree

Los arboles QuadTree es un tipo de estructura de datos en el que cada nodo tiene a su vez cuatro nodos, es algo similar a un árbol binario, pero en vez de dos ramas por cada nodo, tiene es 4 ramas.

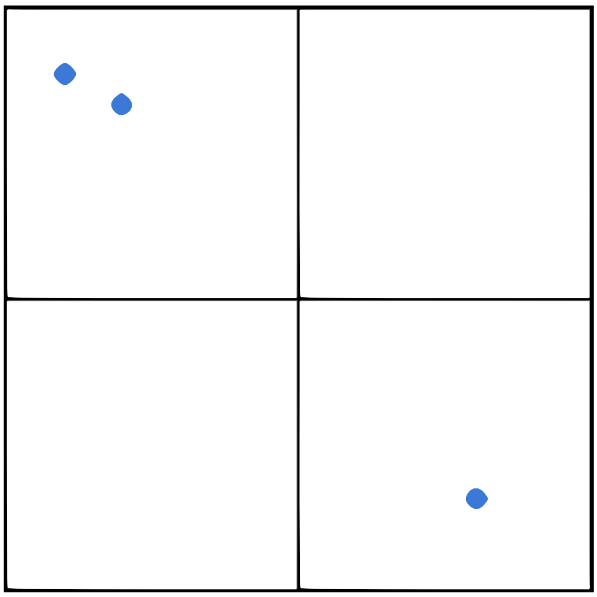


Este tipo de estructura puede ser utilizada para saber el posicionamiento de un objeto en un espacio bajo la frase “Divide y vencerás”, con la única diferencia de que no estaremos dividiendo en 2 partes el problema sino en 4, y cada una de estas partes en otras 4, y así sucesivamente hasta hallar una profundidad optima para que la solución del problema sea eficaz.

Digamos que tenemos en un espacio 3 objetos, y queremos hallar su posición para prevenir posibles colisiones.

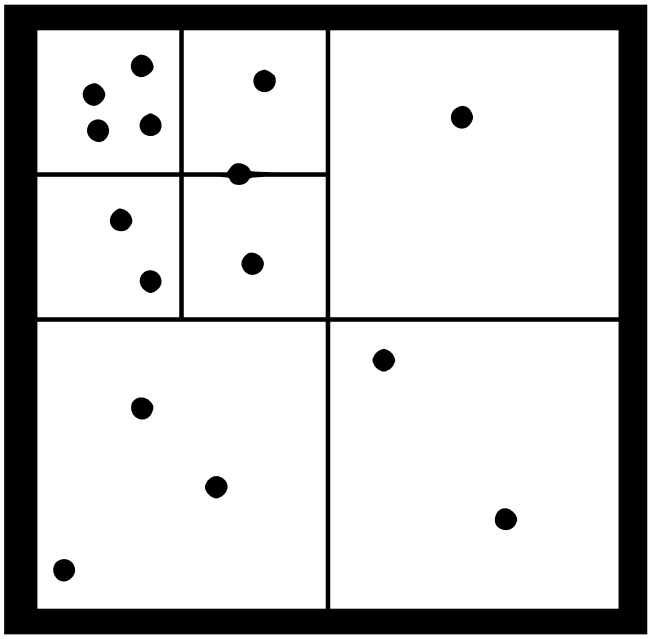


Utilizando QuadTree podemos dividir el espacio en 4 cuadrantes generando un nodo padre (espacio total) y 4 nodos hijos (cuadrantes del espacio).



Teniendo los objetos separados por cuadrantes ya podremos implementar un código que prevenga posibles colisiones solo en el primer cuadrante, ya que en los otros 3 no son posible colisiones debido a que no solo hay un objeto, y para poder colisionar debe existir al menos 2, como el primer cuadrante.

Ahora, si tenemos muchos mas objetos, se aplica lo mismo, se divide el cuadrante en otros 4 cuadrantes para así analizar posibles colisiones solo en los cuadrantes que se encuentran 2 o más objetos.

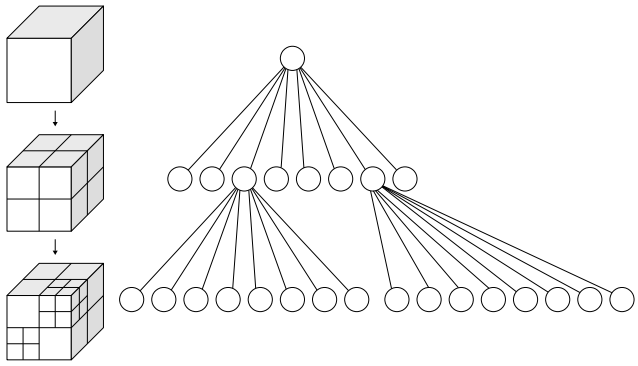


En el caso de que un objeto quede en medio de 2 cuadrantes, se toma como objeto del cuadrante padre, en el cual ya se analizaran sus posibles colisiones.

3.2 Arboles OctaTree

Los arboles OctaTree tienen mucha semejanza con los Arboles QuadTree, en el artículo anteriormente mencionado hablábamos de un espacio 2D, donde lo que hacíamos era dividir este espacio en 4 partes.

Pero ya con los arboles OctaTree hablamos es de un espacio 3D donde ya vemos el nodo pare como un objeto tridimensional (cubo). Este nodo padre genera 8 nodos hijos también en forma tridimensional (cubos), y se analizan los objetos de manera similar que en el artículo pasado.



Cada nodo corresponde a un solo cubo y tiene exactamente ocho nodos.

El árbol OctaTree es muy utilizado en los video juegos para el posicionamiento de objetos en un espacio tridimensional, para una mayor eficacia los arboles solo analizan posibles colisiones en los nodos que hay 2 o mas objetos, si en un nodo se encuentra solo un objeto, este es descartado ya que para que exista una colisión deben haber 2 o más objetos.

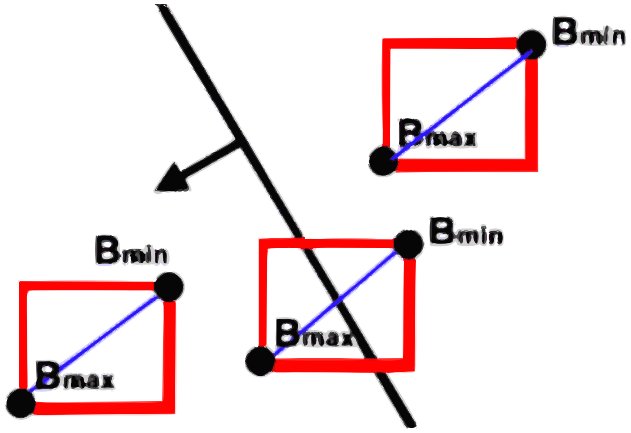
3.3 Arboles AABB

AABB por sus siglas en idioma inglés, Axis Aligned Bounding Boxes.

Es una estructura de datos de subdivisión espacial y un conjunto de algoritmos usados principalmente para realizar intersecciones y calcular distancias de forma eficiente en objetos geométricos en 3D con un número finito de puntos.

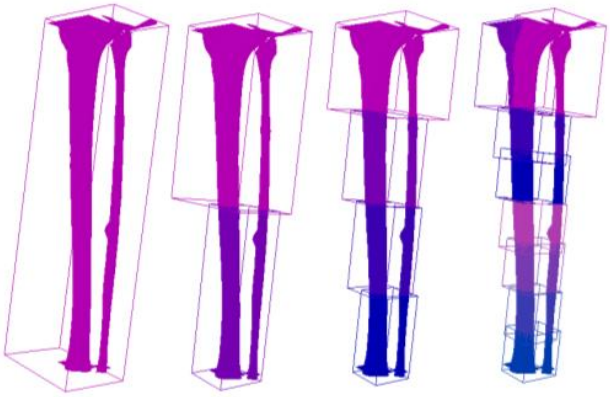
Sobre la practica: Cada AABB define 3 intervalos en los ejes X,Y,Z. Si alguno de estos ejes de solapan, los AABB se interceptan

.



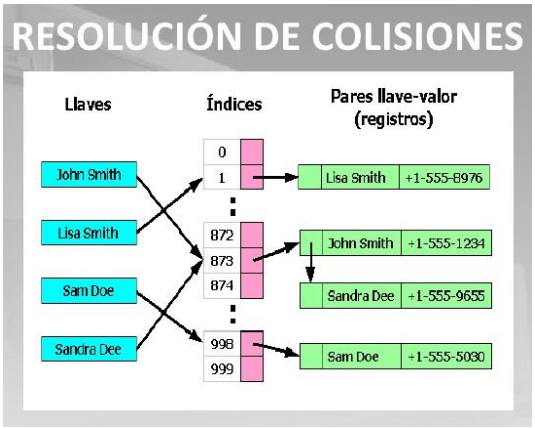
En cada nivel el AABB se subdivide a lo largo del eje mayor

Se subdivide con un plano que pasa por la mediana de los puntos proyectados sobre ese eje.



3.4 Tablas de Hash Espaciales

Las tablas hash de direccionamiento abierto pueden almacenar los registros directamente en el array. Las colisiones se resuelven mediante un *sondeo* del array, en el que se buscan diferentes localidades del array (*secuencia de sondeo*) hasta que el registro es encontrado o se llega a una casilla vacía, indicando que no existe esa llave en la tabla.



Las secuencias de sondeo más utilizadas son:

3.4.1 Sondeo lineal

En el que el intervalo entre cada intento es constante (frecuentemente 1). El sondeo lineal ofrece el mejor rendimiento del caché, pero es más sensible al aglomeramiento.

3.4.2 Sondeo cuadrático

En el que el intervalo entre los intentos aumenta linealmente (por lo que los índices son descritos por una función cuadrática. El sondeo cuadrático se sitúa entre el sondeo lineal y el doble hasheo.

3.4.3 Doble *hasheo*

En el que el intervalo entre intentos es constante para cada registro, pero es calculado por otra función hash. El doble hasheo tiene pobre rendimiento en el caché pero elimina el problema de aglomeramiento. Este puede requerir más cálculos que las otras formas de sondeo.

REFERENCIAS

Referenciar las fuentes usando el formato para referencias de la ACM. Léase en http://bit.ly/2pZnE5g Vean un ejemplo:

1. Adrigm, Teoria de Colisiones 2D: QuadTree. Junio 30 de 2013:

   https://www.genbetadev.com/programacion-de-videojuegos/teoria-de-colisiones-2d-quadtree
2. Mike James, QuadTrees And OctaTrees. Junio 30 de 2015:

   http://www.i-programmer.info/programming/theory/1679-quadtrees-and-octrees.html
3. Octre, Wikipedia. (Imagen):

   https://en.wikipedia.org/wiki/Octree#/media/File:Octree2.svg
4. Arbol AABB, Wikipedia:

   https://es.m.wikipedia.org/wiki/Árbol_AABB
5. Tabla Hash, Wikipedia

   https://es.m.wikipedia.org/wiki/Tabla_hash
6. Hector Navarro, Detección de Colisiones:

   http://ccg.ciens.ucv.ve/~hector/icg/collision_detection.pdf