AVOIDING COLLISION: A SPATIAL STUDY OF POSITION ON AUTOMATED ROBOTIC BEES

Rafael Mateus Carrión
Universidad Eafit
Colombia
rmateusc@eafit.edu.co

Daniel Otero Gómez
Universidad Eafit
Colombia
doterog@eafit.edu.co

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

ABSTRACT
The purpose of this research paper is to propose a solution to avoid the collision of automated robotic bees. Georeferences as geodesic coordinates are essential to recognize and solve this problem, therefore, they will be present throughout the report. Moreover, an analysis of the subproblems is included in the first section, as well, as possible solutions for themselves. The final solution that we find more efficient to solve this problem was using an octree-based data structure, with ArrayList of LinkedList of Bees as a base. Using this solution, we were able to solve the problem with an efficient time of execution. We include a comparison between to data structures and ArrayList of LinkedList and an ArrayList.

In addition, while solving this problem we ascertain that the size of the space were robotic bees were in was a really important factor for the time complexity of the code, as the area has to be imaginary divided more times to have the size wanted.

1. INTRODUCTION

Undoubtedly, technology is each time more present in our daily life, and automated robots are a clear example of it. Robotic bees are now a reality, and even though their functioning is still controlled by humans, automated robotic bees could be a possibility in the near future. For this to happen, preventing the collision between them is an important aspect to be solved first.

There are some challenges in order to solve this problem accurately. It is important to take into account that this is a three-dimensional problem, hence, the chances of collision are innumerable. Consequently, the algorithms that will be implemented should be analyzed in detail, considering that there are high odds of inaccuracy.

Keywords
Collision, coordinates, robotic bees.

ACM CLASSIFICATION Keywords
Information systems→Data management systems→ Database design and models→ Graph-based database models →Hierarchical data models

2. PROBLEM

In a few words, explain the problem, the impact that has in society and why is important to solve the problem.

The main problem is to avoid collisions between near bees. Within it, we can find another set of problems while trying to fix the central one, such as locating the other bees position with accuracy and retracing the movement of the bees in order to make sure that they do not crash.

Solving this problem can have a great impact on the agriculture sector, and may lead to a technology revolution all around the environment. Likewise, it is probable that this improvement contributes to other problems involved in automated robots.

3. RELATED WORK

3.1 Collision detection

It is the detects when an object hits something or when two objects hit each other. In this specific case, the objective is to avoid the collision between two non-stationary objects. This problem can be solved either mathematically or graphically. For this problem, solving it mathematically is more efficient as there will be a mathematical model that describes the position of the objects, in this case, the bees.

If we imagine bees as points, it will only be necessary state that the distance between two points should be a non-zero result, in order to avoid a collision, therefore, if we apply a mathematical formula to do so, the result will be:

$$\mathrm{dist}(P_0, P_1) = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$$

Being P0 and P1 the bees and x, y, z the coordinates of each one respectively. And the result of the distance between both points has to be different than zero, as zero will mean that both points are on the same coordinate, which means they are colliding. [1]

3.2 Obstacle avoidance

In case of a possible collision between two bees, an algorithm for avoiding a collision is important, each bee should be able to dodge if a crash is imminent.

Velocity, distance and the size of each bee is important to make the algorithms as well as the mathematical calculation on how to not crash between them. Algorithms of aggregation and disaggregation can be implemented, in order to change the position or the velocity of each bee. If the position is changed, the path that the bees were following will be changes, hence, the collision will be avoided, and if the velocity is changed, it will be easier to change its direction. It is important to understand that online calculation should be made by the bee to avoid all crashes. [2]

3.3 Spatial Partition
It is a technique used to simplify the process of identifying movement and collisions in video games. This procedure consists on dividing a given space on quadrants in a way in which such subdivisions will not overlap with each other. This kind of approach of solving the problem is a great way of making the program faster when it has a great number of entities in it.

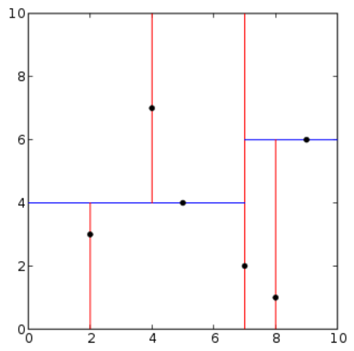
The purpose of subdividing the space is to avoid comparing the location of an entity with every other entity present in the environment and determining which entities are nearby. It simplifies the process by only comparing the position of the entities located in the same quadrant. As this is a very common way of approaching the problem, there are several data structures that solve the problem of dividing the space in their own way, as some of them may result efficient to a specific kind of circumstances.

The K-D Trees are a type of data structure that make space partition in a way that range searches, and nearest neighbor searches are done in a more efficient way. A K-D Tree is a class of binary space partition tree, that works as it follows:

Each leaf of the tree acquires the function of a k-dimensional point. Each non-leaf node can be identified as a subdivision of the plane. Every point located at the left of the hyperplane will be located in the left subtree of the node. Likewise, the points located at the right of the hyperplane will be located at the right of the subtree.

3.4 Retracing Position
Identifying when two or more bees are nearby is only half of the work. Our main goal is to prevent them from colliding between each other. In order to do that is necessary to adjust the position of the bees and locate them in a another in which there is a fewer possibility of colliding with one another. To make the solution as simple as possible we will like to involve the K-D tree again.

As it was said before, the K-D Tree divide the plane in several subdivision depending on the root point. A more graphical way to see it is the one shown below.



The diagram is composed of several squares formed by the vertical and horizontal subdivisions created by the position of the bees.

Let's assume that the diagram shown above is a real case scenario of the position of some bees. We propose to establish a minimum distance between bees to avoid crashes. Let’s suppose that this distance will be two units. As you may observe the bees located at the position (7,2) and (8,1) are extremely close, to the point of violating the 2 unit range established by us before, which means that they could collide in a certain point.

As we can observe, this points are surrounded by squares. One between them, other between the right border and the (8,1) bee, and other at the left of the (7,2) bee. The rectangles at the sides represent the area in which they could move without colliding with other bee. So the perfect solution to the problem would be to move the position of one bee towards the square who has a greater area.

Correspondingly, the (7,2) bee will change its position in a way that the distance of every other bee located in the axis of any of the two subdivisions of the plane denoted by itself violates the minimum distance range.

4. ArrayList of Bees

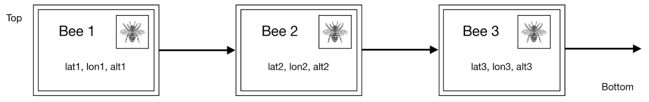


Figure 1: ArrayList of Bees. A Bee is a class that contains the coordinates of each robotic bee: latitude, longitude and altitude.

4.1 Operations of the data structure

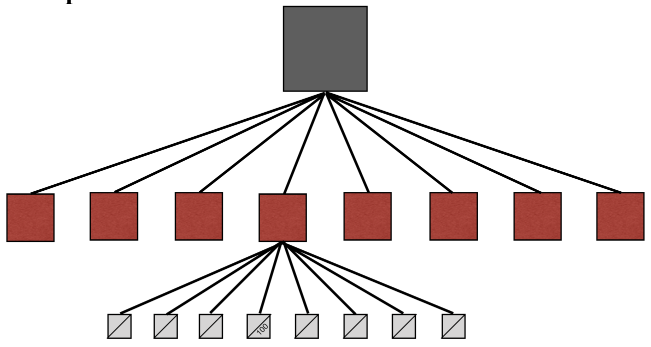


Figure 2: Divides the total area where the Bees are, and divides it into 8 until the diagonal of each area is less or equal than 100 (Octree).

4.2 Design criteria of the data structure

When creating this data structure, we thought of optimizing the execution time of each one of the methods, we sought to reduce the complexity of the project as much as we could. Hence, we used collection frameworks that optimize our code the most, depending on our necessity. Additionally, we simplified the code to avoid time wasting and we attempted to reduce memory usage.

4.3 Complexity analysis

Method	Complexity
Reader	O(n)
createBee	O(n)
getMaxMin	O(1)
choque	O(n)
Octree	O(n)
hashing	O(n)

Table 1: Table to report complexity analysis of each method.

4.4 Execution time

	Average Time (ms)
4 Bees	1
10 Bees	8
15 Bees	6
100 Bees	29
150 Bees	31
1000 Bees	82
1500 Bees	100
10000 Bees	126
15000 Bees	356
100000 Bees	250
150000 Bees	397
1000000 Bees	2520
1500000 Bees	6635

Table 2: Execution time of the operations of the data structure for each data set.

4.5 Memory used

Conjunto de Datos	Consumo de Memoria
4Abejas	29MB
10Abejas	1MB
15Abejas	1MB
100Abejas	1MB
150Abejas	1MB
1000Abejas	2MB
1500Abejas	3MB
10000Abejas	2MB
15000Abejas	3MB
100000Abejas	18MB
150000Abejas	39MB
1000000Abejas	136MB
1500000Abejas	26MB

Table 3: Memory used for each operation of the data structure and for each data set data sets.

4.6 Result analysis

Clearly, regarding the average time spent among the different data sets, there is a direct relation respecting the size of the data set used and the time until the algorithm has finished. If we take a look to the complexity table, this will make complete sense, due to the fact that most of the methods involved on the process solution have a complexity of O(n).

In relation to the memory use of each data set, we believe, that there is a direct relation between the amount of steps that the code has to go through before it finishes. In more simple words, there could be a million bees, but if the do not divide the Octree in any sub-Octrees, the memory use will not be as large as a data set of a hundred bees that need tens of sub-Octrees to get to the end.

5. ArrayList of LinkedList of Bees

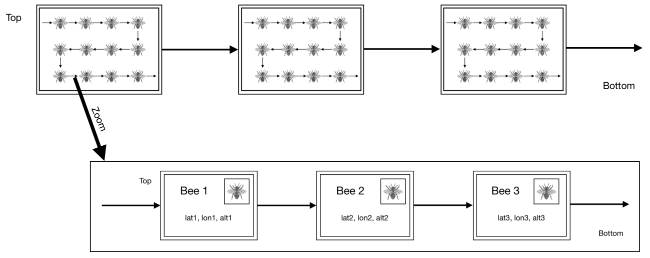


Figure 3: ArrayList of LinkedList of Bees. Each position of the ArrayList contains a LinkedList of Bees. A Bee is a class that contains the coordinates of each robotic bee: latitude, longitude and altitude.

5.1 Operations of the data structure

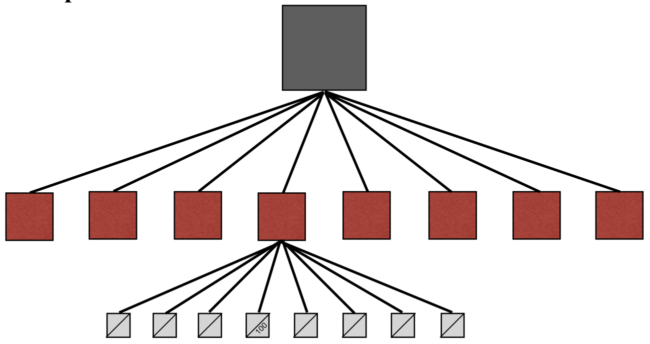


Figure 4: Divides the total area where the Bees are, and divides it into 8 until the diagonal of each area is less or equal than 100 (Octree).

5.2 Design criteria of the data structure

We created this data structure aiming on reducing the time complexity, therefore we simplified the code as much as we could. Besides, as the problem involved a three dimensional space, an octree was the most appropriate to use. And we use various collection frameworks to improve the execution time of the data structure.

5.3 Complexity analysis

Method	Complexity
leer	O(n)
splitString	O(1)
getMaxMin	O(1)
choque	O(n)
octree	O(n)
hashing	O(1)
nuevoOctree	O(1)

Table 5: Table to report complexity analysis of each method.

5.4 Execution time

Data Set	Time (ms)
4	0
10	0
15	0
100	3
150	2
1000	9
1500	11
10000	9
15000	12
100000	174
150000	293
1000000	2095
1500000	453

Table 6: Execution time of the operations of each data set.

5.5 Memory used

Data Set	MemoryUsed (MB)
4	0
10	0
15	0
100	1
150	3
1000	10
1500	59
10000	24
15000	39
100000	107
150000	151
1000000	227
1500000	126

Table 7: Memory used for each operation of the data structure and for each data set.

5.6 Result analysis

We determine that using an ArrayList of LinkedList for the data structure would reduce the execution time of the code. Consequently, it can be seen a reduction on the new execution time table, compared to the previous one. We also realize that the time execution does not only depended on the size of the dataset but also on the area the bees were in, that is why it takes more time to execute the dataset of 1’000,000 bees than the 1’500,000 bees.

6. CONCLUSIONS

During this report we found an efficient solution to avoid collision of robotic bees. This code print all the robotic bees that are in a distance equal or less than a 100 meters away. In order to do so, the design of the data structure is an octree, it is implemented using an ArrayList of LinkedList of Bees, a Bee is a class that contains the coordinates of each robotic bee: latitude, longitude and altitude.

After various attempts on optimizing the problem, regarding time and memory usage, we ascertain that using an ArrayList of LinkedList of Bees made our solution more efficient than an ArrayList of Bees than an ArrayList of Bees. Since a LinkedList had more time efficient methods for, especially, adding and removing items.

Additionally, we conclude that the time it took to run the entire code not only depended of the method we used and the quantity of bees, but also to the area we were working with, that is the reason why in the time tables, even though some data sets of bees are bigger than others, the take less time to execute.

6.1 Future work

We would like to find out a solution to improve the time executions of the code when the area that is entered is too big, in order to make the code as efficient as possible.

REFERENCES

1. Thompson, J. Collision detection. 2019.

2.Xiangru, C, Fengxia, L, Zhaohan, L. and Yue, Y. Obstacle avoidance behavior of swarm robots based on aggregation and disaggregation method. School of Computer Science, Beijing Institute of Technology, China, 2017.

3. Vapnik, V. N.; Chervonenkis, A. Ya. (1971). "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities". *Theory of Probability & Its Applications*.

4. Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching"