Adaptive Rapidly-Exploring-Random-Tree-Star (RRT*) -smart: algorithm characteristics and behavior analysis in complex environments

Jauwairia Nasir, Fahad Islam, Yasar Ayaz

Angaben zur Veröffentlichung / Publication details:

Nasir, Jauwairia, Fahad Islam, and Yasar Ayaz. 2013. "Adaptive Rapidly-Exploring-Random-Tree-Star (RRT*) -smart: algorithm characteristics and behavior analysis in complex environments." *Asia-Pacific Journal of Information Technology and Multimedia* 2 (2): 39–51. https://doi.org/10.17576/apjitm-2013-0202-04.

https://doi.org/10.17576/apjitm-2013-0202-04

http://www.ftsm.ukm.my/apjitm
Asia-Pacific Journal of Information Technology and Multimedia
Jurnal Teknologi Maklumat dan Multimedia Asia-Pasifik
Vol. 2 No. 2, December 2013 : 39 - 51
e-ISSN: 2289-2192

ADAPTIVE RAPIDLY-EXPLORING-RANDOM-TREE-STAR (RRT*) - SMART: ALGORITHM CHARACTERISTICS AND BEHAVIOR ANALYSIS IN COMPLEX ENVIRONMENTS

JAUWAIRIA NASIR
FAHAD ISLAM
YASAR AYAZ

ABSTRACT

Rapidly Exploring Random Trees (RRT) are regarded as one of the most efficient tools for planning feasible paths for mobile robots in complex obstacle cluttered environments. The recent development of its variant: RRT is considered as a major breakthrough as it makes it possible to achieve optimality in paths planning. However, its limitations include the infinite time it takes to reach the optimal solution and a very slow rate of convergence. Just recently the authors have introduced RRT*-Smart which is a rapid convergence implementation of RRT* for improved efficient path planning both in terms of planning time as well as path cost. This paper presents a new scheme for RRT*-Smart that helps it to adapt to various types of environments by tuning its parameters during planning based on the information gathered online. The paper also includes detailed explanation of the algorithm's characteristics and statistical analysis of its behavior in different environment types including mazes, narrow passages and obstacle cluttered environments in comparison with RRT*. Navigation experiments using the real Pioneer 3-AT Mobile Robot provide a proof of the concept.*

*Keywords***:** *RRT*, Adaptive Sampling, Robot Motion Planning, Biasing Radius, Path Optimization, Biasing Ratio.*

INTRODUCTION

The forthcoming era of domestic robotics comes with high expectations from robots to acquire the ability to assist humans in various complex tasks of daily life. For an autonomous robot to safely and optimally navigate through obstacle cluttered everyday terrains such as homes and offices is one of the most fundamental of requirements towards the achievement of this aim.

Motion planning for mobile robots is mainly concerned with finding a path for a robot or an autonomous agent in such a way that the robot or the agent will follow that path to move to the goal configuration from its initial configuration without colliding with obstacles or other agents in the environment. Over the last thirty to forty years, many motion planning algorithms have been introduced with some of the popular ones being grid-based algorithms (Kanehara, 2007), visibility graphs (Latombe, 1991), potential field algorithms (Hwang, 1992), neural networks (Yang, 2004), and sampling based algorithms (Geraerts, 2004). Algorithms may be compared and their pros and cons may be weighed based upon factors such as computational time, computational complexity, cost heuristics, planning time etc.

A-Star algorithm has been one of the most popular and the most widely used techniques since its introduction. It is being used in computer games to autonomous vehicles. Rapidly Exploring Random Tree (RRT) (Lavalle, 1998) introduced in 1998 by Lavelle is considered another popular and widely used algorithms. In fact, sampling based algorithms, RRT being one of them, generally gained much popularity because of their less computational complexity and their ability to find paths without specific information of the obstacles in the configuration space. Instead, they make obstacle free trajectories leading to a roadmap between sampled configurations in the obstacle free region. Though RRT has the advantage in providing the quickest first path and also probabilistic completeness but it does not ensure asymptotic optimality. In 2011, Karaman and Farazolli overcame this problem of asymptotic optimality by introducing the extended version of RRT known as RRT*. Rapidly Exploring Random Tree Star made it possible to reach an optimal solution but to do so, it has been proven to take an infinite time (Karaman, 2011). RRT*-Smart which is used for solving the limitation of infinite time and provides a significantly faster rate of convergence proved to of benefit both in terms of cost and time as compared to RRT* in various environments (Islam, 2012). This is achieved by introducing two new concepts Intelligent Biasing and Path Optimization. This study presents a dynamic biasing ratio scheme for RRT*-Smart which leads to more efficient results and also makes the algorithms parameter, biasing ratio, independent of the environment leading to a balanced trade-off between the exploration rate and biasing rate.

This paper is organized into the following sections. Section 2 presents a brief background of RRT* and RRT*-Smart. Section 3 discusses the characteristics of the latter algorithm in detail in addition to their effect on the behavior and efficiency of RRT*-Smart. Section 4 presents the dynamic scheme for the algorithm while Section 5 discusses the complexity of the algorithm in detail. Comparison between the two algorithms in complex environments using t-test is presented in Section 5 and Section 6 covers the hardware implementation of the algorithm on P3-AT. Section 7 presents possible future avenues.

ALGORITHM DESCRIPTION

This section describes the RRT* motion planning algorithm and its rapid convergence implementation RRT*-Smart.

RRT*

RRT* is one of the recent incremental sampling based algorithm which is quick to find an intial path and then later works to improve and optimize this path as the execution takes place (Karaman, 2011).

Let X defines the configuration space in which X_{goal} , X_{obs} and X_{free}=X/$X_{obstacle}$ is the goal region, the obstacle region and the obstacle-free region respectively. RRT* works to find an input u: [0:T] ϵ U that yields a feasible path x(t) ϵ X_{free}. This path starts from x(0) = x-initial to x(T)= goal following the system constraints. A tree *T*= (V, E) constituting of a set of vertices V sampled from the obstacle-free region X_{free} and edges E that connect these vertices together is being maintained by RRT* as it tries to find this solution. Algorithm 1 is the Pseudocode describing RRT*. This algorithm makes use of the following set of procedure.

Sampling: In this procedure a state z_{rand} ϵ X_{free} is randomly sampled from the obstacle-free region.

Distance: This procedure returns the cost of the path in terms of euclidean distance between two states considering the region between them is obstacle free.

Nearest Neighbor: The procedure Nearest (*T*, z_{rand}) returns the node from *T*=(V, E) which is the nearest from z_{rand} in terms of the cost which is calculated by the distance function.

Steer: The procedure Steer (z_{rand}, $z_{nearest}$) works to solve for a control input u[0,T] that drives the system from x(0)=z_{rand} to x(T)=$z_{nearest}$ along the path x: [0,T] → X giving z_{new} at a distance Δq from $z_{nearest}$ towards z_{rand} where Δq is the incremental distance.

Collision Check: The function Obstaclefree(x) is in charge to find out whether a path x:[0,T] lies in the obstacle-free region X_{free} for all t=0 to t=T.

Near-by Vertices: The function Near(*T*, z_{rand}, n) returns a set of the nearby neighboring nodes that lie in a ball of volume (β (logn/n)) around z_{rand} where β is a constant that depends on the planner.

Insert node: The procedure Insertnode(z_{parent}, z_{new}, *T*) adds a node z_{new} to V in the tree *T* =(V, E) and joins it to an already existent node z_{parent} as its parent, and adds this edge to E. A cost which is the sum of the cost of its parent and the Euclidean cost that is returned by the Distance function between z_{new} and its parent z_{parent} is assigned to $z_{new.}$

Rewire : The function Rewire(T, Z_{near}, z_{min}, z_{new}) checks if the cost to the nodes in Z_{near} is less through z_{new} as compared to their older costs. If that is a case for any node, that nodes parent z_{parent} is changed to $z_{new.}$

Apart from ensuring probablistic completeness, RRT* also guarantees asymptotic optimality which is in contrast to its predecessor RRT and other augmented versions of RRT. Although, based on the fact that it has made it possible to approach an optimal solution, it is considered a landmark sampling based algorithm but it has been proven mathematically that it reaches the said solution in infinite time [Karaman, 2011]. Approaching the optimal solution at a significantly faster rate, RRT*-Smart looks to outrun these limitations.

```
Algorithm 1: T = (V, E) ← RRT*(z_init)
1 T ← InitializeTree();
2 T ← InsertNode(Ø, z_init, T);
3 for i=0 to i=N do
4    z_rand ← Sample(i);
5    z_nearest ← Nearest(T, z_rand);
6    (x_new, u_new, T_new) ← Steer (z_nearest, z_rand);
7    if Obstaclefree(x_new) then
8       Z_near ← Near(T, z_new, |V|);
9       z_min ← Chooseparent (Z_near, z_nearest, z_new, x_new);
10      T ← InsertNode(z_min, z_new, T);
11      T ← Rewire (T, Z_near, z_min, z_new);
12 return T
```

RRT*-SMART

Now we briefly describe the RRT*-Smart algorithm along with the two proposed key concepts: Intelligent Sampling and Path Optimization. Initially, random search of the state space takes place as RRT* does until the first path is found. Once this initial path is found, it works to optimize this path by interconnecting the directly visible nodes in this path. Biasing points for intelligent sampling are yielded from this optimized path. At these biasing points, sampling takes place at regular intervals, which are governed by a constant *b* that in turn depends upon the biasing ratio explained later. As the algorithm progresses, this process continues and the path keeps on being optimized. The biasing shifts towards the new path whenever a shorter path is found. Algorithm 2 outlines this process.

```
Algorithm 2: T = (V,E) ← RRT*Smart(z_init)
1 T ← InitializeTree();
2 T ← InsertNode(Ø, z_init, T);
3 for i=0 to i=N do
4    if i=n+b, n+2b, n+3b…. then
5       z_rand ← Sample(i, z_beacons);
6    else
7       z_rand ← Sample(i);
8       z_nearest ← Nearest(T, z_rand);
9     (x_new, u_new, T_new) ← Steer (z_nearest, z_rand);
10     if Obstaclefree(x_new) then
11        Z_near ← Near(T, z_new, |V|);
```

```
12       $z_{min} \leftarrow$ Chooseparent ($Z_{near}$, $z_{nearest}$, $z_{new}$, $x_{new}$);
13       $T \leftarrow$ InsertNode($z_{min}$, $z_{new}$, $T$);
14       $T \leftarrow$ Rewire ($T$, $Z_{near}$, $z_{min}$, $z_{new}$);
15       **if** InitialPathFound **then**
16           $n \leftarrow i$;
17       ($T$, directcost) $\leftarrow$ PathOptimization($T$, $z_{init}$, $z_{goal}$);
18       **if** (directcost$_{new}$ < directcost$_{old}$)
19           $z_{beacons} \leftarrow$ PathOptimization($T$, $z_{init}$, $z_{goal}$);
20 **return** $T$
```

The lines 1, 2, 3 and 7 to 14 execute in the same way as the corresponding ones in RRT* do. The function *InitialPathFound* , in line 15, returns the iteration number *n* at which the first path is found. The algorithm starts biased sampling based on this information. This biased sampling then starts with an interval time defined by the constant *b*. The function *(T, directcost)* $\leftarrow$ *PathOptimization(T,* z_{init}*,* z_{goal}*)* determines an optimized path by directly connecting the nodes in the path that are visible to each other i.e.there is no obstacle between the nodes and are directly connectable without any intermediate node. The function returns its cost in terms of euclidean distance (line 17). In lines 18-19, the function $z_{beacons} \leftarrow$ *PathOptimization(T,* z_{init}*,* z_{goal}*)* returns the *beacons* (the nodes which form the basis for intelligent sampling) if the new cost is less than the old cost; otherwise the old beacons keep on biasing the tree. In lines 4-5, $z_{rand} \leftarrow$ *Sample (i,* $z_{beacons}$*)*, samples are being generated at the beacons within a ball of radius $R_{beacons}$ centered at $z_{beacons}$ which is known as biasing radius. The intelligent sampling takes place with a certain percentage once the initial beacons are found i.e. after every few samples that are placed in the normal way as for RRT* (lines 7-9), one sample is spawned in the region surrounding the beacons. This percentage is defined by biasing ratio.This is to be discussed later.

ALGORITHM CHARACTERISTICS

In this section, the two most important characteristics of the algorithm, biasing radius and biasing ratio, are explained in detail with tabular examples to give a better understanding of how the two factors affect the efficiency of the algorithm.

BIASING RADIUS

Biasing radius is the radius of sphere within which biasing takes place around Z_{beacon}. The radius R_{beacon} can be chosen according to the planner's requirements. For a relatively larger R_{beacon} in a configuration Space, the path has a greater chance of moving quickly towards an optimum path as the biasing starts, and as the iterations continue, the convergence rate slows down. On the other hand, when the radius R_{beacon} is kept small for the same configuration space, the convergence rate of the path towards an optimum path will be slower as the biasing starts but once an optimum enough path is found, it is certain to reach an optimized path and also at a faster rate towards the end as compared to when the radius R_{beacon} was kept large.

This might be seen as a trade off between the planning time and the navigation time. If, for example, a motion planning and navigation problem are put up, which could be solved by using this algorithm and which is more concerned about reducing planning time rather than navigation time, then the better solution is a larger radius and vice versa. However, there is a limit to increasing the radius as if it is increased to a large value, the algorithm's behaviour becomes identical to that of RRT*. For one of the many environments used for experimentation, the experimental results for various biasing radius vs number of iterations to reach the same cost is presented in a tabulated form in Table 1. As the size of radius is increased beyond a certain level, the number of iterations required to reach the optimum cost increases significantly hence approaching the trend of RRT* as for biasing radius of 25 in the table shown below.

A biasing radius in the range of 10-15 is used for all the experiments performed in a configuration space of the same size. A good approximation is to use a biasing radius in accordance with the size of the configuration space.

TABLE 1. No. of iterations for different biasing radius to achieve the same cost of 450 at a biasing ratio of 2.

Biasing Radius	Number of Iterations	Cost
11	4300	540
13	4600	540
15	5200	540
17	8000	540
25	40000	540

BIASING RATIO

Biasing ratio determines the number of times a sample will be spawned directly at a beacon instead of being sampled normally. For a constant biasing ratio throughout the entire planning phase, the choice depends upon the planner. It must be noted that changing this ratio does affect the trend of reaching optimality.

The effect of choosing different constant biasing ratio for the same obstacle environment is evident from the results shown in table 2. These results have been obtained by using one of the many representative environments used for experimentation and analysis.

The results shown above demonstrate that for the same environment, there exists an optimized value of biasing ratio that balances the rate of biasing and exploration to give the optimum/near optimum cost. Any value of biasing ratio that is above or below this optimized value will get the optimum/near-optimum solution using a larger number of iterations.

There are two biasing schemes for the algorithm : static biasing ratio scheme and dynamic biasing ratio scheme. The basic idea behind the generic scheme of dynamic biasing ratio is to cater for the limitations of constant biasing ratio scheme.

TABLE 2. No. of iterations for different biasing ratio to achieve the same optimal cost of 408

Biasing Ratio	Number of Iterations	Cost
5	23000	408
7	17300	408
10	21600	408

GENERIC DYNAMIC BIASING RATIO SCHEME

This is the second and the latest biasing scheme introduced for RRT*-Smart. By introducing intelligent sampling, a tradeoff has been set between the rate of convergence and the rate of exploration as explained earlier in this section. To reach an optimal solution, it is not only necessary to converge at a faster rate through strong biasing but the exploration of the configuration space is equally important. The complexity of an environment is directly related to this challenge of choosing a suitable biasing ratio to optimize the output. It has been seen that in complex environments including a large number of obstacles, the algorithm employing static biasing ratio scheme may give the worst results. To overcome this challenge, a scheme of dynamic biasing ratio has been presented. Instead of being static, it would keep on changing as the number of iterations take place depending upon some parameters. It is approximated to give the best results in almost all scenarios.

A scheme has been developed based upon the heuristic that the dynamic biasing ratio should be a function of the obstacle-free space X_{free} and the number of iterations n. For the dynamic biasing scheme

$$\text{Biasing ratio} = (n / X_{free}) \times \text{constant} \qquad (1)$$

The factor n/X_{free} determines the space density at any point in time. When the environment is not explored, minimal biasing takes place. As the space density increases with the increase in n, biasing ratio also increases. Note that, at a high value of n, majority of the configuration space has been explored so at this point the algorithm focuses at intelligent biasing instead of exploration.

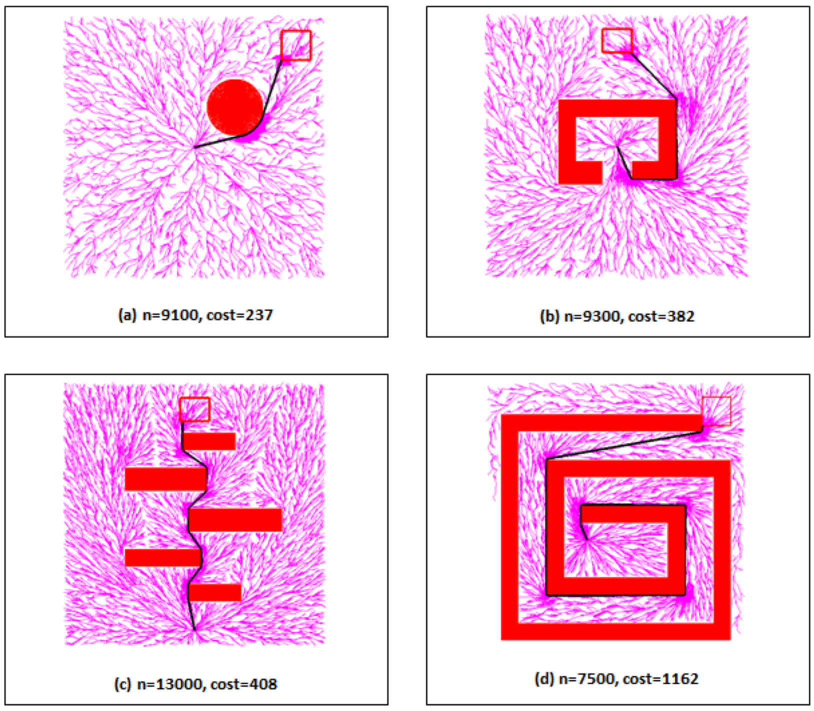


FIGURE 1. RRT*-Smart in various obstacle Environments with constant biasing ratio

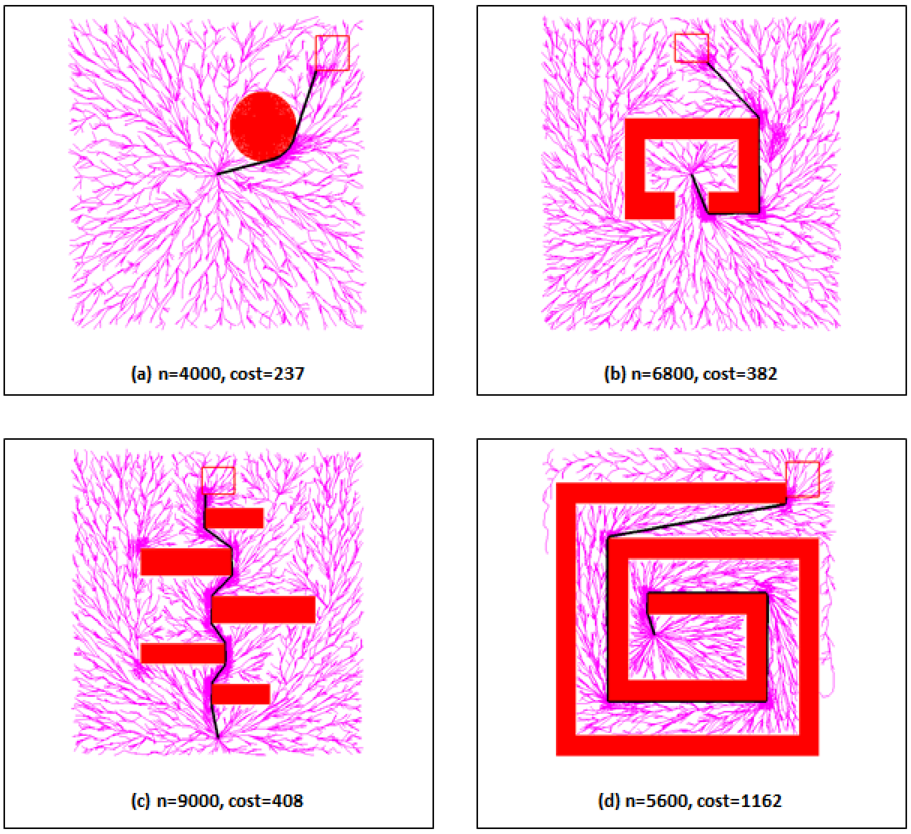


FIGURE 2. RRT*-Smart in various obstacle Environments with dynamic biasing ratio

Some representative environments comparing both the schemes are shown in Figure 1 and Figure 2. RRT*-Smart achieves optimization for each of the four scenarios first using the static biasing ratio in Figure 1 and then a dynamic biasing ratio in Figure 2 which is calculated with the help of Equation (1). For dynamic biasing ratio, as the number of iterations increases, the biasing ratio increases, i.e. the rate at which the nodes are being placed at beacons is now much faster and keeps on accelerating. This implies that as the space is being explored, the biasing increases thus reaching an approximation to provide optimization for the tradeoff between the space exploration rate and intelligent sampling rate. A comparison between the two schemes shows that the latter is able to reach the optimum cost in fewer iterations in all four environments.

This scheme is intended to achieve an optimization for the tradeoff between the biasing rate and the exploration rate in most environments, thus, it makes the algorithm independent and more efficient by choosing a biasing ratio according to the environment itself.

COMPLEXITY ANALYSIS

The Space and Time Complexity of RRT* as demonstrated in (Karaman, 2011) is given by O (n) and O (nlogn). Mathematical analysis shows that the Space and Time complexity of RRT*-Smart also comes out to be the same but for acquiring the same optimality, the value of n, in case of RRT*-Smart, is significantly reduced; hence it yields better performance than RRT*.

SPACE COMPLEXITY

Space complexity of any algorithm is the amount of memory that is required by the algorithm to execute. Clearly it can be seen by the discussion that RRT*-Smart requires n number of memory configurations for n number of iterations to execute. Hence showing a linear Space Complexity of O (n) just like RRT*.

TIME COMPLEXITY

Time complexity of an algorithm is defined as the amount of time that is required by the algorithm to execute a problem of size n. The additional steps of path optimization, intelligent sampling and the collision checking method of interpolation that have been introduced in RRT*-Smart have complexities that are insignificant enough to not have an effect on the complexity of the algorithm. Rather, by the analysis of the T-Notation for both RRT* and RRT*-Smart, it can be observed why for the same number of iterations the time for RRT*-Smart is significantly less as compared to RRT*.

COMPLEXITY OF PATH OPTIMIZATION STEP

Path optimization technique is applied whenever a new smaller path as compared to the previous one is found.

TABLE 3. A comparison of values of delta and *n* in three different obstacle scenarios

Test Environments	Optimized Cost	Iterations Required (*n*)	Delta	Result
Circular	237	9100	17	Delta<<n
Potential	382	7500	22	Delta<<n
Cluttered	408	13000	55	Delta<<n

Let us assume that the new path is found for Delta number of times. The computational complexity of this step be given by:

Delta x Direct cost

Where:

Direct cost = {f x yf/2 x (Cost of each collision check)}
yf = number of nodes in the RRT* path found for each iteration f.
f = number of times the iterative process is repeated for each path until all the visible nodes are directly connected.

The computational cost of each collision check between the nodes under consideration will always be some finite value. As the path will move towards optimization, the direct cost will reduce with time. It is interesting to note here that the upper bound of Delta depends upon the particular environment but will always be significantly less than the total number of iterations n and would always be a finite value for reaching an optimal solution in a particular obstacle scenario. This is because the number of times the path or the paths which lead to optimal solution get optimized will be finite and significantly small in comparison to n. Hence, rendering this term is ineffective in terms of computational complexity as it is independent of the number of iterations n.

Delta <<< n

This is supported by statistical results for three different obstacle scenarios shown in Table 3. In each of these scenarios, the algorithm is executed until the optimized path is found.

COMPLEXITY OF INTELLIGENT SAMPLING STEP

Let p be a constant which defines the number of times a node will be placed at a beacon. This depends on the constant b. The greater the biasing ratio, the greater will be the value of p for the same configuration space and number of iterations. So the complexity of this time may be given by p*c where c is a constant time to place a sample.
Reduced complexity of Sample, Nearest and Steer Step:

Each of these steps will now execute (n-p) times as p times the nodes are being placed directly at the beacons zbeacons whereas n is equal to the total number of iterations. Though this does not affect the O-Notation in any way but explains the reduced computational complexity and one of the factors that leads to reduced execution time.

Thus, it can be concluded that the O-Notation for the time complexity of RRT*-Smart is unaffected by the additional steps though performance has been improved as

1. n is significantly reduced and
2. Execution of RRT* sample step, nearest step and steer step has been reduced by a factor p and replaced by a single step with the same computational complexity as that of RRT* sample step and a path optimization step which has an insignificant contribution as Delta <<< n.

TABLE 4. A comparison of values of delta and *n* in three different obstacle scenarios

Environment	Algorithm	Iterations	Minimum	Maximum	Average	Standard Deviation	t value	z
Maze (Figure 3 a, c)	RRT*-Smart	2000	664	672	668	2.97	28.8	2.31
	RRT*		720	727	722	2.95		
Narrow Passage (Figure 3 b, d)	RRT*-Smart	2500	597	607	602	4.34	15.2	2.31
	RRT*		631	637	633	2.28		
Cluttered with 5 obstacles (Figure 4 a, e)	RRT*-Smart	2000	575	582	578	2.77	14.4	2.31
	RRT*		601	610	606	3.21		
Cluttered with 50 obstacles (Figure 4 b, f)	RRT*-Smart	2000	603	609	607	2.49	11.0	2.31
	RRT*		621	627	624	2.45		
Cluttered with 100 obstacles (Figure 4 c, g)	RRT*-Smart	2000	583	591	588	3.13	36.3	2.31
	RRT*		662	672	666	3.70		
Cluttered with 200 obstacles (Figure 4 d, h)	RRT*-Smart	2500	601	605	603	2	18.8	2.31
	RRT*		631	639	635	3.29		

COMPARISON OF RRT* WITH RRT*-SMART IN COMPLEX ENVIRONMENTS

After the analysis of the algorithm characteristics, introduction of dynamic biasing scheme for RRT*-Smart and complexity analysis, we would now compare the two algorithms in three different environments including a maze, a narrow passage and a cluttered environment (with increasing number of obstacles) each solved for at least 5 times. Figure 3 shows the results for one particular instance out of the five experiments performed in a Maze and a narrow passage environment for both RRT*-Smart and RRT*. In Figure 3 (a,c), the red hollow box represents the goal region while in Figure 3 (b,d), S and G denote the starting point and the goal position respectively. Similarly, Figure 4 compares the two algorithms in the cluttered environment, again for one particular instance, with increasing number of obstacles. In each figure, the obstacles are represented in red color, the goal region is shown by a red box and the trajectory is represented in black. The results are then summarized in Table 4 followed by a statistical analysis of t-Student test for testing equality of the means of the two methods. This statistical test called t-Student test is often used by the researchers and scientists to assess whether two groups significantly differ from one another. We performed an unpaired t-test between the two sets of samples, one for RRT*-Smart and the other for RRT*. Each set consists of 5 values of path costs calculated after solving that particular environment for 5 times considering the stochastic nature of the methods and the costs in each case contributes to the data that is used to perform this test. The minimum, maximum, average and standard deviation values, the t-value for each experiment along with the reference tabulated value z for 95 % confidence level and at 8 degrees of freedom are presented in Table I. Comparing the value of t with the tabulated value of 2.31 (p=0.05) going upto a tabulated value of 5.04 (0.001), we see that the value of our t exceeds these in each case. Thus, according to the principles t-test theory, the difference between the means is very significant and so clearly RRT*-Smart provides significantly improved costs as compared to RRT* for the same number of iterations.

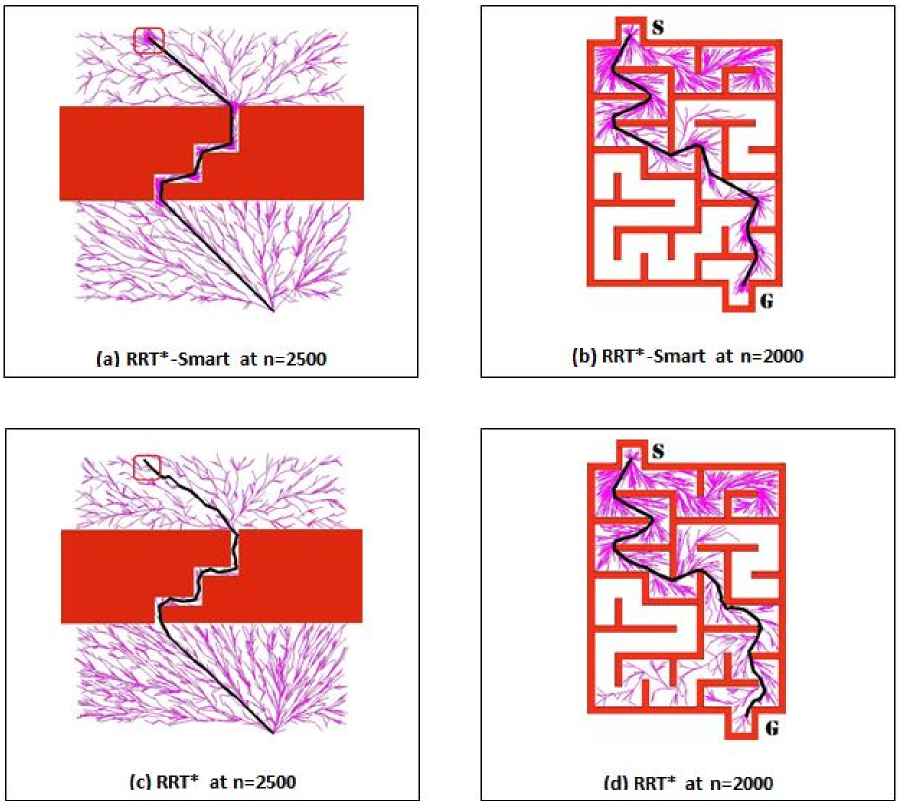


FIGURE. 3. A comparison of RRT* and RRT*-Smart using simulation results in environments with maze and narrow path.

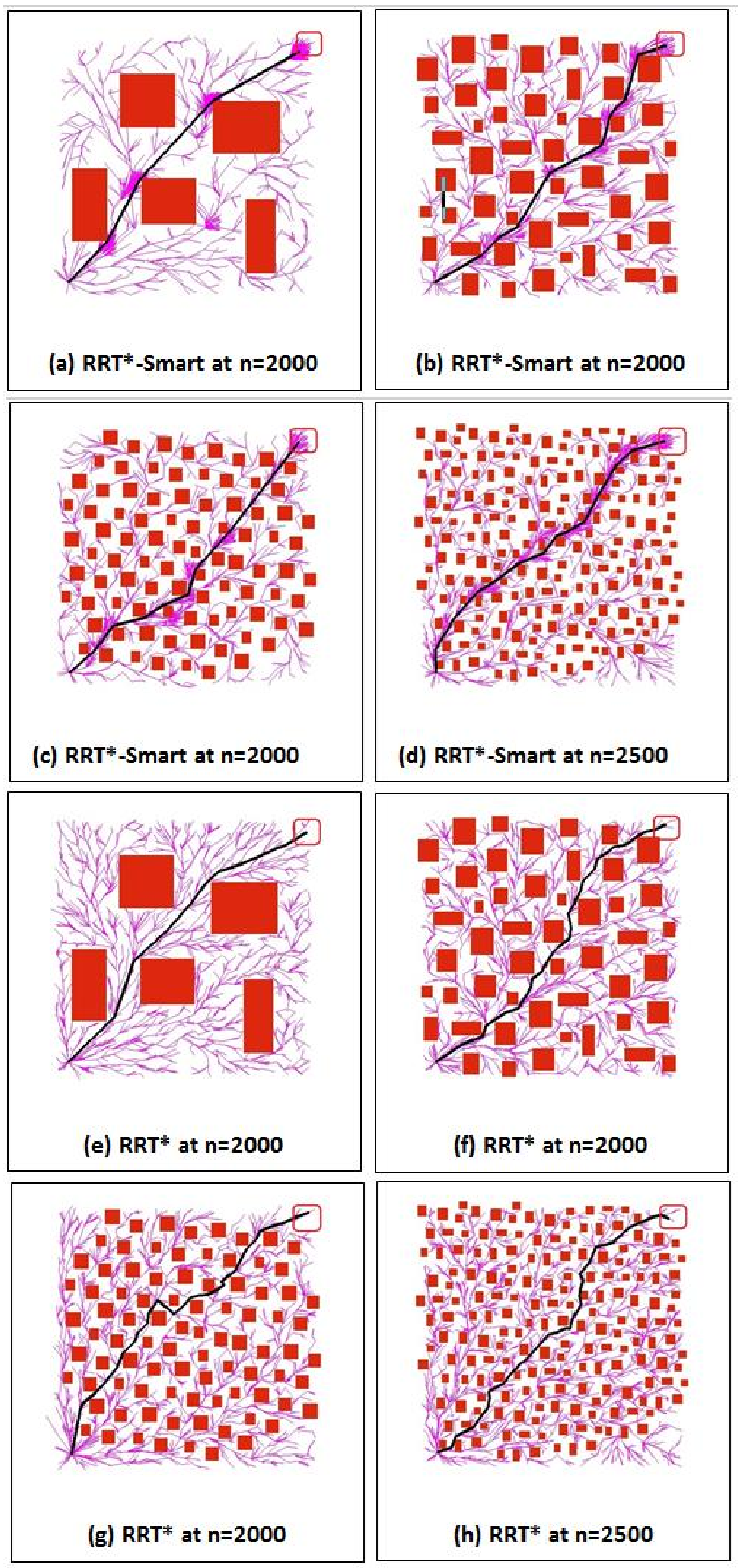
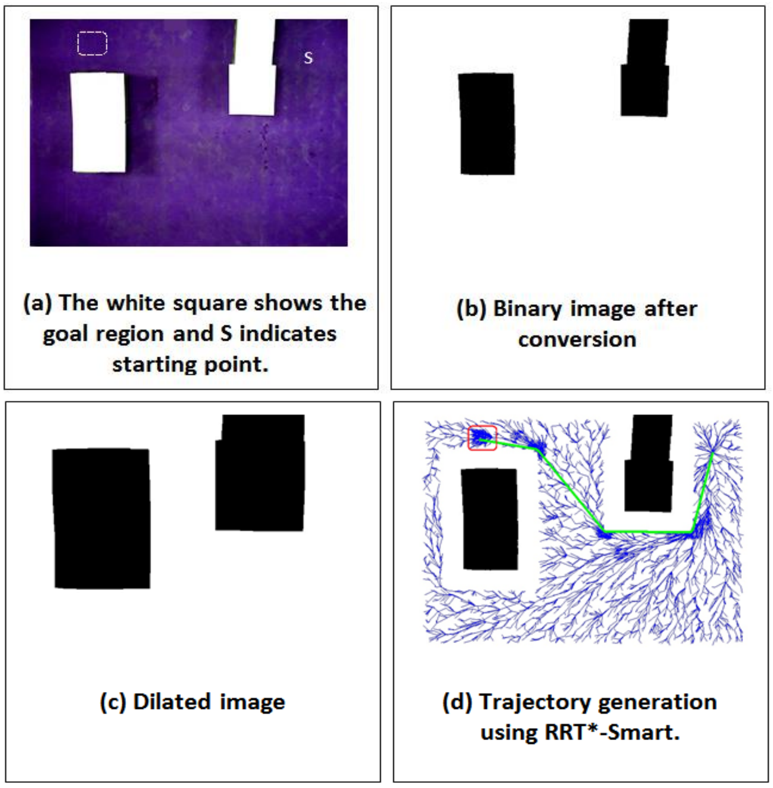


FIGURE 4. A comparison of RRT* and RRT*-Smart using simulation results in cluttered environments with increasing number of obstacles

HARDWARE IMPLEMENTATION

Now the results of hardware implementation of RRT*-Smart algorithm on Pioneer 3AT mobile robotic platform controlled by Player Project to prove its viability is given. An overhead camera is used for environment perception. The image of an environment is taken from the camera as in Figure 5(a). This image is converted into binary image and using image dilation technique the obstacle size is enlarged keeping into consideration the kinodynamic constraints (Oliveira Vaz, 2010) of the robot shown in Figure 5(b) and (c) respectively. Then the RRT*-Smart algorithm functions to come up with a collision free near-optimum path is shown in Figure 5(d). The robot follows this generated trajectory.



(a) The white square shows the goal region and S indicates starting point.

(b) Binary image after conversion

(c) Dilated image

(d) Trajectory generation using RRT*-Smart.

FIGURE 5. Results of hardware implementation of RRT*-Smart algorithm on Pioneer 3AT mobile robotic platform.

CONCLUSION

Since its introduction, the sampling based algorithms have been popular in motion planning and navigation problems. Significant contribution was made by RRT* as it achieves asymptotical optimality apart from probabilistic completeness, but infinite time is needed because of the slow convergence rate. RRT*-Smart, a recent algorithm, tries to overcome these limitations by introducing intelligent sampling and path optimization.

The dynamic biasing ratio scheme for RRT*-Smart has been proven for being able to further improves its efficiency apart from making it independent of the environment. Also, the complexity analysis of RRT*-Smart and its characteristics could impact the behavior of the algorithm. A rigorous comparison has also proved the efficiency of the latter algorithm over RRT* in complex environments. Thus, it is proven easy to implement algorithm on hardware platforms.

ACKNOWLEDGEMENT

The authors would like to thank Alejandro Perez and Matthew Walter of MIT CSAIL for their help regarding RRT* which much improved our understanding of the algorithm.

REFERENCES

Geraerts, R. 2006 Sampling-based Motion Planning: Analysis and Path Quality. PhD thesis, Utrecht University.

Hwang, Y.K. and Ahuja, N. 1992. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, vol. 8(1): 23-32.

Islam, F., Nasir, J., Malik, U., Ayaz, Y., and Hasan, O. 2012. RRT*-Smart: Rapid Convergence Implementation of RRT* Towards Optimal Solution. *Proceedings of IEEE International Conference of Mechatronics and Automation.* Chengdu, China: IEEE, pp.1651-1656.

Kanehara, M. Kagami, S., Kuffner, J.J., Thompson, S., and Mizoguhi, H. 2007. Path shortening and smoothing of grid-based path planning with consideration of obstacles. *IEEE International Conference on Systems, Man and Cybernetics*, 2007 (ISIC), Montreal, pp.991-996.

Karaman, S. and Frazzoli, E. 2011. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*, vol. 30(7):846-894.

Latombe, J.C. 1991. *Robot Motion Planning.* Boston:Kluwer Academic Publishers.

Lavalle, S.M. 1998. Rapidly-exploring random trees: A new tool for path planning. *Technical Report*. Computer Science Dept, Iowa State University, 98–11.

Vaz,O.D.B, Inoue, R.S. and Grassi Jr., V. 2010. Kinodynamic Motion Planning of a Skid-Steering Mobile RobotUsing RRTs. *Latin American Robotics Symposium and Intelligent Robotics Meeting*, Centro Universitario da FEI Sao Bernardo do Campo, Brazil, pp. 73–78.

Yang, S.X., Luo, C., 2004. A neural network approach to complete coverage path planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.34(1):718-724

Jauwairia Nasir,
Fahad Islam,
Yasar Ayaz.
Robotics and Intelligent Systems Engineering (RISE) Lab,
Department of Robotics and Artificial Intelligence,
School of Mechanical and Manufacturing Engineering (SMME),
National University of Sciences and Technology (NUST),
Main Campus, Sector H-12, Islamabad, 44000, Pakistan.
08beejnasir@seecs.edu.pk, 08beefahad@seecs.edu.pk, yasar@smme.nust.edu.pk