THREE-DIMENSIONAL MATRICES IMPLEMENTED WITH LINKED LISTS TO PREVENT COLLISIONS

Juan Sebastián Pérez Salazar
Universidad Eafit
Colombia
jsperezs@eafit.edu.co

Yhoan Alejandro Guzmán García Universidad Eafit
Colombia
yaguzmang@eafit.edu.co

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

ABSTRACT
The main problem we want to solve are the collisions that robotic bees can suffer in certain areas where they are found. The project is planned for a scenario in which the bees are almost extinct or in their entirety, so they are replaced with robotic bees. This problem is important because its correct functioning translates into a good support of the robotic bees life time. A good operation of those robotic bees can reduce the costs caused when colliding with each other. The related problems are all focused on a correct structure for the detection of collisions, however our structure is based on preventing these collisions. We propose in this report the use of simple arrays, linked lists and three-dimensional matrices for an effective solution to the problem posed. At the end of the implementation, we conclude that the structure achieves what was expected, and that the memory expense is justified in order to achieve a higher speed because this algorithm is applied in a context where answers are needed in less than 1 second.

Keywords
Collisions, complexity, correct algorithm, best case, data structures, execution time, innovation.

ACM CLASSIFICATION Keywords

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

1. INTRODUCTION

Currently, one of the most important actors in the ecosystem of our planet, the bee, is dying in a massive way. The importance of the bee is that it is key in the pollination process of the plants, which makes it necessary to carry out different types of crops. Therefore, the death of bees, due to the use of pesticides and other factors such as deforestation and climate change, directly affect farmers.

This alternative poses a challenge at a technological and algorithmic level, so we are concerned with solving a problem derived from this alternative: the collision between drones.

2. PROBLEM

The problem are the collisions that can occur between bees that are in operation. It is important to avoid this type of event, since it can generate large expenses and inefficiencies. This project seeks mainly to avoid such collisions.

3. RELATED WORK

3.1 Data structures and algorithms for nearest neighbor search in general metrics spaces

In this related work is considered the computational problem of finding nearest neighbors in general metrics spaces. Of particular interest are spaces that may not be conveniently embedded or approximated in Euclidian space, or where the dimensionality of a Euclidian representation is very high.

Also relevant are high-dimensional Euclidian settings in which the distribution of data is in some sense of lower dimension and embedded in the space.

Solution
The solution for this problem is the vp-tree (vantage point tree). It is introduced in several forms, together with associated algorithms, as an improved method for these difficult search problems. Tree construction executes in O(nlog(n)) time, and search is under certain circumstances and in the limit, O(log(n)) expected time.

The theoretical basis for this approach is developed, and the results of several experiments are reported. In Euclidian cases, kd-tree performance is compared [1].

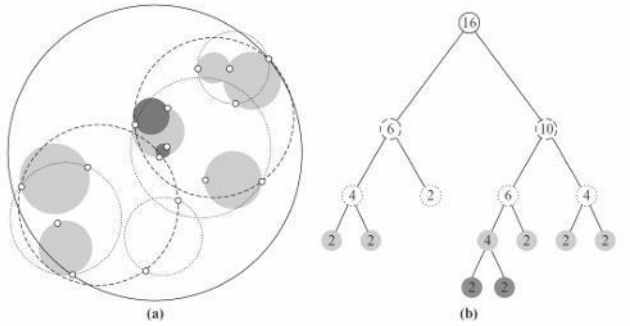

Figure 1: Analysis and process of vp-tree

3.2 Collision-free path planning in free path planning in multi-dimensional environments

Reliable path-planning and generation of collision-free trajectories has become an area of active research over the past decade where the field robotics has probably been the most active area. Different methods are sought to solve the problem

of collision of the robot and we consider that the most appropriate is the following:

Solution

The basic RRT (rapidly-exploring random tree) algorithm constructs a tree T by using nodes and links that gradually growing in a random fashion; they stop once start and goal configurations become joined.

The RRT algorithm starts exploring the neighborhood of q_{start} as the root of T to find a collision-free trajectory. If this tree's growth reaches goal configuration, or a maximum number of iterations is reached, the algorithm stops and a path is drawn from q_{goal} to q_{start}. A continuous path can thus be established starting from the branches, extending through the tree trunk and finally reaching the root [2].

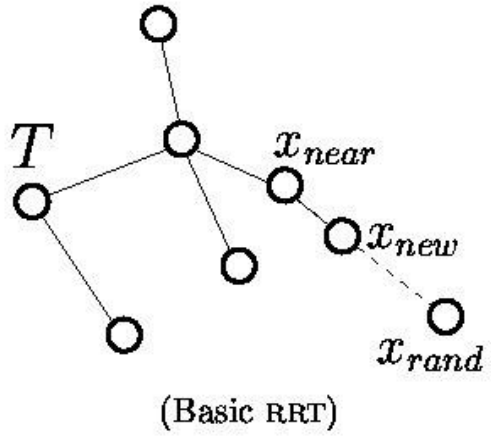

Figure 2: RRT basics procces

3.3 Incremental 3D Collision Detection with Hierarchical Data Structures

3D collision detection is the most time consuming component of many geometric reasoning applications. Any improvements on the efficiency of the collision detection module may have a great impact on the overall performance of these applications. Most efficient collision detection algorithms in the literature use some sort of hierarchical bounding volumes, such as spheres or oriented bounding boxes, to reduce the number of calls to expensive collision checks between polygons.

Solution

Based on an observation from the spatial coherence between two consecutive collision queries, we propose an incremental scheme that can be applied to the existing algorithm to reduce the number of overlap tests between two BV's. We use the list from the previous query as a starting point to generate the separation list for the current query. If coherence between two consecutive queries exists, then only a small portion of the list needs be updated at each time. Three types of results may occur when updating a separation node. A node may (a) move down, the node may (b) stay still, or (c) move up in the recursion tree. [3]

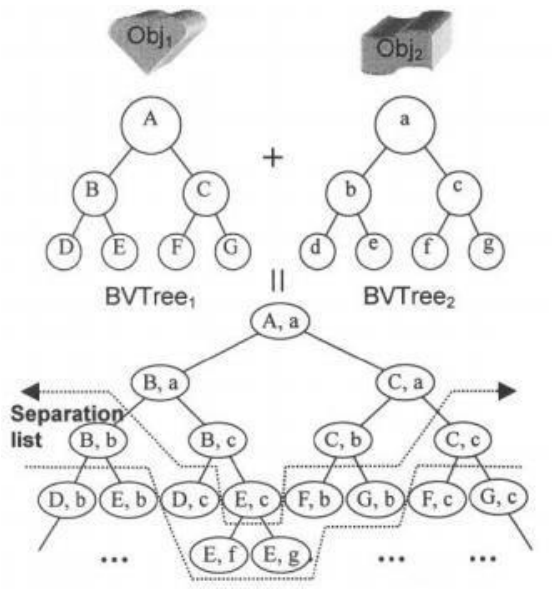

Figure 3: Comparison of two Bvtree with separation lists

3.4 Planning of Collisions Free Trajectories for Multiple UAVs using the Speed Profile

In this article they solve a 3D movement planning problem for multiple unmanned aerial vehicles, sharing space with non-cooperative aerial vehicles. A new speed profile is sought for the different UAVs involved in the collision. This article presents a new heuristic approach that will allow finding suboptimal solutions in less time than optimal media, thanks to the search for solutions in a discrete space.

Solution

The solution to this problem is based on two algorithms:

First, the Search in Tree algorithm aims to find a first collision-free solution, although it will not be the one that minimizes the cost function presented previously. This solution will serve the Tabú Search algorithm as a starting point. The Tree Search will provide us with a solution in which vehicles travel at the maximum speed that assures them of a collision-free trajectory.

After that the algorithm of Search in Tree finds a solution to the problem but does not consider the cost function. The algorithm of Search Tabú modifies the solution that the Tree Search found, minimizing the cost. The Tabú Search improves the result of a local search method, using memory structures to avoid local minimums. [4]

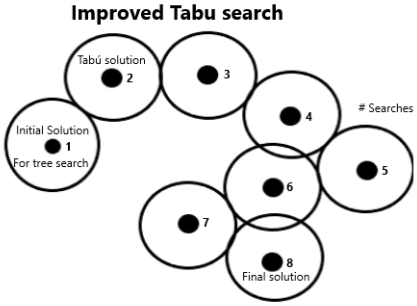

Figure 4: Improved tabú search with a tree search initial solution

4. THREE-DIMENSIONAL MATRICES IMPLEMENTED WITH LINKED LISTS

The selected data structure is "Three-Dimensional Matrices implemented with Linked Lists", this is a new way to implement tree dimensional structures to solve the problem; the following figure is the process of this algorithm.

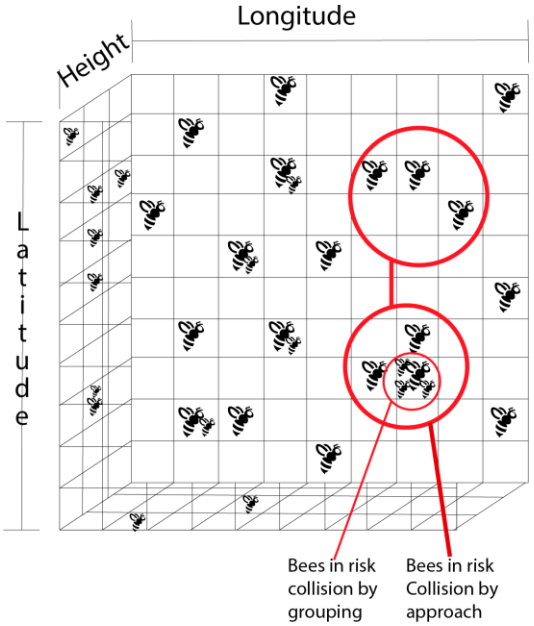

Figure 5: Prevention process of bees in collision risk, from Three-Dimensional Matrices implemented with ArrayList.

4.1 Operations of the data structure

4.1.1 Analysis and reading of data

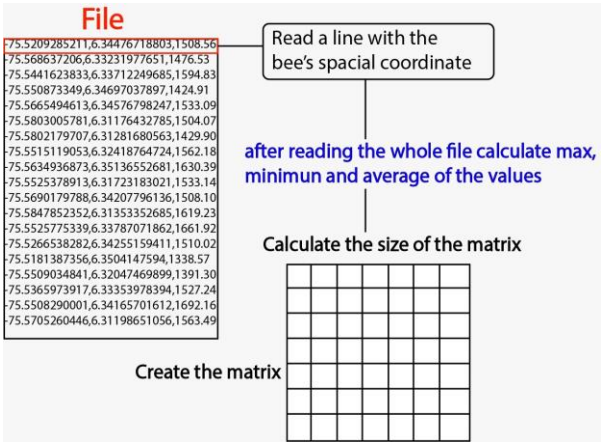

Figure 6: Process of analysis and reading of data

4.1.2 Adding bees to the Three-Dimensional matrix

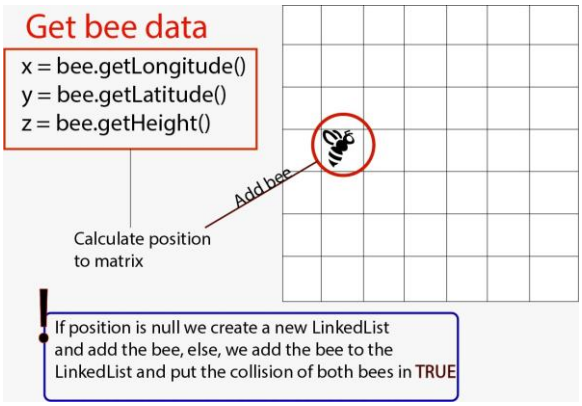

Figure 7: Process of adding bees to the matrix

4.1.3 Detect possible collisions between bees

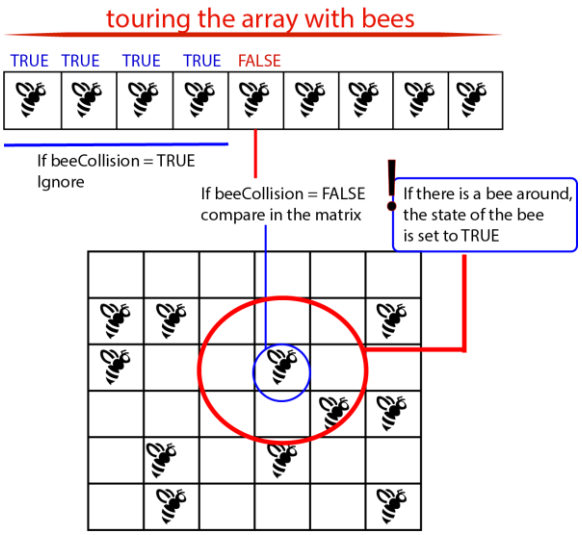

Figure 8: Process of detect possible collisions between bees

Note: This process is also done in heights, so all the cubes that are accessed in the radius of the bee conform a sphere made out of cubes.

4.2 Design criteria of the data structure

It is decided to use this data structure for the following reasons: It is a highly effective data structure for what is sought. It has a multiple implementation, not only works for the case exposed in the report. This structure compensates the memory expense with a high speed which is what is sought when collisions must be prevented. When you have millions of bodies at risk of collision, you should keep in mind that the first thing is to give quick answers. The reason for the low care with memory, is that this structure does not seek to keep data in an elongated manner, only seeks to alert potential collision risks in a repetitive manner.

4.3 Complexity analysis

Method	Complexity
Analysis and reading of data	O(n)
Add bees to matrix	O(n)

Detect possible collisions	O(n)
SaveFile	O(n)

Table 1: Complexity of operations of the data structure

4.4 Execution time in milliseconds

Data set (Number of bees)	Best time (ms)	Worst time (ms)	Average time (ms)
4	0	0	0
10	0	0	0
100	1	4	2
1000	7	9	8
10000	13	13	13
100000	35	37	36
1000000	195	198	196

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

4.5 Memory used in megabytes

Data set (Number of bees)	Best memory (mb)	Worst memory (mb)	Average memory (mb)
4	0	0	0
10	0	0	0
100	0	0	0
1000	0	0	0
10000	2	2	2
100000	4	4	4
1000000	25	28	26

Table 3: Memory used by the data structure for each data set.

4.6 Result analysis

As predicted in the execution time theoretically calculated, the time needed grows in a linear way (O (n)), so does the memory consumption. This is algorithm is very efficient considering that it can be even 10 times faster than an oct-tree in an average case, and much more faster in the oct-tree's worst case where all bees are inserted in one spot. It is important to mention that this data structure has no worst case, and its best case is actually, where all bees are in one spot, or in this case, in one cube.

6. CONCLUSIONS

We conclude that this data structure is efficient for an area where quick answers are needed. Therefore, it has been concluded that the little memory sacrifice that has been made is justified to achieve a necessary speed in the prevention of collisions. Really fast results were obtained in times for large amounts of data and it has been concluded that for the field of collision prevention this data structure can be much more efficient than using Octree or HashMaps. Finally, it is concluded that there was a great improvement compared to the first data structure that was used, since a complexity of O (n^2) could be reduced to O (n).

6.1 Future work

The work to be developed in the future for this data structure is to reduce memory consumption, being able to create a more dynamic matrix, in which a large number of empty spaces can not remain and thus save space in memory. In this way the structure would be effective and not only efficient. We also propose a structure that is adaptable for more areas in which large amounts of data are needed in a few seconds.

You can see our code in the next link:

https://github.com/jsperezsalazar2001/ST0245-032/tree/master/proyecto/codigo

REFERENCES

1. Yianilos, P. Data structures and algorithms for nearest neighbor search in general metrics spaces, (ND) 11 p., page 1.
2. Francis, E., Mendez, L., Sofrony, J. Collision-free path planning in free path planning in multi-dimensional environments, Ingeniería e investigación, VOL 31, 2011, pages 5-14.
3. Chen, J., Li, T. Incremental 3D Collision Detection with Hierarchical Data Structures, 1998, pages 139-141.
4. Rebollo, J., Maza, I. Ollero, A. Planning of Colision Free Trajectories for Multiple UAVs using the Speed Profile, Universidad de Sevilla, 2009, pages 2-6