2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Learning-Based Motion Planning with Mixture Density Networks

Yinghan Wang, Xiaoming Duan, and Jianping He

***Abstract*— The trade-off between computation time and path optimality is a key consideration in motion planning algorithms. While classical sampling based algorithms fall short of computational efficiency in high dimensional planning, learning based methods have shown great potential in achieving time efficient and optimal motion planning. The SOTA learning based motion planning algorithms utilize paths generated by sampling based methods as expert supervision data and train networks via regression techniques. However, these methods often overlook the important multimodal property of the optimal paths in the training set, making them incapable of finding good paths in some scenarios. In this paper, we propose a Multimodal Neuron Planner (MNP) based on the mixture density networks that explicitly takes into account the multimodality of the training data and simultaneously achieves time efficiency and path optimality. For environments represented by point clouds, MNP first efficiently compresses point clouds into a latent vector by encoding networks that are suitable for processing point clouds. We then design multimodal planning networks which enables MNP to learn and predict multiple optimal solutions. Simulation results show that our method outperforms SOTA learning based method MPNet and advanced sampling based methods IRRT* and BIT*.**

***Index Terms*— Motion planning, multimodal motion planner, point cloud, mixture density networks**

I. Introduction

Motion planning is one of the core research topics in robotics [1]. Motion planning algorithms aim to find a collision free path in an environment given the initial and goal configurations of the robot. The desirable properties of a motion planning algorithm include: (1) the algorithm is guaranteed to generate a collision free path if it exists (completeness), and the path found by the algorithm should have the low path cost (optimality); (2) the algorithm finds a path in real time (time efficiency); (3) the algorithm should be able to generate plans in high dimensional configuration space (generalization to high dimensions).

Numerous motion planning algorithms have been developed in the past decades, such as the artificial potential fields method [2], the vector field histogram method [3] and their improved versions [4] [5]. These algorithms are simple and useful, but not suitable for high dimensional planning. Sampling based methods such as RRT [6] try to find a collision free path through repetitively sampling in the configuration space. RRT* [7], IRRT* [8] and BIT* [9] are further developed to improve the optimality of the generated paths. While sampling based methods are able to find a collision free and near-optimal path and can be used to plan in high dimensions, there is a trade-off between the computing time and optimality. It often takes a lot of time for the sampling based methods to find an optimal path, which makes them not suitable for real time planning.

With the advancement and widespread application of machine learning algorithms [10] [11], learning based motion planning algorithms have been developed and have shown promising performance. To improve the efficiency for sampling based methods, the authors in [12] and [13] develop algorithms that learn and predict the sample distributions to guide the sampling process. The work [14], [15] and [16] utilize deep networks that learn from expert experiences and predict motions iteratively to improve the efficiency of planning. However, there are still problems with existing methods. The ability to process environment information properly is the key for generalization and the point clouds are suitable to represent obstacles in environments, while [14] does not take the environment information as the input and [15] [16] process point clouds in an inefficient way. On the other hand, it is common that there are multiple solutions for motion planning problems, but [15] and [16] lack the ability to learn and generate multiple solutions which may cause problems when learning from expert experiences.

In this paper, we propose a learning based motion planning algorithm called Multimodal Neural Planner (MNP) which takes in point clouds, robot's initial and goal configurations and plans a near-optimal path efficiently by iteratively generating the next configuration. The planner consisting of two networks can learn and generate multiple solutions. The environment encoding networks utilize PointNet [17] to compress the input point clouds into a latent vector efficiently, and the planning networks take in the latent vector and the robot's current and goal configuration and predict the next configuration. Inspired by [18], we describe the next configuration by a Gaussian mixture model whose parameters are determined by the output of planning networks. The Gaussian mixture model enables the planning networks to learn and predict multiple solutions. Based on the above two networks, we further utilize the bi-directional planning and the replanning mechanism to form a complete, near-optimal and efficient planner. Simulation results show that our method is able to solve motion planning problems in high dimensions and outperforms the SOTA learning based planner MPNet [15] and advanced sampling based planners such as IRRT* and BIT*.

The authors are with the Department of Automation, Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai, China, 200240. Email:{wyhboos, xduan, jphe}@sjtu.edu.cn.

This work was supported in part by Shanghai Pujiang Program under Grant 22PJ1404900, the Natural Science Foundation of Shanghai under Grant 23ZR1428900, and the National Natural Science Foundation of China under grant 62373247 and 62303314. (*Corresponding author: Xiaoming Duan.*)

II. Related Works

Imitation learning is one of the popular ways to solve motion planning problems through learning from expert' experiences. Many of the existing imitation learning methods adopt the end-to-end training framework to generate control commands from the input sensor information. In [19]–[23], the mobile robot learns from the expert's demonstrations with supervised learning. The expert's demonstrations are either generated by humans [20] [22] or by classical controllers [19] [21]. Several works also study how to improve the performance of the generated control command. In [24], the authors design a command-conditional controller that generates the steering command at road intersections. Active learning has been utilized in [25] to improve the ability to avoid obstacles which enhances safety. In [26], the low level controller is instructed by the high level planned route through the intention-net. In this work, we focus on planning a global path instead of designing learning based controllers.

Since sampling based methods are complete and optimal, they are widely used in the motion planning. To improve their efficiency, methods have been developed to guide the sampling process. In [27] [28] a probability map is generated for sampling and in [29] transformer is utilized to mark out the sampling area. However, [27] [29] are only suitable for environments in low dimensions. In [12] [13], the sample distribution has been predicted. In [30] the sample location is directly predicted, but it is unable to learn and predict multiple solutions. In [31] deep networks are used to help select the parent node and expand node of the search tree, it takes environment map as input while our method uses point clouds which is easier to obtain by sensors. By contrast, we propose a method that directly generates the path iteratively without using a search tree, and it can generate multiple solutions and be applied in high dimensional environments.

The deep networks are also utilized to directly generate paths for motion planning problems. In [32], the Waypoint Planning Networks predicts waypoints with deep networks and connects them by A* algorithm. In [33], the Pathnet directly generates the path by Generative Adversarial Networks [34]. However, the above two methods are suitable only for 2D planning. In [14] the path is generated by iteratively predicting the next configuration with LSTM [35], but it does not take environment information as input which makes it unable to generalize to new environments.

The Motion Planning Networks (MPNet) [16] [15] take the point clouds as input and iteratively predict next configuration to form a path. However, there are two main problems with MPNet. First, MPNet encodes point clouds with fully connected networks which is not suitable for processing disordered point clouds. Second, MPNet is trained on expert's data using regression techniques, and it cannot learn from and predict multiple solutions. Our method overcomes these critical issues. PG-RRT [36] is another sampling based method with deep networks that use mixture density networks to generate sampling distributions similar to ours. However, PG-RRT aims to solve the discontinuity in motion planning, and our method enables networks to learn and predict multiple solutions.

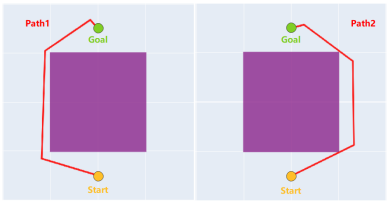


Fig. 1: Two different but near-optimal solutions for a simple motion planning problem.

In this paper, we mainly reveal an important yet overlooked issue with existing solutions to learning-based motion planning problems. That is, when learning from the expert data generated by sampling based methods, existing methods do not consider the fact that multiple different solutions with similar optimality could coexist in the data set. Due to this, they cannot mimic the behavior of the expert sampling based methods. To address this issue, we propose a mixture density network based multimodal planning network, which learns from the training data with multimodal property and can produce multiple solutions.

III. Problem Formulation and Motivation

A. Problem formulation

Let the robot configuration space be $C \subset \mathbb{R}^d$ where $d \in \mathbb{N}$ is the dimension of the configuration space. The configuration space C is comprised of the collision space C_{col} and the free space C_{free} where $C_{\text{free}} = C \setminus C_{\text{col}}$. The robot's physical environment space is called the workspace and is denoted by $W \subset \mathbb{R}^m$ where m is the dimension of the workspace. Similarly, the workspace is comprised of the collision space W_{obs} (regions occupied by obstacles) and the free space W_{free} where $W_{\text{free}} = W \setminus W_{\text{obs}}$. Let the robot's initial configuration be $c_{\text{init}} \in C_{\text{free}}$ and the robot's goal configuration space be $C_{\text{goal}} \subset C_{\text{free}}$, then the motion planning problem can be described as follows. Given the initial configuration and the goal configuration space, find a path represented by an ordered list $\tau = \{c_0, \ldots, c_i, \ldots, c_l\}$ such that $c_0 = c_{\text{init}}$, $c_l \in C_{\text{goal}}$ and $c_i \in C_{\text{free}}$ for $i = 0, \ldots, l$. Moreover, the line segment between c_i and c_{i+1} should lie entirely in the free configuration space C_{free}. A path that satisfies the conditions above is a feasible solution to the motion planning problem. We can evaluate the path quality and obtain a best one based on a cost function. Therefore, a good planning algorithm is expected to find a path that has the minimum cost value as measured by some cost function.

B. Multimodal property of motion planning solutions

The solution path generated by motion planning algorithms is generally not unique, i.e., there may exist different solution paths that have the same lowest cost values and all of them are in the solution set of the problem. We call such property of the motion planning problems the multimodal

property. Fig. 1 depicts a scenario where two different but near-optimal solutions are generated by RRT* for a 2D planning problem.

The performance of deep imitation learning based motion planning algorithms is profoundly affected by their ability to fit the expert data. In the above situation, it should learn all the choices that can possibly be made by the expert (e.g., RRT*). However, existing methods in [14]–[16], [30] which use a simple mean squared error (MSE) loss function to train the network are not capable of generating multimodal solutions. As mentioned in Section II, MPNet obtains a path by iteratively predicting the next configuration c_{next} based on the current configuration c_{current}, goal configuration c_{goal} and the environment information Z. In MPNet, the training data set generated by a sampling based method such as RRT* consists of tuples $(c_{\text{current}}, c_{\text{goal}}, Z, c_{\text{next}})$ where c_{current}, c_{goal}, Z are inputs and c_{next} is the output. During the training process, the following MSE loss function is utilized

$$\min_{\theta} \sum_{i=1}^{N} ||\hat{c_i}(\theta) - c_i| \,|_2^2,$$

where θ is the parameters of the planning networks, N is the number of training data, $\hat{c_i}(\theta)$ and c_i are the predicted and labeled configurations, respectively. While the MSE loss is simple and intuitive, it restricts the planning networks to generate only one solution for the motion planning problem. Moreover, when there are more than one solution in the training dataset, networks with the MSE loss fail to learn any of the solutions because an "average solution" leads to a minimum cost. For the simple 2D path planning problem in Fig. 1, with the MSE loss, the planning networks can only learn the average of the paths and outputs the decision of going up north at the initial position. This is unacceptable because it leads to collision with the obstacle. To alleviate this issue, MPNet utilizes dropout, which is proven to be a Bayesian approximation in [37], to promote randomness. However, as shown in Fig. 3 in our later experiment section, MPNet fails to predict the multimodal solution when the size of the training data becomes larger, where a 0.5 dropout probability is applied to each layer of the neural network [15]. Overall, MPNet with the MSE loss is not able to correctly learn and generate multiple solutions and can result in error when learning from multiple solutions.

In practice, it is quite common to get multiple solutions for one motion planning problem when generating training data using sampling based method. The multimodal problem not only exists in situations shown in Fig. 1 where there are two choices at the starting position, but it can also show up whenever multiple equally valid choices are available at a given location. To make planning networks smarter and more efficient, we need to develop a method that overcomes the above problems and allows the planning networks to learn and predict multiple solutions. The proposed multimodal planning networks in Section IV-B have the ability to learn multiple solutions by utilizing Mixture Density Networks.

IV. METHOD

The proposed framework is shown in Fig. 2. The proposed Multimodal Neural Planner consists of two parts, environment encoding networks (Enet) and the planning networks (Pnet). The Enet aims to encode the environment information which is represented by point clouds into a latent vector. We design our Enet by referring to PointNet [17] which is well suited for processing point clouds. The Pnet takes the latent vector obtained from the Enet as input and iteratively generates waypoints to form a path. As the expert data are multimodal, we utilize the Mixture Density Networks [18] to learn from the expert and generate waypoints.

A. Environment encoding networks

The use of point clouds to represent environment information is practical and common [15]. The point clouds characterize the geometric features of the environment, and extracting and understanding those features is very important for the downstream motion planning task. In this work, we develop environment encoding networks to process the point clouds and convert them into a latent vector

$$Z = \text{Enet}(X_{\text{cloud}}; \theta_e),$$

where $X_{\text{cloud}} = \{x_1, \ldots, x_n\}$ is the set of point clouds with $x_i \in W$, and θ_e is the parameters of the Enet.

To learn the environment representation, MPNet utilizes the multi-layer perception (MLP) to process the point clouds. However, since point clouds do not have a fixed order, MLP is not an efficient method to process them. To overcome this issue and similar to [38], we use the PointNet [17] to obtain global features, where a shared MLP and a maxplooling layer are utilized. The architecture of the Enet is shown in Fig. 2. For the training of the Enet, unlike [15], [16], [30], [36], we do not use the encoder-decoder approach with unsupervised learning. Instead, we train the Enet and the Pnet together. The reason for this is that the Enet should be well informed of the downstream motion planning task in order for it to retain critical environmental features relevant to the specific task, and when trained together with the Pnet, it receives error feedback from the output of the Pnet based on which the parameters can be adjusted.

B. Multimodal planning networks

After obtaining the latent vector Z through the Enet, the multimodal Pnet is designed to generate a full path iteratively. The architecture of the Pnet is also shown in Fig. 2. The Pnet takes in Z, the current configuration c_t and the goal configuration c_{goal}, and it predicts the next configuration c_{t+1}. Then, c_t is replaced with c_{t+1} to generate a full path iteratively. As mentioned in Section III-B, there are multiple different solutions to a motion planning problem, and the Pnet must be able to learn and predict these solutions. To achieve this, we utilize a probability distribution to describe the predicted next configuration. Specifically, we adopt the Mixture Density Networks (MDN) which utilize deep networks to represent the parameters of the Gaussian Mixture Model (GMM) as the backbone of our Pnet. In our Pnet,

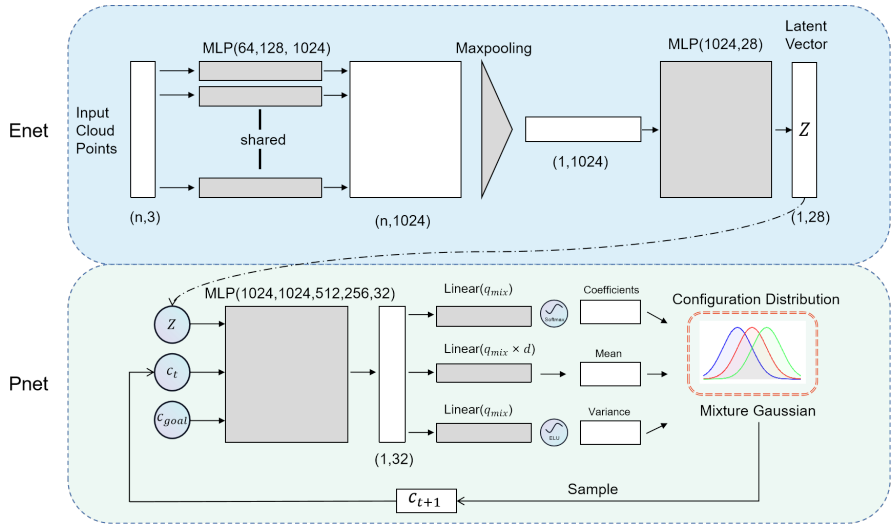


Fig. 2: The overview of the structure of the proposed method. Enet takes n point clouds as input and each point is processed by the same MLP. Then, a maxpooling operation is performed along the dimension of the number of point clouds n. Finally, another MLP is applied to obtain the latent vector Z. Pnet takes in the latent vector Z, the current configuration c_t and the goal configuration c_{goal}, through MLP and specific activation functions, and it generates the coefficients, mean and variance of the mixture Gaussian distribution with which we can obtain the next configuration c_{t+1} by sampling. Finally, we replace c_t with c_{t+1} and run Pnet iteratively to obtain a complete path.

MDN predicts the distribution of the next configuration with GMM. The GMM is determined by the following parameters: coefficients α_i of each Gaussian distribution, mean μ_i of each Gaussian distribution and variance σ_i of each Gaussian distribution. Note that we assume that each component of the output vector is independent and its variance is described by a common σ_i for each Gaussian. Under such an assumption, the GMM can still approximate any distribution [18] given enough Gaussian distribution. Since the parameters α and σ must satisfy certain constraints, different activation functions are used. We apply the Softmax activation to obtain α so that each α_i is non-negative and the sum of α_i's is one. For σ, we use the ELU activation instead of the EXP activation for better numerical statbility [39]. The parameters μ are obtained directly from the output of networks. Through the Pnet, we can obtain the distribution of the robot's next configuration as

$$p(c|\alpha,\mu,\sigma) = \sum_{i=1}^{q_{\text{mix}}} \alpha_i \frac{1}{(2\pi)^{d/2}\sigma_i^d} e^{-\frac{||c-\mu_i||^2}{2\sigma_i^2}}, \qquad (1)$$

where c is the robot configuration in dimension d, and q_{mix} is the number of Gaussian distributions. The parameters α, μ and σ are the output of the Pnet

$$(\alpha,\mu,\sigma) = \text{Pnet}(c_t, c_{\text{goal}}, Z; \theta_p), \qquad (2)$$

where θ_p is its parameters. Finally, we sample from (1) to obtain the next configuration $\hat{c}_{t+1} = \text{Sample}(p(c|\alpha,\mu,\sigma))$.

As discussed in Section IV-A, we train both the Enet and the Pnet together in a supervised end-to-end manner. Since the Pnet outputs a probability distribution, the negative logarithmic likelihood (NLL) is adopted as the loss function, and the optimization problem can be described by

$$\min_{\theta_e,\theta_p} -\sum_{i=1}^{N} \ln p(c_{t+1}),$$

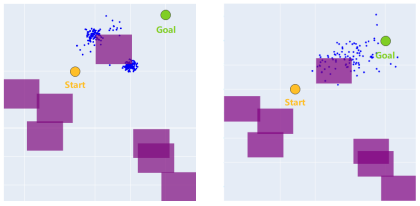
where N is the number of training data points and $p(c_{t+1})$ is obtained from (1) and (2). The training data is organized into the form of $(c_t, c_{\text{goal}}, X_{\text{cloud}}, c_{t+1})$ with $(c_t, c_{\text{goal}}, X_{\text{cloud}})$ as input and c_{t+1} as output, and the training data is acquired from sampling based methods such as RRT* with sufficient running time. More training details will be discussed in Section V. The proposed motion planning method Multimodal Neuron Planner (MPN) is composed of the Enet and the Pnet. It uses a similar algorithmic framework to MPNet which generates path by iteratively predicting the next configuration and uses bi-directional planning and replanning mechanisms.

V. Results

We present the results of simulation studies in this section. First, we show the multimodal property of the proposed MNP by visualizing the trained GMM in Section V-A. Then, we comprehensively compare the MNP with the SOTA MPNet and advanced sampling based methods such as BIT* and Informed-RRT* (IRRT*) using multiple different robot models in different environments in Section V-B. Finally, we compare above algorithms for the motion planning task of a 7-DOF Panda Arm in Section V-C.

A. Comparison of algorithms' multimodal property

To evaluate how well our multimodal Pnet learns and generates multiple solutions, we repetitively generate the next configuration for a point robot model in simple 2D environments and plot the distributions. A case is shown



(a) MNP (b) MPNet

Fig. 3: Distributions of the predicted next configuration by MNP and MPNet. The obstacles are marked in purple. For MNP, the distribution is obtained by repetitive sampling, and for MPNet, these points are generated through multiple forward propagation where stochastic property is introduced by the dropout layer.

in Fig. 3, MNP generates multiple reasonable next configurations, while MPNet generates relatively monotonous configurations that lie around the straight line to the goal even when the dropout layer is applied. We also compare the collision rate of the generated configurations which profoundly influences the algorithm's efficiency. Our multimodal Pnet has a lower collision rate at 20% while MPNet has 42%. The above shows that MNP can learn and predict multiple solutions and outperforms MPNet.

B. Comparisons using different robot models

In this subsection, we compare the proposed MNP, MPNet and advanced sampling based methods IRRT* and BIT* in 2D and 3D environments with different robot models including the point mass, a rigid body, a two-link manipulator and a three-link manipulator. To avoid complete folding, the angle range of the link manipulators is set to $(-0.75\pi, 0.75\pi)$. Training data are from 900 environments called seen environments, and both seen and unseen environments will be used to assess how well the learned results generalize. For each of 900 seen environments and 100 unseen environments, we generate 10 random initial and goal configurations for comparison. Fig. 4 shows the paths planned by MNP with different robot models. Table I and Table II show the success rate and computation time, respectively.

From Table I we can see that without RRT as the replanner, the success rate of MPNet(NR) decreases rapidly as the dimension of the configuration space increases, while MNP(Origin) retains a high success rate. For the 2D 3-link manipulator where $C \subset \mathbb{R}^5$, the success rate of MNP(Origin) reaches 93.3% while MPNet(NR) only has 38%. Table II shows that MNP(RRT) takes the minimum computation time for planning compared with MPNet(HR), IRRT* and BIT*. For the 2D point mass, MPN(RRT) is 4x faster than MPNet(HR) and 6x faster than BIT* with a 100% success rate. For the 2D rigid body, MPN(RRT) is 2x faster than MPNet(HR) and 13x faster than BIT* with a 99.8% success rate. For the 2D 3-link manipulator, MPN(RRT) is 4x faster than MPNet(HR) and 20x faster than BIT* with a 99.4% success rate. Comparison results in unseen environments also show that MPN(RRT) can generalize to new environments. In the higher configuration dimension, the improvement of MNP(RRT) with respect to MPNet and BIT* becomes more significant. Note that, for training MNP and MPNet with the 2-link and 3-link manipulators, we remove the trivial data where initial and goal configurations can be connected directly using a straight line from the training dataset, which seems to be helpful for improving the performance of MNP.

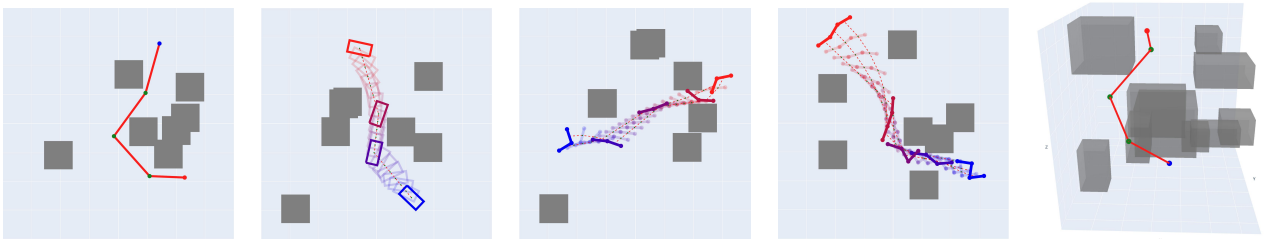
C. Comparisons using a 7-DOF Panda Arm

We further evaluate MNP's performance in high dimensions using a 7-DOF Panda Arm. We evaluate the methods in 10 random environments with 30 random initial and goal configurations for each environments which are not in training dataset. Fig. 5 shows a trajectory planned by MNP(RRT) that Panda Arm follows from the initial to the goal configuration. Table III shows the comparison results. The success rate of MNP(RRT) is 96.7%, which is as high as BIT* whose maximum computation time is set to 60s. MNP(RRT) achieves minimum computation time which is 1.7x faster than MPNet(HR) and 11x faster than BIT*(10%).

The overall comparison results show that our proposed MNP can learn and generate multiple solutions and it takes less computation time than MPNet and advanced sampling based methods such as IRRT* and BIT* to obtain a near-optimal solution. MNP can also generalize to new environments thanks to the Enet's ability to process point clouds. Finally, our method is effective for planning in high dimensional configuration space such as in robotic arm motion planning problems.

VI. Discussion

In this paper, we address the multimodal property of motion planning problems and its influence on motion planning algorithms based on deep imitation learning, and we propose to use the mixture Gaussian model with mixture density networks to learn from multimodal solutions. In MPNet, the authors use dropout layers to introduce stochasticity, but their aim is to generate different configurations for replanning, and they do not focus on learning multimodal solutions. Although the dropout mechanism is proven to be a Bayesian approximation and has the potential to learn multimodal solutions [37], its ability to learn multimodal solutions degenerates rapidly as the size of training dataset increases. For the MPNet where the dropout probability is set to be 0.5 for each layer, it demonstrates strong ability to learn multimodal solutions when the size of the training dataset is smaller than 10k. However, its performance degrades a lot when the size of the dataset increases to 100k and 1000k. When the dataset grows lager, MPNet seems to learn the "average" of the multimodal solution as shown in Sec V-A and Fig. 3, which results in a high collision rate of the predicted configurations and low efficiency of the algorithm. Increasing the complexity of the network may improve the effectiveness of the dropout mechanism, but the network may become very inefficient. For example, the Pnet of MPNet has 3.76M parameters, but it fails to learn multimodal solutions with the point mass model in 2D environments.



(a) 2D point. (b) 2D rigid. (c) 2D 2-link. (d) 2D 3-link. (e) 3D point.

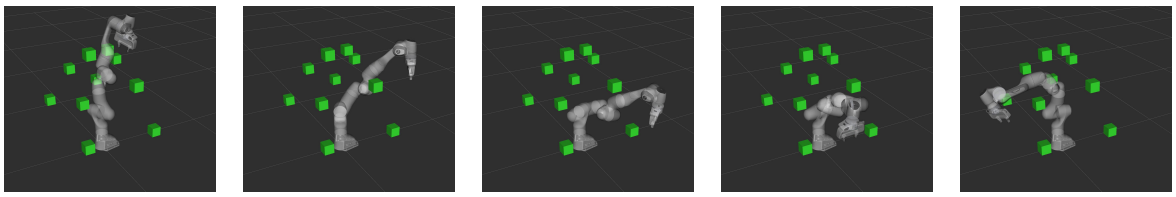
Fig. 4: Paths planned by MNP in different environments. Initial and goal configurations are in blue and red, respectively.

Method	Environment & Robot Model									
	2D Point		2D Rigid		2D 2-link		2D 3-link		3D Point	
	seen	unseen	seen	unseen	seen	unseen	seen	unseen	seen	unseen
MPNet(NR)	0.81	0.79	0.777	0.736	0.524	0.542	0.380	0.386	0.762	0.753
MNP(Origin)	0.967	0.966	0.829	0.833	0.922	0.94	0.933	0.947	0.872	0.891
MPNet(HR)	**1**	0.99	0.995	0.997	0.997	0.997	0.979	0.954	0.999	0.999
MNP(RRT)	**1**	**1**	0.998	0.997	0.998	**1**	0.994	0.991	0.999	**1**
IRRT*	**1**	**1**	0.998	0.994	0.994	0.998	0.977	0.983	0.999	**1**
BIT*	**1**	**1**	**1**	**1**	**1**	**1**	**0.999**	**0.999**	**1**	**1**

TABLE I: Mean success rate comparison results. MPNet(NR) is MPNet with neuro replanning and MPNet(HR) is MPNet with hybrid replanning. MNP(Origin) has no replanner and MNP(RRT) has RRT as its replanner. For the top two methods of the table, since RRT is not used for replanning, they directly show the performance of the learned networks.

Method	Environment & Robot Model									
	2D Point		2D Rigid		2D 2-link		2D 3-link		3D Point	
	seen	unseen	seen	unseen	seen	unseen	seen	unseen	seen	unseen
MPNet(NR)	0.025	0.026	0.0342	0.0301	0.03608	0.0336	0.0731	0.0747	0.0167	0.0211
MNP(Origin)	0.0058	0.0069	0.0158	0.0168	0.0106	0.0105	0.0167	0.0162	0.00989	0.0104
MPNet(HR)	0.029	0.033	0.0608	0.0552	0.0568	0.0530	0.238	0.194	0.0267	0.0303
MNP(RRT)	**0.0066**	**0.0075**	**0.0276**	**0.0318**	**0.0169**	**0.0127**	**0.0555**	**0.0339**	**0.0156**	**0.0153**
IRRT*	0.116	0.147	0.345	0.211	0.207	0.172	0.565	0.542	1.556	1.622
BIT*	0.0395	0.0565	0.377	0.134	0.201	0.186	1.082	1.129	0.980	0.946

TABLE II: Mean computation time comparisons. Methods are compared in seen and unseen environments. Informed-RRT* and BIT* stop when the path length is within 110% of MNP(RRT)'s. Since MPNet(NR) and MNP(Origin) have low success rates and their computation time are of limited significance, we focus on the comparison of the four remaining methods.



(a) Initial. (b) Step 1. (c) Step 2. (d) Step 3. (e) Goal.

Fig. 5: Paths planned by MNP for Panda Arm where green cubes are obstacles.

Method	Metrics		
	Time	Length	Suc Rate
MPNet(HR)	2.67	**0.954**	0.897
MNP(RRT)	**1.56**	0.962	0.967
BIT*(40%)	5.55	1.125	0.97
BIT*(10%)	16.55	1	**0.983**

TABLE III: 7-DOF Panda Arm results. BIT*(40%) stops when path length is within 140% of MNP(RRT)'s. The maximum planning time is 60s. BIT*(10%) is the baseline for length.

VII. Conclusion

In this paper, we proposed a learning-based motion planning method called Multimodal Neuron Planner (MNP). MPN is comprised of environment encoding networks (Enet) and multimodal planning networks (Pnet). We utilize the mixture Gaussian model to describe the distribution of the configuration, which enables Pnet to learn and predict multiple solutions. Simulation results verified MNP's ability to learn and produce multiple solutions and demonstrated that MNP outperforms the SOTA learning based method MPNet and advanced sampling based method IRRT* and BIT*.

REFERENCES

[1] Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo, and J. Gu, "The obstacle detection and obstacle avoidance algorithm based on 2-D lidar," in *IEEE International Conference on Information and Automation*, 2015, pp. 1648–1653.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[3] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[4] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2347–2354.

[5] I. Ulrich and J. Borenstein, "VFH*: Local obstacle avoidance with look-ahead verification," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 2505–2511.

[6] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[9] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 3067–3074.

[10] Y. Zhai, X. Ding, X. Jin, and L. Zhao, "Adaptive lssvm based iterative prediction method for nox concentration prediction in coal-fired power plant considering system delay," *Applied Soft Computing*, vol. 89, p. 106070, 2020.

[11] X. Ding, H. Wang, J. He, C. Chen, and X. Guan, "On the sample complexity of observers for unknown linear systems with biased dynamics estimations," *International Journal of Robust and Nonlinear Control*, 2023.

[12] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 7087–7094.

[13] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3654–3661.

[14] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 3965–3972.

[15] 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, 2020.

[16] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *International Conference on Robotics and Automation*, 2019, pp. 2118–2124.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[18] C. M. Bishop, "Mixture density networks," 1994.

[19] Y. Liu, A. Xu, and Z. Chen, "Map-based deep imitation learning for obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 8644–8649.

[20] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 204–211.

[21] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1527–1533.

[22] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 2759–2764.

[23] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *Conference on Robot Learning*. PMLR, 2023, pp. 967–977.

[24] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4693–4700.

[25] B. Xiong, F. Wang, C. Yu, F. Qiao, Y. Yang, Q. Wei, and X.-J. Liu, "Learning safety-aware policy with imitation learning for context-adaptive navigation," in *International Workshop on the Semantic Descriptor, Semantic Modeling and Mapping for Humanlike Perception and Navigation of Mobile Robots toward Large Scale Long-Term Autonomy*, 2019, pp. 38–47.

[26] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intention-Net: Integrating planning and deep learning for goal-directed autonomous navigation," in *Conference on Robot Learning*, 2017, pp. 185–194.

[27] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[28] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 9535–9541.

[29] J. J. Johnson, L. Li, A. H. Qureshi, and M. C. Yip, "Motion planning transformers: One model to plan them all," *arXiv preprint arXiv:2106.02791*, 2021.

[30] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 6582–6588.

[31] B. Chen, B. Dai, Q. Lin, G. Ye, H. Liu, and L. Song, "Learning to plan in high dimensions via neural exploration-exploitation trees," in *International Conference on Learning Representations*, 2020.

[32] A.-I. Toma, H. A. Jaafar, H.-Y. Hsueh, S. James, D. Lenton, R. Clark, and S. Saeedi, "Waypoint planning networks," *arXiv preprint arXiv:2105.00312*, 2021.

[33] A. M. Watt and Y. Yoshiyasu, "Pathnet: Learning to generate trajectories avoiding obstacles," in *IEEE International Conference on Image Processing*, 2020, pp. 3194–3198.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] E. Lyu, T. Liu, J. Wang, S. Song, and M. Q.-H. Meng, "Motion planning of manipulator by points-guided sampling network," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, 2023.

[37] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016, pp. 1050–1059.

[38] R. Strudel, R. G. Pinel, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," in *Conference on Robot Learning*, 2021, pp. 355–364.

[39] A. Brando Guillaumes, "Mixture density networks for distribution and uncertainty estimation," Master's thesis, Universitat Politècnica de Catalunya, 2017.