# MOBILE ROBOT PATH PLANNING FOR INDOOR USE

**Agris Nikitenko, Aleksis Liekna, Ilze Andersone, Martins Ekmanis, Evalds Urtans**
Riga Technical University, Department of Systems Theory and Design
agris.nikitenko@rtu.lv, aleksis.liekna@rtu.lv, ilze.andersone@rtu.lv, martins.ekmanis@rtu.lv, evalds.urtans@gmail.com

**Abstract.** The paper presents practical and theoretical results acquired during a scientific project addressing multi-robot control and management challenges with possible application in greenhouse automation solutions. The main focus of the paper is path planning of mobile robots addressing computation time and memory constraints in embedded robotic systems. The paper presents an analysis that is based on practically experienced cases and proposes modifications of the discussed RRT-based methods in order to ensure better quality of the planning result as well as saved time in specific cases. A key factor of the compareson analysis is time and memory usage that usually are limited in embedded devices like small scale mobile robots. The paper also presents experiment results collected using a prototype robotic system.

**Keywords:** embedded planning, indoor robotics, real time path planning.

## Introduction

Path planning is one of the central tasks to be solved in mobile robotics. Along with the method groups like Potential field planning [1] and Combinatorial planning [2], Rapidly Exploring Random Tree [3] (RRT) planning has found its application in mobile robotics. While it does not ensure optimal solutions and does not guarantee solution at all it provides sufficient performance [4] for most applications in mobile robotics.

Since first implementations RRT planning has experienced a variety of modifications [4] that differ with implementations of particular algorithm steps and provide different overall performance under particular constraints [5]. However, in real applications the robotic systems are operating under memory, computation and real time constraints, which, as a consequence, pose limitations on the used algorithms and techniques. Memory limitation is not a critical problem since as shown later usually a raw RRT plan does not require storing more than few hundreds to few thousands planning tree vertexes in memory while time and computing power limitations are more sensitive especially for embedded robotic systems. Within this paper we propose a slight modification of the well-known RRT algorithms focusing on space search under memory and time limitations.

The paper is organized as follows: Section II gives a brief overview of the RRT algorithm and related work regarding increase of RRT modifications, Section III presents the proposed modification RRT-Wave, Section IV provides experimental evaluation of the proposed method, Section VI gives conclusions and insight of future work.

## Related work

The RRT was introduced as a planning technique for wide range of motion planning problems [3], which can accommodate particular kinematic or geometric constraints of a given system. The RRT "base" algorithm is outlined in Figure 2. The planning goal is to generate a path from the initial configuration $q_0$ to the goal configuration $q_g$. At each iteration i, a random configuration $q_{rnd}$ is selected. Then the closest configuration $q_c$ from the graph is found and algorithm tries to extend the planning graph towards $q_{rnd}$, by adding an arc from $q_c$ towards $q_{rnd}$ with length d. Thereby a new configuration $q_i$ is added to the planning graph. This step is depicted in Figure 1. The planning stops when the newly added configuration $q_i$ is in a predefined proximity from the goal configuration $q_g$. A number of variations of the RRT exist [4], which provide better performance under particular constraints [5]. There are a number of other modifications of the initial algorithm for addressing different application domains. Hereby we consider RRT-Connect version of this algorithm group [6].

As indicated in [7] the sampling-based planning techniques are well applicable in wide variety of domains, but rapidly can become ineffective if the planning problem is specified by complex dynamic or kinematic constraints that results in computational overheads slowing down the planner and increasing the overall planning time. The [7] presents an approach that employs heuristics in combination with greedy search technique, which results in better performance over complex and constraint parts of search spaces such as narrow passages, where alternative search techniques are combined.

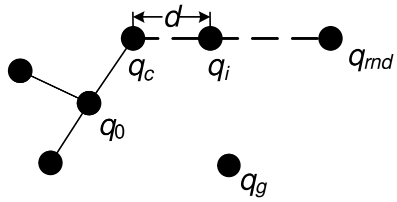



Fig. 1. **RRT extending towards $q_{rnd}$**

**Algorithm 1: Base RRT**

```
RRTmain()
Tree = q.0
q.rnd = q.0
while Distance (q.rnd , q.g) <ErrTolerance do
        q.target = SampleTarget()
        q.nearest = NearestVertex (Tree , q.target)
        q.rnd = ExtendTowards (q.nearest,q.target)
        Tree.add(q.rnd)
end while
return Trajectory (Tree,q.rnd)
SampleTarget()
if Rand() < GoalSamplingProb then
        return q.g
else
        return RandomConfiguration()
end if
```

Fig. 2. **RRT "base" algorithm [10]**

As indicated in [8] the RRT planning technique, when applied in domains with complex kinodynamic constraints, may be computationally costly due to the extend step, which requires computation of kinodynamic equation for each of possible planning tree vertex candidates. The [8] proposes to apply adaptive sampling strategy RG-RRT, which takes into account local reachability, as defined by differential constraints, while building the tree. The proposed approach is based on observation that random selection of points and their checking on possible collisions is much faster than adding extra nodes to the planning tree, which adds both, time cost by having a larger tree. Thereby, adding kinematic obstacles and checking collisions with them on randomly sampled vertex candidates reduce the necessary time for generation as well as the size of the planning tree.

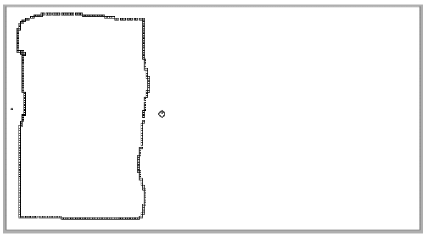
The [9] proposes heuristically guided RRT or hRRT. This method bias the search toward low cost paths using quality measure based on the cost of the path from the root node and estimation of the optimal cost to the goal. Unfortunately, as indicated in [10], this approach has been verified only on rather simple problems with discrete cost states and limited scalability and performance in complex environments. To overcome the mentioned drawbacks the [10] proposes T-RRT with extension of cost function from discrete to continuous values space guiding the sampling into less costly regions allowing to apply the technique in more complex domains like multidimensional robot motion planning.

According to the RRT "base" algorithm, the main effort is necessary for collision detection and the nearest neighbour search, which can be reduced by application of massive parallel computing or optimized searching techniques [11; 12].

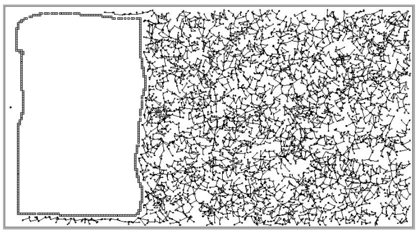
As an alternative this paper presents a sampling space limitation approach and provides experimental analysis over different practically observed situations in mobile robotics domain.

## Materials and methods

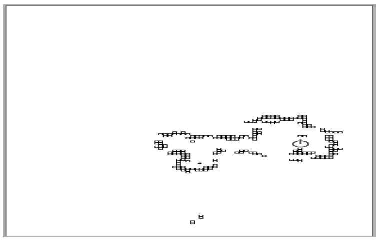
While according to reports the RRT and its variants are well suited for wide variety of robot motion planning, within our research we faced several rather simple practical planning situations, where the used RRT-Connect (a slight modification of the "base" RRT) [3; 6] failed a number of times before delivered a consistent plan. Two of them are presented in the following figures:



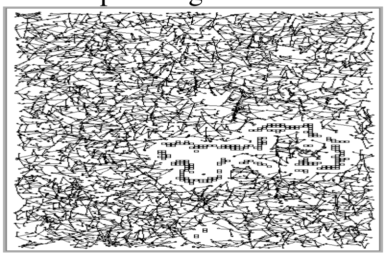
A – planning area 24m x 13m with a single large obstacle



A– planning area 24m x 13m with a single large obstacle planning result – failure.



B – Planning area in small warehouse



B – Planning area in small warehouse with planning result – failure.

Fig. 3. **RRT-Connect results on practical planning problems in 2D space**

During practical experiments we noticed that the planner since it produces a planning tree tends to be weak in cases that could be classified as inverted bug traps. A good example of such a situation is depicted in Figure 3, example B, where there is a free space pocket surrounded by obstacles with close exit to outer environment. As a result the planner in many cases rapidly exits the pocket and cannot return to pocket exploration, thereby leaving the pocked unexplored and goal unreached. The situation can be avoided by significant increasing of the samples generated. However, in this case the effectiveness of the planner is reduced according to its time complexity – $O(\log n)$, where $n$ – number of vertexes [11; 12]. Thereby, to increase the effectiveness of the planner, it is necessary to make it more focused on reaching the goal in order to limit the generation of new vertex candidates. To do so, we propose to start sampling within the area in close proximity from the goal and starting configuration – an initial sampling space that includes starting and goal configurations. When the pre-set sampling density is reached the sampling area is widened. Thereby, the sampling like a wave flushes over the whole planning space. The modified algorithm is outlined in Fig 4.

**Algorithm 1: Base RRT**

```
RRTmain()
Tree = q.0
q.rnd = q.0
WaveIndex = 0
while Distance (q.rnd , q.g) < ErrTolerance do
        q.target = SampleTarget(Tree.count, WaveIndex)
        q.nearest = NearestVertex (Tree , q.target)
        q.rnd = ExtendTowards (q.nearest,q.target)
        Tree.add(q.rnd)
end while
return Trajectory (Tree,q.rnd)
SampleTarget (Tree.count, WaveIndex)
Area = Rectangele(0,0,0,0)        //x_min,y_min,x_max,y_max
if Rand() < GoalSamplingProb then
        return q.g
else
        Area = AreaCalculation(Tree.count,WaveIndex)
        return RandomConfiguration(Area)
end if
```

Fig. 4. **RRT "base" algorithm [10]**

The main difference is the target generation procedure SampleTarget, where a sampling area is calculated and only then the sample is being generated within the area. Thereby, the key is the sampling area calculation function. Within the paper we propose to drive the sampling process by increasing the sampling area by a predefined step, when the number of samples within the area has reached the previously predefined maximum value. At the same time the maximum number of samples is also increased by a predefined value. Thereby, we get an effect of wave. Let us have the planning area 100 x 100 units, maximum allowed samples per wave 10000 and the area edge length increment of 25 units every time the maximum allowed number of samples is reached. It can be easily noticed that by each wave the area increases quadratically while the maximum number of samples linearly. The effect is depicted in the graph in Fig. 5.

Thereby, the total sampling density is decreasing by each wave and, what is more important, the density is higher if the area unit is closer to the initial planning area. It means that the planner pays more attention to closer areas and less to more distant ones. While theoretically such an approach could lead to a local minimum, the practical experiments show that the planning tasks that require crossing the whole area through complex labyrinth with local minima points are very rare.

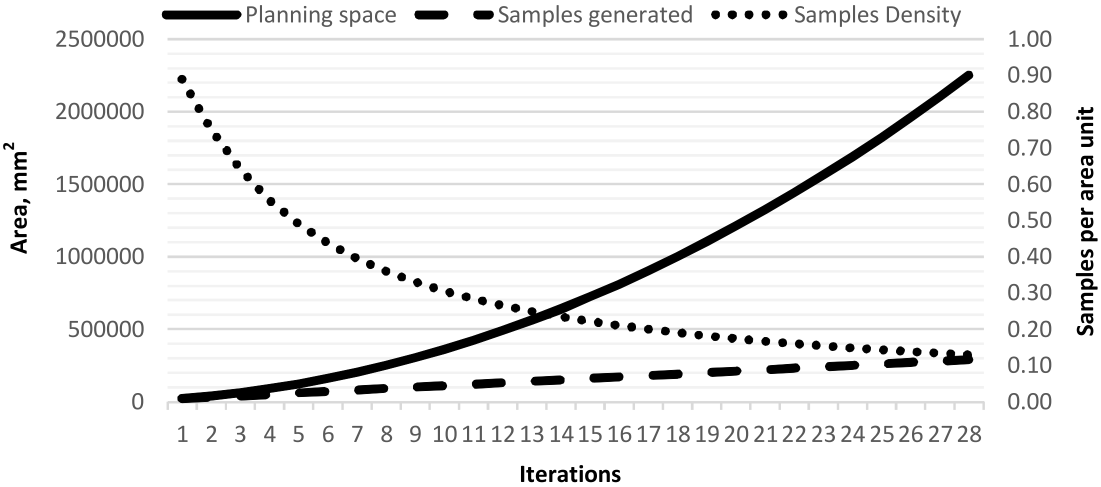



Fig. 5. **Sampling density (samples per area unit)**

## Results and discussions

In order to determine the effectiveness of the proposed modifications, the modified RRT-Wave and the initial RRT-Connect are compared in a number of planning tasks. For experimental purposes both, real and simulated environments were used. The real ones were acquired by the mobile robotic system based on iRobot Roomba 560 vacuum cleaner controlled by IntelAtom based computer, which implements all the necessary planning and motion control routines [13]. The used robotic system is depicted in Fig. 6. In figure Fig. 7 the used environments are appropriately marked. Each of the algorithms is examined with different sampling probability distribution (probability of goal selection as target) and run 20 times for the same situation, thus providing an average number of vertices, time and success rate values. The experiments were conducted on Intel iCore3 based PC with appropriate UI developed in VisualStudio 2010. The actual number of ms spent for planning is only for comparison purposes because of specifics of every individual implementation. The proposed RRT-Wave is also implemented for use on RaspberryPI based mobile robotic system.

Table 1

**Empty area 24.0 m x 13.0 m**

| Probability | RRT-Connect | | RRT-Wave | |
|---|---|---|---|---|
| | Time in ms | Number of vertexes | Time in ms | Number of vertexes |
| $p$ = 0.1 | 14.00 | 171.70 | 24.00 | 216.00 |
| $p$ = 0.3 | 12.40 | 119.00 | 10.40 | 129.20 |
| $p$ = 0.6 | 5.70 | 73.70 | 10.60 | 72.10 |
| $p$ = 0.9 | 7.40 | 60.50 | 6.70 | 60.00 |

Table 2

**U-Shape 24.0 m x 13.0 m**

| Probability | RRT-Connect | | RRT-Wave | |
|---|---|---|---|---|
| | Time in ms | Number of vertexes | Time in ms | Number of vertexes |
| $p = 0.1$ | 541.70 | 598.90 | 2053.70 | 1477.20 |
| $p = 0.3$ | 731.70 | 717.80 | 2234.90 | 1530.70 |
| $p = 0.6$ | 817.10 | 710.20 | 3201.40 | 1545.90 |
| $p = 0.9$ | 2051.50 | 702.80 | 5881.00 | 1229.00 |

Table 3

**Practical situation – 1, 8.2 m x 12.0 m**

| Probability | RRT-Connect | | | RRT-Wave | | |
|---|---|---|---|---|---|---|
| | Time in ms | Number of vertexes | Success rate | Time in ms | Number of vertexes | Success rate |
| $p = 0.1$ | 359.17 | 401.17 | 60 % | 1 329.90 | 1 262.60 | 100 % |
| $p = 0.3$ | 1 243.80 | 782.80 | 50 % | 567.10 | 514.90 | 100 % |
| $p = 0.6$ | 2 002.00 | 924.80 | 100 % | 2 283.33 | 762.67 | 90 % |
| $p = 0.9$ | 1 054.86 | 343.14 | 70 % | 465.67 | 191.50 | 60 % |

Table 4

**Practical situation – 2, 8.2 m x 12.0 m**

| Probability | RRT-Connect | | | RRT-Wave | | |
|---|---|---|---|---|---|---|
| | Time in ms | Number of vertexes | Success rate | Time in ms | Number of vertexes | Success rate |
| $p = 0.1$ | 239.2 | 509.90 | 100 % | 44.7 | 64.6 | 100 % |
| $p = 0.3$ | 348.40 | 588.10 | 100 % | 50.4 | 72.3 | 100 % |
| $p = 0.6$ | 470.10 | 457.70 | 100 % | 67.4 | 65.5 | 100 % |
| $p = 0.9$ | 1056.22 | 351.00 | 90 % | 223.6 | 81.2 | 100 % |

Table 5

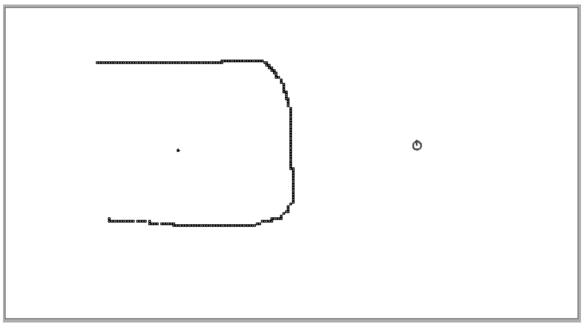
**Simulated situation – 3, 14.5 m x 14.5 m (Warehouse)**

| Probability | RRT-Connect | | | RRT-Wave | | |
|---|---|---|---|---|---|---|
| | Time in ms | Number of vertexes | Success rate | Time in ms | Number of vertexes | Success rate |
| $p = 0.1$ | 6407.4 | 558.6 | 100 % | 5149.4 | 336.6 | 100 % |
| $p = 0.3$ | 7516.6 | 496.1 | 100 % | 6433.2 | 373.7 | 100 % |
| $p = 0.6$ | 11259.9 | 585.8 | 100 % | 6599.7 | 300.7 | 100 % |
| $p = 0.9$ | 21336.8 | 485.7 | 100 % | 9874.8 | 243.6 | 100 % |

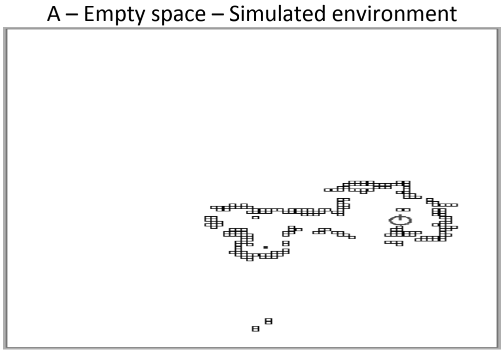
Fig. 6. **Autonomous mobile robotic system for experiment data acquireing**

The used planning areas are depicted in the following figures:

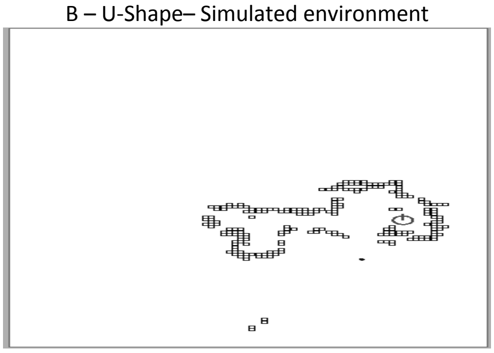
A – Empty space – Simulated environment



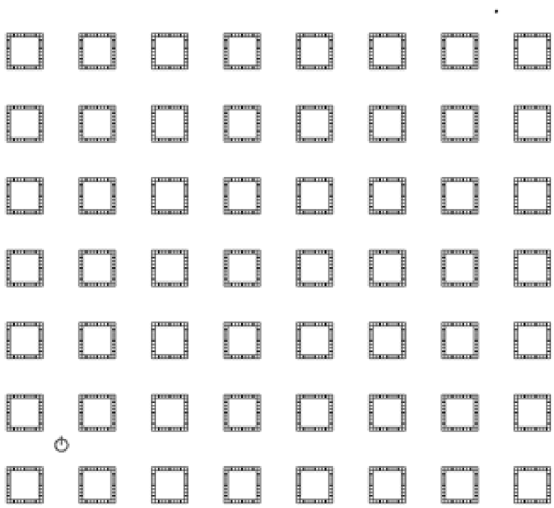
B – U-Shape– Simulated environment



C – Practical Situation 1– Real environment



D – Practical Situation 2– Real environment



E – Practical Situation 3 (Warehouse) – Simulated environment

Fig. 7. **Used experimental environments**

## Conclusions and discussions

1. The proposed RRT planner modification RRT-Wave allows focusing the planner on the planning goal by incrementally increasing the point sampling area at the same time increasing the total allowed samples, which allows reducing the number of generated vertexes and thus reduces the total planning time.
2. Thereby, the sampling density is decreasing, i.e., the number of total samples is the same for each area increment, which is added to the already existing samples thus the sampling density closer to the goal and start is higher than in more distant sectors.
3. As a consequence the proposed planner is more appropriate for use in embedded systems, which usually are very sensitive to extra memory and the computation time needed to solve the planning problems. Thereby, it is well suited for application in service robots for indoor use like greenhouses.
4. This slight modification allows increasing the overall planning effectiveness in terms of a smaller number of generated vertices and as a consequence less time spent for planning. The experimental

results show that the classical tests on u-shape still are challenging while more practical experimental examples clearly show higher performance of the RRT-Wave both, in terms of the generated vertices and the planning time.

**Acknowledgement**

The paper is created within RTU funded project “Multi robot floor cleaning systems” ZI-2013/2.

**References**

1. LaValle S. M., Planning Algorithms, 2006, Cambridge University Press, ISBN 0-521-86205-1.
2. Latombe J.C. Robot motion planning, 1991, Springer, ISBN 9780792391296,
3. LaValle. S.M. Rapidly-exploring random trees: A new tool for path planning. In Technical Report No. 98-11. October 1998.
4. Abbadi A., Matousek R., Minar P., Soustek P. RRTs Review and Options, computational Engineering in Systems Applications, Volume II, 2011, IAASAT Press, pp. 194-199, Iasi, Romania. ISBN: 978-1-61804-014-5, ISSN: 2223-9812.
5. Abbadi A., Matousek R. RRTs Review and Statistical Analysis. International Journal of Mathematics and Computers in Simulation, 6, 2012.
6. Kuffner J.J., LaValle S.M.. RRT-Connect: An Efficient Approach to Single-Query Path Planning, in proceedings of International Conference on Robotics and Automation (ICRA), 2000.
7. Bekris K.E., Kavraki L. Informed and Probabilistically Complete Search for Motion Planning under Differential Constraints, First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR), Chicago, IL, 2008.
8. Shkolnik A., Walter M., Tedrake R. Reachibility – guided sampling for planning under differential constraints, in proceedings of Robotics and Automation ICRA’09, 2009. pp. 2859-2865.
9. Urmson C., Simmons R. Approaches for heuristically biasing RRT growth, in proceedings of IEEE/RSJ International conference on Intelligent Robots and Systems, 2003.
10. Jaillet L., Cortes J., Simeon T. Transition – based RRT for Path Planning in Continuous Cost Spaces, in proceedings of IEEE/RSJ International conference on Robotics and Systems, 2008.
11. Svenstrup M., Bak T., Andersen H.J. Minimising Computational Complexity of the RRT Algorithm.A Practical Approach, Proceedings of IEEE International Conference on Robotics and Automation.
12. Bialkowski J., Karaman S., Frazzoli E. Shanghai International Conference Center, 2011, Massively Parallelizing the RRT and the RRT, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2011. pp. 5602-5607
13. iRobot Roomba® 500 Series Product Resources: Anatomy. [online][13.04.2014] Available at: http://www.irobot.com/us/Support/For_Home/Roomba_500_Resources/Anatomy.aspx.