ALGORITHMIC CONTROL AND SUPERVISION FOR THE PREVENTION OF COLLISION BETWEEN ROBOTIC BEES

Hamilton Smith Gómez Osorio
Universidad EAFIT
Colombia
hsgomezo@eafit.edu.co

Santiago Isaza Cadavid
Universidad EAFIT
Colombia
sisazac@eafit.edu.co

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

ABSTRACT

Robotic bees are the future of agriculture in their process of production, that is why makes necessary having control and supervision of the exact location of these in order to achieve an optimal behavior.

KEYWORDS

Sorting and Searching, Data structures, Hashing, Data Management.

ACM CLASSIFICATION System

CCS → Theory of computation → Design and analysis of algorithms → Data structures design analysis → Sorting and searching.

1. INTRODUCTION

Faced with the decline of the population of bees that is happening nowadays and the importance of these in the process of pollination and in the agricultural sector it is possible to say that there is a risk in agricultural crops, so it is necessary to find a solution to this problem. This is how the idea of creating robotic bees was born, which can help in this process and, to supervise them, and control their behavior, develop and implement and algorithm that prevents their collision.

2. PROBLEM

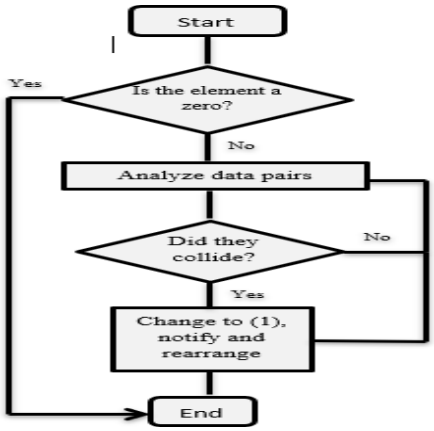
The implemented robotic bees in agriculture for the pollination process can collide if they are less than 100 meters from other bees, that is why is so important solve the problem in order to achieve and optimal behavior and an improvement in their processes.

3. RELATED WORKS

3.1 Radio Frequency Identification System (RFIS)

When there are several labels (used to store the information) and readers (read, change and verify the information on the label) in the same channel and signal transmission, a collision problem occurs due to mutual interference between the labels and the readers.

Solution: an anti-collision algorithm based on a matrix and a coding scheme. The decoded data is established in a matrix and then the reader is responsible for processing the data row by row, analyzed in pairs and finding a collision is replaced by a value (1), otherwise set a zero (0). After replacing the rows, the collisions are extracted and the following rows are analyzed until the algorithm finishes.



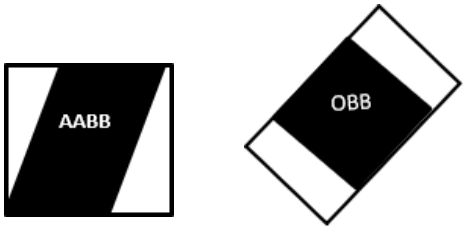
[1]

Figure 1: RFID Flowchart

3.2 Bounded volumes to detect collisions

Very used in graphic computing when making videogames. It is based on the use of basic geometric shapes bounding more complex figures and using the intersection of these to determine when a collision occurs; thanks to the figures you have control of the objects when there is movement or perspective changes, a solution used in AABB (Axis-

aligned bounding box) and OBB (Oriented bounding box).



[2]

Figure 2: Bounding figures

3.3 Octree

To analyze collisions between multiple elements in a plane, this method is used, which subdivide a delimited space into rectangles of equal size and then divides the space again until it has a relatively smaller area in which is easier and faster to compare positions of one object and another, considering its volume and coordinates. [3]

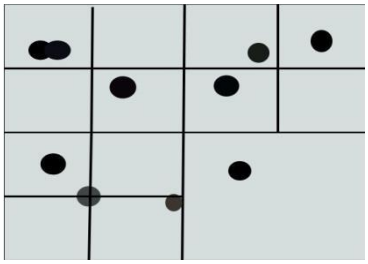


Figure 3: Octree subdivisions

3.4 Data structure spatial hashing

It consists of dividing a zone into cubes with a specific measure, considering the maximum and minimum value of coordinates, and then organizing the objects that are inscribed within them in a list of reference to the index of the box in which it is located. [4]

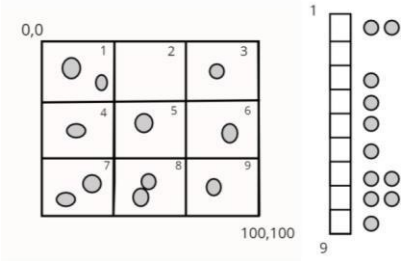


Figure 4: Subdivision and classification by boxes.

4. Structure to use: Three-dimensional Matrix

4.1 Operations

- **Create the cells:** we use this to define the cells in which the bees will be found. In each cube will be found the bees stored in a stack, these bees being the same cube are the ones that are at collision risk.
- **Detect collision:** given the coordinates of a bee it will be added to a certain stack with those bees that are at risk collision.
- **Adjacent bees:** determines which bees are adjacent cubes and end up being at collision risk.

4.2 Design criteria for the data structure

This structure is efficient compared to other presented previously, since form the construction of the algorithm we can define the size and number of cells depending on the number of objects that can collide. Thus, it does not need to analyze any position of one bee respect to another, since the fact of belonging to the same cell already represents a collision risk. So with this data structure memory is saved (compared to a comparison between all bees) avoiding any sort of classification or query, besides, making the algorithm faster.

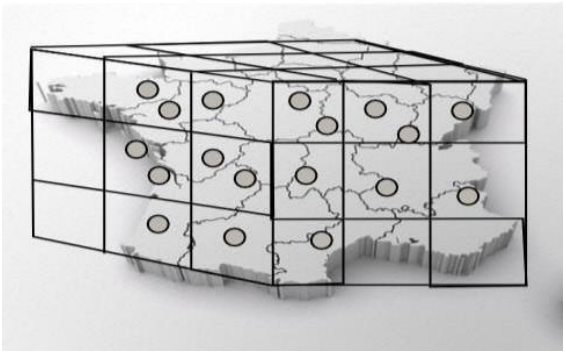


Figure 5: Surface bounded by boxes

Implementation approach

In the Java programming language, a solution was implemented through the formation of a three-dimensional matrix. That represents the divisions of a space in cells. The volume of these cubes is determined by a diagonal of 100 meters and 57 meters in their edges, later the bees are classified in stacks depending on the cube in which they are to return those in the same cube and recognize those at risk of collision.

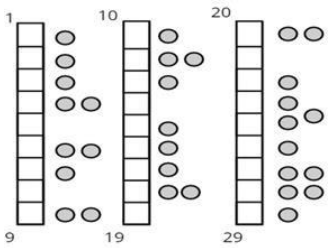


Figure 6: Organization of bees according to their position

4.3 Complexity

Method	Complexity
areaDeUbicacion()	O(n)
abejasAdyacentes()	O(m)
detectarColisiones()	O(n + m)

Table 1: Complexity

4.4 Operations time

Method	Average time(ms)						
	Number						
	4	10	100	1.000	10.000	100.000	1'000.000
areaDeUbicacion()	0,6	3,6	3,6	11,6	28,4	275,8	3.315,2
detectarColisiones()	0	4,2	11	8,2	37,8	131,8	673,6
guardarArchivo()	1,6	3,2	7,4	5	112,8	770	6.060,2

Table 2: Time

4.5 Memory

Number of bees	Memory
10	1
100	2
1000	4
10000	15
100000	80
1000000	245

Table 3: Memory used in MB

4.6 Analysis

Collision detect with ArrayList	Collision detect LinkedList	Collision detect 3D Matrix
O(n^3)	O(n^2)	O(n+m)

Table 4: Different complexities

The obtained results, compared with the previous solutions show a real improvement, since the complexity of the algorithm 's time is reduced. Besides, the best data structure depends of the type of problem you are working on.

About the memory use and time, we can see that a bigger the number of bees means an increase of the time. This is clearly because is the biggest data set.

Bees	Results
4	4
10	4
100	24
1000	279
10000	9546
100000	99184
1000000	1000000

Table 5: Obtained results using the different data sets

5 Conclusions

Throughout the report we talked about different possible solutions to the collision problem, from Octree to small divisions using bounded figures. All of them viable, but the solution made in this work also means a possible and convenient solution, because it allows to analyze and obtain those bees that are at risk of collision using a practical method such as three-dimensional matrices and allowing satisfactory results of rapid calculation and manageable

5.1 Future Work

We would like to improve the implementation of the search of adjacent bees since we consider that it can be done in a more efficient way. Also, improving the data types to use in order to waste less memory.

ACKNOWLEDGEMENTS

We are especially grateful to the national education project Ser Pilo Paga of the Government of Colombia and to the Fundación scholarship of Universidad EAFIT for financially support our education.

We would also like to thank Daniel Mesa, teaching assistant of the Data Structures and Algorithms I course at Universidad EAFIT, for his advice in the learning process, for his contributions in class and his guidance in the realization of this project

REFERENCES

[1]. Liu, B. and Su, X. An Anti-Collision Algorithm for RFID Based on an Array And Encoding Scheme. Information, 2018, 2078-2489. Retrieved August 25, 2018 from Eafit University: https://bit.ly/2PEzPhx

[2]. Dinas, S. and Bañón J. M. A literature review pf bounding volumes hierarchy focused on collision detection. Ingeniería Competitiva, 2015, 49-62. Retrieved August 25, 2018, from Eafit University: https://bit.ly/2BMI9sD

[3]. Nevala, E. Introduction to Octrees. GameDev.net, 2018

https://gamedev.net/articles/programming/general-and-gameplay-programming/introduction-to-octrees-r3529/.

Accessed September 23, 2018.

[4]. Spatial hashing implementation for fast 2D collisions. The mind of Conkerjo, 2013. https://conkerjo.wordpress.com/2009/06/13/spatial-hashing-implementation-for-fast-2d-collisions/. Accessed September 23, 2018.

[5]. How to efficiently remove duplicate collision pairs in spatial hash grid? Stack Overflow, 2015. https://stackoverflow.com/questions/31124702/how-to-efficiently-remove-duplicate-collision-pairs-in-spatial-hash-grid. Accessed September 23, 2018.