2022 7th International Conference on Automation, Control and Robotics Engineering (CACRE) | 978-1-6654-0387-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/CACRE54574.2022.9834188

Adaptively Dynamic RRT*-Connect: Path Planning for UAVs Against Dynamic Obstacles

1st Yicheng Chen
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
yicheng@buaa.edu.cn

2nd Lingling Wang* (Corresponding author)
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
wangling0908@buaa.edu.cn

***Abstract*—Path planning for Unmanned Aerial Vehicles (UAVs), especially in three-dimensional environments with dynamic obstacles, is an active area of research. In recent years, RRT-based algorithms have been attracting significant interest, and various improvements are introduced to RRT to make it applicable in such scenarios. Some methods leverage re-planning mechanisms to avoid collision with the dynamic threats, however, most of them fail to fully reuse historical information, hence having limitations in saving the planning costs. This paper presents Adaptively Dynamic RRT*-Connect (ADRRT*-Connect), a novel RRT-based path planning algorithm that enables UAVs to fly safely in three-dimensional environments with dynamic threats. To improve efficiency in sampling new nodes, we propose a strategy to automatically adjust the heuristic factor based on feedback from the sampling results. For avoiding collision with dynamic threats, we introduce a pruning-reconnecting mechanism to repair the path when new obstacles emerge. Our approach is economical in the consumption of tree nodes. In comparison to existing benchmarks, simulations have shown that our proposed algorithm only requires 3.5% new nodes to repair the path in re-planning.**

***Index Terms*—Unmanned Aerial Vehicle, Path Planning, Rapidly-exploring Random Tree, Dynamic Environment**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have been attracting growing research interests due to their wide applications on areas including photography, delivering [1], mapping [2], and mobile edge computing [3]–[5], etc. As the operating range of UAVs gradually extends to lower altitudes with complex surroundings, dense obstacles and dynamic threats from three-dimensional environments may pose challenges to the safety of UAVs. Path planning aims at finding a path from the start point to the goal point while protecting the UAVs from collision with the obstacles [6]. Accordingly, how to perform dependable path planning for UAVs emerges as an topic worth discussion.

Various path planning methods have been proposed to solve the above issue such as A* algorithm [7], Artificial Potential Field [8], and Probabilistic Roadmaps [9]. Among diverse path planning methods for UAVs, Rapidly-exploring Random Tree (RRT) [10]—a sampling-based search algorithm—attracts considerable attention. Compared with other algorithms, RRT has advantages for it seldom falls into local optimum and it does not require discretization of the configuration space [11], which makes it an ideal algorithm for three-dimensional path planning for UAVs. Since RRT was proposed [12], there have been many related advances. For example, J.J. Kuffner [13] improves the efficiency of RRT by incrementally building two rapidly-exploring random trees rooted at the start and the goal configurations. C. Urmson [14] utilizes a heuristic quality function to guide the search which improves the speed of finding a path. Sertac Karaman proposes RRT* [15], the first asymptotically optimal RRT which analyzes the behavior of the cost of the solution returned by stochastic sampling and applies modification on the random tree generated. More extensions and improvements have been presented in [16]–[20].

Most previous variants of RRT cannot be directly used in dynamic planning, however, in real-world scenarios information about the environment is often incomplete and updated with time. When UAVs encounter emerging obstacles, to avoid collision, a straightforward approach [21], noted as traditional re-planning method, is to regard the changed surroundings as brand-new, discard the old trees and re-run a full planner. This method, ignoring the historical information obtained in previous planning, tends to be computationally-expensive for it often takes many samples before finding a feasible solution. Therefore, there is an interest in developing algorithms that reduce the cost of re-planning through information reuse. To address such problem, Olzhas Adiyatov proposes RRT*D [21], which retains the useful part of the tree after a dynamic obstacle breaks the solution path. The key idea of RRT*D is to remove only part of the nodes and try to repair the broken path instead of re-generating a whole path. Although RRT*D greatly reduces the computational cost, its unidirectional reconnecting mechanism limits its efficiency.

Aiming at addressing drawbacks in existing approaches, we propose Adaptively Dynamic RRT*-Connect (ADRRT*-Connect) to advance research in path planning for UAVs in three-dimensional environments with dynamic obstacles. ADRRT*-Connect applies an adaptive sampling strategy to automatically guide the growth of the random tree, thus improving the efficiency of the heuristic. A bidirectional growth and reconnecting mechanism allows ADRRT*-Connect to efficiently reuse previous search efforts in re-planning. Therefore, ADRRT*-Connect requires fewer samples before finding a feasible solution compared with current re-planning methods.

II. Related Work

A. Static planning algorithm

The basic RRT explores the configuration space by growing a tree rooted at the start node p_{init}, as is visualized in Figure 1. At each step, RRT randomly chooses a point p_{rand} within the configuration space and tries to grow the tree towards p_{rand} by a distance Δp. If the growth is not baffled by obstacles, a new node p_{new} is added to the tree. This process is repeated until the tree touches the goal point p_{goal}.

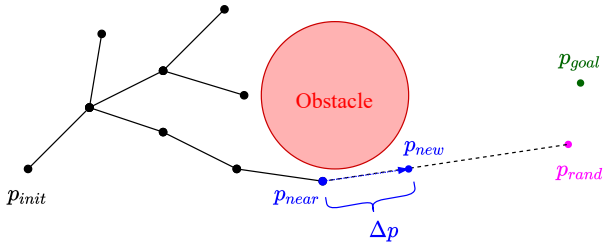


Fig. 1. RRT expands a new node.

RRT*, extending RRT to provide almost-surely asymptotically optimal solutions, uses `Extend*` to grow the tree, as is shown in Algorithm 1. When a new node p_{new} is added to tree T, nodes within a certain area from p_{new} are selected as P_{near}, then `ReselectParent` and `Rewire` are performed to optimize the connection. `ReselectParent` selects the best parent for p_{new}. Then `Rewire` examines each $p_{near} \in P_{near}$, and re-sets p_{near}'s parent to p_{new} by `SetParent` if such change can reduce the path cost from p_{init} to p_{near}.

Algorithm 1 Extend*

Function `Extend*` (T, p_{rand})
$p_{nearest} \leftarrow$ `Nearest` (T, p_{rand})
$p_{new} \leftarrow$ `Steer` $(p_{nearest}, p_{rand})$
if `ObstacleFree` $(p_{new}, p_{nearest})$ **then**
  if `DisCost` $(p_{new}, p_{goal}) < threshold$ **then**
    $p_{new} \leftarrow p_{goal}$
    $T \leftarrow$ `addVertex` (T, p_{new})
    $Flag \leftarrow Reached$
    $T \leftarrow$ `addVertex` (T, p_{new})
    $P_{near} \leftarrow$ `findNeighbor` (T, p_{new}, r)
    $T \leftarrow$ `ReselectParent` (T, p_{new}, P_{near})
    $T \leftarrow$ `Rewire` (T, p_{new}, P_{near})
    $Flag \leftarrow Advanced$
  end if
  $Flag \leftarrow Trapped$
end if
Return $Flag$

Since RRT* first achieves asymptotically optimal, later researches [20], [22] focus on improving the speed of finding the initial solution and converging towards the optimum. Of various improvements, one representative advance is the conversion of unidirectional search to bidirectional search, such as RRT*-Connect [23]. RRT*-Connect combines the ideas of RRT* and RRT-Connect. It uses two separate trees T_a and T_b initialized respectively at the start and goal states. If an extend step is valid and a new node p_{new} is created, the algorithm tries to grow the other tree towards the newly generated node until the growth is baffled by obstalces, as is shown in Algorithm 2. RRT*-Connect alternately grows two trees towards each other, hence finds paths more quickly than RRT*.

Algorithm 2 Connect*

Function `Connect*` (T, p_{target})
while `Extend*` $(T, p_{target}) = Advanced$ **do**
  `Extend*` (T, p_{target})
end while
if $Flag = Reached$ **then**
  $\sigma \leftarrow$ `ConnectTree` $()$
end if

B. Dynamic re-planning algorithm

RRT*D [21] is a highly efficient path planning algorithm which can be used in dynamic environments. It aims at saving the re-planning cost. Traditional re-planing methods discards the whole old trees, consequently loses information including results from obstacle collision detection routines and exploration of the configuration space by the tree. Thus, RRT*D tries to reuse such information.

As shown in Algorithm 3, initially, RRT* is executed, and tree T and path σ are generated. Afterwards, the UAV starts to move, and it's current position $p_{current}$ is synchronously updated. If an obstacle is detected to break any path segment from $p_{current}$ to p_{goal}, as is shown in Figure 2, the movement is temporarily suspended, then comes the re-planning process, which is the key part of the algorithm. `SelectBranch` discards the nodes from p_{init} till $p_{current}$ and their offspring, thus creating a subtree rooted at $p_{current}$, which is called T_{parent}. `ValidPath` focuses on the previous solution path and removes all the nodes colliding with the obstacle and their offspring, as a result, retains the usable part of the solution path connected to p_{goal}. This part is named $\sigma_{separate}$, starting at $p_{separate}$ and ending at p_{goal}. In `Regrow`, T_{parent} grows until connecting with $\sigma_{separate}$, then the broken path is repaired.

When evaluating the reconnecting mechanism of RRT*D, it is quite natural to find out that it has two main limitations:

- The reuse of valid information is not sufficient. When discarding nodes, the previous tree structure is only half retained: The first half of the tree ($p_{current}$ till obstacle) is retained, however, the last half of the tree (obstacle till p_{goal}) is almost totally abandoned, only the last half of the path $\sigma_{separate}$ is retained. Namely, the exploration information of the last half of the tree is mostly lost.
- The growing strategy of the tree is relatively less efficient. Compared with bidirectional searching algorithms such as RRT-Connect, RRT*D uses a rather inefficient grow strategy to lead the growth of the tree, which results in

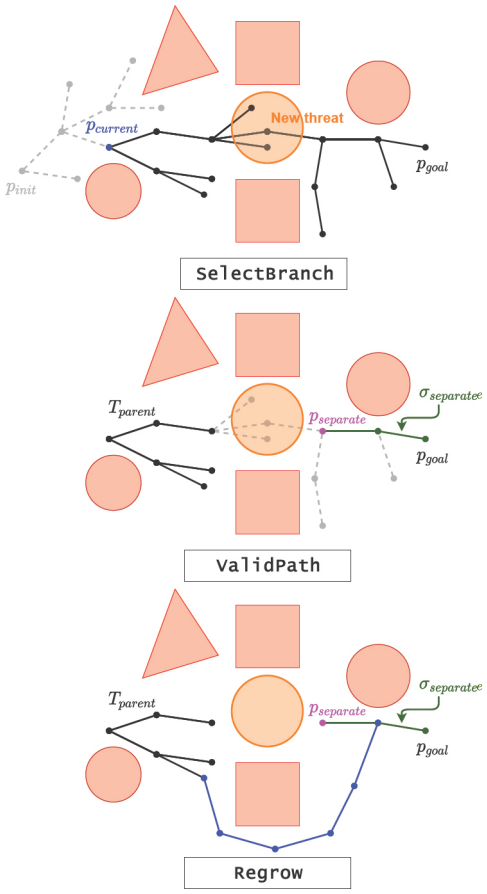


Fig. 2. Visual illustrations of the three key functions in RRT*D.

Algorithm 3 RRT*D

```
1: T, σ ← RRT*(p_init)
2: p_current ← p_init
3: InitMovement()
4: while p_current ≠ p_goal do
5:     D ← UpdateObstacles()
6:     if DetectCollision(σ, p_current) then
7:         StopMovement()
8:         T ← SelectBranch(T, p_current)
9:         p_separate, σ_separate ← ValidPath(σ)
10:        T ← Regrow(T, p_separate)
11:        σ ← SolutionPath(T, p_current)
12:        ResumeMovement()
13:    end if
14:    p_current ← NextNode(σ)
15: end while
```

more computational cost both in the initial grow phase and the regrow phase.

Based on the analysis of these baseline algorithms, RRT*-Connect is highly efficient in growing the tree but cannot be directly used in dynamic re-planning, while RRT*D provides a cost-saving strategy for re-planning against emerging obstacles but is limited in efficiency due to its reconnecting mechanism. Inspired by these algorithms, we introduce a highly efficient reconnecting mechanism, and propose a novel adaptive sampling strategy, thus form a path planning algorithm which can be used in dynamic environments. We name the proposed algorithm as Adaptively Dynamic RRT*-Connect (ADRRT*-Connect).

III. Adaptively Dynamic RRT*-Connect

A. Adaptive sampling

Sampling is the key procedure of RRT and its variants. There is no heuristics in RRT when expanding new node. In algorithms like Goal-bias RRT [14], heuristics are introduced to the process of expanding new node: When sampling in the configuration space, the probability of directly sampling the goal point (this behavior is noted as greedy sampling) is ϵ and the probability of randomly sampling is $(1-\epsilon)$. ϵ is the heuristic factor and is constant in current algorithms. We introduce sigmoid function and use $y(x)$ instead of the constant factor ϵ, and propose a mechanism to adaptively adjust $y(x)$ based on feedback from the results of tree growth, thus form the idea of adaptively sampling. The corresponding pseudocode is shown in Algorithm 4 and 5. In Algorithm 4, `GreedyProbability` corresponds to the function $y(x)$, which is

$$y(x) = \frac{1}{1+e^x}. \tag{1}$$

Algorithm 4 AdapSample

```
1: Function AdapSample(x)
2: ε ← GreedyProbability(x)
3: if rand(1) < ε then
4:     p_rand ← GreedySample()
5:     SampleType ← Greedy
6:     p_rand ← RandomSample()
7:     SampleType ← Random
8: end if
9: Return p_rand, SampleType
```

In order to adaptively adjust y, we set the following rule to adjust x (line 10-19 in Algorithm 5): Each time we apply greedy sampling in tree expansion, if there are no obstacles between the newly-sampled node and the nearest node, we increase x by Δx, otherwise we decrease x by Δx. Δx is a hyperparameter and is set empirically. The motivation is that if a greedy sampling is successful, it means that there are no obstacles between the nearest node and the goal point, and thus the probability of greedy sampling is supposed to be increased so as to reach the goal point as soon as possible or hit an obstacle and thus bypass it.

The above mechanism uses the results of the tree growth to automatically adjust x. x is initialized as 0 in each mission. There are two main advantages of using sigmoid function to map x to y:

- The sigmoid function converts x, which range is $(-\infty, +\infty)$, to y, which range is $(0, 1)$ and can be used as a probability of an event.

Algorithm 5 ARRT*-Connect

Function ARRT*-Connect(T_a, T_b)
if isempty(T_a) **then**
  $T_a \leftarrow$ InitTree(p_{init})
end if
if isempty(T_b) **then**
  $T_a \leftarrow$ InitTree(p_{goal})
end if
while $Attempts \leq MaxAttempts$ **do**
  $p_{rand}, SampleType \leftarrow$ AdapSample(x)
  if Extend*(T_a, p_{rand}) $\neq Trapped$ **then**
    if $SampleType = Greedy$ **then**
      $x \leftarrow x + \Delta x$
    end if
    Connect*(T_b, p_{new})
    if $SampleType = Greedy$ **then**
      $x \leftarrow x - \Delta x$
    end if
  end if
  Swap(T_a, T_b)
end while
Return T_a, T_b, σ

- y can react to the changes of x with different sensitivity at different stages of tree growth. When x is near to 0, the value of y is sensitive to the change of x, when x gains a large absolute value, y tends to saturate, i.e., y is relatively insensitive to change of x. Such adaptive sensitivity is proved to guarantee more efficient sampling.

The effectiveness of adaptively sampling is validated by simulation in Section IV.

B. Pruning-reconnecting mechanism

In order to apply RRT-based algorithms in dynamic environments where there are emerging obstacles, we propose a pruning-reconnecting mechanism to flexibly guide the growth of the random tree, as is shown in Algorithm 6.

Algorithm 6 ADRRT*-Connect

1: $T_a, T_b, \sigma \leftarrow$ ARRT*-Connect([ ],[ ])
2: $p_{current} \leftarrow p_{init}$
3: InitMovement()
4: **while** $p_{current} \neq p_{goal}$ **do**
5: $D \leftarrow$ UpdateObstacles()
6: **if** DetectCollision($\sigma, p_{current}$) **then**
7: StopMovement()
8: $T_a \leftarrow$ RemoveNodes($T_a, p_{current}$)
9: $T_b \leftarrow$ RemoveNodes(T_b, p_{goal})
10: $T_a, T_b, \sigma \leftarrow$ ARRT*-Connect(T_a, T_b)
11: ResumeMovement()
12: **end if**
13: $p_{current} \leftarrow$ NextNode(σ)
14: **end while**

If emerging obstacles break the path, RemoveNodes is executed. This function deletes the nodes and their offspring in the tree which are in collision with the obstacles. Thus T_a and T_b are pruned. By doing this, the useful information in the two trees are retained as much as possible. After RemoveNodes is executed, the broken path can be repaired by ARRT*-Connect, as detailed in III-A. The workflow of ADRRT*-Connect is depicted in Figure 3.

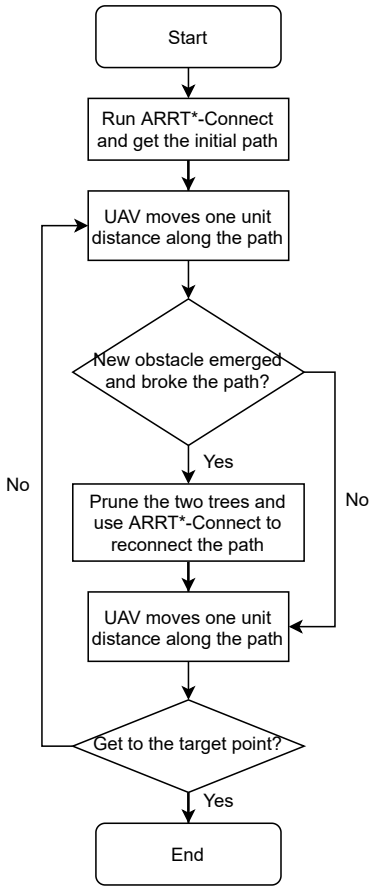


Fig. 3. Workflow of ADRRT*-Connect.

The comparison of ADRRT*-Connect and RRT*D is illustrated in Figure 4. The proposed ADRRT*-Connect differs from RRT*D in the three following aspects:

- Our proposed algorithm leads a bidirectional growth of two trees simultaneously, as is shown in step 1, while RRT*D grows only one tree from the start point towards the goal point. Our bidirectional growth mechanism guarantees faster planning speed.
- When emerging obstacles break the path, as demonstrated in step 2-3, our proposed algorithm deletes the tree nodes that collide with the obstacles, whereas in the same circumstance RRT*D deletes more nodes (the grey ones). Our algorithm reuses more historal information.
- When reconnecting, as is shown in step 4, in our algorithm, the path can be rebuilt as soon as the two trees join together. However, in RRT*D, the first half of the

tree must connect to the remained path on the right. Such unidirectional search is less efficient.

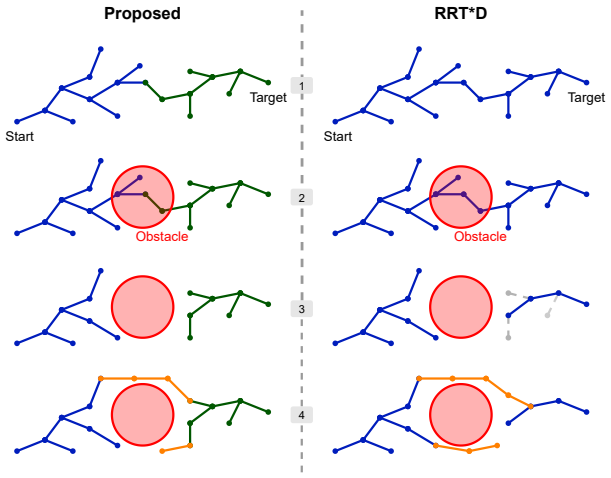


Fig. 4. The comparison of our proposed algorithm and RRT*D.

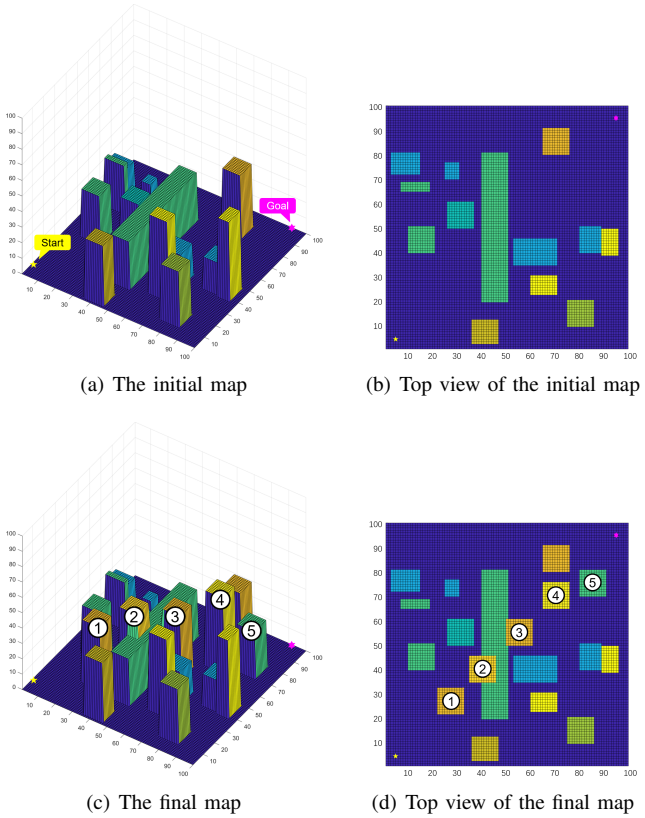
IV. Simulation and Discussion

When applying a path planning algorithm for a real-world object like a UAV, the general approach to perform the collision check is to inflate the obstacles, regard the UAV as a volumeless point, and examine the interaction of the point and the expanded obstacles. In the following simulations, the obstacles are regarded as inflated so that we can treat the UAV as a volumeless point.

The simulation conditions for the series of experiments are set as follows: In a three-dimensional map with a size of $100 \times 100 \times 100$, a series of static obstacles are distributed, as is visualized in Figure 5(a). When the simulation starts, an initial path is first generated, and then the UAV starts to move. During the movement of the UAV, five dynamic threats will arise one after another. Figure 5(c) shows the map where all five dynamic threats have emerged. The UAV cannot anticipate the geometric information of the obstacles. After one threat appears, the UAV immediately obtains the geometric information of the obstacle, and the obstacle will no longer change location. Due to the emergence of five dynamic threats, the UAV may perform 0-5 re-planning in one mission: When the initial path does not conflict with all 5 dynamic obstacles, there is no need to perform re-planning. Whenever an obstacle breaks the current path, a re-planning is executed. The UAV follows the initial solution of each re-planning and moves at a constant speed across the whole mission. The time consumption of getting the initial solution path is negligible.

A. Path planning in dynamic environments

A straightforward demonstration of a mission with five times of re-planning is presented in Figure 6. This figure shows that our proposed algorithm is able to perform dependable path re-planning in environments with dense static obstacles and dynamic threats. Each time a new obstacle appears and



(a) The initial map (b) Top view of the initial map

(c) The final map (d) Top view of the final map

Fig. 5. Illustrations of the initial map and the map where all dynamic obstacles have emerged. Markers 1-5 represent the order in which the obstacles appear.

breaks the current path, ADRRT*-Connect is able to repair the path and guide the UAV to continue to move towards the goal point.

B. Effectiveness of adaptive sampling

The planning cost is reflected in two aspects, the sampling times (number of attempts) and the sampling success rate. We define the sampling success rate η as the ratio of number of tree nodes to total sampling times. A larger η means more efficient sampling because it costs fewer times of attempts to build a tree of the same scale compared with a smaller η.

To verify the effectiveness of the adaptive sampling proposed in section III-A, we compare the planning results from ADRRT*-Connect when the adaptive mechanism is enabled and disabled separately. In the group where adaptive sampling is enabled, x is initialized as zero and Δx is set to 0.3. In the group where adaptive sampling is disabled, $\epsilon \equiv 0.5$. In the same condition, 1000 independent replicate tests are carried out separately for each group. Figure 7 shows the comparison of planning results when adaptive sampling is enabled and disabled. We notice that, except for the first re-planning, the adaptive mechanism allows our algorithm to use fewer sampling tries while achieving a higher sampling success rate. On average, when adaptive sampling is applied, ADRRT*-Connect requires 6.7% fewer times of sampling and produces 7.3% greater sampling success rate than that

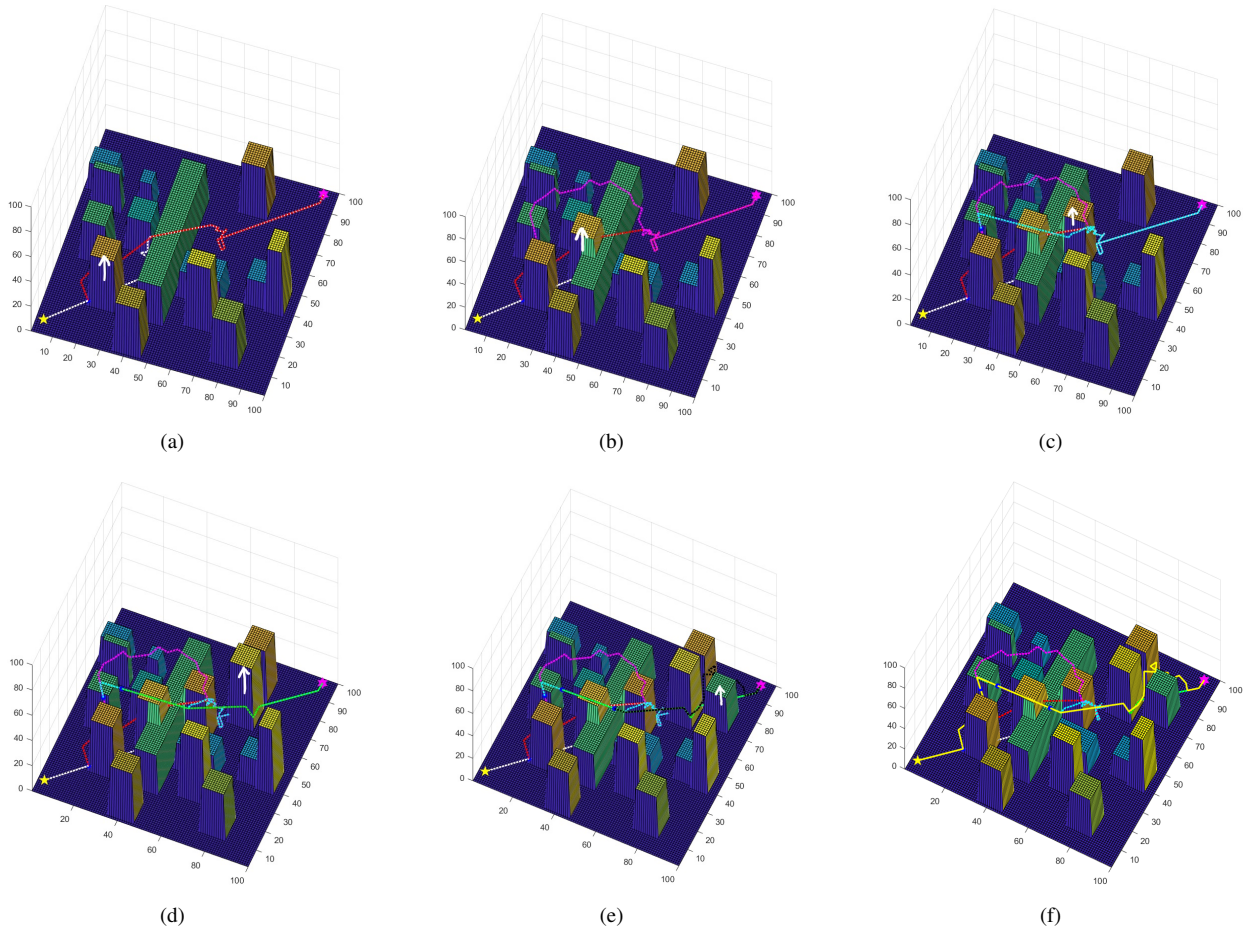


Fig. 6. The whole procedure of a mission in which five re-planning are executed. (a) First re-planning. (b) Second re-planning. (c) Third re-planning. (d) Fourth re-planning. (e) Fifth re-planning. (f) Final results. The yellow five-pointed star marks the start point and the magenta six-pointed star marks the goal point. The white arrow marks the newly emerged obstacle each time. The initial path is in white dotted line and the final path is in yellow line. In each re-planning, the updated path is plotted in red, magenta, cyan, green, and black, respectively.

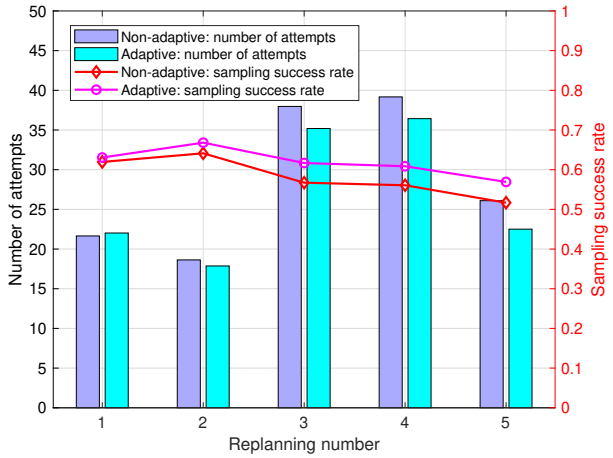


Fig. 7. Planning results comparison of ADRRT*-Connect when adaptive sampling is turned on and off separately.

without adaptive sampling. Such reduction in sampling times means that the adaptive mechanism is conducive to less computational burden, which is of vital significance in real-world scenarios because on-board computers are often limited in processing resources. The effectiveness of the proposed adaptive mechanism is verified.

C. Comparison of re-planning cost

In re-planning, the number of newly sampled tree nodes is an important metric to evaluate the re-planning efficiency. The fewer the new nodes, the better the exploitation of existing information. To verify the advantages of ADRRT*-Connect in terms of re-planning efficiency, We compare ADRRT*-Connect, RRT*D [21], and the traditional re-planning algorithm [21] in the same scenario. Due to the random factor's influence in the algorithms, 1000 independent replicate trials were performed for each algorithm, and the average number of new nodes in the random tree at five re-planning periods was calculated for each algorithm. The comparison results of the three algorithms are shown in Table I.

In Table I, each row is the average result of 1000 independent replicate trials. Based on the results, with our proposed ADRRT*-Connect, whenever the dynamic threat breaks the

TABLE I
NUMBER OF NODES REQUIRED IN FIVE TIMES OF RE-PLANNING WITH TRADITIONAL RE-PLANNING METHOD, RRT*D AND OUR PROPOSED ALGORITHM

Obstacle	Traditional re-planning method	RRT*D	Proposed
1	84	135	14
2	119	203	12
3	69	560	22
4	250	426	22
5	538	1086	13
Average	212	482	17

current path, the random tree only needs to add a modest number of new nodes to restore the connection. This feature benefits from the sufficient re-use of historical information in the pruning-reconnecting mechanism. On average, the proposed algorithm only costs 3.5% of new nodes compared with RRT*D, indicating that our approach builds a much lighter tree and consumes less memory compared with the baseline algorithms. The effectiveness of our approach is clear.

V. CONCLUSION

In this paper, we introduce ADRRT*-Connect, a novel path planning algorithm for UAVs against dynamic obstacles in three-dimensional environments. To break the limitations of existing algorithms that information reuse is insufficient, we make the following two enhancements to further related research: (1) We offer an adaptive sampling approach in which the heuristic factor is dynamically adjusted based on feedback from the sampling outcomes. (2) We design a highly efficient pruning-reconnecting mechanism to repair the path when new threats appear, ensuring the safety of the UAVs. Simulations have proved that, with these improvements, our proposed algorithm achieves higher sampling efficiency and costs only 3.5% new nodes when re-planning compared with existing approaches. We have introduced an adaptive mechanism to the process of sampling new nodes. And there still could be some work worth doing on this research in the future. Later researchers, for example, can explore more adaptive mechanisms in different stages of the planning process, such as adaptively adjusting the growth direction or the step length of the random tree.

ACKNOWLEDGMENT

This work was supported in part by grants from Beihang University first-class virtual simulation course construction project (No.42020200), National Natural Foundation of China (No.61601010), and the Aviation Science Foundation (No.20160551002).

REFERENCES

[1] V. Spurnỳ, T. Báča, M. Saska, R. Pěnička, T. Krajník, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 125–148, 2019.

[2] N. Mahdoui, V. Frémont, and E. Natalizio, "Communicating multi-uav system for cooperative slam-based exploration," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 325–343, 2020.

[3] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2015.

[4] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[5] H. Nawaz, H. M. Ali, and A. A. Laghari, "Uav communication networks issues: a review," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, pp. 1349–1369, 2021.

[6] S. Ahmed, B. Qiu, F. Ahmad, C.-W. Kong, and H. Xin, "A state-of-the-art analysis of obstacle avoidance methods from the perspective of an agricultural sprayer uav's operation scenario," *Agronomy*, vol. 11, no. 6, p. 1069, 2021.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[9] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[10] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[11] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, 2020.

[12] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, 1999, pp. 473–479 vol.1.

[13] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[14] C. Urmson and R. Simmons, "Approaches for heuristically biasing rrt growth," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1178–1183.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] S. M. LaValle, "Motion planning: Wild frontiers," *IEEE Robotics Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[17] O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in *2013 IEEE international conference on mechatronics and automation*. IEEE, 2013, pp. 354–359.

[18] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[19] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[20] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[21] O. Adiyatov and H. A. Varol, "A novel rrt*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2017, pp. 1416–1421.

[22] M. P. Strub and J. D. Gammell, "Advanced bit*(abit*): Sampling-based planning with advanced graph-search techniques," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 130–136.

[23] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "Rrt-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.