Defence Science Journal, Vol. 73, No. 3, May 2023, pp. 322-331, DOI : 10.14429/dsj.73.17888

A Novel Goal-oriented Sampling Method for Improving the Convergence Rate of Sampling-based Path Planning for Autonomous Mobile Robot Navigation

Sivasankar Ganesan[#,*], Senthil Kumar Natarajan[#] and Asokan Thondiyath[$]

[#]*Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu - 626 005, India*
[$]*Indian Institute of Technology Madras, Chennai, Tamil Nadu - 600 036, India*
[*]*E-mail: sivasankarg@mepcoeng.ac.in*

ABSTRACT

Autonomous Mobile Robots' performance relies on intelligent motion planning algorithms. In autonomous mobile robots, sampling-based path-planning algorithms are widely used. One of the efficient sampling-based path planning algorithms is the Rapidly Exploring Random Tree (RRT). However, the solution provided by RRT is suboptimal. An RRT extension known as RRT* is optimal, but it takes time to converge. To improve the RRT* slow convergence problem, a goal-oriented sampling-based RRT* algorithm known as GS-RRT* is proposed in this paper. The focus of the proposed research work is to reduce unwanted sample exploration and solve the slow convergence problem of RRT* by taking more samples in the vicinity of the goal region. The proposed research work is validated in three different environments with a map size of 384*384 and compared to the existing algorithms: RRT, Goal-directed RRT(G-RRT), RRT*, and Informed-RRT*. The proposed research work is compared with existing algorithms using four metrics: path length, time to find the solution, the number of nodes visited, and the convergence rate. The validation is done in the Gazebo Simulation and on a TurtleBot3 mobile robot using the Robotics Operating System (ROS). The numerical findings show that the proposed research work improves the convergence rate by an average of 33 % over RRT* and 27 % over Informed RRT*, and the node exploration is 26 % better than RRT* and 20 % better than Informed RRT*.

Keywords: Goal-oriented sampling; Mobile robot navigation; Path planning algorithm; Rapidly exploring random tree

1. INTRODUCTION

Autonomous Mobile Robots (AMR) are used in industrial automation[1], agriculture[2], and healthcare applications such as patient food delivery, hospital disinfection, and transportation[3-6]. To carry out their tasks in an active environment, the autonomous mobile robot needs to have competent and intelligent path planning algorithms[7-8]. The main objective of the path planning algorithm is to determine a feasible path from the initial position to the desired position by avoiding obstacles in its path[9-10]. Due to their higher efficiency in real-world applications, sampling-based path planning algorithms are widely used in robotics[11]. There are many reported sampling-based algorithms in the literature[12]. The Probabilistic Roadmap (PRM)[13] and the Rapidly Exploring Random Tree (RRT)[14] are the most prominent sampling-based path planning techniques. The PRM technique is a multiple-query approach that works well for issues when the environment is known in advance. In reality, the environment is not always known in advance, so the algorithm is not always effective. For a single query problem, the RRT is probabilistically complete and faster than the PRM[15]. However, the RRT algorithms' path is a non-optimal solution, which is its primary limitation. RRT*[16], an RRT extension, offers an asymptotically optimal solution. Since the RRT* sampling technique uniformly selects a sample configuration from the entire configuration space, it leads to slow convergence, which is due to pure excess exploration[17]. In this paper, we are proposing a new method to improve the slow convergence problem of the RRT* path planning algorithm.

In general, two fundamental approaches are considered to improve the slow convergence problem of the RRT*[18]: the first involves the sampling procedure[19-21], and the second involves path optimisation[22-24]. Of the two approaches, modifying the sampling procedure is the one being considered in this proposed research work. A review of the various sampling techniques in the Rapidly-Exploring Random Trees can be found in the work of Veras[25], *et al.* The RRT* sampling process is biased using the Artificial Potential Fields (APF) technique to attract the samples towards goal[26]. The primary benefit of using the artificial potential function is that it reduces the sample distribution, thereby speeding up the RRT* convergence process. Goal-biased sampling methods[27-28] are based on the sampling process towards the goal region of the search space with a certain probability. A goal-directed method is proposed that collects two random samples concurrently and uses the sample that is closest to the goal position to expand the tree[29]. A probability distribution is used to generate samples[30], with samples closer to

Received : 04 February 2022, Revised : 10 January 2023
Accepted : 02 February 2023, Online published : 12 May 2023

the goal having a higher probability which improves the speed and robustness. The initial path is determined via goal-biased bounded sampling[31] within the boundary of the connected region. A goal direction-based sampling technique is used to improve the convergence of the RRT*[32]. Even though all these methods converge more quickly than RRT*, they are trapped in the obstacle-surrounded environment, which results in a larger path length than RRT*. To avoid the trapping problem, the GO-RRT path planning algorithm includes a node counting approach[33]. However, the node counting technique does not rely on the number of samples used in the path planning algorithm, resulting in performance reduction. To improve the performance, a method called Goal-oriented Sampling-based RRT* (GS-RRT*) is proposed in this paper, which incorporates the simple random selection method[34] into our previous work on Goal-oriented RRT*(G-RRT*) algorithm[35]. The focus of the proposed research work is to reduce the unwanted sample exploration to solve the slow convergence problem of RRT* by taking more samples near the vicinity of the goal region. At the same time, it generates a few samples in the other regions using a simple random selection method to avoid the local minima problem and boost the success rate.

The remainder of the paper is organized as follows: the problem statement is discussed in Section 2. Section 3 presents the proposed GS-RRT* algorithm. Section 4 presents the simulation results, and Section 5 discusses the real-time experimental evaluation. Section 6 presents the conclusions and future research directions.

2. PROBLEM DEFINITION

Consider a configuration space (C-space) denoted as $C \subseteq R^N$ that contains all the robot's possible configurations in N-Dimensions. The position of the robot is described as a configuration $Z \in C$. The obstacle space is defined as C_{obs} and obstacle-free space is defined as $C_{free}=C \setminus C_{obs}$. If $Z_{init} \in C_{free}$ denotes the initial configuration and $Z_{goal} \in C_{free}$ the goal configuration, then the path through C-space is denoted by $\sigma:[0,1] \rightarrow C$. The path planning problem is concerned with determining a path $\sigma:[0,1] \rightarrow C_{free}$ that begins at $\sigma(0)=Z_{init}$ and ends at $\sigma(1) \in Z_{goal}$ and $\sigma(\tau) \in C_{free} \forall \tau \in [0,1]$.

2.1 Feasible Path Planning Problem

For a given path planning problem (Z_{init}, Z_{goal}, C_{free}), feasible path planning is concerned with determining a feasible path if exists and reporting a failure if no such path exists.

Let the set of all possible paths over C denoted by S and the set of all feasible paths by $S_{feasible}$.

2.2 Optimal Path Planning Problem

For a given path planning problem (Z_{init}, Z_{goal}, C_{free}), optimal path planning is concerned with determining a feasible path σ^* such that $plen(\sigma^*)=\min\{plen(\sigma):\sigma \text{ feasible}\}$ if exists and if no such path exists report failure. Where $plen:S \rightarrow R_{\geq 0}$ is a path length function.

3. Goal-oriented Sampling RRT*(GS-RRT*)

The proposed Goal-oriented Sampling RRT* path planning Algorithm is explained in this section.

3.1 Goal-oriented Sampling Procedure

The sampling scheme is the brain of the sampling-based path planning algorithm, which plays an important role in finding the initial path planning rapidly. The RRT* algorithm explores the whole configuration space (C-space) as shown in Fig 1(a) using a uniform sampling method to find a feasible path from an initial node Z_{init} until it gets to the goal node Z_{goal}. This means that the RRT* sampling method generates the samples in the redundant area. Therefore, to improve the sampling scheme and to minimize the redundant sampling area, the proposed GS-RRT* generates more samples near the goal region as given in Fig 1(b). When compared to RRT* uniform sampling, GS-RRT* search space is more oriented towards the goal configuration Z_{goal}. As a result, it discovers an initial path with fewer node visits.

Algorithm 1 provides pseudo code for the proposed GS-RRT* sampling process *'GS sample'*, which returns a random node configuration Z_{rand} using the steps given below. The Euclidean distance between the current node configuration and the goal node configuration is used to compute the ball radius R shown in Fig. 1(b) (Step1). Then, a random number is generated between 0 and 1 using the *'Rand'* function (Step 2). According to the value of the node counting variable *node_count* (Step 3), the ball radius R is changed. If the node

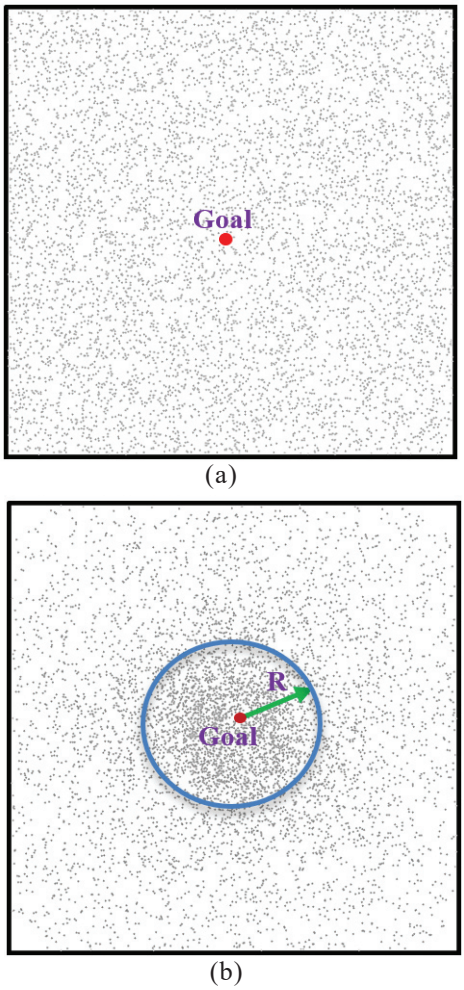

Figure 1. Samples generated with a goal as a center by: (a) RRT* and (b) GS-RRT* with 10000 samples.

count condition *node_count*<τ is met, the ball radius *R* is transformed into *R*×*rand* (Step 4). If the node count criterion *node_count*<τ is not met, Goal-oriented sampling with the default radius is performed (Step 6). A constant $\tau=a\times k$ is calculated using the path planning algorithm's number of samples *k* and a biasing percentage *a*. The number of samples *k* used in the experiment is 10000, and the biasing percentage *a* is selected as 0.2 based on the previous experimental study[35].

Algorithm 1

GS sample **procedure** (Z_{new}, Z_{goal})

1: $R \leftarrow norm(Z_{new}, Z_{goal})$;
2: $rand \leftarrow RandomNumber$; // between 0 to 1
3: **if** $node\_count < \tau$ **then**
4: $R \leftarrow R \times rand$;
5: **endif**
6: $Z_{rand} = sampleINBall(Z_{goal}, R)$;
7: return Z_{rand}

3.2 GS-RRT* Framework

Algorithm 2 gives the pseudocode for the proposed GS-RRT* framework. The proposed GS-RRT* framework combines the advantages of goal-oriented sampling and uniform sampling using a simple random selection[34] function *'Rand'* (step 3). A Goal-oriented Sampling procedure, function *'GS sample'* generates a random sample Z_{rand} (Step 4). To avoid the local minima problem and boost the success rate, the proposed research work generates a few samples from the entire *C*-space like RRT* using *'Uniformsample'* function (Step 6). The function '*nearest*' identifies and returns the node $Z_{nearest}$, the node closest to Z_{rand} from the vertices v based on the path length using a distance function (Step 7). The function *'steer'* drive the path σ from the node Z_{rand} to the nearest node $Z_{nearest}$ through a new node Z_{new} at a distance δ from the node $Z_{nearest}$ towards the node Z_{rand} (Step 8). The *'obstaclefree'* function determines whether the path σ is in the obstacle-free region or not (Step 9). If the edge between the nodes $Z_{nearest}$ and Z_{new} is in the obstacle collision-free zone, the function *'near_nodes'* finds and returns the nearby node Z_{near} that is in a ball with a volume of (β(log*n*/*n*)) around the node Z_{rand}, where β is a constant that varies depending on the planner (Step 10). The *'updateParent'* function[35] compares the path length of each path and selects the parent node Z_{parent} (Step 11). Then the *'add'* function adds the node Z_{new} to the tree *T* and assign the node Z_{parent} as its parent (Step 12). The 'rewire' function[35] checks the node path length and updates the tree *T* (Step 13).

Finally, the GS-RRT* establishes a feasible optimal path σ* between Z_{init} and Z_{goal} using a tree *T(v,e)* with a set of vertices *v* and edges *e.* Thus, the proposed GS-RRT* takes the sample points from the reduced configuration space through the *'GS sample'* function without wasting more samples. At the same time, it generates a few samples using *'Uniformsample'* function to avoid the local minima problem and boost the success rate.

Algorithm 2

GS - RRT*Framework

1: $v \leftarrow \{Z_{init}\}; e \leftarrow \varnothing; Z_{new} \leftarrow Z_{init}$;
2: **for** $i = 0$ to k **do**
3: **if** $Rand() > 0.5$ **then**
4: $Z_{rand} \leftarrow GS\,sample(Z_{new}, Z_{goal})$;
5: else
6: $Z_{rand} \leftarrow Uniformsample()$;
7: $Z_{nearest} \leftarrow nearest(Z_{rand}, v)$;
8: $(Z_{new}, \sigma) \leftarrow steer(Z_{nearest}, Z_{rand})$;
9: **if** $obstaclefree(\sigma)$ **then**
10: $Z_{near} \leftarrow near\_nodes(Z_{new}, T)$;
11: $(Z_{parent}, \sigma_{parent}) \leftarrow updateParent(Z_{near}, Z_{nearest}, Z_{new}, \sigma)$;
12: $T \leftarrow add(T, Z_{new}, Z_{parent}, \sigma_{parent})$;
13: $T \leftarrow rewire(T, Z_{new}, Z_{near})$;
14: **end if**
15: **end for**
16: **return** T;

3.3 Probabilistic Completeness

3.3.1 Theorem: The proposed GS-RRT Path Planning Algorithm is Probabilistic Complete*

For a given path-planning problem (Z_{init},Z_{goal},C_{free}), consider constants $x_0 \in$ R and d>0 and it depends on Z_{init} and Z_{goal}, such that $P(\{V^{GSRRT^*} \cap Z_{goal} \neq \varnothing\}) > 1\text{-}e^{-dx}\ \forall x > x_0$ Where V^{GSRRT^*} is the proposed GS-RRT* path planning algorithm vertices.

Proof: For any given environment, the proposed GS-RRT* path planning algorithm returns a connected graph, so the proposed GS-RRT* follows the RRT*probabilistic completeness.

3.4 Complexity Analysis

3.4.1 Time Complexity Analysis

The time complexity is measured by the number of process calls to the procedure that takes the longest time.

Let O_n^{ALG} represent the number of times the *'obstaclefree'* procedure call made by the given algorithm in *n* iterations.

Lemma 1. $O_n^{(GSRRT^*)} \in O(logn)$.

Proof. Let $U_n^{GSRRT^*}$ and $R_n^{GSRRT^*}$ denote the number of times the *'obstaclefree'* procedure call made in *'updateParent'* and the *'rewire'* procedure, then

$$O_n^{GSRRT^*} = U_n^{GSRRT^*} + R_n^{GSRRT^*} \quad (1)$$

The time complexity analysis of the proposed GS-RRT* algorithm is the same as the RRT* algorithm because the *'updateParent'* and the *'rewire'* procedures are not changed, $U_n^{GSRRT^*} \in O(\log n)$ and $R_n^{(GSRRT^*)} \in O(logn)$. Hence $O_n^{(GSRRT^*)} \in O(logn)$.

3.4.2 Space Complexity Analysis

Space complexity is defined as how much memory is used by a specific algorithm.

The proposed GS-RRT* path planning algorithm builds a tree $T_n=(v_n, e_n)$ and the size of memory consumed by the proposed GS-RRT* algorithm is found using the tree T_n.

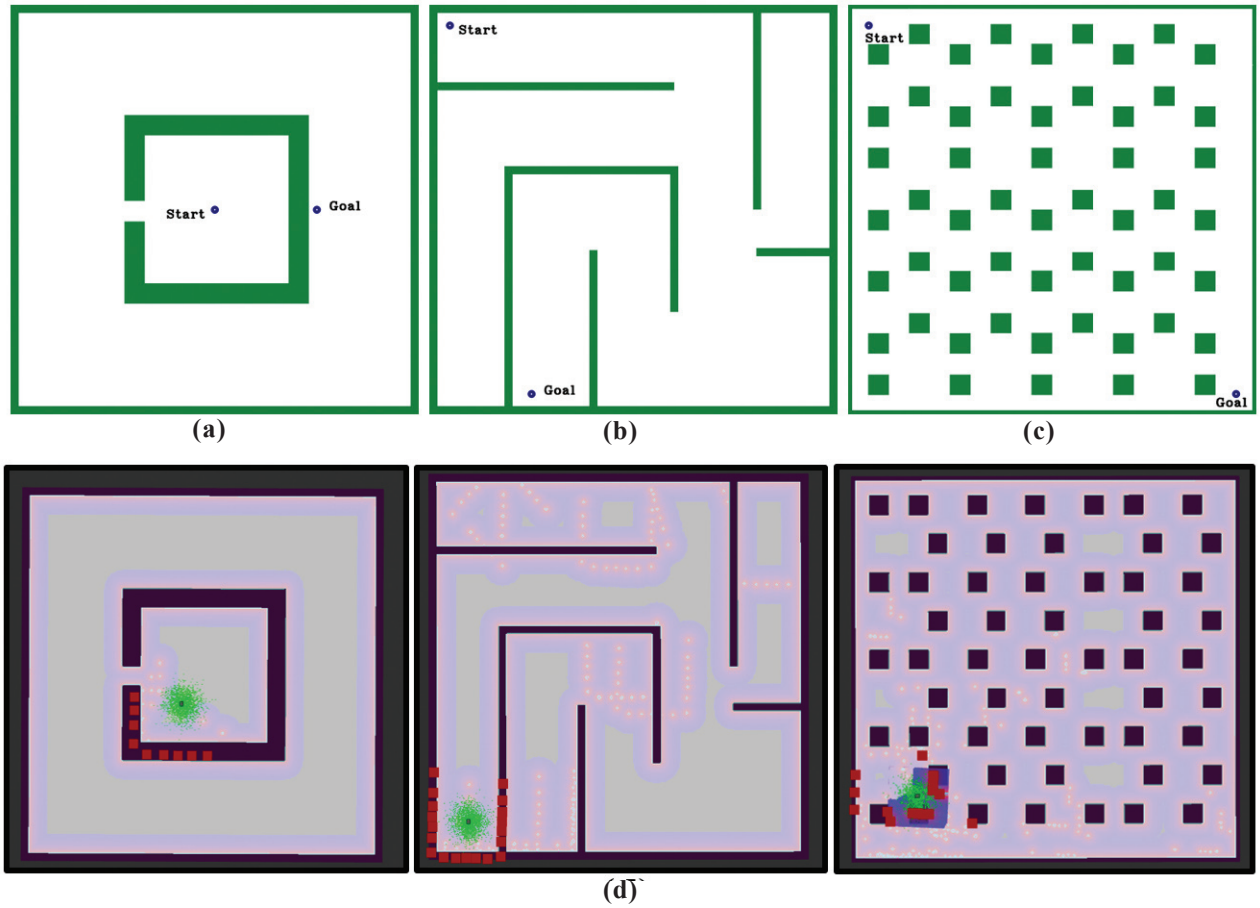

Figure 2. Environments: (a) Narrow, (b) Maze, and (c) Cluttered in (d) Rviz tool.

Lemma 2. $|T_n| \in O(n)$.

Proof: The size of the tree $|T_n|$ constructed by the proposed GS-RRT* algorithm is computed as $|T_n| = |v_n| + |e_n|$. If the tree $|T_n|$ visit 'n' nodes, then $|v_n| = n$, and $|e_n| = n-1$. Therefore, the size of the tree is

$|T_n| = |v_n| + |e_n| = 2n-1$. Hence $|T_n| \in O(n)$.

4. SIMULATION EXPERIMENT SETUP

To verify the effectiveness of the proposed GS-RRT* algorithm, a simple TurtleBot3 Autonomous Mobile Robot (AMR) is used in the gazebo simulation experiments. The proposed GS-RRT* is compared with the existing algorithms RRT, G-RRT, RRT*, and Informed-RRT* in three different environments namely narrow, maze, and cluttered environments as shown in Fig 2(a),(b)&(c) respectively. The 2D map is considered with a size of 384*384 pixels and 0.05 pixels/m resolution in the Gazebo simulations. Three performance metrics are used to compare the different path planning algorithms: path length denoted as p, the number of nodes visited denoted as n_v, and time to discover the solution denoted as t. Experimental simulations are conducted by employing the algorithms written in C++ code on an Intel i5-7500 CPU with 8 GB RAM, running on Ubuntu 18.04 with the Robot Operating System (ROS) Melodic. Because of the randomness of the sampling-based techniques, all of the algorithm's statistical data is generated from 100 independent runs. First, an environment is created in the Gazebo simulator using the building editor GUI as shown in Fig 2(a). A corresponding environment map is constructed with the help of the turtlebot3_teleoperation and the Simultaneous Localization and Mapping (SLAM) package[36].

The map_server package is used to store the created map and open the map when required. The ROS Visualization (RViz) tool is used to view the created environment map, as illustrated in Fig 2(b). The TurtleBot3 robot is localized in a given environment map with the help of the Adaptive Monte Carlo Localization (AMCL) algorithm, which uses a particle filter to locate the TurtleBot3's position and TurtleBot3 odometry information. The TurtleBot3 wheel optical encoders and IMU sensors, as well as the LiDAR, are used to update the TurtleBot3 position on the given environment map. The proposed GS-RRT* algorithm is implemented with a step size of 6.

Figure 3 depicts the path constructed with 10,000 samples in the three environments using RRT* and GS-RRT* algorithms. As illustrated in Fig. 3(a), a path is generated in RRT* after more node visits. Whereas GS-RRT* generates a path with a lower number of node visits, as shown in Fig. 3(b).

Statistical information of the path length p, time t, and the number of nodes visited n_v by various popular existing algorithms in the considered three environments is presented in Table 1. The proposed algorithm and all the baseline algorithms listed in Table 1 are offline planning algorithms, and they are

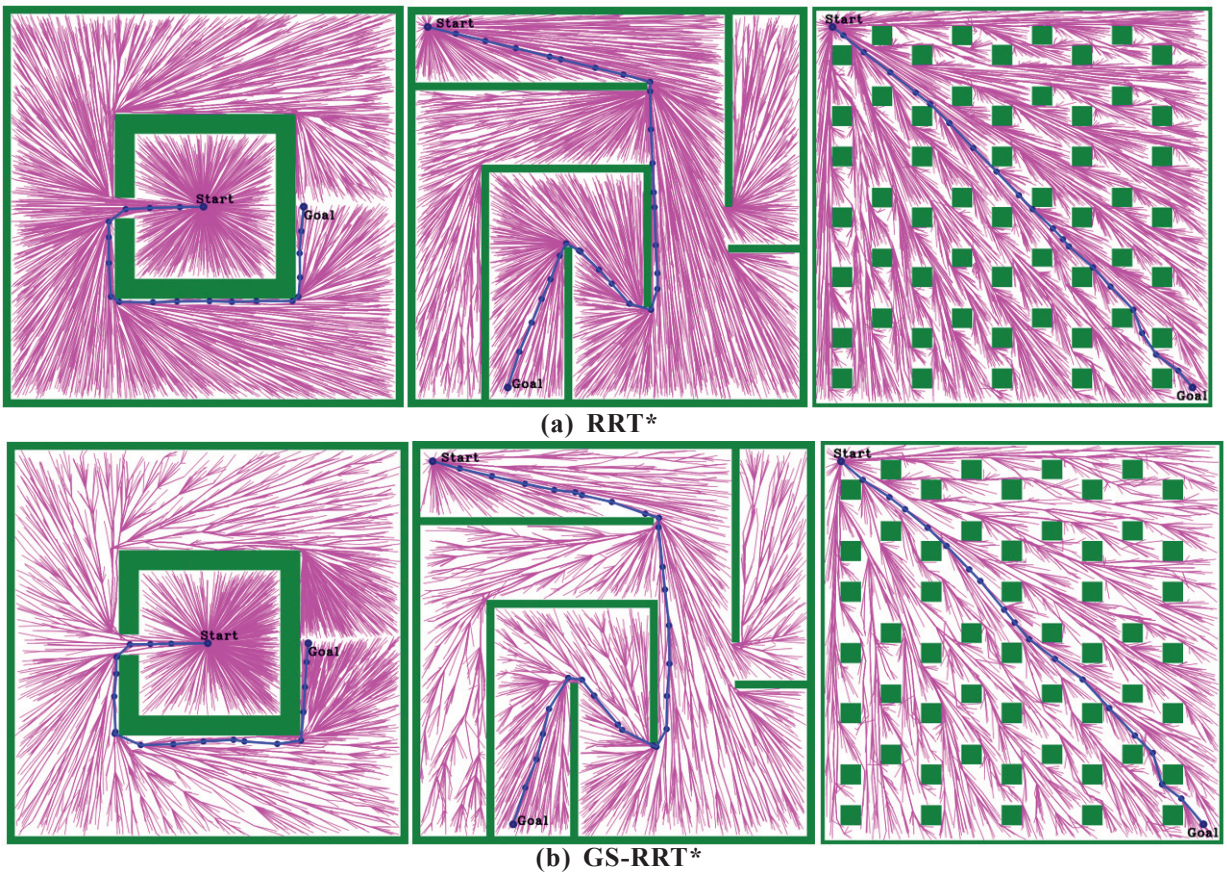

Figure 3. Path generation using (a) RRT* and (b) GS-RRT* with 10000 samples.

Table 1. Performance comparison of the GS-RRT* with some popular existing algorithms

Environment	Performance measure	RRT[12]	G-RRT[27]	RRT*[14]	Informed RRT*[19]	GS-RRT* (Proposed)
	Sampling strategy	**Uniform**	**Biased**	**Uniform**	**Adaptive**	**Biased**
Narrow	p (units)	1760	1624	1162	1194	1144
	$t(s)$	0.82	0.66	1.93	1.64	1.60
	n_v	7946	6583	7932	7236	6057
Maze	p (units)	2398	2285	1877	1928	1928
	$t(s)$	0.84	0.82	2.01	1.62	1.14
	n_v	8241	6704	8241	7384	6320
Cluttered	p (units)	1724	1717	1310	1297	1289
	$t(s)$	0.79	0.78	1.43	1.267	1.06
	n_v	8430	7200	8053	8015	6196

implemented using the TurtleBot3 robot's differential drive mechanism. The Euclidean metric is used to find the distance calculation for all the algorithms. However, different sampling strategies are used for all the algorithms, as listed in Table 1.

The path length performance of the proposed GS-RRT* algorithm with the existing algorithms is shown in Fig 4(a). When compared to all of the other algorithms in the three environments, the proposed GS-RRT* path length is the shortest. It is 2 % less compared with RRT* and 4 % less compared with Informed RRT*. The time performance of the proposed GS-RRT* algorithm with the existing algorithms is shown in Fig. 4(b). The graph clearly shows that the proposed GS-RRT* takes 20 % less time to build a path than the RRT*. The number of nodes visited by the proposed GS-RRT* algorithm with the existing algorithms is shown in Fig 4(c). It is clear from the plot that the number of nodes visited by the proposed GS-RRT* is less than RRT* by 26 % and 15 % less compared with Informed RRT*. The convergence rate of the proposed GS-RRT* and RRT* is computed in the three different environments. The convergence rate[26] is defined as $\frac{\text{plen}(\sigma_{init})-\text{plen}(\sigma^*)}{t_{opt}-t_{init}}$ (units/sec), whereas σ_{init} is the initial feasible path computed in t_{init} and σ^* is optimal path computed in t_{opt} time. The convergence rate (units/sec) of the proposed GS-RRT* is 0.5, 0.63, and 0.23 in the narrow, maze, and cluttered

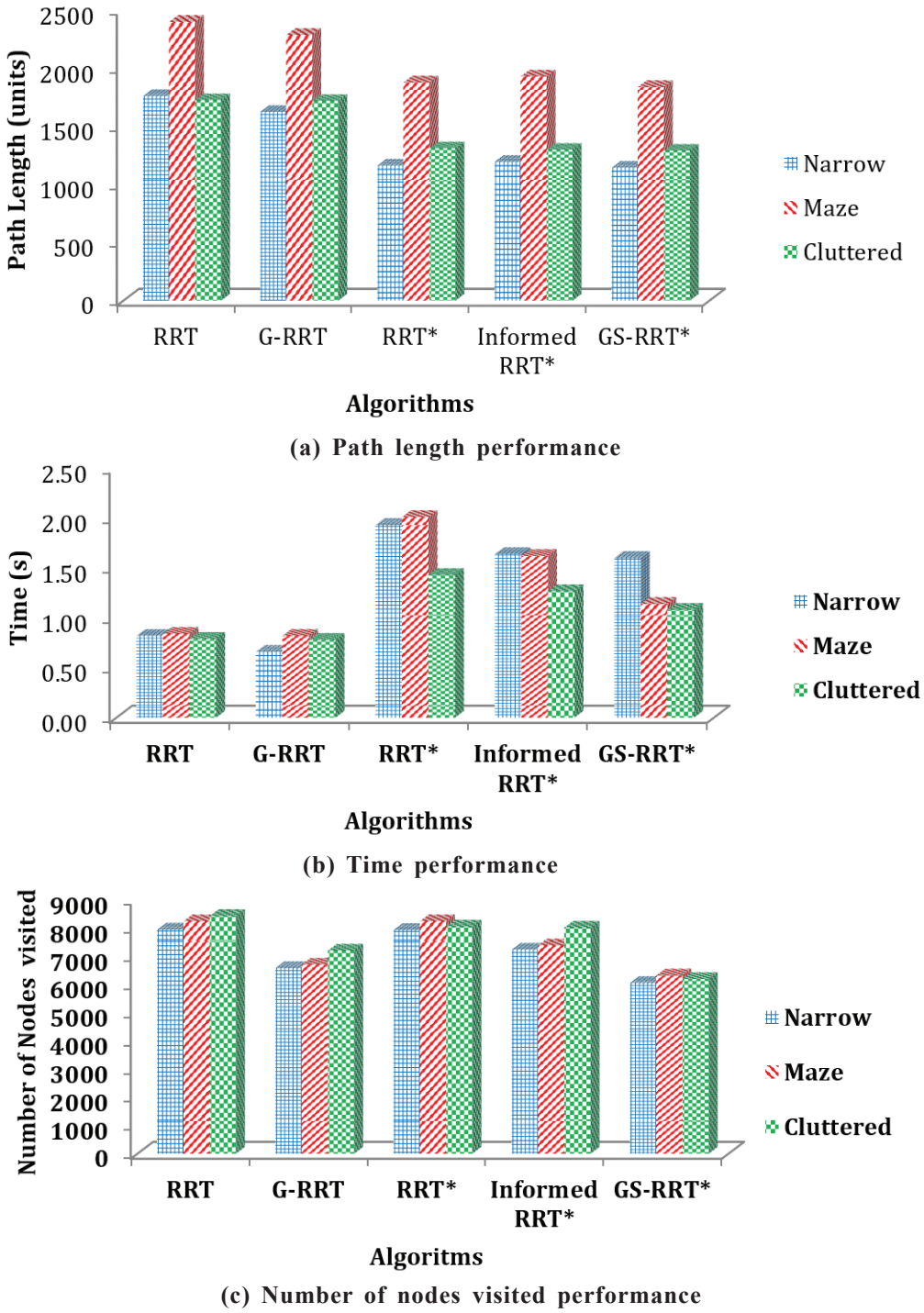

Figure 4. Performance of the proposed GS-RRT* algorithm with traditional algorithms concerning the path length, time, and the number of nodes visited.

environments, respectively, whereas the RRT* convergence rate is 0.42, 0.34, and 0.27, respectively. The proposed GS-RRT* convergence rate is 17 % improved over RRT* algorithm in the narrow environment and 60 % in the maze environment. At the same time, the convergence rate is 16 % less compared with RRT* in the cluttered environment. This is due to less difference between the initial to optimal time of the proposed GS-RRT*. Thus, the numerical results show that the proposed GS-RRT* framework improves the convergence rate by an average of 33 % over RRT* and 27 % over Informed RRT*; node exploration is 26 % better than RRT* and 20 % better than Informed RRT*; planning time improves by 34 % over RRT* and 17 % over Informed RRT*; and path length improves by 2 % over RRT* and 3 % over Informed RRT*, while maintaining the same computational complexity as RRT*.

5. REAL-TIME EXPERIMENTAL VALIDATION

In a real-time environment, the ROS Melodic and the TurtleBot3 robot are used to validate the proposed GS-RRT*

Table 2. Specifications of the TurtleBot3 robot with technical details

Specification	Technical detail
Single board computer	Raspberry Pi
Microcontroller	32-bit ARM Cortex®-M7
I/O Communication	10/100 Ethernet, 802.11n WiFi, and Bluetooth 4.1
Size (L x W x H)	138 mm x 178 mm x 192 mm
Speed	Max velocity linear 0.22 m/s and angular 2.84 rad/s
Weight	1kg
Actuator	DYNAMIXEL XL430-W250
Payload	15kg

path planning algorithm. The specifications of the TurtleBot3 robot are given in Table 2. A real environment map used in the proposed GS-RRT* algorithm evaluation is shown in Fig 5(a). The TurtleBot3 robot moves from the starting position to the goal location autonomously using the proposed GS-RRT* global path planning algorithm with a step size of 6. To handle both static and dynamic obstacles, the proposed GS-RRT* algorithm is used as a global path planning algorithm

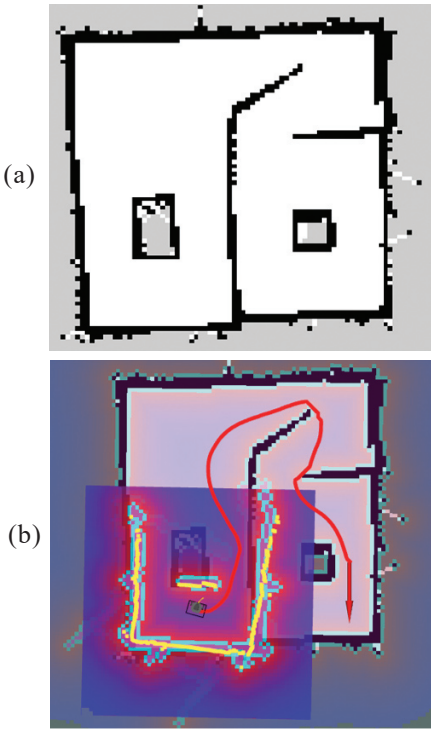

Figure 5. Navigation of TurtleBot3 Robot in the real-time environment using Rviz.

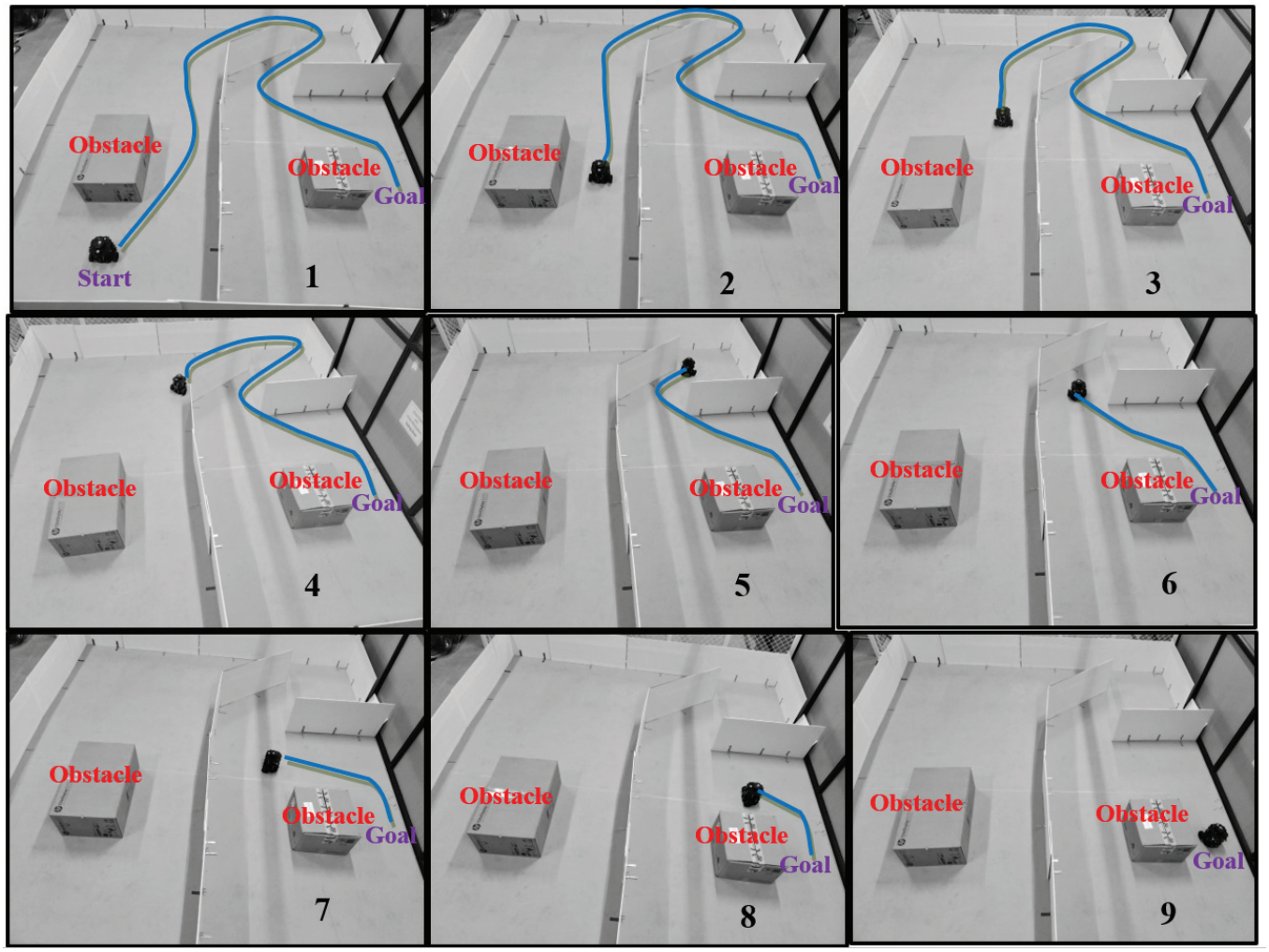

Figure 6. Navigation of TurtleBot3 robot in the real-time environment (1to9).

and a Dynamic-Window Approach (DWA) is employed as a local planning algorithm. As shown in Fig 5(b), the Rviz tool is used to display the generated path using the proposed GS-RRT* algorithm.

As shown in Fig. 6 (1 to 9), the generated path using the proposed GS-RRT* is verified in a real-time experiment using the TurtleBot3 robot. During the real-time experiment, the generated path is the same as the path obtained by the Gazebo Simulation results. While the generated global path plan is identical to that plan generated in the Gazebo simulation, the TurtleBot3 robot's real-time path execution differs slightly from the Gazebo simulation. This is due to the factors such as positional sensing error and wheel slippage on the robot. However, the results obtained are encouraging, as is the ease with which it was implemented.

6. CONCLUSIONS

Among sampling-based path-planning algorithms, RRT* is an optimal path-planning algorithm, but it suffers from a slow convergence problem. To address this problem, this paper proposed a Goal-oriented Sampling-based RRT* algorithm called GS-RRT*. The focus of the proposed GS-RRT* is to reduce unwanted sample exploration and solve the slow convergence problem of RRT* by taking more samples in the vicinity of the goal region. To avoid the local minima problem and boost the success rate, the proposed research work generates a few samples using uniform sampling using a simple random selection method. The proposed research work is compared with existing RRT* and Informed RRT* baseline algorithms using four metrics: path length, time to find the solution, the number of nodes visited, and the convergence rate. The numerical results demonstrate that the proposed GS-RRT* framework improves the convergence rate by 33 % over RRT* and 27 % over Informed RRT* while retaining the same computational complexity as RRT*. The proposed GS-RRT* framework is also validated in a real-time laboratory environment with the TurtleBot3 robot acting as a global planner, and it can be extended to act as a local planner. The proposed GS-RRT* path planning algorithm can be employed in real-time autonomous mobile robot navigation applications, such as warehouse applications.

REFERENCES

1. Gültekin Ö.; Cinar E,; Özkan K. & Yazıcı, A. Multisensory data fusion-based deep learning approach for fault diagnosis of an industrial autonomous transfer vehicle. *Expert Syst. Appl*., 2022, **200**, 117055. doi: 10.1016/j.eswa.2022.117055.
2. Cao, X.; Zou, X.; Jia, C.; Chen, M. & Zeng, Z. RRT-based path planning for an intelligent litchi-picking manipulator. *Comput. Electron. Agricult.*, 2019, **156**, 105-118. doi:10.1016/j.compag.2018.10.031.
3. Seidita, V.; Lanza, F.; Pipitone, A. & Chella, A. Robots as intelligent assistants to face COVID-19 pandemic. *Briefings in Bioinformat.*, 2021, **22**(2), 823-831. doi: 10.1093/bib/bbaa361.
4. Shen, Y.; Guo, D.; Long, F.; Mateos, L.A.; Ding, H.; Xiu, Z.; Hellman, R.B.; King, A.; Chen, S.; Zhang, C. & Tan, H. Robots under COVID-19 pandemic: A comprehensive survey. *IEEE Access*, 2020, **9**, 1590-1615. doi: 10.1109/ACCESS.2020.3045792.
5. Wang, X.V. & Wang, L. A literature survey of the robotic technologies during the COVID-19 pandemic. *J Manufacturing Syst.*, 2021,**60**, 823-836. doi: 10.1016/j.jmsy.2021.02.005.
6. Pani, A.; Mishra, S.; Golias, M. & Figliozzi, M. Evaluating public acceptance of autonomous delivery robots during COVID-19 pandemic. *Transport. Res. part D: Transport and Environ.*, 2020, **89**, 102600. doi: 10.1016/j.trd.2020.102600.
7. Karur, K.; Sharma, N.; Dharmatti, C. & Siegel, J.E. A survey of path planning algorithms for mobile robots. *Vehicles*, 2021, **3**(3), 448-468. doi:10.3390/vehicles3030027.
8. Rajchandar, K.; Baskaran, R.; Padmanabhan Panchu, K. & Rajmohan, M. An approach to improve multi-objective path planning for mobile robot navigation using the novel quadrant selection method. *Def. Sci. J.*, 2021, **71**(6). 748-761. doi:10.14429/dsj.71.16563.
9. Galceran, E. & Carreras, M. A survey on coverage path planning for robotics. *Robotics and Autonomous Syst.*, 2013, **61**(12), 1258-1276. doi:10.1016/j.robot.2013.09.004.
10. Jasmine, X.A. & Shantha, S.R. Behavior architecture controller for an autonomous robot navigation in an unknown environment to perform a given task. *Int. J. Physical Sci.*, 2015, **10**(5), 182-191. doi:10.5897/IJPS2014.4242.
11. Singh, G.K. & Gopal, A. Path planning in the presence of dynamically moving obstacles with uncertainty. *Def. Sci. J.*, 2010, **60**(1). 55-60. doi: 10.14429/dsj.60.107.
12. Elbanhawi, M. & Simic, M. Sampling-based robot motion planning: A review. *IEEE Access*, 2014, **2**, 56-77. doi: 10.1109/ACCESS.2014.2302442.
13. Kavraki, L.E.; Svestka, P.; Latombe, J.C. & Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 1996, **12**(4), 566-580. doi: 10.1109/70.508439.
14. LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. *Iowa State University, Tech. Rep,* (TR: 98–11), 1998. http://msl.cs.illinois.edu/~lavalle/papers/Lav98c.pdf (Accessed on 2008-06-30).
15. Svenstrup, M.; Bak, T. & Andersen, H.J. Minimising computational complexity of the RRT algorithm a practical approach. *In* IEEE International Conference on Robotics and Automation, 2011, 5602–5607. doi: 10.1109/ICRA.2011.5979540.
16. Karaman, S. & Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Res.*, 2011, **30**(7), 846-894.

doi:10.1177/0278364911406761.
17. Li, Y.; Wei, W.; Gao, Y.; Wang, D. & Fan, Z. PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Syst. Appl.*, 2020, **152**, 113425. doi:10.1016/j.eswa.2020.113425.
18. Akgun, B. & Stilman, M. Sampling heuristics for optimal motion planning in high dimensions. *In* IEEE/RSJ international conference on intelligent robots and systems, 2011, 2640-2645. doi: 10.1109/IROS.2011.6095077.
19. Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M. & Muhammad, M.S. RRT*-SMART: A rapid convergence implementation of RRT. *Int. J. Adv. Robotic Syst.*, 2013, **10**(7), 299. doi:10.5772/56718.
20. Kim, D.; Lee, J. & Yoon, S.E. Cloud RRT∗: Sampling cloud based RRT∗. *In* IEEE International Conference on Robotics and Automation, 2014, 2519-2526. doi: 10.1109/ICRA.2014.6907211.
21. Gammell, J.D.; Barfoot, T.D. & Srinivasa, S.S. Informed sampling for asymptotically optimal path planning. *IEEE Transact. Robotics*, 2018, **34**(4), 966-984. doi: 10.1109/TRO.2018.2830331.
22. Jeong, I.B.; Lee, S.J. & Kim, J.H. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Syst. Appl.*, 2019,**123,** 82-90. doi:10.1016/j.eswa.2019.01.032.
23. Wang, J.; Li, B. & Meng, M.Q.H. Kinematic constrained bi-directional rrt with efficient branch pruning for robot path planning. *Expert Syst. Appl.*, 2021, **170**, 114541. doi:10.1016/j.eswa.2020.114541.
24. Liao, B.; Wan, F.; Hua, Y.; Ma, R.; Zhu, S. & Qing, X. F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate. *Expert Syst. Appl.*, 2021, **184**, 115457. doi:10.1016/j.eswa.2021.115457.
25. Véras, L.G.D.; Medeiros, F.L. & Guimaráes, L.N. Systematic literature review of sampling process in rapidly-exploring random trees. *IEEE Access*, 2019, **7**, 50933-50953. doi: 10.1109/ACCESS.2019.2908100.
26. Qureshi, A.H. & Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, 2016, **40**(6), 1079-1093. doi:10.1007/s10514-015-9518-0.
27. Shen, J.; Fu, X.; Wang, H. & Shen, S. Fast path planning for underwater robots by combining goal-biased Gaussian sampling with focused optimal search. *Comput. Electrical Eng.*, 2021, **95**, 107412. doi:10.1016/j.compeleceng.2021.107412.
28. Kiesel, S.; Gu, T. & Ruml, W. An effort bias for sampling-based motion planning. *In* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017, 2864-2871. doi: 10.1109/IROS.2017.8206118.
29. Moses E, D.B. & Anitha, G. Goal-directed approach to autonomous motion planning for unmanned vehicles. *Def. Sci. J.*, 2017, **67**(1). 45-49, doi: 10.14429/dsj.67.10295.
30. Mohammed, H.; Romdhane, L. & Jaradat, M.A. RRT* N: An efficient approach to path planning in 3D for static and dynamic environments. *Advanced Robotics*, 2021, **35**(3-4), 168-180. doi: 10.1080/01691864.2020.1850349.
31. Noreen, I.; Khan, A.; Ryu, H.; Doh, N.L. & Habib, Z. Optimal path planning in cluttered environment using RRT*-AB. *Intelligent Service Robotics*, 2018, **11**(1), 41-52. doi:10.1007/s11370-017-0236-7.
32. Ganesan, S.; Natarajan, S.K. & Srinivasan, J. A global path planning algorithm for mobile robot in cluttered environments with an improved initial cost solution and convergence rate. *Arabian J. Sci. Eng.*, 2022, 1-15. doi:10.1007/s13369-021-06452-3.
33. Kang, G.; Kim, Y.B.; Lee, Y.H.; Oh, H.S.; You, W.S. & Choi, H.R. Sampling-based motion planning of manipulator with goal-oriented sampling. *Intelligent Service Robotics*, 2019, **12**(3), 265-273. doi:10.1007/s11370-019-00281-y.
34. Wang, J.; Chi, W.; Li, C.; Wang, C. & Meng, M.Q. Neural RRT*: Learning-based optimal path planning. *IEEE Transact. Automation Sci. Eng.*, 2020, **17**(4), 1748-1758. doi: 10.1109/TASE.2020.2976560.
35. Ganesan, S.; Natarajan, S.K. & Thondiyath, A. G-RRT*: Goal-oriented sampling-based RRT* path planning Algorithm for mobile robot navigation with improved convergence rate. *In* Advances in Robotics - 5th International Conference of The Robotics Society, Association for Computing Machinery, New York, USA, Article 23, 1–6. doi: 10.1145/3478586.3478588.
36. Huang, Y. & Gupta, K. RRT-SLAM for motion planning with motion and map uncertainty for robot exploration. *In* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, 1077-1082. doi: 10.1109/IROS.2008.4651183.

ACKNOWLEDGMENT

The authors would like to thank the DST-TEACHERS ASSOCIATESHIP FOR RESEARCH EXCELLENCE (TARE) Grant (TAR/2021/000214) for their support.

CONTRIBUTORS

Mr Sivasankar Ganesan obtained M.E. in Embedded and Real-Time Systems from Anna University, Coimbatore and is working as a Selection Grade Assistant Professor in the Department of Electronics and Communication Engineering at Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, India. His areas of interest are robotics and embedded systems.
He has contributed to the current study's conceptualisation, experimentation, validation, and paper writing.

Dr Senthil Kumar Natarajan obtained his PhD degree and is working as a Senior Professor and Head in the Department

of Electrical and Electronics Engineering at Mepco Schlenk Engineering College, Sivakasi. His areas of interest include Electronics Engineering, Power Converters, Microprocessors, Microcontrollers, Embedded Systems, Fuzzy and neural networks, Control systems, etc.
He has contributed to the validation of the methodology, the review and editing of the article, and the supervision of the current study.

Dr Asokan Thondiyath obtained his PhD in Mechanical Engineering from the IIT, Madras and is working as a Professor in the Department of Engineering Design at IIT, Madras, India. His areas of research are: Robotics, Product design, and Medical device design.
He has contributed to the validation of the methodology, the review and editing of the article, and the supervision of the current study.