UDC 004. 896:681.5 **doi: 10.20998/2522-9052.2020.3.15**

Andrii Protsenko, Valerii Ivanov

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

COMPARATIVE ANALYSIS OF RRT-BASED METHODS FOR PATHFINDING IN UNDERGROUND ENVIRONMENT

Abstract. The importance of finding a path for autonomous moving robots is indispensable, because the successful achievement of the target location depends on the solution of this problem. There are a large number of different methods of finding the way, which differ in the accuracy of work, speed, the need for additional equipment. Underground environments, such as mines and tunnels, differ from other structures and open space, and therefore, require different approach when performing pathfinding, as narrow, curved passages and heterogeneous structure could render some of the pathfinding methods ineffective. However, methods based on rapidly exploring random trees (RRT) maintain their effectiveness because they are unaffected by the complexity of the environment. In this article presented a comparison of the three RRT-based methods: RRT, RRT-connect and RRT*.

Keywords: robot; pathfinding; autonomy; RRT.

Introduction

Performing tasks in underground spaces always carries certain risks for the workers and engineers on site. The use of robots for performing remote work helps to reduce, prevent or mitigate these risks. To successfully perform assigned tasks, robot must be able to reach the designated location by solving the pathfinding problem. There are many methods for pathfinding, however, due to complexity and heterogeneity of environment usage of sampling methods carries more advantages over the others. Sampling methods are unique in that planning is done by randomly sampling the configuration space. Sampling methods do not guarantee finding a solution, if any exists, that is, they do not provide completeness. Instead, they provide a lesser idea of completeness - probabilistic completeness. The solution will be given, if any, with sufficient algorithm execution time (infinite runtime in some cases). That's could cause problems in some cases, however it grants independence from the complexity of the surrounding obstacles.

In our previous work [1] we noted how RRT-based search algorithms differ from others in their simplicity and compactness. They could provide solution fast and without the usage of a lot of computing power. Usability of RRT-based solutions for long-term autonomy problems in complex environments is shown in [2]. In [3] authors used RRT* for local path planning during mine exploration. Those examples show the practicality of the RRT-based algorithms. However, those algorithms differ in speed and complexity, and some of them are more situational than the others.

The purpose of the work is to analyze speed and efficiency of the RRT-Based algorithms (RRT, RRT-Connect, RRT*).

Rapidly-Exploring Random Trees

RRT is an efficient data structure and sampling scheme for quick, large-scale space searches. Developed by LaValle [4], this approach had the key of shifting the search toward unexplored areas of space. For the first configuration q_{init} tree with K vertices build in K-1 steps, on each of which random configuration q_{rand} is selected. After selection of q_{rand} the algorithm looks for the vertex closest to it q_{near}, and creates an edge between these two vertices, then the next one is selected q_{rand}. RRT algorithm is shown on Fig. 1.

```
BUILD_RRT(q_init)
    1   T.init(q_init);
    2   for k = 1 to K do
    3   q_rand ← RANDOM_CONFIG();
    4   EXTEND(T,q_rand);
    5   Return T
EXTEND(T,q)
    1   q_near←NEAREST_NEIGHBOR(q,T);
    2   If NEW_CONFIG(q, q_near,q_new) then
    3   T.add_vertex(q_new);
    4   T.add_edge(q_near,q_new);
    5   If q_new = q then
    6    Return Reached;
    7   else
    8    Return Advanced;
    9   Return Trapped;
```

Fig. 1. The RRT algorithm

The RRT-Connect planner is designed specifically for path planning problems that involve no differential constraints [5]. In this case, the need for incremental motions is less important. It differentiates from basic RRT by usage of two trees instead of one, and swaps between them until they are connected. RRT-Connect uses CONNECT heuristic instead of the EXTEND function from basic RRT. Instead of attempting to extend a RRT by a single step, the CONNECT heuristic iterates the EXTEND step until q or an obstacle is reached. RRT-Connect algorithm is shown on Fig. 2.

The RRT* [6] algorithm differs from the basic RRT only in the way that it handles the EXTEND procedure. However, it connects the new vertex, q_{new}, to the vertex that incurs the minimum accumulated cost up until q_{new} and lies within the set q_{near} of vertices returned by the NEAREST_NEIGHBOR procedure. This way it can return the shortest solution out of possibly achievable with the current number of maximum nodes. RRT-Connect algorithm is shown on Fig. 3.

```
CONNECT (T,q)
    repeat
    S←EXTEND(T,q);
    until not (S = Advanced)
    Return S;
RRT_CONNECT_PLANNER(q_init,q_goal)
    T_a.init(q_init); T_b.init(q_goal);
    for k = 1 to K do
    q_rand ← RANDOM_CONFIG();
    if not (EXTEND(T_a,q_rand) = Trapped) then
    If (CONNECT(T_b,q_new) = Reached) then
    Return PATH(T_a, T_b);
    SWAP(T_a, T_b)
    Return Failure
```

Fig. 2. The RRT-connect algorithm

```
RRT_STAR_PLANNER(q_init,q_goal)
    T.init(q_init, q_goal);
    for k = 1 to K do
    q_rand ← RANDOM_CONFIG();
    COST(q_rand) ← DIST (q_rand, q_near)
    q_best, q_neighbors ← FIND_NEIGHBORS(T, q_rand)
    for q' in q_neighbors
    if COST(q_rand) + DISTANCE (q_rand, q') < COST(q')
    COST (q') = COST(q_rand) + DISTANCE (q_rand, q')
    EXTEND(T,q')
    Return T
```

Fig. 3. The RRT* algorithm

Experiments

For the experiments was used map of the Ivsilikat mine, located in Republic of Tatarstan, Russia. The map is shown on Fig. 4.

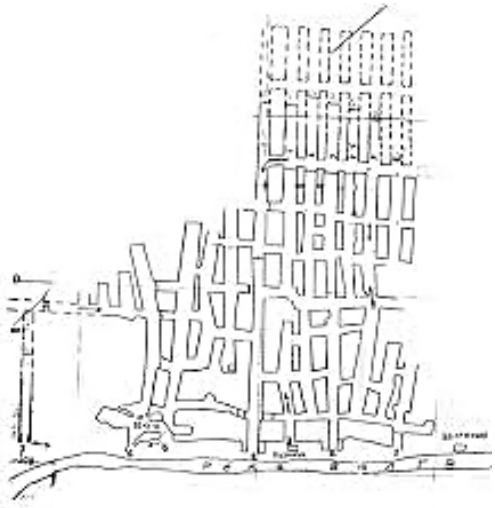

Fig. 4. Ivsilikat mine map

This map offers a vast, heterogeneous structure with a lot of small openings and long passages, so it optimal for the purpose testing performance of path-finding algorithms.

Due to random and somewhat unpredictable behaviors of RRT-based algorithms, number of iteration for each part of the experiments was set to 1000. Six experiments were conducted in total, with different amounts of maximum nodes (1024 and 2048) for each algorithm. Behavior of different algorithms is shown on the Fig. 5. As can be observed, their behavior varies differently from each other.

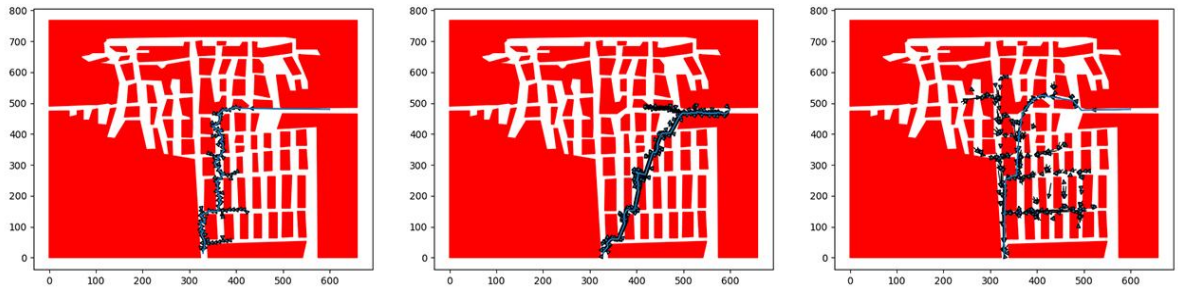

Fig. 5. Behavior of RRT-Based algorithms, from left to right: RRT, RRT-Connect, RRT*

Results of simulation (time/shortest path) with maximum number of nodes set to 1024 for the RRT, RRT-Connect, RRT* are shown on Fig. 6, 7 and 8 respectively. For the RRT with 1024 nodes, average time in which path was found is, 2.3 seconds, with average length of the path 776.3. Shortest path, achieved in 2.996 seconds is 694.8, shortest time, for the path 768.9, is 0.973 seconds. Algorithm failed to find the path 253 times (25.3% of total number).

For the RRT-Connect with 1024 nodes, average time in which path was found is, 3.3 seconds, with average length of the path 709.3. Shortest path, achieved in 3.06 seconds is 668, shortest time, for the path 702, is 2.561 seconds. Algorithm failed to find the path 901 times (90.1% of total number).

For the RRT* with 1024 nodes, average time in which path was found is, 29.3 seconds, with average length of the path 693.3. Shortest path, achieved in 59.724 seconds is 616.7, shortest time, for the path 726, is 7.137 seconds. Algorithm failed to find the path 241 times (24.1% of total number).

Results of simulation (time/shortest path) with maximum number of nodes set to 2048 for the RRT, RRT-Connect, RRT* are shown on Fig. 9, 10 and 11 respectively. For the RRT with 2048 nodes, average time in which path was found is, 3.3 seconds, with average length of the path 780.3. Shortest path, achieved in 1.503 seconds is 688.6, shortest time, for the path 778, is 0.878 seconds. Algorithm failed to find the path 120 times (12.0% of total number).

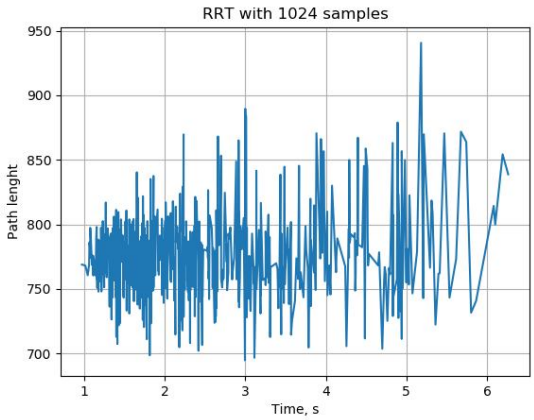

Fig. 6. Results of the simulation for the RRT with 1024 nodes

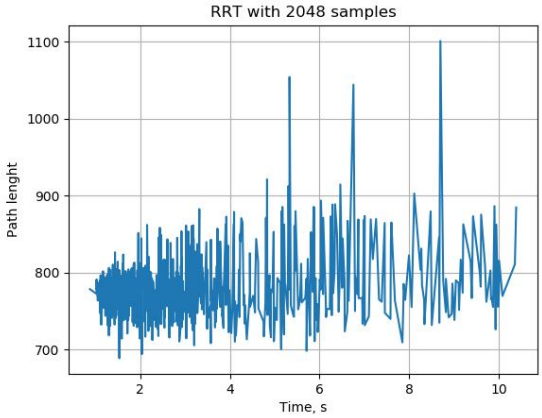

Fig. 9. Results of the simulation for the RRT with 2048 nodes

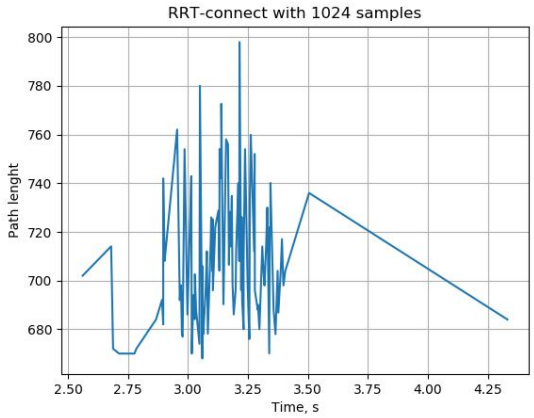

Fig. 7. Results of the simulation for the RRT-Connect with 1024 nodes

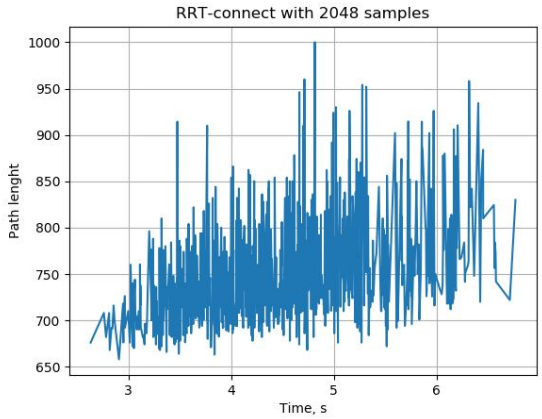

Fig. 10. Results of the simulation for the RRT-Connect with 2048 nodes

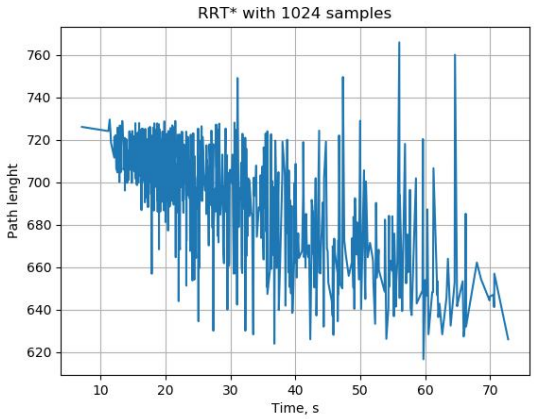

Fig. 8. Results of the simulation for the RRT-Star with 1024 nodes

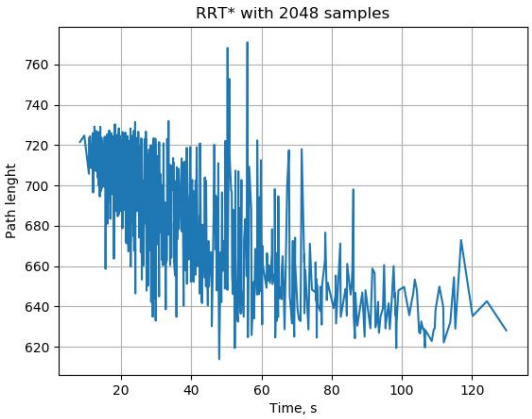

Fig. 11. Results of the simulation for the RRT-Star with 2048 nodes

For the RRT-Connect with 2048 nodes, average time in which path was found is, 4.3 seconds, with average length of the path 749,3. Shortest path, achieved in 2.908 seconds is 658, shortest time, for the path 676, is 2.632 seconds. Algorithm failed to find the path 50 times (5.0% of total number). For the RRT* with 2048 nodes, average time in which path was found is, 37.3 seconds, with average length of the path 687.3. Shortest path, achieved in 47.972 seconds is 613.8, shortest time, for the path 721.5, is 8.321 seconds. Algorithm failed to find the path 132 times (13.2% of total number). Average time and path for the RRT with 2048 nodes increased compared to the one with 1048, in a trade for decrease of failure rate.

Conclusions

Due to basic RRT algorithm being first of it's kind, the average results of RRT 1024 experiment are used as example to compare the results of other experiments. Comparison table for all six experiments is shown on Fig. 12. As seen in the 1024 experiment, RRT-Connect showed better average time and average path that RRT, however it's success rate is tremendously low. As can be seen from the Fig. 12, the it is most dependable on the number of nodes. While other methods deceased failure rate in about a half, failure rate for RRT-Connect with 1024 nodes is 86.1% higher than the RRT-Connect with 2048 nodes. RRT*, on the other hand, has higher success

rate and shorter path than RRT, but takes 12.3 times more seconds to complete. As seen inn the 2048 experiment, RRT-Connect has increased time and path compared to 1024 nodes experiment due to increased number of nodes on each tree, however it trades it for the best success rate in this series of experiments. RRT*, on the other hand, has the shortest path out of all the methods, but it's time further increased.

In conclusion, the choice of the pathfinding method should be made based on the computing power and time available for the task. For the getting optimal path, RRT* offer the best solution. For fast and simple pathfinding RRT-Connect seems like a better choice.

Name	Max. Nodes	Average time, s	Average path	Failure rate, %
RRT	1024	2.3	776.3	25.3
RRT-Connect	1024	3.3	709.3	90.1
RRT*	1024	29.3	693.3	24.1
RRT	2048	3.3	780.3	12
RRT-Connect	2048	4.3	749.3	5
RRT*	2048	37.3	687.3	13.2

Fig. 12. Algorithm effectiveness comparison table

References

1. Protsenko, A. & Ivanov V. (2019), "Classical methods of path planning for mobile robots", *Control, navigation and communication systems,* Vol. 3 (55), pp. 143-151, DOI: https://doi.org/10.26906/SUNZ.2019.3.143.
2. Barfoot, T. D., Stenning, B., Furgale, P., & McManus, C. (2012), "Exploiting Reusable Paths in Mobile Robotics: Benefits and Challenges for Long-term Autonomy", *Ninth Conf. on Comp. and Robot Vision,* DOI: https://doi.org/10.1109/crv.2012.58.
3. Dang, T., Khattak, S., Mascarich, F., & Alexis, K. (2019), "Explore Locally, Plan Globally: A Path Planning Framework for Autonomous Robotic Exploration in Subterranean Environments", *19th International Conference on Advanced Robotics (ICAR).* DOI: https://doi.org/10.1109/icar46387.2019.8981594.
4. LaValle, S.M. (1998), "Rapidly-exploring random trees: A new tool for path planning", DOI: https://doi.org/10.1.1.35.1853.
5. Kuffner, J.J., & LaValle, S.M. (2000), "RRT-connect: An efficient approach to single-query path planning", *IEEE Int. Conf. on Rob. and Aut. Symp. Proc. (Cat. No. 00CH37065)*, Vol. 2, pp. 995-1001, DOI: https://doi.org/10.1109/ROBOT.2000.844730.
6. Karaman, S. & Frazzoli, E. (2011), "Sampling-based algorithms for optimal motion planning", *The Int. Journal of Robotics Research*, *30*(7), pp. 846–894, DOI: https://doi.org/10.1177/0278364911406761.

Received (надійшла) 12.06.2020
Accepted for publication (прийнята до друку) 19.08.2020

About the authors / Відомості про авторів

Проценко Андрій Андрійович – аспірант кафедри комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки, Харківський національний університет радіоелектроніки, Харків, Україна;
Protsenko Andrii – PhD student of Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: andrii.protsenko@nure.ua; ORCID ID: http://orcid.org/0000-0001-8754-7444.

Іванов Валерій Геннадійович– кандидат технічних наук, професор кафедри системотехніки, Харківський національний університет радіоелектроніки, Харків, Україна;
Valerii Ivanov – PhD (C) of Technical Sciences, Professor of Department of Systems Engineering (SysEng), Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: avaleriy.ivanov@nure.ua; ORCID ID: http://orcid.org/0000-0002-6419-3759.

Порівняльний аналіз методів, що базуються на RRT для пошуку шляху у підземних структурах

А. А. Проценко, В. Г. Іванов

Анотація. Важливість пошуку шляху для автономних рухомих роботів незмінна, адже від вирішення цієї проблеми залежить успішне досягнення цільового розташування. Є велика кількість різних методів пошуку шляху, які відрізняються точністю роботи, швидкістю, необхідністю в додатковому устаткуванні. Підземні середовища, такі як шахти та тунелі, відрізняються від інших споруд та відкритого простору, а отже, вимагають іншого підходу при здійсненні пошуку маршруту, оскільки вузькі вигнуті проходи та гетерогенна структура можуть зробити деякі методи проходження маршруту неефективними. Однак методи, засновані на швидкому дослідженні випадкових дерев (RRT), зберігають свою ефективність, оскільки на них не впливає складність навколишнього середовища. У цій статті представлено порівняння трьох методів на основі RRT: RRT, RRT-connect та RRT*.

Ключові слова: робот; пошук маршрутів; автономія; RRT.

Сравнительный анализ методов, основанных на RRT для поиска пути в подземных структурах

А. А. Проценко, В. Г. Иванов

Аннотация. Важность поиска пути для автономных подвижных роботов неизменна, ведь от решения этой проблемы зависит успешное достижение целевого расположения. Есть большое количество различных методов поиска пути, которые отличаются точностью работы, скоростью, необходимостью в дополнительном оборудовании. Подземные среды, такие как шахты и тоннели, отличаются от других сооружений и открытого пространства, а следовательно, требуют иного подхода при осуществлении поиска маршрута, поскольку узкие изогнутые проходы и гетерогенная структура могут сделать некоторые методы прохождения маршрута неэффективными. Однако методы, основанные на быстром исследовании случайных деревьев (RRT), сохраняют свою эффективность, поскольку на них не влияет сложность окружающей среды. В этой статье представлено сравнение трех методов на основе RRT: RRT, RRT-connect и RRT*.

Ключевые слова: робот; поиск маршрутов; автономия; RRT.