# PREVENTING COLLISIONS BETWEEN ROBOTIC BEES USING DATA STRUCTURES AND ALGORITHMS

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 is the collisions that can occur in robotic bees in certain areas in which they are found. This problem is important because the proper functioning of this leads to a good life support, taking into account that the project is planned for a scenario in which the bees are almost extinct or in its entirety, and reduce project costs with respect to how much it can generate of waste that the bees produced collide with each other and become useless. The related problems are: Data structures and algorithms for nearest neighbor search in general metrics spaces, Collision-free path planning in free path planning in multi- dimensional environments, Incremental 3D Collision Detection with Hierarchical Data Structures and Planning of Collisions Free Trajectories for Multiple UAVs using the Speed Profile.

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

**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 is the collision that occurs between bees that are in operation. It is important to avoid this type of events, as it can produce large expenses and inefficiencies in what is sought by the project. By giving a solution to this problem, the effectiveness of the project of the robotic bees can be improved and they will have a more lasting use than previously planned.

## 3. RELATED WORK

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

They consider 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].

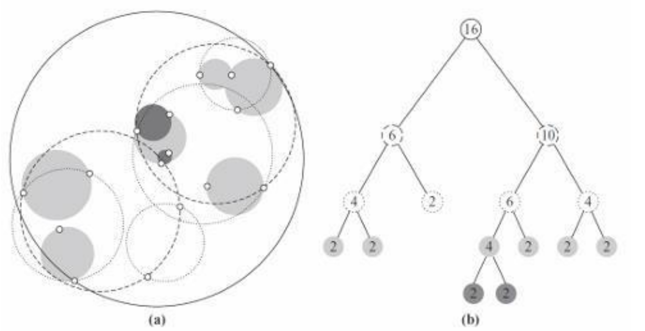


**Figure 1:** Analysis and process of vp-tree

### 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].

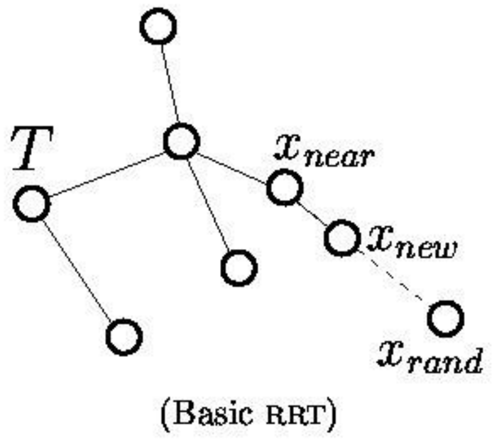


**Figure 2:** RRT basics procces

**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]

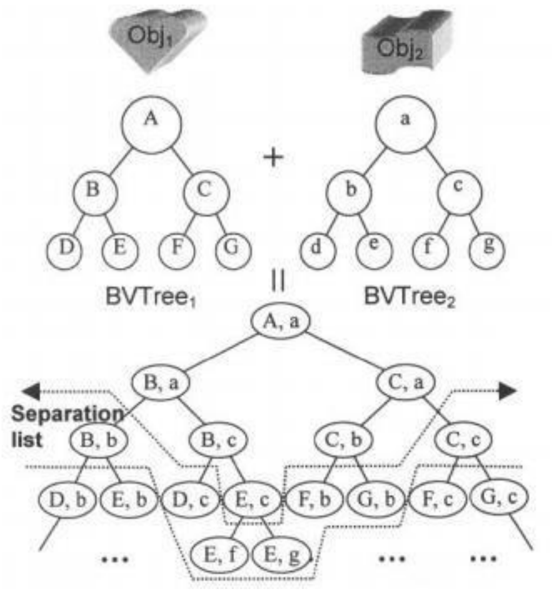


**Figure 3:** Comparison of two Bvtree with separation lists

**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]

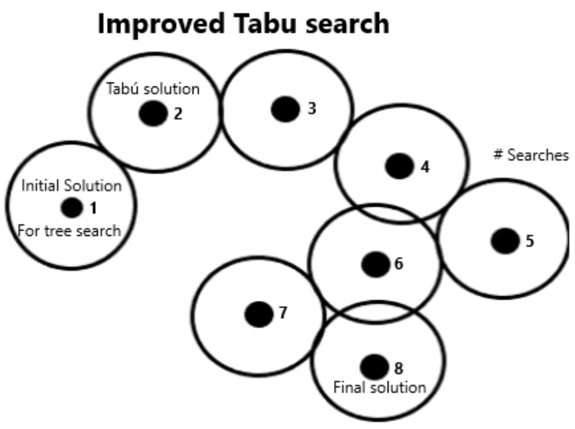


**Figure 4:** Improved tabú search with a tree search initial solution

## 4. Searching arrays

The selected data structure is "Searching arrays"; the following figure is the process of this algorithm.

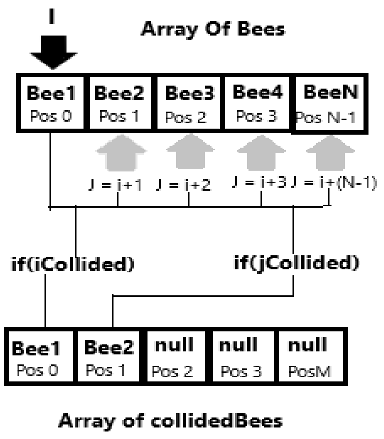


**Figure 5:** Prevention process of collided bees, from searching arrays.

### 4.1 Operations of the data structure

#### 4.1.1 Calculate the distance between two bees

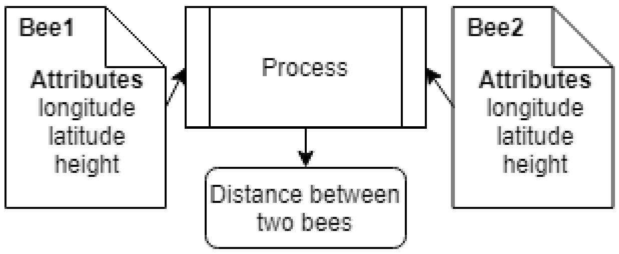


**Figure 6:** Process of calculate the distance between two bees

#### 4.1.2 Evaluate if a bee is in danger of collision

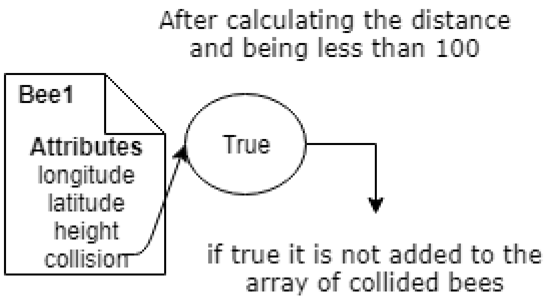

After calculating the distance and being less than 100
Bee1
Attributes
longitude
latitude
height
collision
True
if true it is not added to the array of collided bees

**Figure 7:** Process of evaluate if a bee is in danger of collision

#### 4.1.3 Add a bee to another array

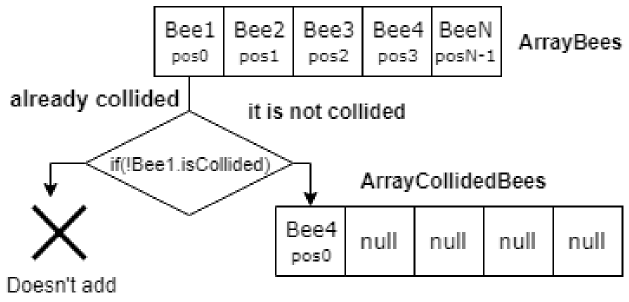


**Figure 8:** Process of add a bee to another array

### 4.2 Design criteria of the data structure

It is decided to use this data structure for 4 reasons, it is easy to interpret, easy to implement, it is efficient for small values and efficient in memory consumption. this helps the program in general to be efficient in the time it takes to execute the respective operations.

### 4.3 Complexity analysis

| Method | Complexity |
|---|---|
| Distance | O(1) |
| ReadFile | O(n) |
| DetectCollisions | O($n^2$) |
| SaveFile | O(n) |

**Table 1:** Table to report complexity analysis

**4.4 Execution time**

| Data set (Number of bees) | Best time (ms) | Worst time (ms) | Average time (ms) |
|---|---|---|---|
| 4 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 |
| 100 | 1 | 2 | 1 |
| 1000 | 6 | 8 | 6 |
| 10000 | 569 | 683 | 631 |
| 100000 | Undefined | Undefined | Undefined |
| 1000000 | Undefined | Undefined | Undefined |

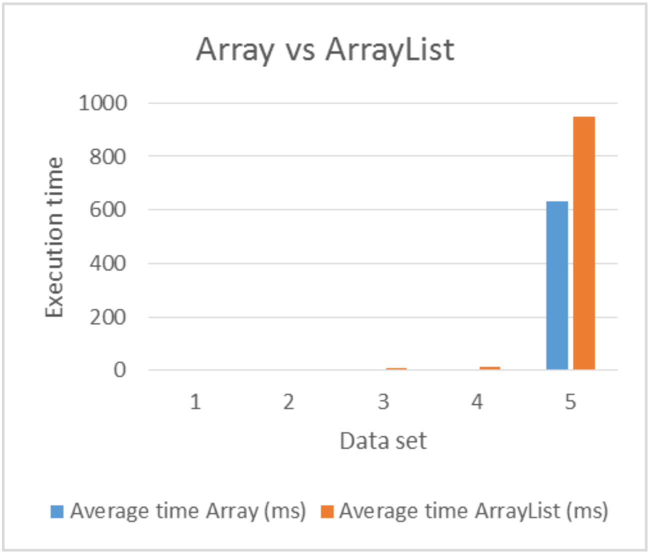
**Table 2:** Execution time of the data structure for each data set.

**4.5 Memory used**

| Data set (Number of bees) | Best memory (bytes) | Worst memory (bytes) | Average memory (bytes) |
|---|---|---|---|
| 4 | 75498 | 95486 | 79876 |
| 10 | 86256 | 102608 | 87560 |
| 100 | 98840 | 146548 | 91465 |
| 1000 | 624344 | 729936 | 694671 |
| 10000 | 1985088 | 2043568 | 2036378 |
| 100000 | Undefined | Undefined | Undefined |
| 1000000 | Undefined | Undefined | Undefined |

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

**4.6 Result analysis**



**Graphic 1:** Comparison of execution time between Searching Array and ArrayList.

**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.