Benchmark Analysis of Sampling Methods for RRT Path Planning

Gilang Nugraha Putu Pratama [1,*], Oktaf Agni Dhewa [2], Ardy Seto Priambodo [3], Faris Yusuf Baktiar [4], Rizky Hidayat Prasetyo [5], Mentari Putri Jati [6], Indra Hidayatulloh [7]
[1,2,3,4,5] Department of Electrical and Electronics Engineering, Universitas Negeri Yogyakarta, Yogyakarta, Indonesia
[6] Department of Electro-Optical Engineering, National Taipei University of Technology, Taipei, Taiwan
[7] Glasgow Caledonian University, Glasgow, Scotland
Email: [1] gilang.n.p.pratama@uny.ac.id, [6] t111999402@ntut.edu.tw, [7] jraeeuny@gmail.com
*Corresponding Author

Abstract*—Path planning is a crucial aspect of mobile robot navigation, ensuring that robots can safely travel from their initial position to the goal. In real-world applications, path planning is essential for autonomous vehicles, drones, warehouse robots, and rescue robots to navigate complex environments efficiently and safely. One effective method for path planning is the Rapidly-exploring Random Tree (RRT) algorithm, which is particularly practical in maze-like environments. The performance of RRT depends on the sampling methods used to explore the maze. Sampling methods are important because they determine how the algorithm explores the search space, affecting the efficiency and success of finding an optimal path. Poor sampling can lead to suboptimal or infeasible paths. In this study, we investigate different sampling strategies for RRT, specifically focusing on uniform sampling, Gaussian sampling, and the Motion Planning Network (MPNet) sampling. MPNet leverages a neural network trained on past environments, allowing it to predict promising regions of the search space quickly, unlike traditional methods like RRT that rely on random exploration without prior knowledge. This makes MPNet much faster and more efficient, especially in complex or high-dimensional spaces. Through a benchmarking analysis, we compare these methods in terms of their effectiveness in generating feasible paths. The results indicate that while all three methods are effective, MPNet sampling outperforms uniform and Gaussian sampling, particularly in terms of path length. The mean path length generated, based on a sample size of 30, is 13.115 meters for MPNet, which is shorter compared to uniform and Gaussian sampling, which are 18.27 meters and 18.088 meters, respectively. These findings highlight the potential to enhance path planning algorithms using learning-based sampling methods.

Keywords*—RRT, Path Planning, Sampling Methods, Benchmarking, Deep Learning

I. INTRODUCTION

Navigation is a necessary aspect of mobile robots, as it enables them to operate autonomously in dynamic and complex environments [1]. In real-world applications, navigation is crucial for ensuring that robots can perform tasks effectively without human intervention, whether it's for delivering goods in a warehouse, conducting inspections in hazardous areas, or monitoring crops in agriculture. Successful navigation allows robots to interact with their surroundings, avoid obstacles, and make real-time decisions to reach their destinations. It involves the ability of mobile robots to move through environments while avoiding obstacles and reaching their goals [2]. Safe and efficient path planning is mandatory in many mobile robot applications, such as industrial automation [3], search and rescue missions [4], surveillance [5], and even agriculture [6], where robots must navigate cluttered or unpredictable terrains.

To achieve effective navigation, path planning algorithms play a central role by generating feasible routes that guide the robot from its starting point to the target location. Among the various path planning methods, one of the most widely used is the Rapidly-exploring Random Tree (RRT) algorithm. RRT is particularly well-suited for environments with complex or unknown geometries, where traditional grid-based methods may struggle to efficiently explore the search space. The RRT is widely used for mobile robots, especially in intricate and high-dimensional spaces [7]-[10]. It was introduced by Steven M. LaValle in 1998 to address the limitations of traditional path planning methods, which often struggle to explore large and complex environments [11]. Technically speaking, RRT builds a tree of possible paths from the starting point toward the goal, making it a reliable method for navigating dynamic environments.

Despite its prominence, the performance and efficiency of the RRT algorithm are greatly influenced by the sampling methods used to explore the search space. It is fair to state that an effective sampling strategy is necessary, as it determines the quality of the generated paths and the time taken to reach the goal. Different sampling methods, such as uniform and Gaussian sampling, offer varying levels of efficiency, affecting the success rate of the navigation task.

Uniform sampling involves selecting random points from the search space with a uniform probability distribution. Every point in the space has an equal chance of being chosen [12]-[14]. Wang et al. use RRT with uniform sampling to ensure effective exploration of different regions. The method has practical applications for disinfecting robots in environments with complex obstacles, emphasizing efficiency and reliability in path planning [15].

On the other hand, Gaussian sampling involves generating random points around a mean with a certain standard deviation. Typically, a pair of samples are generated where one sample is near a randomly chosen point, and the other is farther away [16]-[18]. Some related research includes a study conducted by Yuncheng and Jie, who use the Gaussian sampling method for RRT path planning. Their

approach extends to higher-dimensional spaces, such as 3D, for practical applications in robot motion planning [19].

Meanwhile, deep learning-based sampling methods are revolutionizing the field of robotics [20]. Deep learning enables machines to adapt to intricate environments. It has significantly improved the capabilities of robots in perception, decision-making, and path planning [21], [22]. One notable application of deep learning in robotics is the development of the Motion Planning Network (MPNet), which has the potential to enhance the performance of path planning algorithms [23], [24]. MPNet leverages deep neural networks to predict promising regions of the search space, leading to more efficient and accurate path planning.

In this study, we will compare and benchmark the performance of RRT using different sampling methods—uniform, Gaussian, and MPNet—in a maze-like scenario. This comparison will help us understand the impact of deep learning on sampling efficiency and path quality in complex environments. The remainder of this paper is organized as follows. The necessary methods for RRT and the sampling methods are briefly explained in Section II. In Section III, we discuss the results of our simulations, covering the model building, simulation preparation, and analysis of the results. Finally, future work and the conclusions of this paper are presented in Section IV.

II. RRT Path Planning and Sampling Methods

This section covers several topics. Subsection II.A introduces path planning with RRT, providing a brief overview of basic planning and the importance of appropriate sampling methods. Subsections II.B and II.C focus on the uniform and Gaussian sampling methods, discussing their respective advantages and disadvantages. Finally, subsection II.D presents MPNet as a more effective alternative for sampling.

A. RRT Path Planning

The RRT algorithm was introduced by Steven M. LaValle in 1998 to handle high-dimensional motion planning problems [11]. The key feature of RRT is the ability to efficiently explore complex, high-dimensional spaces, making it useful for real-time path planning in many robotics applications [7]-[10].

The RRT works by incrementally growing a tree rooted at the start configuration. In each iteration, a random sample, q_rand, is generated in the configuration space. The nearest node in the tree, q_near, is identified, and a new node, q_new, is created by moving from q_near toward q_rand by a fixed step size, ϵ. The fixed step size (ε) affects the exploration efficiency and path smoothness. A smaller ε allows the tree to explore the space in finer detail, potentially leading to smoother and more precise paths, but at the cost of increased computation time. Conversely, a larger ε enables faster exploration by covering more space with each step, but it can result in less optimal, jagged paths.

It should be noted that the new node is added to the tree as long as it lies in a free space. It continues this process until a node is added near the goal configuration, successfully generating a path [11].

The main steps in RRT can be summarized as follows:

1. Sample a random point q_rand in the configuration space.
2. Find the nearest node in the existing tree, q_near.
3. Move from q_near toward q_rand by a step size ϵ to generate q_new.
4. Check if q_new is in the free space. If it is, add it to the tree.
5. Repeat until the goal is reached or a specified number of iterations is completed.

The efficiency in high-dimensional spaces makes RRT ideal for robot motion planning, autonomous vehicle navigation, and manipulation tasks. Nevertheless, in order to achieve such an efficiency, RRT needs a suitable sampling method [25]. They are crucial in RRT as they dictate how the configuration space is explored, by strategically selecting points in the space, the tree can grow efficiently, reaching different regions and avoiding unnecessary detours [26]. Proper sampling method improves the ability of RRT to navigate complex environments, such as those with narrow passages or obstacles. These methods also help balance exploration and exploitation, ensuring that the path toward the goal is both feasible and optimized. The right sampling strategy can significantly improve the performance of RRT and convergence rate.

B. Uniform Sampling Method

Uniform sampling in RRT is a method where samples are generated randomly and uniformly across the entire configuration space [12]. Uniform sampling ensures that the tree grows in an unbiased manner, covering all regions of the space equally. This approach works well in large, open areas, as it prevents the algorithm from being overly concentrated in any specific region and ensures broad exploration [13]. Uniform sampling assists the RRT algorithm in efficiently discovering feasible paths while avoiding obstacles and dead ends. It is also easy to implement and does not require additional computations for generating samples [14].

However, uniform sampling has notable limitations, especially in more complex environments. In environments with narrow passages or cluttered spaces, the random nature of uniform sampling often results in many samples being placed in open areas rather than in critical regions near narrow pathways. Since uniform sampling does not prioritize these narrow, solution-critical regions, it can take significantly more iterations to successfully navigate through tight spaces. Additionally, in large open areas, uniform sampling can lead to redundant tree growth because the algorithm may unnecessarily explore regions that are already well-covered, instead of focusing on areas closer to the goal. This inefficiency results in slower path generation and a less optimal exploration strategy, especially in environments with varying complexity. Thus, while uniform sampling is easy to implement, its lack of bias towards important regions can make it less effective in environments where strategic exploration is needed.

C. Gaussian Sampling Method

Gaussian sampling, as its name suggests, is a method where samples are drawn from a Gaussian distribution, focusing more on areas near obstacles or regions of interest [16]. This allows the RRT algorithm to concentrate its search in challenging regions, improving its performance in environments with complex geometries, such as narrow passages, where uniform sampling might struggle. By concentrating samples near obstacles, Gaussian sampling

helps the RRT algorithm explore critical, constrained areas with higher precision, improving its chances of finding feasible paths through tight spaces [17]. This method reduces the number of iterations needed to find a valid path compared to uniform sampling, especially in constrained spaces. The reason for this efficiency improvement lies in the fact that Gaussian sampling increases the likelihood of placing samples near difficult-to-navigate areas, such as obstacles and bottlenecks, where uniform sampling might overlook due to its equal treatment of the entire space. Focusing on these critical areas allows the algorithm to explore the essential regions earlier, shortening the search time.

However, Gaussian sampling may require fine-tuning of key parameters, such as the variance of the distribution, which controls how spread out or concentrated the samples are around obstacles. To optimize these parameters in practice, techniques like cross-validation or gradient-based optimization can be employed, adjusting the variance to strike a balance between broad exploration and precise exploitation [18]. A properly tuned variance ensures that the algorithm effectively navigates through both open spaces and tight corridors. In this scenario, Gaussian sampling improves efficiency by focusing samples around walls and narrow doorways, allowing the mobile robot to quickly find paths through the complex layout without being overwhelmed by irrelevant open areas.

While Gaussian sampling is particularly useful in scenarios where obstacles and tight spaces dominate the configuration space, its ability to selectively focus on critical regions is what gives it an edge over uniform sampling in terms of reducing the number of iterations and overall computational cost.

D. Motion Planning Networks

It is developed to tackle the limitations of traditional motion planning algorithms, which often struggle in high-dimensional spaces and complex environments. Traditional path planning algorithms like RRT are computationally intensive and slow when dealing with intricate obstacles. In 2019, A.H. Qureshi et al. introduced MPNet, which combines neural networks with classical sampling-based motion planning techniques. This hybrid approach enables the system to learn from past experiences and generalize to unseen environments [27]. Instead of randomly sampling points in every new environment, as done in methods like Gaussian and uniform sampling, MPNet is trained on a variety of past environments, allowing it to understand common path structures and efficiently apply this knowledge in new scenarios. When planning paths, MPNet focuses its exploration on the most promising regions, guided by its learned model, eliminating the inefficiency of random, potentially unproductive samples [28]. This focused exploration leads to the generation of shorter and smoother paths, reducing the need for post-processing to optimize the path. This innovation enables faster and more efficient path planning, as it can predict feasible paths even in challenging scenarios [29].

Furthermore, MPNet excels in high-dimensional spaces, where traditional methods face an exponential increase in complexity. By predicting points based on its understanding of the environment, MPNet avoids the inefficiencies of random sampling in such complex scenarios. As MPNet generalizes across different environments, it applies its learned knowledge to new spaces without needing to start from scratch, making it highly adaptable [30]. Finally, this informed sampling dramatically reduces the computational load, resulting in faster path planning and enabling real-time performance where traditional methods would struggle due to time constraints.

Technically, MPNet consists of two main modules. The first module encodes the input map environment into a compact representation using a basis point set encoding approach [30]. This encoded representation is smaller than the original map, which is particularly beneficial in real-time scenarios, where map environments are typically large and sparse. Encoding the map reduces input data sparsity, lowers computational complexity, and shortens the training time. The encoded environment is then stored as a map object.

The second module is a feed-forward neural network, consisting of an input layer, one or more hidden layers, and an output layer. Each hidden layer includes a fully connected layer, a ReLU (Rectified Linear Unit) activation layer, and a dropout layer. The ReLU activation layer introduces non-linearity, enabling the network to capture and represent complex, non-linear relationships in the data [31], [32]. The dropout layer helps prevent overfitting by randomly setting a fraction of input units to zero during training.

III. Results and Discussion

In this section, we will implement various sampling methods for RRT path planning on a mobile robot navigating a maze-like map. The robot starts from an initial position and must reach a designated goal. However, before proceeding, we need to prepare the data for training and validation, and create the MPNet model, as detailed in subsection III.A. Once the model is built, we can generate a new map with a defined start and goal for the robot, then apply the different sampling methods for RRT path planning. In subsection III.B, each sampling method is tested over 30 runs to generate a path using RRT. Finally, subsection III.C presents the analysis, evaluating parameters such as path length and initialization time to determine the most suitable sampling method among uniform, Gaussian, and MPNet for this scenario.

A. Build the Model

Before proceeding, we first need to explain the specifications of our simulation. We use MATLAB on Ubuntu 22.04 LTS to simulate and gather 80000 maze-like maps, splitting them in a 50:50 ratio for training and validation. These maps are represented as binary occupancy grids with two colors: black for stationary obstacles or walls and white for free space.

The data is trained for 50 epochs, with shuffling applied at each epoch. Additionally, the Adam optimizer is employed due to its reliability and computational efficiency [33]. It is a robust algorithm for training deep neural networks, aimed at minimizing regret—a measure used to evaluate the performance of adaptive learning rate optimization algorithms [34]. It uses AdaGrad's approach to adapt learning rates for each parameter based on the gradients' history, helping with sparse data. At the same time, it incorporates RMSProp's ability to adjust learning rates based on recent gradient magnitudes, making it effective in non-stationary settings. Then, by combining these techniques, Adam

balances fast convergence with robust performance, especially in deep learning tasks [35].

The training is performed over 50 epochs, with a total of 15250 iterations. As shown in Fig. 1, the training and validation losses are plotted on a logarithmic scale.

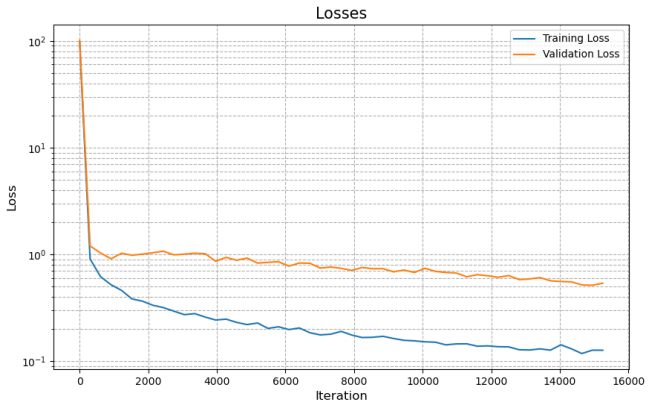

Fig. 1. The training and validation loss

The losses are decreasing over time, and by the end of the 15250 iterations, the training loss is 0.127, while the validation loss is 0.538. Since the validation loss is higher than the training loss, suggests potential overfitting. However, it is common and acceptable for the validation loss to be slightly larger than the training loss. Overfitting becomes a concern if the validation loss continuously increases as the training loss decreases. Since this is not the case here, the model is considered acceptable.

B. Simulation on Binary Occupancy Map

First, we need to define the binary occupancy map, as shown in Fig. 2.

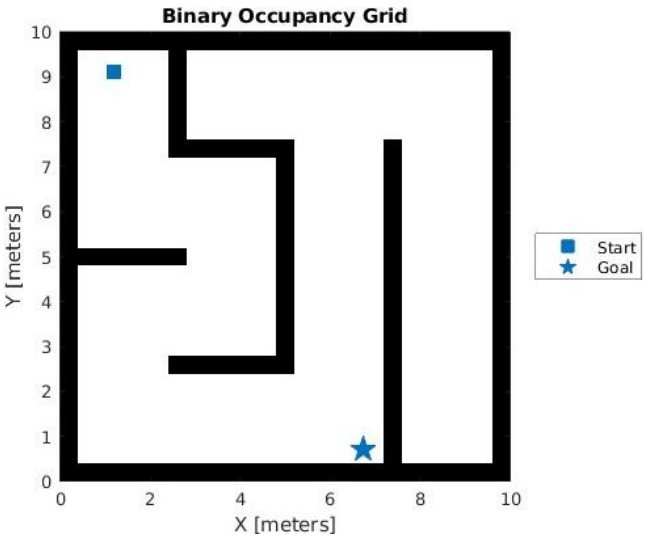

Fig. 2. Binary occupancy map

The map has dimensions of 10 by 10, where the mobile robot's initial position is at (7, 1), represented by a blue square, and the goal is at (1, 9), marked by a blue star. There are walls obstructing the path, and the generated path must efficiently guide the mobile robot from the start to the goal while avoiding these obstacles.

The next step is to apply the RRT algorithm using different sampling methods to generate a path toward the goal. The results are shown in Fig. 3, Fig. 4, and Fig. 5 for uniform sampling, Gaussian sampling, and MPNet, respectively.

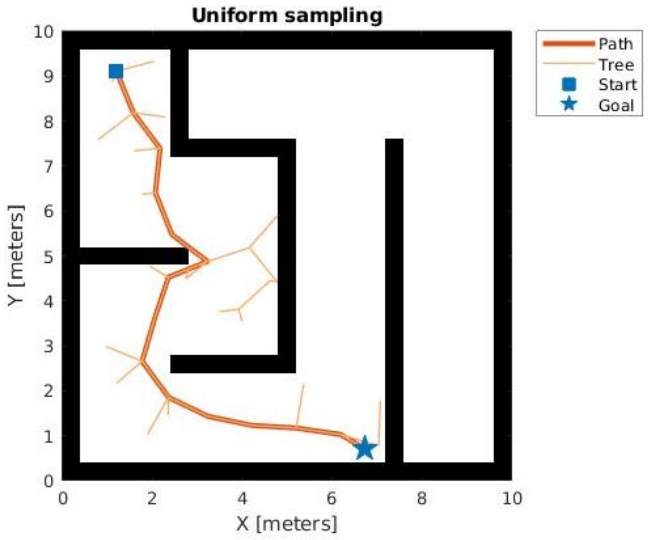

Fig. 3. Path generated with uniform sampling

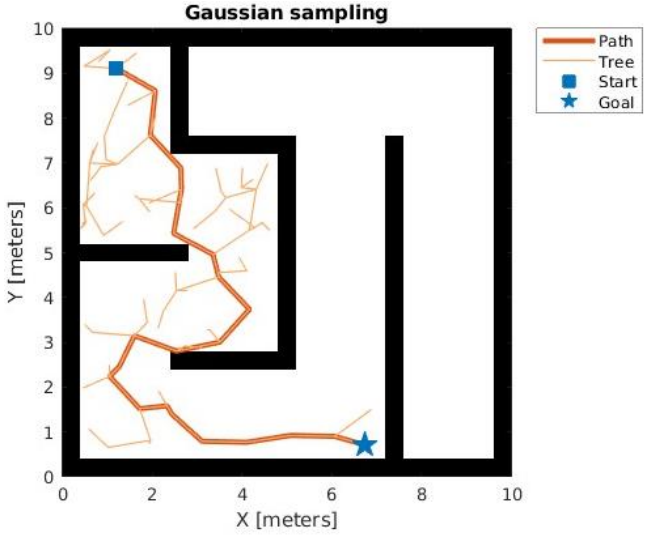

Fig. 4. Path generated with gaussian sampling

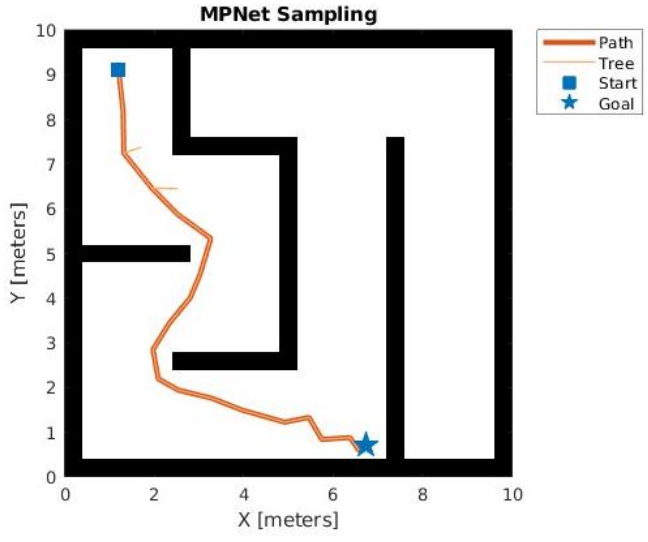

Fig. 5. Path generated with MPNet

While all sampling methods successfully generate a path, it is clear from the figures that MPNet outperforms the others. RRT with MPNet efficiently samples points to create a direct path to the goal. In contrast, both uniform and Gaussian sampling tend to be less efficient, as they select points farther from the goal. Additionally, the paths generated with uniform and Gaussian sampling are too close to the walls, increasing the risk of collision. Therefore, the path generated by RRT with MPNet is more efficient and safer compared to those produced by uniform and Gaussian sampling.

C. Benchmark Analysis

In this section, we will run simulations for the sampling methods to generate additional paths. The RRT with each sampling method will generate 30 paths.

It is evident that the path generated in each iteration will differ due to the inherent randomness of RRT. More importantly, from Fig. 6, we can see that MPNet (right) provides more convergent paths compared to the others. Both uniform (left) and Gaussian sampling (center) yield more variation than MPNet. We can further analyze this using statistical data for key parameters. Several parameters will be considered for analyzing the performance of these sampling methods.

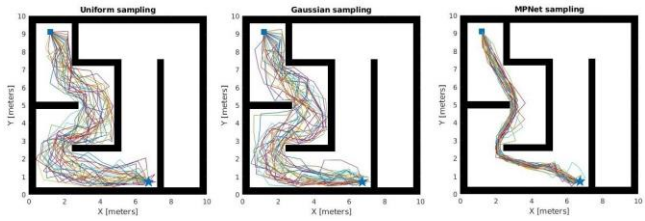

Fig. 6. Path comparison for each sampling method

The first parameter is the success rate of RRT with each sampling method in generating valid paths from the initial position to the goal, which indicates the effectiveness of the algorithm. Statistically, the success rate for each sampling method in generating valid paths is 100%, as shown in Fig. 7. This high success rate is largely attributed to the relatively simple structure of the map used in this scenario, which lacks highly complex or challenging obstacles.

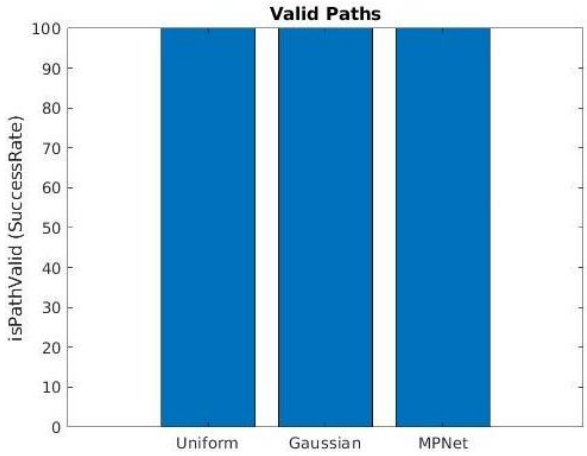

Fig. 7. Success rate for finding valid paths

The straightforward nature of the maze-like map allowed all sampling methods to consistently generate paths without failure. This implies that all of the methods can be implemented in our scenario for generating paths in a maze-like map with a sample size of 30. However, in more complex environments with tighter constraints or intricate obstacles, the success rate may vary depending on the sampling method used. The next parameter to consider is the path length, which is important because we aim for the paths to be both effective and efficient. In this case, shorter paths tend to be more efficient, and vice versa. After running the simulations, we gathered the statistics for the path lengths for each sampling method, as presented in Table 1.

Table 1. Path length

Sampling Methods	Path Length (meters)		
	Mean	Median	Standard Deviation
Uniform sampling	18.27	18.427	2.0727
Gaussian sampling	18.088	17.726	2.0159
MPNet	13.115	12.936	0.7231

Based on Table 1, it can be concluded that MPNet not only provides the shortest path but also tends to be consistent, with a low standard deviation. The mean and median path lengths using MPNet are 13.115 meters and 12.936 meters, respectively. These values are lower compared to uniform sampling, with a mean of 18.27 meters and a median of 18.427 meters, as well as Gaussian sampling, with a mean of 18.088 meters and a median of 17.726 meters. The standard deviation for MPNet is also the smallest. Specifically, the standard deviations for uniform sampling, Gaussian sampling, and MPNet are 2.0727 meters, 2.0159 meters, and 0.7231 meters, respectively. This indicates that the path lengths generated with MPNet are more consistent compared to the other two methods. The statistical visualization is shown in Fig. 8, where the box plot diagram for path lengths is presented.

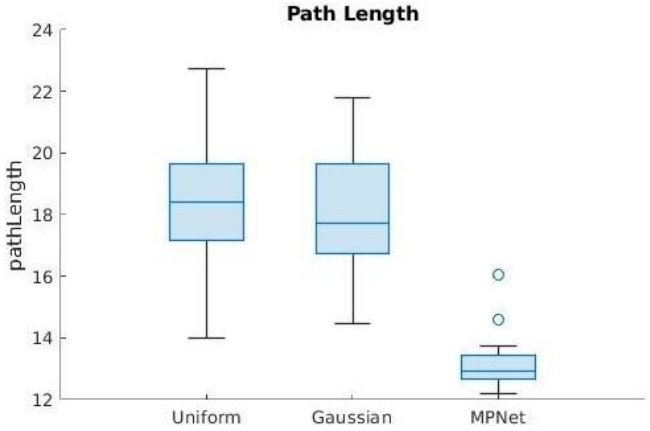

Fig. 8. Box plot diagram for path length

A shorter path and consistent performance is crucial for real-world applications such as autonomous navigation. Shorter paths directly reduce the time and energy required for a robot to reach its destination, which is especially critical in scenarios like drone delivery, autonomous driving, or industrial robots where efficiency translates to operational cost savings and faster task completion. Consistent path lengths, indicated by a lower standard deviation, also ensure more predictable behavior in complex environments, which is important for safety, especially in dynamic and unpredictable settings.

As seen in Fig. 7, the median (Q2) for MPNet is the lowest, followed by Gaussian and then uniform sampling. It also shows that there are outliers for MPNet, where the path lengths exceed 14 meters. However, these outliers are still smaller than the median values for both uniform and Gaussian sampling, suggesting that MPNet yields better overall results. Additionally, Fig. 7 shows that the longest path is generated using uniform sampling, with a length exceeding 22 meters. In practical terms, longer paths and higher variability can lead to increased energy consumption, slower task execution, and reduced reliability, all of which are undesirable in time-sensitive or resource-limited applications. Thus, the ability to consistently generate shorter and smoother paths highlights its effectiveness in real-world navigation tasks.

The last parameter to consider is the initialization time, which indicates how long it takes to initialize the function and execute it. The initialization time is measured during the execution of the planner. A shorter initialization time is desirable so that the generated path can be quickly built. Based on a sample size of 30 from the simulation, the statistics for initialization time are presented in Table 2.

Table 2. Initialization time

Sampling Methods	Initialization Time (ms)		
	Mean	Median	Standard Deviation
Uniform sampling	5.74	5.74	0
Gaussian Sampling	11.908	11.908	0
MPNet	5.41	5.41	0

As shown in Table 2, MPNet requires the shortest initialization time at only 5.41 milliseconds, whereas uniform and Gaussian sampling take 5.74 milliseconds and 11.908 milliseconds, respectively. Interestingly, the initialization times are consistent for each method, with a standard deviation of 0. This consistency means that the minimum, median, mean, and maximum values are the same for all of them. As a result, when visualizing the data in a box plot, we get straight lines, as shown in Fig. 9.

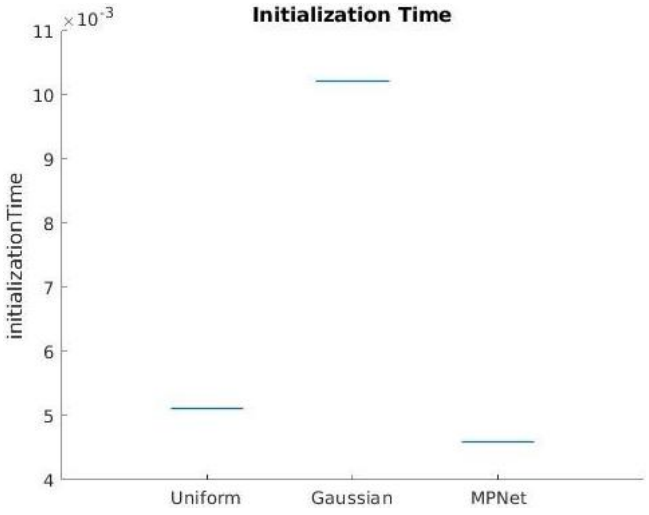

Fig. 9. Box plot diagram for initialization time

In Fig. 9, we can see that the minimum, mean, median, and maximum values are all the same, and there are no outliers.

IV. Conclusions and Future Works

In this paper, we conduct a benchmark analysis of sampling methods for RRT path planning. The sampling methods—uniform sampling, Gaussian sampling, and MPNet—all successfully generate valid paths with RRT. By analyzing data from a sample size of 30 generated paths, we conclude that MPNet produces shorter paths compared to the other methods. The mean path length generated by MPNet is 13.115 meters, which is shorter than the paths produced by uniform sampling at 18.27 meters and Gaussian sampling at 18.088 meters. Moreover, MPNet tends to generate more consistent paths, as evidenced by its low standard deviation of 0.7231 meters. In contrast, the standard deviations for uniform sampling and Gaussian sampling are larger, at 2.0727 meters and 2.0159 meters, respectively. Additionally, the initialization time for MPNet is the fastest, requiring only 5.41 milliseconds, compared to 5.74 milliseconds and 11.908 milliseconds for uniform and Gaussian sampling, respectively. The ability of MPNet to generate shorter and more consistent paths with faster initialization times makes it highly suitable for use in time-sensitive and resource-constrained environments, such as autonomous navigation in warehouses where a shorter path can significantly enhance the efficiency of robots tasked with retrieving or transporting goods, reducing operational time and energy consumption.

In the future, there are several directions for further research. We plan to implement RRT path planning on an actual mobile robot, and we are also considering developing a hybrid sampling method that combines Gaussian and uniform sampling. Such advancements could further optimize path planning for various real-world autonomous systems, pushing the boundaries of what these technologies can achieve in practical applications.

References

[1] N. Azhar and P. T. Aji, “Enhance the Balance of Quadruped Robot using CMPS12,” *Journal of Robotics, Automation, and Electronics Engineering*, vol. 1, no. 2, pp. 100-115, 2024, https://doi.org/10.21831/jraee.v1i2.170.

[2] O. Wahyunggoro, H. H. Triharminto, and A. I. Cahyadi, "Safe Robot Path Planning and Obstacle Avoidance using Efficient Genetic Algorithm,” *International Journal on Electrical Engineering and Informatics*, vol. 15, no. 3, pp. 387-400, 2023, https://doi.org/10.15676/ijeei.2023.15.3.2.

[3] H. Zhang, Y. Wang, J. Zheng and J. Yu, "Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments," *IEEE Access*, vol. 6, pp. 53296-53306, 2018, https://doi.org/10.1109/ACCESS.2018.2871222.

[4] S. H. Alsamhi *et al*., “UAV computing-assisted search and rescue mission framework for disaster and harsh environment mitigation,” *Drones*, vol. 6, no. 7, p. 154, 2022, https://doi.org/10.3390/drones6070154.

[5] G. M. De Lima Filho, A. Passaro, G. M. Delfino, L. De Santana and H. Monsuur, "Time-Critical Maritime UAV Mission Planning Using a Neural Network: An Operational View," *IEEE Access*, vol. 10, pp. 111749-111758, 2022, https://doi.org/10.1109/ACCESS.2022.3215646.

[6] H. S. Hewawasam, M. Y. Ibrahim and G. K. Appuhamillage, "Past, Present and Future of Path-Planning Algorithms for Mobile Robot Navigation in Dynamic Environments," *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, pp. 353-365, 2022, https://doi.org/10.1109/OJIES.2022.3179617.

[7] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi and I. Ahmedy, "Hybrid RRT: A Semi-Dual-Tree RRT-Based Motion Planner," *IEEE Access*, vol. 8, pp. 18658-18668, 2020, https://doi.org/10.1109/ACCESS.2020.2968471.

[8] J. Zhang, Y. An, J. Cao, S. Ouyang and L. Wang, "UAV Trajectory Planning for Complex Open Storage Environments Based on an Improved RRT Algorithm," *IEEE Access*, vol. 11, pp. 23189-23204, 2023, https://doi.org/10.1109/ACCESS.2023.3252018.

[9] W. Lan, X. Jin, T. Wang and H. Zhou, "Improved RRT Algorithms to Solve Path Planning of Multi-Glider in Time-Varying Ocean Currents," *IEEE Access*, vol. 9, pp. 158098-158115, 2021, https://doi.org/10.1109/ACCESS.2021.3130367.

[10] M. Cao, X. Zhou and Y. Ju, "Robot Motion Planning Based on Improved RRT Algorithm and RBF Neural Network Sliding," *IEEE Access*, vol. 11, pp. 121295-121305, 2023, https://doi.org/10.1109/ACCESS.2023.3327915.

[11] S. M. LaValle, "Motion Planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 79-89, 2011, https://doi.org/10.1109/MRA.2011.940276.

[12] L. G. D. O. Véras, F. L. L. Medeiros and L. N. F. Guimarães, "Systematic Literature Review of Sampling Process in Rapidly-Exploring Random Trees," *IEEE Access*, vol. 7, pp. 50933-50953, 2019, https://doi.org/10.1109/ACCESS.2019.2908100.

[13] M. Faroni and D. Berenson, "Motion Planning as Online Learning: A Multi-Armed Bandit Approach to Kinodynamic Sampling-Based Planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6651-6658, 2023, https://doi.org/10.1109/LRA.2023.3311262.

[14] I. Mitsioni, P. Tajvar, D. Kragic, J. Tumova and C. Pek, "Safe Data-Driven Model Predictive Control of Systems With Complex Dynamics," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 3242-3258, 2023, https://doi.org/10.1109/TRO.2023.3266995.

[15] H. Wang, X. Zhou, J. Li, Z. Yang, and L. Cao, "Improved RRT* Algorithm for Disinfecting Robot Path Planning," *Sensors*, vol. 24, no. 5, p. 1520, 2024, https://doi.org/10.3390/s24051520.

[16] I. Ahmad, M. Liaquat, F. M. Malik, H. Ullah and U. Ali, "Variants of the Sliding Mode Control in Presence of External Disturbance for Quadrotor," *IEEE Access*, vol. 8, pp. 227810-227824, 2020, https://doi.org/10.1109/ACCESS.2020.3041678.

[17] Y. Huang and H. H. Lee, "Adaptive Informed RRT*: Asymptotically Optimal Path Planning With Elliptical Sampling Pools in Narrow Passages," *International Journal of Control, Automation, and Systems*, vol. 22, pp. 241-251, 2024, https://doi.org/10.1007/s12555-022-0834-9.

[18] S. Kaden and U. Thomas, "Optimizing Mobility of Robotic Arms in Collision-free Motion Planning," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 49, 2021, https://doi.org/10.1007/s10846-021-01407-0.

[19] Li Yuncheng and Shao Jie, "A revised Gaussian distribution sampling scheme based on RRT* algorithms in robot motion planning," *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pp. 22-26, 2017, https://doi.org/10.1109/ICCAR.2017.7942654.

[20] A. Faust *et al*., "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113-5120, 2018, https://doi.org/10.1109/ICRA.2018.8461096.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, pp. 1-40, 2016, https://www.jmlr.org/papers/volume17/15-522/15-522.pdf.

[22] A. Giusti *et al*., "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661-667, 2016, https://doi.org/10.1109/LRA.2015.2509024.

[23] A. H. Qureshi and M. C. Yip, "Deeply Informed Neural Sampling for Robot Motion Planning," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6582-6588, 2018, https://doi.org/10.1109/IROS.2018.8593772.

[24] A. H. Qureshi, Y. Miao, A. Simeonov and M. C. Yip, "Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48-66, 2021, https://doi.org/10.1109/TRO.2020.3006716.

[25] J. D. Gammell, T. D. Barfoot and S. S. Srinivasa, "Informed Sampling for Asymptotically Optimal Path Planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966-984, 2018, https://doi.org/10.1109/TRO.2018.2830331.

[26] Z. Wang, P. Li, Z. Wang and Z. Li, "APG-RRT: Sampling-Based Path Planning Method for Small Autonomous Vehicle in Closed Scenarios," *IEEE Access*, vol. 12, pp. 25731-25739, 2024, https://doi.org/10.1109/ACCESS.2024.3359643.

[27] A. H. Qureshi, A. Simeonov, M. J. Bency and M. C. Yip, "Motion Planning Networks," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118-2124, 2019, https://doi.org/10.1109/ICRA.2019.8793889.

[28] L. Li, Y. Miao, A. H. Qureshi and M. C. Yip, "MPC-MPNet: Model-Predictive Motion Planning Networks for Fast, Near-Optimal Planning Under Kinodynamic Constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4496-4503, 2021, https://doi.org/10.1109/LRA.2021.3067847.

[29] A. H. Qureshi, J. Dong, A. Baig and M. C. Yip, "Constrained Motion Planning Networks X," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 868-886, 2022, https://doi.org/10.1109/TRO.2021.3096070.

[30] S. Prokudin, C. Lassner and J. Romero, "Efficient Learning on Point Clouds With Basis Point Sets," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4331-4340, 2019, https://doi.org/10.1109/ICCV.2019.00443.

[31] M. Ghafoor and T. Akutsu, "On the Generative Power of ReLU Network for Generating Similar Strings," *IEEE Access*, vol. 12, pp. 52603-52622, 2024, https://doi.org/10.1109/ACCESS.2024.3387306.

[32] Y. Takeishi, M. Iida, and J. Takeuchi, "Approximate spectral decomposition of Fisher information matrix for simple ReLU networks," *Neural Networks*, vol. 164, pp. 691-706, 2023, https://doi.org/10.1016/j.neunet.2023.05.017.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Arxiv*, 2015, https://doi.org/10.48550/arXiv.1412.6980.

[34] H. Iiduka, "Appropriate Learning Rates of Adaptive Learning Rate Optimization Algorithms for Training Deep Neural Networks," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13250-13261, 2022, https://doi.org/10.1109/TCYB.2021.3107415.

[35] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159, 2011, https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf.