Optimization of the RRT Algorithm Applied in Autonomous Planning for the 2D Robot Navigation

Yingrui Xie[1*] and Tianle Yang[2]

[1]College of Electrical Engineering, The University of Liverpool, L69 7ZX, UK
[2]College of Mechanical and Electronic Engineering, Huazhong Agricultural University, 430070, China
**Corresponding author: 1592628127@qq.com*

Abstract
Path planning in robotics usually needs to find high-quality route schemes. Improved RRT algorithm, such as RRT*, random sampling is traditionally used to gradually approximate the problem domain. This enables RRT* to find a more suitable route scheme than RRT. But did not change the order in which their search depends on approximation. Any sequence of random samples approximates the problem domain in each direction at the same time, but it may be very inefficient when containing solutions. This report unifies and extend the basic RRT algorithm to develop an Optimized RRT algorithm which combined of RRT and Greedy search. The optimized RRT algorithm can help find an ideal path within a short time period and save a great number of calculations during a 2D robot navigation. In different cases, the comparison with normal RRT and RRT* will also be presented in the report.

Keywords
Optimized RRT Algorithm, Normal RRT, Greedy Research, 2D Navigation

1. Introduction

This research mainly focuses on optimizing the basic RRT algorithm. As for the normal RRT algorithm, which will firstly establish an undirected graph in the informed map through sampling, and then find the relatively optimal path through the search method [1]. It is to search from a certain point, sample and build a map at the same time. Most of the research like Batch Informed Trees (BIT*) [2], they reduce unnecessary searches by narrowing the scope of the search range. This research didn't simplify the searching procession. Instead of doing the same thing, this report will show a new way to optimize the basic RRT algorithm. This report will provide a brand-new thinking direction for the follow-up researches. In the work, the optimized RRT algorithm developed and studied in this report is composed of RRT and Greedy search, which can help find an ideal path within a short time period and save a great amount of calculations during a 2D robot navigation simultaneously.

2. Ease of Use Related works

A lot of RRT-based approaches, like RRT and RRT*, using incremental sampling to build trees in barrier free space to solve continuous path planning problem. These methods avoid a priori discretization of the problem domain and allow them to run indefinitely until a solution is found. The search depends on the sample sequence and have a certain randomness. So that some searches need to use heuristics to order the search. Heuristics can also be used to focus the search, as in Anytime RRTs and Informed RRT* but it does not change the order of the search [3,4]. Heuristic guided RRT may include heuristics in the RRT extension step[5]. The states of random sampling are added according to their proportional probability relative to the heuristic value of the tree. The bias of RRT search and the preference are balanced for expanding potential suitable paths. This optimizes the performance in the problem of continuous cost function, however, the probability of wasting computing work in the unnecessary state of the final solution is not zero [6]. Karaman et al. and Akgun and Stillman both use heuristics to accelerate the convergence of RRT*. They rejected samples where heuristic estimation could not give a better solution [7]. These methods avoid unnecessary computation, but the initial search will not be change because the heuristic will not be used until there is a solution. Kiesel et al. (2012) use a two-stage process to order sampling for RRT* [8]. The heuristic is calculated by using the Dijkstra algorithm to solve the rough discretization of the planning problem. To bias RRT* heuristic is used sampling to the area where the solution is found in the problem domain. This increases the possibility of finding a solution quickly but maintains a non-zero sampling probability in the whole problem domain throughout the search process and allows the search effort to be wasted in the unnecessary state of the final solution [9]. The RRT# algorithm uses heuristics to find and update a tree in the graph built incrementally by Rapidly exploring Random Graph [10]. To effectively maintain the best connection to each vertex, it propagates changes throughout the drawing by using LPA*. This can also be implemented as policy iteration [11]. RRT# also uses heuristics to limit this diagram to problem areas that can improve existing solutions. However, these improvements will not change the construction order of the graph itself and may lead to the loss of important and difficult to sample states because they cannot be connected to the tree at present. RRT^x exceeds the propagated rewiring to the dynamic environment by limiting the propagation to the changes that can improve the graphics and making it exceed the threshold specified by the user [12]. This consistency balances local rewiring performed by RRT* and cascaded rewiring performed by RRT# to improve performance [13]. It was pointed out that this can be done effectively by reusing information, but no specific method was given [14]. Different from these methods, this report tries another research orientation—optimizing the search process.

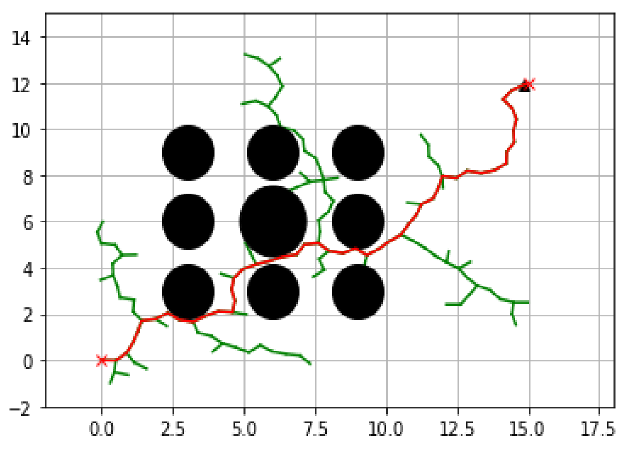
3. Method

During the research in RRT algorithm, it is turned out that the normal RRT can reach the goal state point with a reasonable maximum iteration within a short time period compared with RRT* algorithm. However, the returned path is not an ideal path, which means the path might be long, and seems to be silly, as the sampling of the points in the 2D space is random, while the RRT* algorithm helps find a shortest path with large iterations, all the iterations will be executed, which means though the shortest path might have been found during the earlier iteration, the path will not be returned until all the iterations have been executed, and this characteristic of the RRT* algorithm contributes to the fact that the RRT* cannot provide a short path within a short time period. In this case, an optimized RRT algorithm that combined of RRT and Greedy search is developed in this research. Besides, for the obstacle simulated in this research, the shape of the obstacle is chosen as circular, as for one hand, the distance function is easier to be defined to identify whether the expanded path has a collision with the obstacle, on the other hand, for the obstacles with different shapes, circular can help cover them and calculate the distance between the centre of the circular and the expanded path through a unified approach to identify whether there is a collision.

3.1. Problems to solve

As for the RRT, the main problem is that the returned path has a long length, though the path can be returned within a short time period, which is shown in Figure 1:

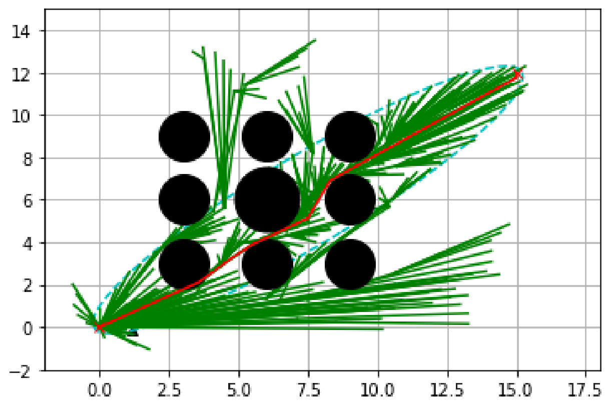
```
The path length: 24.162120089789312, It costs 29.
```



```
The path has returned.
```

Figure 1: The returned path by using RRT (300 iterations, expand distance=0.5)

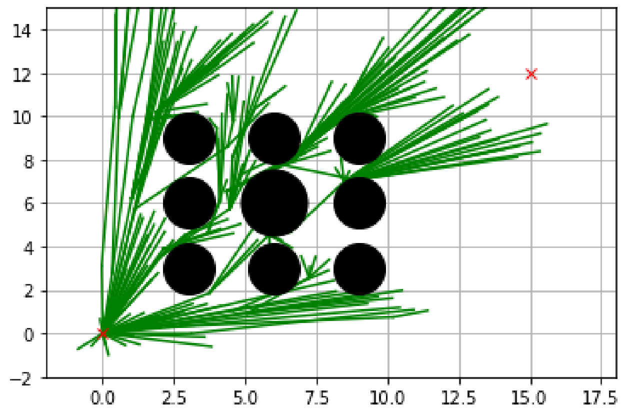
Meanwhile, when applying RRT* in the same informed 2D space, the returned path seems to be the theoretical shortest path, with execution of a large amount of iterations (600 iterations), which is shown in Figure 2:



```
The path has returned.
```

Figure 2: The returned path with RRT* (600 iterations, expand distance=0.5)

Additionally, if the execution of the iterations is not large enough, the RRT* might fail to find the theoretical shortest path, an example with 300 iterations is shown in Figure 3:



```
The path has returned.
```

Figure 3: Failure of RRT* during the 2D navigation (300 iterations, expand distance=0.5)

In this case, the problem to be solved with the optimized RRT is that to ensure the path can be returned within a short time period with small iterations to be executed, and ensure the path can be close to the theoretical shortest path simultaneously.

3.2. Inspired from the Greedy search in A*

As for the greedy search, it is a search algorithm that search for the shortest distance between the start state point and the goal state point with the ignorance of the obstacles between them, which is essentially the connected line between the two points in the 2D space navigation. And for RRT, as the sampling of the points in the 2D space is random, the expand direction of the path might be opposite of the direction toward the goal state point, in this case, to make the expand of the RRT path has a tendency of expanding toward the goal state point, the optimized RRT is expected to prioritize the goal state as the sampling point during each iteration, while the point will be sampled randomly in the 2D space, when there is a collision with the obstacle to help avoid the obstacle. Then for the next iteration, the goal state point will still be the priority of the sampling point, if there is no collision with the obstacle. The plot of such process is shown in Figure 4:

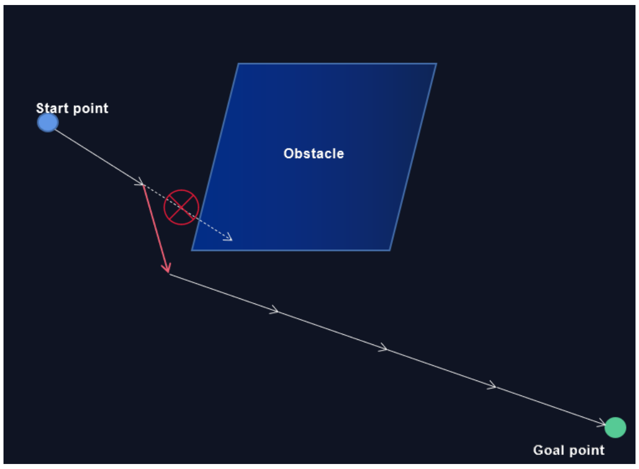


Figure 4: Theoretical expanding process of the optimized RRT, when there is a collision

3.3. The algorithm of the optimized RRT

For the normal RRT, the algorithm is shown as follow:

```
for i in range (max_iteration)
        sampling point = random sampling point

        if no Collision
                newNode is appended to the nodelist

        if the newNode = goal state point
                return the nodelist
```

While for the optimized RRT algorithm, which prioritizes the goal state point as the sampling point, a nested loop needs to be applied in this algorithm, the logic of the algorithm is shown as follow :

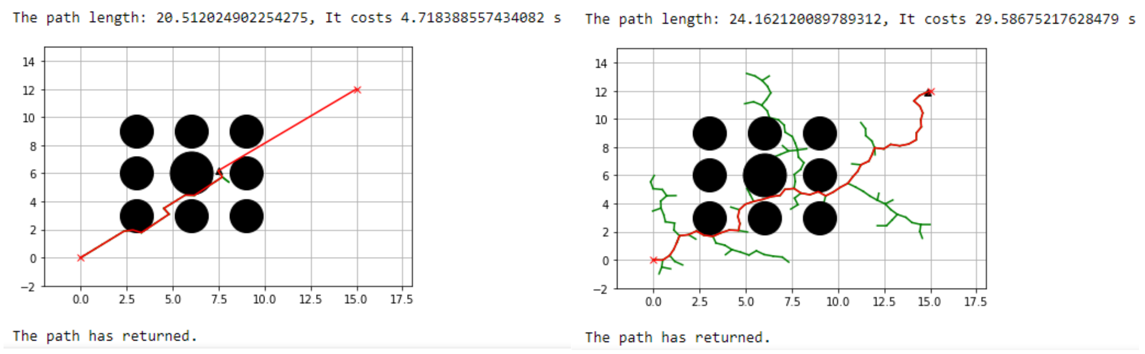
```
for i in range (max_iteration)
        sampling point = goal state point
        if no Collision
                newNode is appended to the nodelist

        if Collision
        for n in range (a large number)   # ensure a new node sampled can be added to node list
            sampling point = random sampling point
            if no Collision
                newNode is appended to the nodelist
                break  # help jump out of the nested loop

    if the newNode = goal state point
        return the nodelist
```

During the optimized RRT algorithm, at the beginning of the external loop, the goal state point is chosen as the sampling point, then the expand distance will decide the new node, if the connection of the new node and the parent node has no collision with the obstacle, the new node will be appended to the node list and become the parent node for the next iteration. However, if there is a collision with the obstacle, the program will get into the nested loop, which is the loop for the random sampling, in this nested loop the iteration number of it should be a large number to ensure that a new node can be appended to the node list by random sampling (during the research, 1000 is chosen as the number of the nested maximum iteration), then if there is no collision, the new random sampled node will be appended to the list, and the 'break' order will help the program jump out of the nested loop, entering the next external iteration, which starts with prioritizing the goal state point as the sampling point.

4. Discussion

4.1. Comparison with normal RRT

As for the optimized RRT, for the informed 2D space with many obstacles and gaps to go through, compared with the normal RRT, the optimized RRT seems to help save the time cost, and obtain an ideal path that is close to the theoretical shortest path simultaneously. And such result is shown as follow (figure 5):

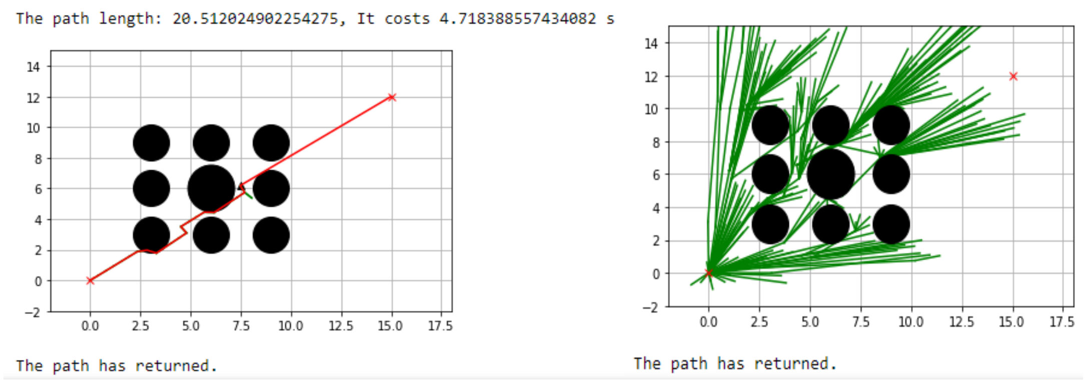


```
The path length: 20.512024902254275, It costs 4.718388557434082 s
The path has returned.
```

```
The path length: 24.162120089789312, It costs 29.58675217628479 s
The path has returned.
```

Figure 5: Comparison between the optimized RRT (left) and normal RRT (right) (300 iterations, expand distance=0.5)

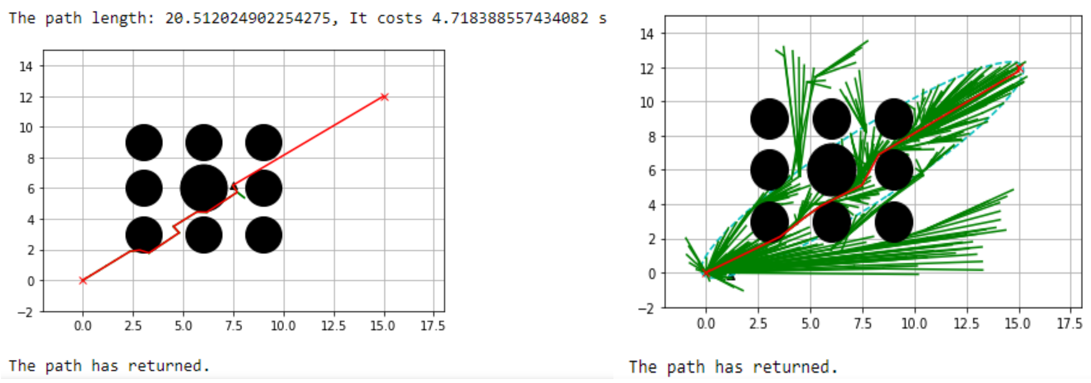
Obviously, the path it obtained is a very short path (length = 20.51) compared with the normal RRT, which obtains a path with length of 24.16. Concurrently, the time cost of the optimized RRT (time cost=4.72s) is shorter than the normal RRT (29.57s).



```
The path length: 20.512024902254275, It costs 4.718388557434082 s
The path has returned.
```

```
The path has returned.
```

Figure 6: Comparison between the optimized RRT (left) and RRT* (right) (300 iterations, expand distance=0.5)



```
The path length: 20.512024902254275, It costs 4.718388557434082 s
The path has returned.
```

```
The path has returned.
```

Figure 7: Comparison between the optimized RRT (left) and RRT* (right) (300 iterations for optimized RRT, 600 iterations for RRT*, expand distance=0.5)

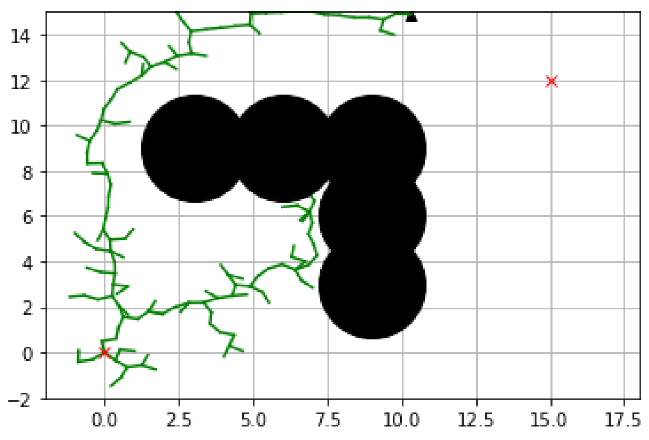
Additionally, in figure 6 and figure 7 , compared with RRT*, there is no doubt that the optimized RRT helps save a large amount of time, as for the optimized RRT just cost nearly 5 seconds, while the RRT* even fails when the maximum iteration is limited within 300. However, for RRT*, the obtained path with enough iterations, which is 600 in the research, the path obtained is theoretically the shortest path. And for the optimized RRT, the path obtained is not the shortest path, but it is an ideal path that is close to the shortest one. All things considered, the advantage of the optimized RRT is obvious, as it can provide an ideal path within a short time period, which means it can provide the robot with an ideal path immediately, when the information of the 2D space is processed, helping the robot react more quickly.

4.2. Optimized RRT algorithm's application in the informed space with cup obstacle

During the research ,as for the optimized RRT algorithm, the expand of the path has a tendency toward the goal state directly, the optimized RRT has a good performance during the search of path in the informed 2D space with complicated obstacles containing many gaps. However, for a 2D space with a cup obstacle, theoretically, the path will directly expand toward the obstacle, then start to bypass the obstacle, until a collision is detected. In this case, such extreme condition is studied, and the comparisons with RRT and RRT* are shown as follow.

To start with, the expand distance is chosen as 0.5, while the maximum iteration is chosen as 300, the results of three algorithms are shown (figure 8, figure 9 and figure 10):

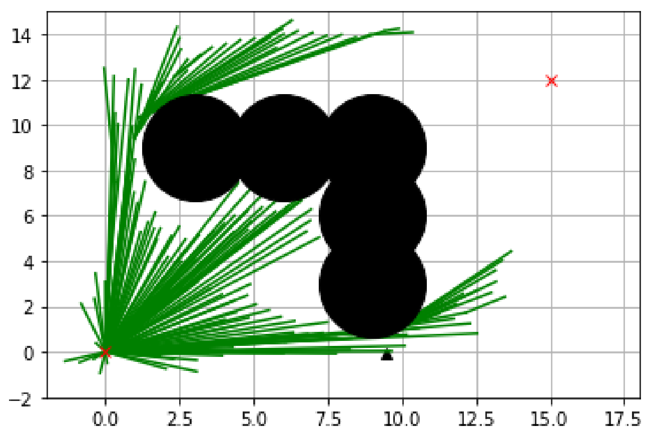
Normal RRT:



```
The path has returned.
```

Figure 8: The performance of normal RRT (failed)

RRT*:

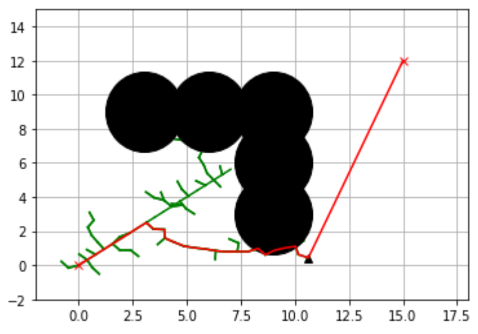


```
The path has returned.
```

Figure 9: The performance of the RRT* algorithm (failed)

Optimized RRT:

```
The path length: 25.363628184253752, It costs 27.554258823394775 s
```



```
The path has returned.
```

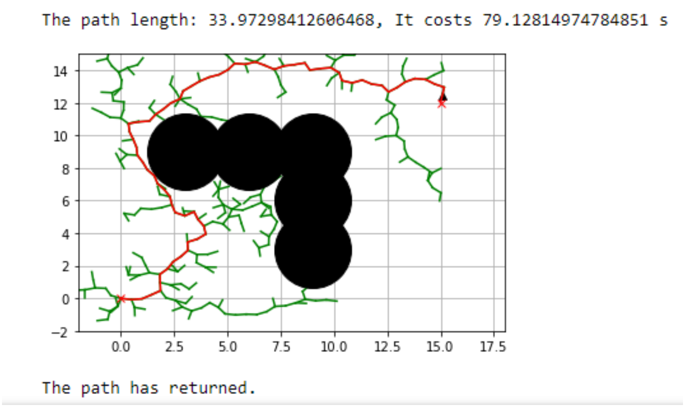
Figure 10: The performance of the optimized RRT (succeeded)

Obviously, within 300 iterations, only the optimized RRT can work, though such path is not short enough, it is still an ideal path with less amount of calculations within a short time and less iterations.

In this case, for both normal RRT and RRT* need more iterations, which means during the competition in the aspect of time cost and amount of calculations, the optimized RRT wins.

Now, increase the iterations to 500, the results of normal RRT and RRT* are shown as follow (figure 11 and figure 12):

Normal RRT:

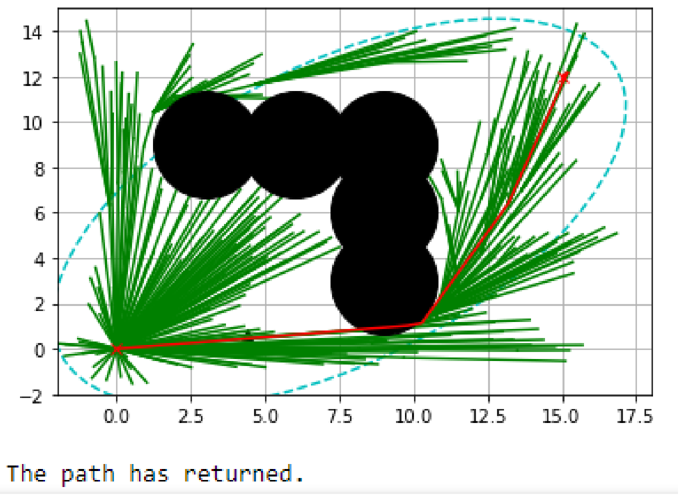


```
The path length: 33.97298412606468, It costs 79.12814974784851 s
```

```
The path has returned.
```

Figure 11: The performance of normal RRT with more iterations (500 iterations)

RRT*:



```
The path has returned.
```

Figure 12: The performance of RRT* with more iterations (500 iterations)

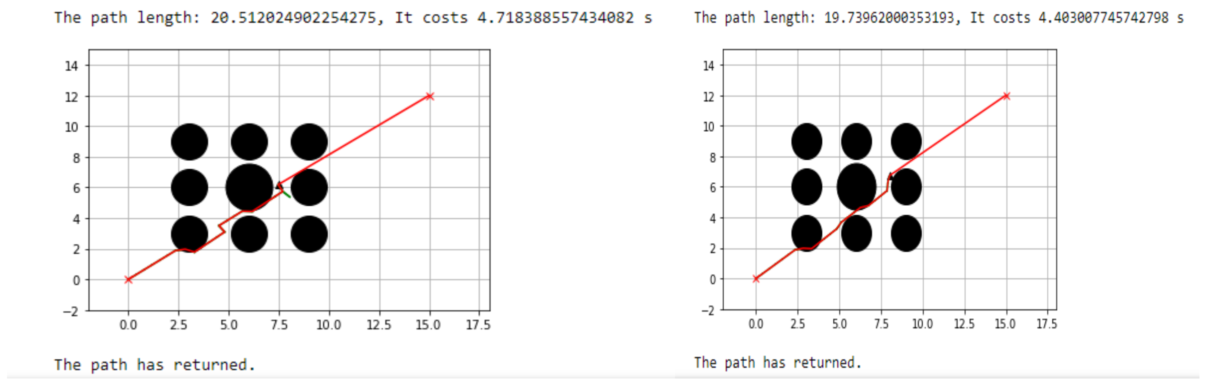
From the results above, the normal RRT actually can be abandoned, as compared with the optimized RRT, not matter the time cost or the length of path, the performance of the normal RRT is worse. And for the RRT*, it is obvious that the obtained path is the shortest, but the large amount of calculation due to the large iterations it needs cannot be ignored, and the path obtained by using the optimized RRT is still an ideal path, though the part of the path that tries to bypass the obstacle is not as short as RRT*, but after bypassing the obstacle, the path is as short as RRT*. In this case, the path obtained from the optimized RRT is still an ideal path, so the development of the optimized RRT is meaningful and more useful compared with RRT* in the field of the autonomous 2D space planning within a short period.

Further research on a dynamic reduction of the area of sampling

As for the algorithms that have been mentioned, the sampling of the points are always in the area of the whole informed 2D space. In this case, if the sampling area is narrowed according to the coordinates of the dynamic parent node and the goal state node, new boundaries of the sampling area is shown in the code as follow :

```
x_randmax = max(newNode.x, self.goal.x)
y_randmax = max(newNode.y, self.goal.y)
x_randmin = min(newNode.x, self.goal.x)
y_randmin = min(newNode.y, self.goal.y)
```

The results of the path obtained by using the optimized RRT with different sampling areas in the complicated 2D space with the obstacles containing many gaps is shown in Figure 13:



```
The path length: 20.512024902254275, It costs 4.718388557434082 s
The path has returned.
```

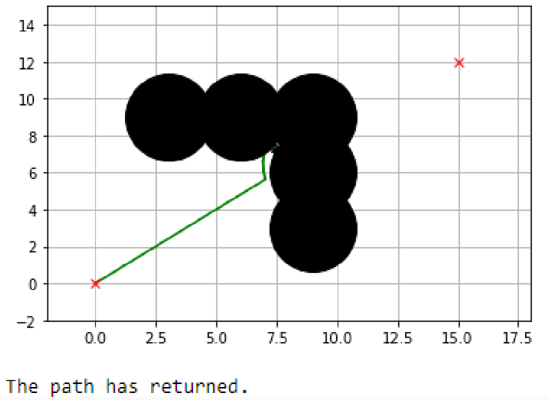
```
The path length: 19.73962000353193, It costs 4.403007745742798 s
The path has returned.
```

Figure 13: The original optimized RRT (left)

The optimized RRT after narrowing the area of sampling (right)

Obviously, by narrowing the area of sampling space, the path obtained seems to be more smooth and slight shorter, the time cost is slight shortened simultaneously, which means narrowing the area of sampling space can help in the optimization of the algorithm.

However, this method of narrowing the sampling area may leads to the failure of the algorithm in some extreme environmental situations, like the many obstacles connected in the shape of a cup (containing no gaps). The failed result of the algorithm after applying the dynamic area of sampling mentioned above is shown in Figure 14:



```
The path has returned.
```

Figure 14: Failure of the optimized algorithm after narrowing the sampling area in the way mentioned above in this extreme condition (no gaps)

Apparently, the narrowing method mentioned above is not reasonable in such extreme environmental situations, and using the whole area as the sampling space can ensure that the optimized RRT algorithm can work within less iterations in any situation, while for the obstacles containing many gaps, such dynamic area of sampling can helps save more time and make the path become more smooth. Additionally, other reasonable methods of narrowing the sampling area that can be applied is different environments are expected to be developed in the future research.

5. Conclusion

All things considered, it is turned out that the main advantage of the optimized RRT is that such algorithm can help find an ideal path within a short time period and save a great amount of calculations during a 2D robot navigation. Though the RRT* algorithm is widely acknowledged in the field of autonomous planning due to the theoretical shortest path it can find, compared with the time and iterations of the calculation it needs, such optimized RRT is useful and meaningful for the 2D robot to move in an ideal path within limited time, and improves the work efficiency during the navigation process in a 2D informed space.

6. References

[1] Zihan Yu, Linying Xiang. (2021) NPQ-RRT : An Improved RRT Approach to Hybrid Path Planning . Complexity. *https://www.hindawi.com/journals/complexity/2021/6633878*

[2] Jonathan D. Gammell (2020) Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search. The International Journal of Robotics Research (IJRR) DOI:10.1177/0278364919890396

[3] Ferguson D and Stentz A (2006) Anytime RRTs. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5369–5375. DOI:10.1109/IROS.2006. 282100.

[4] Gammell JD, Barfoot TD and Srinivasa SS (2018) Informed sampling for asymptotically optimal path planning. IEEE Transactions on Robotics (T-RO) 34(4): 966–984. DOI: 10.1109/TRO.2018.2830331

[5] Urmson C and Simmons R (2003) Approaches for heuristically biasing RRT growth. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), volume 2. pp. 1178–1183. DOI:10.1109/IROS.2003. 1248805

[6] Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research 30(7): 846–894. DOI:10.1177/ 0278364911406761

[7] Akgun B and Stilman M (2011) Sampling heuristics for optimal motion planning in high dimensions. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2640–2645. DOI:10.1109/IROS.2011. 6095077.

[8] Kiesel S, Burns E and Ruml W (2012) Abstraction-guided sampling for motion planning. In: Proceedings of the Fifth Annual Symposium on Combinatorial Search (SoCS). ISBN 978-1- 577-35584-7

[9] Jonathan D. Gammell (2020) Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search. The International Journal of Robotics Research (IJRR) DOI:10.1177/0278364919890396

[10] Arslan O and Tsiotras P (2013) Use of relaxation methods in sampling-based algorithms for optimal motion planning. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 2421–2428. DOI:10.1109/ICRA. 2013.6630906.

[11] Arslan O and Tsiotras P (2015) Dynamic programming guided exploration for sampling-based motion planning algorithms. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 4819–4826. DOI: 10.1109/ICRA.2015.7139869

[12] Arslan O and Tsiotras P (2016) Incremental sampling-based motion planners using policy iteration methods. In: Proceedings of the IEEE Conference on Decision and Control (CDC). pp. 5004–5009. DOI:10.1109/CDC.2016.7799034.

[13] Otte M and Frazzoli E (2014) RRTX : Real-time motion planning/replanning for environments with unpredictable obstacles. In: International Workshop on the Algorithmic Foundations of Robotics (WAFR). Istanbul, Turkey

[14] Gammell JD, Barfoot TD and Srinivasa SS (2018) Informed sampling for asymptotically optimal path planning. IEEE Transactions on Robotics (T-RO) 34(4): 966–984. DOI: 10.1109/TRO.2018.2830331