ISSN 1870-4069

Path Planning Method for Navigation and Exploration with Drones Using the 3D-RRT Algorithm

Camilo Espinosa-Martinez[1], Lina Maria Aguilar-Lobo[1], Oscar J. Suarez[2], Ulises Davalos-Guzman[1], Gilberto Ochoa-Ruiz[3], Fabian Castaño[4], Alberto Ochoa-Zezzatti[5]

[1] Universidad Autónoma de Guadalajara, Mexico

[2] Universidad de Pamplona, Colombia

[3] Tecnologico de Monterrey, Mexico

[4] Pontificia Universidad Javeriana, Colombia

[5] Universidad Autónoma de Ciudad Juárez, Mexico

{camilo.espinosa, lina.aguilar, ulises.davalos}@edu.uag.mx, oscar.suarez@unipamplona.edu.co, gilberto.ochoa@tec.mx, fabian.castano@javerianacali.edu.co, alberto.ochoa@uacj.mx

Abstract. In the past few years, drone development has opened doors to new application areas giving us the chance to carry sensors onto more complex environments, such as cities and forests, in which there are proximity conditions of considerable size that could be categorized as highly dangerous. In this context, the path planning method is commonly used in robotics applications to find a valid sequence to move the drone to a target point. This method aims to find the shortest path length and obtain a safe trajectory avoiding collisions with obstacles. To achieve this objective, we present a novel path planning method for navigation and exploration with drones based on a 3D version of the RRT algorithm. The proposed algorithm developed in Python have two principal contributions, first are used a box model to encapsulate obstacles to avoid collisions and a dynamic range bias for the sampling, and, a giving orientation technique is employed on the exploration steps to reduce the number of computational operations and processing time when is obtained a valid path. Simulations results are performed to validate the algorithm using different scenarios and 3D obstacles randomly located. The results illustrated that the 3D-RRT algorithm finds a valid path avoiding obstacles with benefits on computational cost and better processing time.

Keywords: Path planning, 3D-RRT algorithm, navigation, drones applications.

1 Introduction

Path planning algorithm is a computational method consisting of generating a valid output path free of obstacles from a start point to a target point. Several path planning algorithms have been proposed in the literature [1-4], and in [5, 6] these algorithms were classified into three main categories:

– **Classical Methods**: They can find a solution or ensure a missing solution. However, these methods typically entail very costly and complex computational workloads. Therefore, they are not practical solutions for real environments [7].
– **Sampling-Based Methods**: They run sampling of configurations in space to model obstacles and possible paths. Different techniques have been implemented in past years, and the most relevant is Rapidly-Exploring Random Trees (RRT)[7].
– **Optimization Methods**: They try to solve the problem as a numeric optimization problem, this kind of algorithm start with a set of trajectories (it can be free of obstacles or not), then try to use optimization to find a solution for a valid path free of obstacles and optimal cost function, i.e., length, steps, time, among others. However, some of those cost functions usually have many local minimum [7].

Sampling-based methods are widely used because of their effectiveness and low computational cost on high dimensional spaces [8-11]. These methods use a representative configuration space and build a collision-free road map connecting points sampled from the obstacle-free space. One of the most well-known sample-based methods is the RRT algorithm proposed by [10].

Most RRT algorithm implementations have focused on path planning for holonomic systems in the 2D world, where successful solutions end in commercial products such as automatic vacuum, robots, and sweepers robots [12-15]. However, 3D point array processing is an essential task working in the 3D world to computer obtain a better description. Moving in a 3D space includes a challenge to move through space with obstacles at different heights, and the vehicle could set a path starting at low levels to go up and continue trajectories to reach the target suddenly. The novelty of the present paper is summarized as follows:

1. A path planning method for navigation and exploration with drones based on a 3D version of the RRT algorithm, which uses box encapsulation as a modeling tool to define obstacles in the 3D space. The advantage of this encapsulation is saving computing power and simplifying obstacle management to select the best trajectory avoiding obstacles from the start point to the target point in a 3D surface.

2. The implementation and validation of the proposed algorithm are performed using Phyton.

The rest of the document is organized as follows: In section 2, related works of path planing methods are presented. In Section 3 the classical RRT algorithm is provided.

Section 4 presents our model of the 3D-RRT algorithm, which is the main contribution of this paper. Simulation results are reported in Section 5, illustrating different scenarios. On the other hand, the discussion is presented in Section 6. Finally, conclusions and future works are drawn in Section 7.

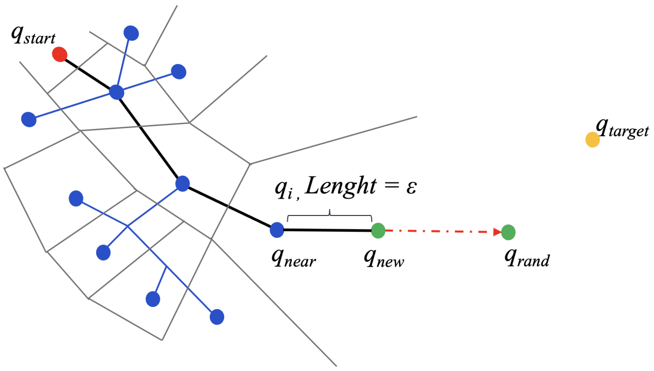


Fig. 1. RRT explore space to find a trajectory P from q_{start} to q_{target} composed by segments q_i, using random points and length ϵ.

2 Related Works

Recently, in the literature, several works related to methods for the path planing problem has been reported. In [18] is presented a path planing system for ground robots in 3D environments using point clouds as input using a 3D range sensors, presenting advantages in efficiency computation time and obtained a safety and effective path. The main reference here is use of special geometries to speed up the computational operations and reduce the subset of sampling space used by RRT algorithms.

In reference [19] is presented an improves RRT-connect algorithm for path planning for urban low altitude UAV with an improves RRT-connect algorithm, using a optimization to search step length, parent node selection and branch orientation to reduce the path length and algorithm time. However this technique is reduced to qualify new branches compared to an angle ideal for UAV. Our proposal is covering a wide range of solution, using searching direction focused around a vector in direction of target, not new point, this speed up the global search, not only new branches.

The work presented in [20] proposes a hybrid algorithm for path planing in complex offshore areas, using an improve of the particle swarm optimization (PSO) for global path planing and Artificial Potential Field (APF) to solve the local minimum problem. In [21] is presented three novel versions of the RRT algorithm with metaheuristics algorithms to solve 3D path planning problem in autonomous UAVs, where are employed the advantages of the two methods. These new hybrid models try to find solutions close to the optima, avoiding obstacles with a efficient execution time and space. However, the metaheuristic-based algorithms are disadvantaged as they demand a predetermined knowledge of intermediate stations.

Another work [22] proposes an improved version of B-RRT, named BPIB-RRT*, that employed a greedy connect a heuristic for the connection of two-directional trees. However, it is still not very successful in exploration. On the other hand, although it is recommended as a 3D path planning method, it is possible to use it mainly in 2D environments. This restriction is mainly valid to all B-RRT versions, and the results shown an execution time higher that our model.

Algorithm 1: RRT

```
1  Input: Initial Configuration with q_start and q_target located in space C ;
   Result: Path valid from q_start point to q_target point
2  τ ← INITTREE ( q_start)
3  while ¬ STOPCRITERIA do
4  |  q_rand ← RANDCONF()
5  |  q_new ← EXTEND (τ , q_rand )
6  |  if q_new = q_target then
7  |  |  return Path
8  |  else
9  |  |  return EMPTY
10 |  end
11 end
```

There are other approaches in the state of art that include dynamic considerations using point-mass model for cluttered environments in specific [23] propose a solution of 3 level to plan and find a feasible trajectory, in the third level use a modified RRT (SST) in a reduced sampling space, the current RRT 3D proposal could give and improvement in computational time for the third level even reducing more the searching volume.

3 RRT algorithm

Taken from [7], the path planing method considers a configuration C in $\mathbf{R^d}$ space where $C \subseteq \mathbf{R^d}$, then C contains all possible configurations in the space and C_{free} contains the set of configurations free of obstacles.

In this context, the state $q \in \mathbf{R^d}$ is the point in the configuration space that indicates the position and direction in the space C. Considering a trajectory P as a series of N configurations joining points q_i linked by $N-1$ segments, where each segment is a direct line segment from q_i to $q_i + 1$ represented by (q_i , q_{i+1}), this trajectory P is presented in equation (1) as follows:

$$P = \cup_{i=1}^{N-1}(q_i, q_{i+1}). \tag{1}$$

Based on equation (1), two points are selected (start point (q_{start}) and a target point (q_{target})). The purpose of RRT algorithm is to find a valid trajectory P (showed as black line in Figure 1), where all segments $q_i \in C_{\text{free}}$.

To achieve this purpose, RRT algorithm will build a tree with line segments found exploring the space of configurations C using q_{rand} points $\in C$ as can be seen in Figure 1.

Algorithm 1 shows the classical RRT proposed in [10]. This algorithm builds a tree of line segments that explores multiple regions of space C using randomly based functions to generate q_i configurations until they meet q_{target} with a good trajectory free of obstacles. The RRT algorithm receives the space C which contains q_{start} and q_{target} and returns a valid path P.

The algorithm initializes the search tree with root in q_{start} (line 1). Then a loop is used to explore over the space, searching the path free of obstacles. Line 2 shows the

Algorithm 2: EXTEND - RRT

1 **INPUT:** τ, q_{rand}
  Result: q_{new}
2 $q_{\text{near}} \leftarrow \text{NEAREST}(\ \tau\ ,\ q_{\text{rand}}\ )$;
3 $q_{\text{new}} \leftarrow q_{\text{near}} + (q_{\text{near}} + \dfrac{\mid {q_{\text{rand}}}_i - {q_{\text{near}}}_i \mid}{\epsilon})$
4 **if** $\text{VALIDSEGMENT}(q_{near}, q_{new})$ **then**
5   ADDSEGMENT
6   **return** q_{new}
7 **else**
8   **return** EMPTY
9 **end**

use of STOPCRITERIA function to stop the loop in one of two possible scenarios; the first one q_{target} has been met from q_{start}, and the second one is a maximum number of iterations in order to prevent infinite loop. Line 3 select a q_{rand} from valid space C, as possible next step in the trajectory. This value is used as a parameter for the EXTEND function. As result a q_{new} is added to tree τ (line 4).

If q_{new} is equal to q_{target}, the complete path has been founded (line 6 and line 7); otherwise, the process continues running until STOPCRITERIA function stops the algorithm. The second scenario in STOPCRITERIA function uses a maximum number of iterations as a parameter; it is usually set to a specific number of iterations (100, 300, 500, 1000, and others) if the algorithm does not find a valid trajectory under this maximum number of iteration returns an empty tree.

The EXTEND function details (line 4 in Algorithm 1) are show in Algorithm 2, where it finds the nearest point q_i to q_{rand} in τ (line 1), then a line segment from q_{near} to q_{rand} is calculated using a factor ϵ in direction of q_{target} (line 2) see Figure 1.

In Algorithm 2, VALIDSEGMENT function determines whether the line segment is hitting an obstacle in the possible trajectory (line 3). If the function finds an obstacle, an empty segment is returned; otherwise, the segment is added to τTREE. Some implementations of the RRT algorithm use a cost function to evaluate the quality of the segments, such as the length and energy consumption on a robot's trajectory; those functions can be used by VALIDSEGMENT function to determine possible segments free of obstacles meet cost function criteria to be added to τTREE.

4 Proposed 3D-RRT Algorithm

The RRT algorithms proposed by [10,17] use a randomized data structure for path planning and the main goal is to find a continuous path from initial point (q_{start}) to target point (q_{target}), where q_{start} and $q_{\text{target}} \in C$ that is the valid configurations space. All the paths found by the traditional RRT algorithm follow the equation (1), and obstacles are modeled as polygonal structures in the 2D space, then C is presented as a possible set of rectangle paths.

The path planning in 3D scenarios need to solve the problem of find a path from start point to target point using the 3D space, not only a 2D solution implemented in a

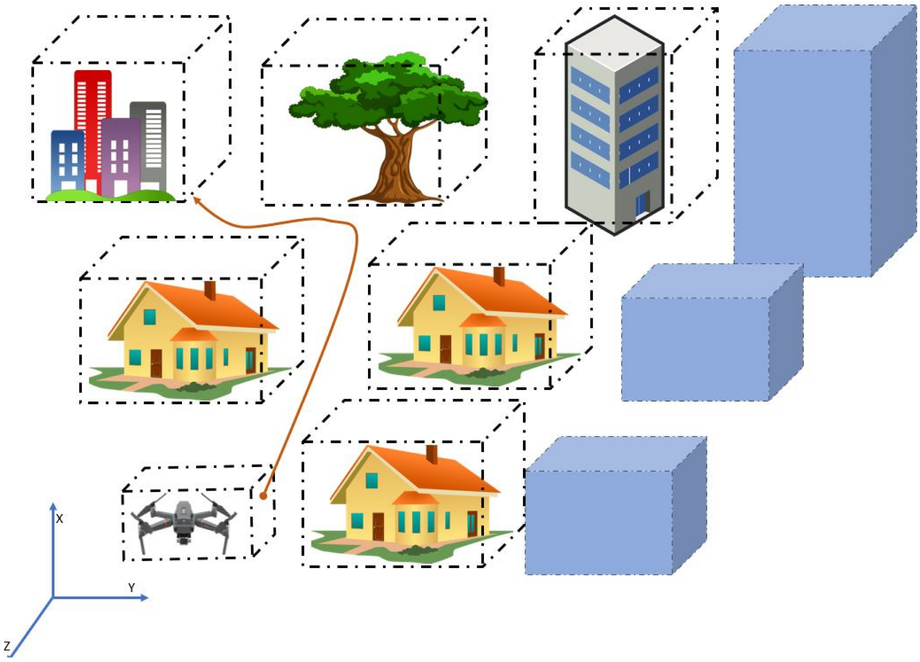


Fig. 2. Box model for obstacles.

specific Z coordinate (that allows robot's movement over the obstacles). That means to produce a real 3D solution path in the space C. In implementations for real world, the systems should be efficient in time, number of steps and iterations.

For example, a robot using 3D planning algorithm should consider the number of iterations and time to find a path as factors that consume computational power and energy. Moreover, in implementations of navigation with drones, the battery became a crucial resource, therefore, an algorithm that help to generate energy saving will be preferred over other implementations.

This paper proposes a novel path planning method for the navigation and exploration with drones based on the 3D version of the RRT algorithm using box models to encapsulate obstacles and the cost function of 3D vectorial distance as vector distance.

Our model is a solution that have the two factors became essential in the path planning method that are, the time and the iterations numbers to find a path free obstacles. Using box models to encapsulate obstacles, the cost function of 3D vectorial distance as vector distance and focus box of possible random points following vector distance to reduce time on the calculus of τTREE.

Our algorithm uses the information of the 3D space from q_{start} point to q_{target} point with various obstacles modeled as a series of rectangular boxes that can be used to set safe trajectories avoiding collisions even with not regular geometry or cubic style obstacles.

The box encapsulation allows different model obstacles from the real world with boxes to simplify finding a valid path in complex scenarios with multiple obstacles

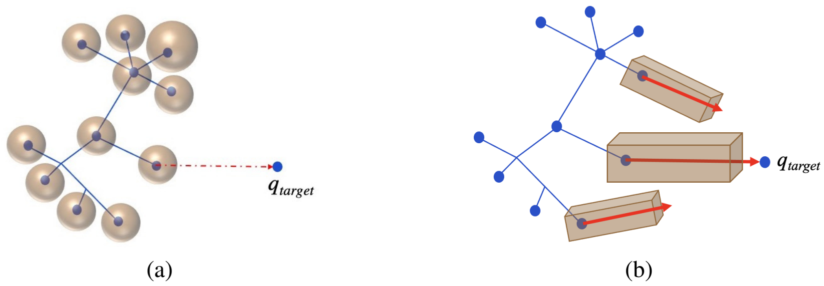


Fig. 3. Dynamic Range Proposed. (a) The Dynamic Range used in [17] uses ranges defined by spheres of radius R. (b) The Dynamic Range proposed in our 3D-RRT uses a box with length ϵ, focused in the direction of the target.

geometry. An example of the box encapsulation model in an urban environment is presented in Figure 2, where it can see that one box could contain a single house, a building, or a group of buildings.

Two features of the RRT algorithm proposed in [7] are considered in our proposal method, which is presented in Figure 4. The first refers to the minimal bias for random function using vector distance in the target direction to generate dynamic ranges that speed up the calculations and converge closer to the target point. The second one refers to an increment of minimal ϵ segments used in the proposed algorithm [7] with vectorial distance as cost function during calculations, allowing exploring the 3D space randomly.

Previous implementations of the RRT algorithms for robot and drone navigation in 3D spaces usually generate the path in the 2D plane then join it to the Z-axis, adding a vertical value. Although this approach can generate a valid route, it does not take advantage of most points through the 3D space available for other routes in multiple configurations approaches.

In this context, our proposal method presents a new form of calculating a valid path in a 3D space. Specifically, the main contributions of this paper are presented in lines 3 ($\mathrm{RANDCONF}$ function) and 4 (EXTEND function) of the RRT Algorithm 1 presented in [10]. The detail of EXTEND function used in our 3D-RRT proposed in the tree generation is presented in Figure 3.

The new 3D-RRT algorithm proposed are illustrated in the block diagram of Figure 4. Is important to say, that in $\mathrm{RANDCONF}$ were introduced the proposed modifications for the new 3D-RRT algorithm (green boxes in Fig4). The $\mathrm{RANDCONF}$ function generates coordinate values in each dimension in the 3D space. This function uses a range [m, n] to generate a pseudo-random value.

The proposed change considers the target's direction, using a vector pointing from the start point to the target point. The range is calculated for each dimension in space C on the target's bias, generating a cubic volume of random points in C. The start point, used to calculate the direction, is updated to the previous nearest value in each iteration; this keeps the quality of segment added.

Previous approaches used the dynamic domain for RRT [17], considering the volume of Voronoi regions with spheres of the previously defined radius. The proposed novelty

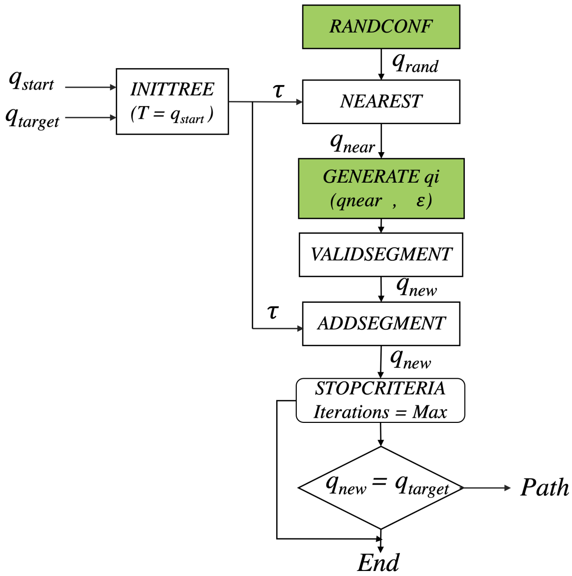


Fig. 4. Proposed 3D-RRT algorithm.

in our model speeds up the algorithm, reduces convergence over Voronoi volumes selected and proximal to the target point, reduces the number of computing operations to build trees; as a result, a worthy improvement on reducing wasting calculations over space in distant regions focus calculations.

The modified algorithm still converges as original RRT [9] without loose effective calculations. This new feature helps on drones implementations where the battery is an essential resource in 3D implementation, and at the same time is impacting in a reduction in the total number of computing operations, in consequence, the algorithm 3D-RRT contributes in the energy consumption saving energy from the battery.

On the other hand, the $\mathrm{VALIDSEGMENT}$ function implemented is used to determine if the segment is in collision with an obstacle. The path found q_{start} - q_{target} avoid obstacles using an encapsulation approach implementing boxes as a modular shape to run calculations.

The drone is encapsulated by a box that preserves the integrity and reduces complexity on calculations, as shown in Figure 2. A system of boxes also models obstacles, then the collision is determined by the calculation of box collision. Figure 2 shows an approximation of the box model and the drone encapsulated; in implementations of the proposed technique, a drone can use sensors to measure the height of the box, consequently the height of the obstacle.

For path planning in drones, it is necessary to use a couple of sensors above and under the chassis to detect the obstacle edges and then the box dimensions, observing that the flying or the grounded obstacles can be avoided using this approach.

The approach introduced in this paper uses a box model in the direction or target and dynamic range over this volume to run random exploration in space C. The $\mathrm{NEAREST}$ function will set the dimension of the box depending on q_{near} and q_{target}. This function

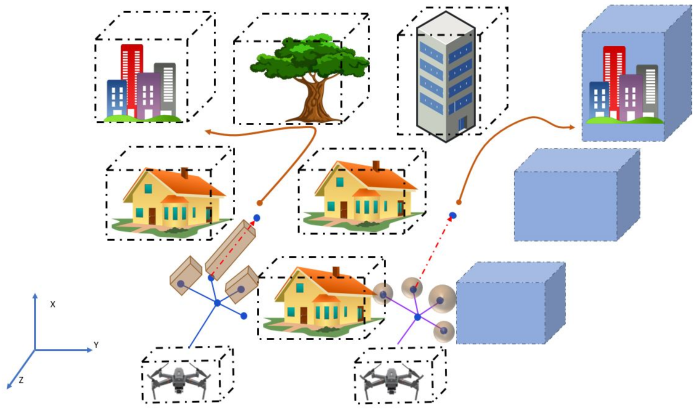


Fig. 5. Proposed 3D-RRT algorithm using boxes as modular shape.

will help to the algorithm search on the more likely and closer configuration. It is important to note that the dynamic box range occurs on each iteration on nearest q_i, so the algorithm keeps covering all space C, focusing on the target's direction.

Line 2 in Algorithm 2, shows detail of incremental ϵ from q_{near} calculated with vectorial distance to q_{rand}. In our model, the distance calculation and the random points are calculated using 3D coordinates in $\mathrm{RANDCONF}$ function, and q_{rand_i} points denote a component of each axis in C (where $i \in C$) were generated with dynamic box range technique as is showed in Fig 3. Thus, q_i points added to τTREE are closer to the distance vector, which represents an advantage in narrow spaces between obstacles.

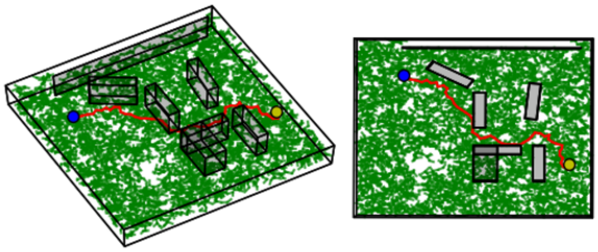
Using focused boxes to calculate new points in Tree and boxes to encapsulate obstacles help to processing the environment to generate a valid path, at first glance Figure 5 shows that dynamics boxes fits better the environments modeled with boxes, similar to maze of boxes, results found out a reductions in time to find a path and number of iteration needed; this approach could be used in real time systems where number of iterations and processing time are critical.

5 Simulation Results

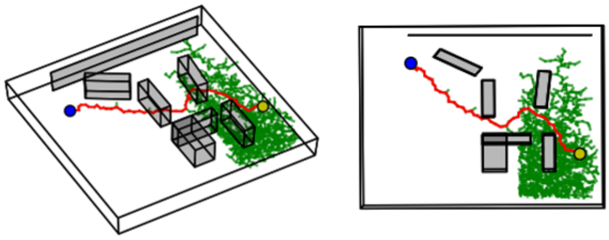
In order to validate the proposed 3D-RRT algorithm, simulations with different scenarios, different numbers of obstacles, and forms of 3D obstacles, which were randomly located, have been tested using Python.

The validation results for different scenarios and forms of 3D obstacles are presented in Figure 8. These figures show the comparison between the results of the RRT algorithm proposed by [10] (Figs. 5, 5 and 5) and our version 3D-RRT algorithm (Figs. 5, 5 and 5). The starting point (q_{start}) is shown in blue and the target point (q_{target}) is shown in yellow.

Note that the RRT algorithm proposed by [10] executes searching of new points to build paths in farther areas, even in regions previously explored, while the results of our

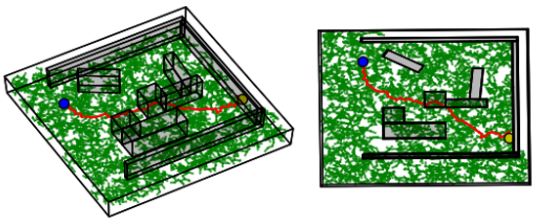


(a)

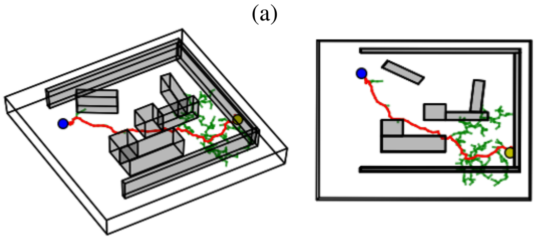


(b)

Fig. 6. Comparative between the results of the RRT algorithm of [10] and our proposed 3D-RRT algorithm for the case 1. Figs 5, 5 and 5 presented the results of RRT algorithm of [10], and Figs 5, 5 and 5 presents the results of our proposed 3D-RRT algorithm.



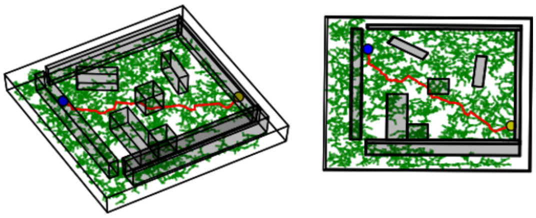
(a)



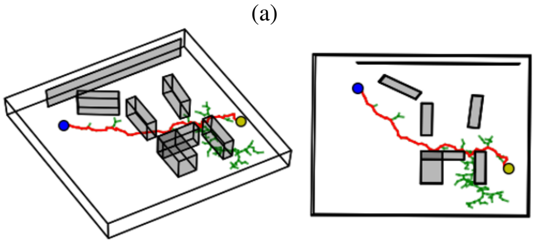
(b)

Fig. 7. Comparative between the results of the RRT algorithm of [10] and our proposed 3D-RRT algorithm for the case 2. Figs 5, 5 and 5 presented the results of RRT algorithm of [10], and Figs 5, 5 and 5 presents the results of our proposed 3D-RRT algorithm.

3D-RRT algorithm show the giving orientation on the exploring steps with the fixed boxes, it reduces the number of computational operations and processing time to obtain a valid path free of obstacles.



(a)



(b)

Fig. 8. Comparative between the results of the RRT algorithm of [10] and our proposed 3D-RRT algorithm for the case 3. Figs 5, 5 and 5 presented the results of RRT algorithm of [10], and Figs 5, 5 and 5 presents the results of our proposed 3D-RRT algorithm.

Table 1. Comparison of the performance between the RRT algorithm in [10] and our 3D-RRT proposal algorithm.

Test Cases		Time (seconds)		Iterations (number)	
Case	**obstacles**	**RRT**	**3D-RRT**	**RRT**	**3D-RRT**
1	7	204.3432	9.7121	7292	4413
2	9	45.2837	0.6609	3593	384
3	9	97.2242	1.0178	5466	556
4	6	86.2973	0.517	4557	276
5	6	160.2973	2.2086	6548	1084

Several simulation cases were executed to test our algorithm and the results are summarized in Table 1 where five cases are presented. It is important refer that although in some cases there is the same number of obstacles, the location of them is different.

This feature produces a different result in terms of performance of the algorithm, because in closes spaces is necessary more computational processing to obtain a valid path. However, we can see that in all test cases, our algorithm has a better performance in terms of execution time and number of iterations that the basic RRT algorithm.

6 Discussion

Based on the simulation results is possible to see that trees growth shows a strong bias in the target point's direction. No branches are exploring lateral edges over limits of space simulated or other areas with few probabilities to add a valid path; this approach avoids executing computational operations in regions most likely too far from the target.

The 3D exploration occurs between and over obstacles; it is not restricted to a 2D plane to increase probabilities to find valid paths. Simulations use different configurations of obstacles to show the response of the 3D algorithm to variations in space C.

The validation results of our method show that the 3D-RRT proposed model has better performance due to the reduction of the time execution and the reduction of the total number of computational operations. Although there are few works on other versions of the 3D-RRT algorithm, all of them explore areas with classic perspective, as mentioned before.

This could include wasting exploration time on farther areas. In our model, a valid path free of obstacles is found, and the use of the box encapsulation technique saves time and computational operations compared with the classic models. The proposed changes saves time and computational operations that produce energy saving in implementation in real world examples like use of drones with 3D-RRT algorithm.

7 Conclusions and Future Works

This paper proposes a novel path planning method using the RRT algorithm for 3D surfaces in Python. The validation results for different randomly located obstacles show that it successfully finds a good valid trajectory, avoiding collisions, and a higher reduction in the execution time and in the number of iterations or computational operations. The 3D exploration gives benefits on computational cost and improvement in narrowed spaces.

Furthermore, an encapsulation approach implements a box model for both obstacles, and the drone is used. The dynamic box encapsulation model generates a bias volume through the space with the most valuable points, as presented, using length as a function of cost. The box could be spread across different Voronoi regions without losing a random exploration feature, which is ideal for spaces like streets and forests.

Future Works could cover the implementations in not urban scenarios like forest, where terrain and obstacles became more irregular, in other hand use of algorithm proposed in laboratory scenarios could help to implement different sensors and strategies to obstacles sampling. The proposed algorithm use fixed obstacles, this mean size and number of obstacles don't change during path calculation, this became an limitation to consider in real implementations. This could be address with a sampling method that generate a fresh environment samples for real implementations but that analysis is out of scope for current proposal.

Acknowledgment. This work was funded by COECYTJAL (Consejo Estatal de Ciencia y Tecnología del Estado de Jalisco) with the found FODECIJAL, Project number 7817, and the UAG (Universidad Autónoma de Guadalajara).

References

1. Khan, A., Noreen, I., Habib, Z.: On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges. Journal of Information Science and Engineering, vol. 33, no. 1, pp. 101–121 (2017)

2. Nazarahari, M., Khanmirza, E., Doostie, S.: Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Systems with Applications, vol. 115, pp. 106–120 (2019). DOI: 10.1016/j.eswa.2018.08.008.
3. Radmanesh, M., Kumar, M., Guentert, P. H., Sarim, M.: Overview of path-planning and obstacle avoidance algorithms for UAVs: A comparative study. Unmanned systems, vol. 6, no. 2, pp. 95–118 (2018). DOI: 10.1142/S2301385018400022.
4. Wu, Q., Lin, H., Jin, Y., Chen, Z., Li, S., Chen, D.: A new fallback beetle antennae search algorithm for path planning of mobile robots with collision-free capability. Soft Computing, vol. 24, no. 3, pp. 2369–2380 (2020). DOI: 10.1007/s00500-019-04067-3.
5. Aguilar, W., Morales, S.: 3D environment mapping using the Kinect V2 and path planning based on RRT algorithms. Electronics, vol. 5, no. 4 (2016). DOI: 10.3390/electronics5040070.
6. Vagale, A., Oucheikh, R., Bye, R., Osen, O., Fossen, T.: Path planning and collision avoidance for autonomous surface vehicles I: A review. Journal of Marine Science and Technology, vol. 26, pp. 1292–1306 (2021). DOI: 10.1007/s00773-020-00787-6.
7. Garcia, N., Rosell, J., Suárez, R.: Aplicacion de algoritmos RRT en la aplicación de movimientos óptimos en robótica. In: Proceedings of the 12th Metaheuristics International Conference, pp. 953–962 (2017)
8. Chen, L., Shan, Y., Tian, W., Li, B., Cao, D.: A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic systems. IEEE/ASME Transactions on Mechatronics, vol. 23, no. 6, pp. 2568–2578 (2018) DOI: 10.1109/TMECH.2018.2821767.
9. Karaman, S., Frazzoli, E.: Incremental Sampling-based Algorithms for Optimal Motion Planning. Robotics Science and Systems VI, vol. 142, no. 2 (2010). DOI: 10.48550/arXiv.1005.0416.
10. LaValle, S.: Rapidly-exploring random trees: A new tool for path planning. Iowa State University, pp. 1–4 (1998)
11. Naderi, K., Rajamäki, J., Hämäläinen, P.: RT-RRT*: A Real-time Path Planning Algorithm Based on RRT*. In: Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, pp. 113–118 (2015). DOI: 10.1145/2822013.2822036.
12. Dong, Y., Camci, E., Kayacan, E.: Faster RRT-based Nonholonomic Path Planning in 2D Building Environments Using Skeleton-constrained Path Biasing. Journal of Intelligent and Robotic Systems, vol. 28, no. 3, pp. 387–401 (2018). DOI: 10.1007/s10846-017-0567-9.
13. Aiswarya, L. Chowdhury, A.: Human Aware Robot Motion Planning Using RRT Algorithm in Industry 4.0 environment. In: 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), pp. 351–358 (2021). DOI: 10.1109/ISR50024.2021.9419511.
14. Noreen, I., Khan, A., Asghar, K., Habib, Z.: A Path-planning Performance Comparison of RRT*-AB with MEA* in a 2-dimensional environment. Symmetry, vol. 11, no. 7, pp. 945–958 (2019). DOI: 10.3390/sym11070945.
15. Wang, J., Chi, W., Li, C., Wang, C., Meng, M.: Neural RRT*: Learning-based Optimal Path Planning. IEEE Transactions on Automation Science and Engineering, vol. 17, no. 4, pp. 1748–1758 (2020). DOI: 10.1109/TASE.2020.2976560.
16. Wang, J., Chi, W., Shao, M., Meng, M.: Finding a High-quality Initial Solution for the RRTs Algorithms in 2D Environments. Robotica, vol. 37, no. 10, pp. 1677–1694 (2019). DOI: 10.1017/S0263574719000195.
17. Yersova, A., Jaille, L., LaValle, S.: Dynamic-domain-RRTs Efficient Exploration by Controlling the Sampling Domain. In: Proceedings of International Conference on Robotics and Automation, pp. 3856–3861 (2005). DOI: 10.1109/ROBOT.2005.1570709.
18. Pérez-Higueras, N., Jardón, A., Rodríguez, Á., Balaguer, C.: 3D Exploration and Navigation with Optimal-RRT Planners for Ground Robots in Indoor Incidents. Sensors, vol. 20, no. 1 (2020). DOI: 10.3390/s20010220.

19. Jin, H., Cui, W., Fu, H.: Improved RRT-connect Algorithm for Urban Low-altitude UAV Route Planning. Journal of Physics: Conference Series, vol. 1948 (2021). DOI: 10.1088/1742-6596/1948/1/012048.
20. Wang, Z., Li, G., Ren, J.: Dynamic Path Planning for Unmanned Surface Vehicle in Complex Offshore Areas Based on Hybrid Algorithm. Computer Communications, vol. 166, pp. 49–56 (2021). DOI: 10.1016/j.comcom.2020.11.012.
21. Kiani, F., Seyyedabbasi, A., Aliyev, R., Gulle, M. U., Basyildiz, H., Shah, M. A.: Adapted-RRT: Novel Hybrid Method to Solve Three-dimensional Path Planning Problem Using Sampling and Metaheuristic-based Algorithms. Neural Computing and Applications, vol. 33, pp. 15569–15599 (2021). DOI: 10.1007/s00521-021-06179-0.
22. Wu, X., Xu, L., Zhen, R., Wu, X.: Biased Sampling Potentially Guided Intelligent Bidirectional RRT Algorithm for UAV Path Planning in 3D Environment. Mathematical Problems in Engineering, vol. 2019 (2019). DOI: 10.1155/2019/5157403.
23. Penicka, R., Scaramuzza, D.: Minimum-time Quadrotor Waypoint Flight in Cluttered Environments. IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 5719–5726 (2022). DOI: 10.1109/LRA.2022.3154013.