Proceedings of the ASME 2025
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE2025
August 17–20, 2025, Anaheim, California

DETC2025-169215

MOTION PLANNING TRANSFORMERS (MP-FORMER): TREAT MOTION PLANNING AS SEQUENCE-TO-SEQUENCE GENERATION

Boyan Li[1], Runqiu Wang[1], Yulin Chen[1], Bohan Feng[1], Qi Zhou[1], Youyi Bi[1,*]

[1]University of Michigan – Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China

ABSTRACT

Motion planning is a fundamental aspect of robotic manipulation. Compared with traditional planners, recent neural planners can achieve better efficiency for high-dimensional tasks such as manipulating multi-joint robot arms but perform unsatisfactorily in long-term tasks with limited generalizability. This paper reinterprets motion planning in a context-aware sequence-to-sequence generation manner and proposes a transformer-based framework called MP-Former. Our framework is designed to learn long-term dependency from demonstration sequences and enables efficient generation of collision-free, near-optimal paths. The experiments demonstrate that MP-Former achieves higher efficiency compared to traditional sampling-based planners, without compromising path quality and success rate, and outperforms previous neural motion planners regarding robustness and generalizability. We further illustrate the superiority of our framework in long-term planning, which prevents the planning from getting stuck into equilibrium points. Though tested primarily with point cloud inputs, MP-Former has potential to integrate multi-modal data such as visual and tactile inputs owing to the flexibility of the transformer architecture.

Keywords: Motion Planning, Neural Planner, Transformer

1. INTRODUCTION

Motion planning is a fundamental capability for AI-driven robots, which aims to find a sequence of configurations to move the robot from source to target, obeying constraints like collision avoidance. With the development of artificial intelligence techniques and their application on robotics, robots are attempting to deal with more realistic and complicated environments and multi-modal long-horizon tasks. Recent advancements in deep learning and computer vision have enabled the extraction of hidden features from the environment, allowing robots to perceive, interpret, and reason about their surroundings more efficiently.

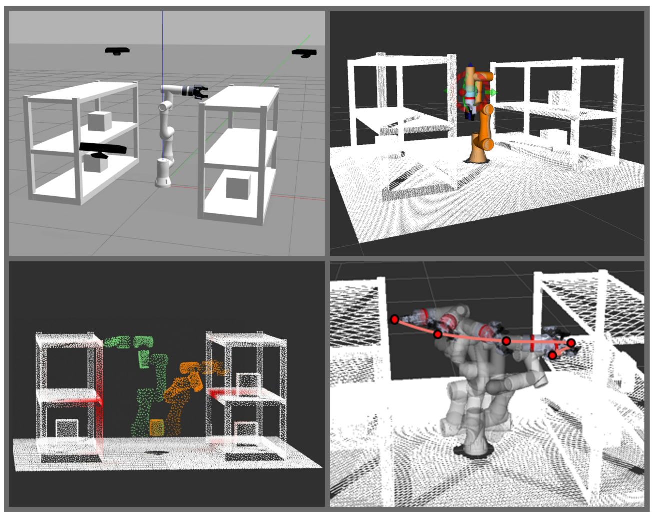

FIGURE 1: MP-FORMER IS TRAINED AND EVALUATED WITH A SYNTHETIC DATASET (UPPER LEFT). WITH THE ENVIRONMENTAL INCOMPLETE POINT CLOUD (UPPER RIGHT) AND THE PLANNING TASK (GREEN INDICATES THE START CONFIGURATION, ORANGE THE GOAL), THE ATTENTION MECHANISM FOCUSES ON THE MOST RELEVANT OBSTACLE INFORMATION IN RED (LOWER LEFT) AND GENERATES THE PATH FOR AGENTS (LOWER RIGHT).

However, traditional planning methods like sampling-based methods cannot integrate these new techniques effectively. Sampling-based methods like variants of RRT method [1–3] utilize random samples to explore the environment and construct a tree to search for the feasible path. Though theoretically guaranteeing completeness [4], such searching methods often suffer from low planning efficiency and can only achieve improved performance through the use of heuristics in biased sampling [1, 2, 5]. In addition, the computational cost of exploring and maintaining a

*Corresponding author: youyi.bi@sjtu.edu.cn search tree remains substantial in high-dimensional configuration spaces [6, 7], especially for tasks involving long-term planning to generate paths over extended distances or through complex environments. This highlights the necessity for more novel and efficient approaches.

Recently, learning-based methods like neural motion planners have been developed to deal with higher-order planning tasks with complicated scenarios and high-dimensional configuration spaces [8, 9]. Unlike the learning-enhanced sampling-based methods utilizing cues to guide biased sampling [10, 11], these algorithms directly predict a sequence of configurations with deep neural networks. Such methods can better incorporate the up-to-date environment feature extraction methods since the extracted latent features directly inform path generation, leading to highly improved planning efficiency and enhanced path quality.

Although achieving significant advances, previous learning-based methods tackle motion planning problems in a one-step-ahead way [12, 13]. Such methods perform well when learning from limited task space, but struggle with long-term planning tasks that acquire path planning and operation across the full workspace of the robotic arm, where error accumulation or premature halting can occur during extended-range path generations. Conversely, some neural motion planners face challenges due to their high demand for demonstrations during training [9], calling for effective techniques to learn sequential dependencies in path generation and achieve better generalizability across diverse scenarios.

To tackle these limitations, we propose Motion Planning Transformers (MP-Former), a novel transformer-based neural motion planner framework that treats path generation as a context-aware sequence-to-sequence (seq2seq) generation process. As shown in the overview of this work in Fig.1, MP-Former takes the environment point cloud as input and returns the planned path consisting of a dense configuration sequence. The framework is composed of 1) a perception module, which extracts features from the input point cloud and generates contextual embeddings of the environment, and 2) a transformer-based path generation module that utilizes environmental embeddings, the preceding path sequence along with the target configuration to generate the configuration sequence. These two modules are combined using a co-attention framework, which facilitates the modeling of pairwise relationships between each configuration and the corresponding local environment, allowing for more fine-grained path generation. We also employ the autoregressive training strategy during the model training to enable long-term dependency learning.

In summary, the main contributions of this paper are three folds:

- The idea of treating motion planning as context-aware sequence-to-sequence (seq2seq) generation is proposed, which theoretically avoids the risk of getting stuck into equilibrium points and enables an enhanced performance on path generation for long-term planning tasks.
- A transformer-based path generation network is developed to implement the seq2seq motion planning idea, seamlessly incorporating contextual information of the environment and prior state sequence to achieve better quality and efficiency in path planning.
- Evaluations demonstrate that our method outperforms other neural motion planners in long-term path generation and generalization under novel scenes with superior path quality and higher robustness.

2. RELATED WORK

2.1. Traditional global motion planners

Global motion planning refers to planning under complete information of the environment. Such planners have been widely used in robotic planning tasks [14–16]. There are several variants of traditional global motion planners. As the most classical searching-based method, A* algorithm simplifies the environment into discrete grid maps and is proven to be optimal and complete [17]. To further deal with continuous space, sampling-based methods such as RRT [18] and its variants, explore the free configuration space with samples and expand a spanning tree to approach the target. Some planners incorporate heuristics or learning techniques to bias sampling to more related regions hence enhancing the sampling efficiency [19–21]. However, their limited compatibility with modern environment perception algorithms often leads to inefficiencies in the expansion process.

2.2. Neural motion planners

Neural motion planners leverage neural networks to learn from traditional planners' demonstrations and generate plans efficiently. Instead of sampling and tree expansion, they directly produce configuration sequences from environmental features and can be supplemented by traditional planners to guarantee the completeness of solutions. Motion Planning Networks (MPNet) [13] first utilizes an encoder-decoder structure, which encodes the environmental information to latent space and uses the decoder iteratively to produce the next step's robot configuration. Further researches try to optimize the structure by either replacing the encoder with PointNet++-based structure [9] to tackle incomplete environmental input or concatenate additional task encoders [8, 22] to enable planning with multi-modal inputs. While greatly improving the planning efficiency, these methods generate the path in a one-step-ahead way, namely, the subsequent planning configuration prediction highly depends on the preceding one. Such dependency can lead to short-sighted performance as inaccuracies accumulate over iterations, increasing the risk of falling into equilibrium points (i.e., keeping generating similar configurations). Consequently, this limits the planner's ability to explore more effective solutions in long-term planning tasks. Our approach addresses this issue by conceptualizing the planning process as a context-aware seq2seq generation task, prioritizing the generation of the entire sequence rather than relying solely on individual predictions.

2.3. Transformer-based multi-modal planners

Transformer has become a powerful computational structure in many relevant topics such as autonomous driving or smart navigation [23, 24] due to its flexibility in multi-modal representation fusion and the parallel processing capability. Although existing studies [25, 26] fully utilize the transformer's ability to fuse multimodal inputs, providing only fixed horizons of waypoints prediction, our proposed method further applies the transformer's ability to generate an indefinite-length path sequentially during the decoding process. Aligning with the idea of seq2seq generation, we consider the planning process as an indefinite-step configuration sequence generation, which is similar to the text generation in natural language processing tasks with a high demand for learning long sequence dependency.

3. TRANSFORMER FOR MOTION PLANNING

3.1. Problem formulation and algorithm overview

MP-former treats motion planning as a seq2seq generation problem and solves it with a co-attention framework. Assume a motion planning task with start configuration c_{start} and target c_{goal}, and already with $n-1$ steps of configuration predictions $c_1...c_{n-1}$ ($n \geq 1$). The path generation module takes the configuration sequence $\boldsymbol{C} = \{c_{start}, c_1, ..c_{n-1}, c_{goal}\}$ and the environmental embedding sequence $\boldsymbol{E} = \{E_0, E_1, ..., E_m\}$ extracted by the perception module, predicts a configuration displacement δc_n and gets the next configuration with $c_n = c_{n-1} + \delta c_n$. By repeating this process, our algorithm generates a complete path for execution. Different from previous one-step-ahead methods, our framework takes all preceding predictions as contextual information to guide the posterior sequence predictions. This approach enables the framework to learn the long-term dependencies within diverse demonstrations, enhancing its ability to generalize across varied tasks. We further discuss in Sec. 5.3 that by maintaining a coherent configuration sequence, our framework prevents the sequence generation from getting stuck into equilibrium points in long-term tasks.

The overall structure of the proposed MP-Former is shown in Fig. 2. The environment and configuration embedding module maps the contextual information from the environment and preceding configurations to separate embedding sequences. The path generation module then uses both sequences to generate new predictions in a sequence expansion manner. Implementation details are discussed below.

3.2. Environment embedding module

In this work, the point cloud captured by the depth camera is used to provide the path generation module with collision constraints, which only contain partial environmental information due to occlusions. We utilize the Point-BERT [27] architecture as our environment embedding module. With pre-training for representation learning and evaluations on various downstream tasks (e.g. classification and segmentation), this module enables comprehensive reasoning on local and global information within the point cloud.

The detailed structure of the environmental embedding module is illustrated on the upper left of Fig. 2. Taking the environment point cloud $\boldsymbol{Pt} = \{pt_i\}_{i=1}^{n}$, the module follows the PointNet++ structure [28] to sample m patch center points $\boldsymbol{Pc} = \{pc_i\}_{i=1}^{m}$ via farthest point sampling (FPS) and generates point patches by grouping the K nearest neighbors (KNN) of each center point in $\boldsymbol{Pc}$. A normalization operation is further utilized to shift each patch to origin-centered patch tokens $\boldsymbol{T} = \{t_i\}_{i=1}^{m}$, retaining only local geometry patterns. With local patch tokens, PointNet-like embedding layers [29] are utilized to convert the points into local geometric embeddings. We further add the positional embedding encoded from the corresponding patch centers' position $\mathbf{C}$ from the raw point cloud to indicate the corresponding absolute location and get the input embeddings $\mathbf{x} = \{x_i\}_{i=1}^{m}$. A Transformer-based backbone is used to process the input embeddings $\mathbf{x}$ with a classification token attached at the start, and output the environmental embeddings, $\mathbf{E} = cat(E_{cls}, \{E_i\}_{i=1}^{m})$, representing the global feature and the fused local features of the input point cloud. We follow most parameter settings with the original paper [27], for further details, please refer to the original work.

3.3. Path generation module

The backbone of the path generation Network is adapted from vanilla autoregressive Transformer architecture [30, 31] with self-attention and cross-attention blocks. The sequence of robot configurations is mapped to configuration embeddings in latent space with multi-layer perceptron, and the sinusoidal positional embeddings are added to inject permutation information. We further insert learned prediction token [**pred**] to which we want to generate new configuration predictions and form the final embedding sequence. This embedding sequence serves as the input to the transformer decoder blocks. Within the blocks, a self-attention layer first calculates the attention map between [**pred**] token and previous configurations, yielding a context vector that encapsulates the information from the entire sequence relevant to the [**pred**] token and serves as a configuration hypothesis in latent space. A normal lower triangular attention map for autoregressive training with extra attention on the target embedding is used as the attention mask for each self-attention block. The hypothesis is further fed into a cross-attention layer, where it interacts with the environmental embeddings from the previous module. The cross-attention mechanism helps the hypothesis to attend to environmental geometric patterns and allows it to make collision-avoidance adjustments based on relevant local geometry or high-level policy adaptation from global environmental patterns. The output of the cross-attention layer is processed by the following self and cross-attention blocks for refinement and finally outputs a final prediction in latent space. A decoder maps the latent prediction back to the robot's configuration space to predict the δc_n, the change (or increment) in the configuration relative to the previous state. This trick to learning increments allows for faster convergence and enhances robustness in performance.

Algorithm 1 represents the pipeline of MP-Former for the inference phase: starting with a sequence including start and goal configuration, we iteratively insert the [**pred**] token before the goal configuration and predict the configuration with the path generation module to form the motion sequence. This prediction loop terminates when the predicted configuration is within the target region or the maximum iterations are reached, returning a coarse path. We also follow [9, 13] to apply Oracle replanner as a post-processor to fix breakpoints of the coarse path using the OctoMap generated from the incomplete point cloud, and to reduce duplicated waypoints.

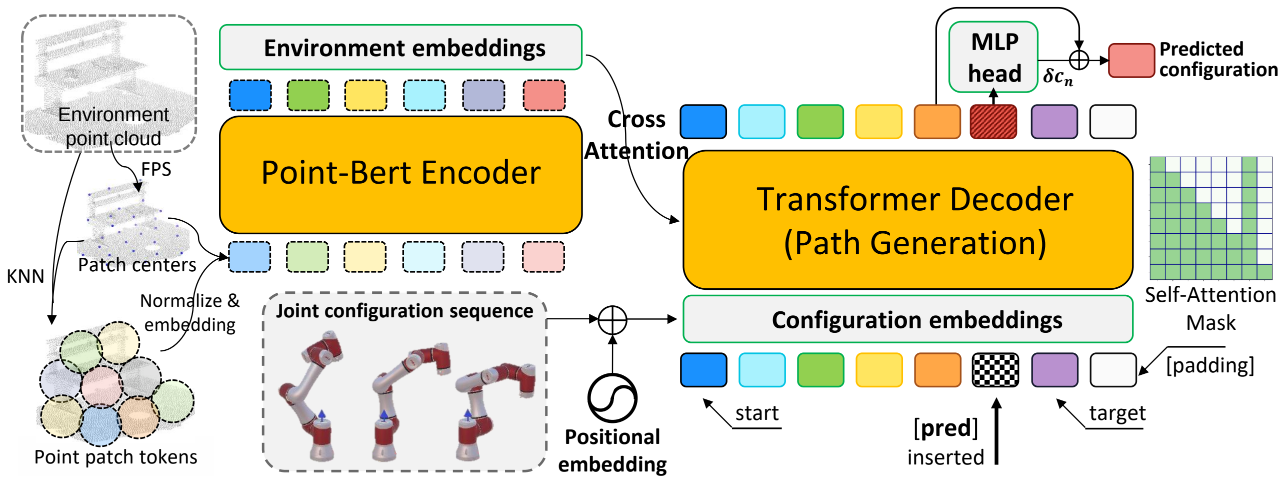

FIGURE 2: THE PIPELINE OF MP-FORMER. THE INCOMPLETE ENVIRONMENTAL POINT CLOUD IS FIRST PARTITIONED INTO PATCHES AND WE UTILIZE THE POINT-BERT BACKBONE [27] TO EXTRACT ENVIRONMENTAL EMBEDDINGS. THEN AN ADAPTED TRANSFORMER DECODER IS USED TO GENERATE THE CONFIGURATION INCREMENT ITERATIVELY IN AN AUTOREGRESSIVE MANNER. THIS PROCESS IS GUIDED BY SELF-ATTENTION WITH PRIOR CONFIGURATIONS AND CROSS-ATTENTION WITH ENVIRONMENTAL EMBEDDINGS.

Algorithm 1 MP-FORMER ALGORITHM IN INFERENCE PHASE

```
Input: Environmental point cloud Pt;
  Start configuration $c_{start}$ & Goal configuration $c_{goal}$;
  Maximum Iteration ($M$) for neural motion planner.
Output: Planned path $C_{pred}$.
  $Pc \leftarrow$ FPS($Pt$)
  $T \leftarrow$ Normalize(KNN($Pc, Pt$)) #generate patch tokens
  $E \leftarrow$ EnvEmbedding($T, Pc$)
  time = 0
  $c_0 = c_{start}$
  $C_{pred} = \{c_{start}, c_{goal}\}$
  for i = 1 to $M$ do
    $C_{pred} \leftarrow$ Insert($C_{pred}$, [pred], index = −2)
    #insert [pred] token before $c_{goal}$
    $c_i \leftarrow c_{time}$ + PathGenerator($E, C_{pred}$)
    if Termination($C_{pred}$) then
      Break
    end if
  end for
  if Not CollisionCheck($C_{pred}, Pc$) then
    $C_{pred} \leftarrow$ OraclePlanner($C_{pred}$) ∪ $C_{pred}$
  end if
  return
```

4. TRAINING STRATEGY

The training dataset of our method is generated in the Moveit environment [32] with RRT* [4] as the oracle planner on a 7-DOF robotic manipulator. It includes two different types of scenes: generic obstacle scene with randomly generated objects and common industrial production scenes, and it consists of a total of 120 different environments with over 180,000 valid paths. We use 50 generic environments as the training dataset and the rest generic and production scenes as the validation and testing dataset shown in Fig.3. To strengthen the ability to deal with long-term tasks, we randomly sample the start and goal configuration on the whole workspace and generate demo paths with dense waypoints.

4.1. Data preparation and autoregressive training

Since the paths generated by the Oracle planner are diverse in length, we utilize the tricks to attach padding to the path sequences, ensuring uniform length across all data samples. This padding allows for efficient batch processing and avoids issues related to varying sequence lengths. In addition, padding is ignored during loss calculation, while using the lower triangle attention mask has naturally disabled the model to attend subsequent tokens, ensuring that the padding does not interfere with the reasoning process. During the training process, the prediction token **[pred]** is substituted for each configuration c_i sequentially, one at a time to facilitate teaching-force training. The model tries to reveal the hidden configuration by attending the environmental embedding sequence and the preceding configuration sequence. This method enables the model to learn stable patterns from long configuration sequences and exploit the attention mechanism to focus on relevant environmental features, thereby enhancing its ability to generate accurate long configuration sequences during inference and generalization by finding similar local environment patterns in novel scenes. Additionally, we introduce noise to the previous configurations during training to prevent overfitting and help learning the ability to recover from mistakes, enhancing prediction stability.

4.2. Network specification and setting of hyperparameters

Our encoder-decoder transformer-based motion planning network comprises $L = 12$ encoder layers and $M = 4$ decoder layers. Each layer features $H = 6$ attention heads with a hidden

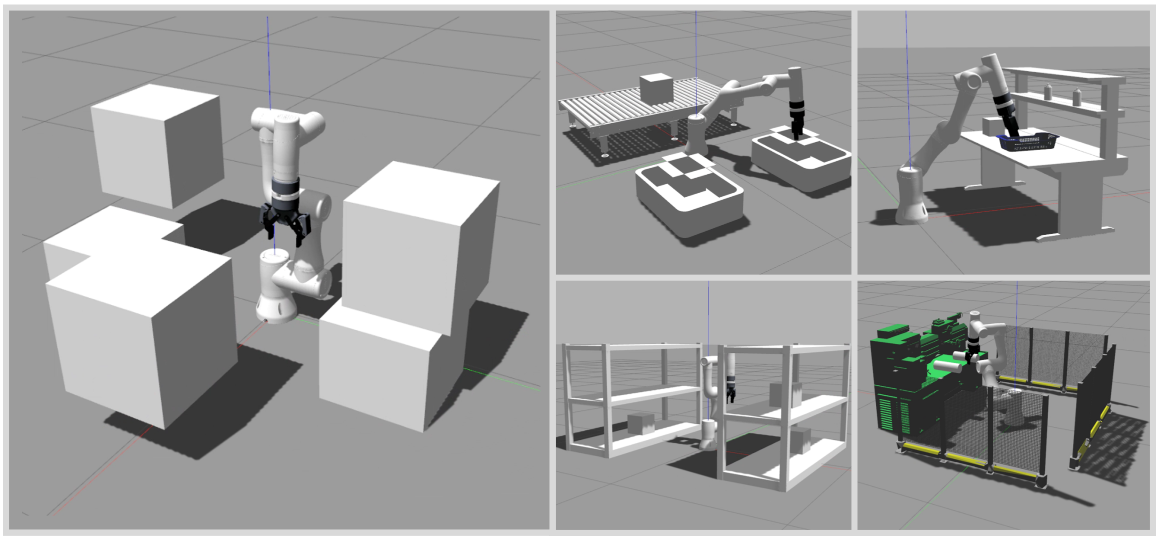

FIGURE 3: EXAMPLES OF ENVIRONMENTS USED FOR DATA COLLECTION. THE LEFT IMAGE ILLUSTRATES A GENERIC OBSTACLE SCENE, WHILE THE FOUR IMAGES ON THE RIGHT DEPICT SEVERAL COMMONLY ENCOUNTERED INDUSTRIAL PRODUCTION SETTINGS USED.

dimension of $d_{model} = 384$. The whole model is trained for 50 epoches using the RAdam optimizer, with a learning rate lr =1e-4 and weight decay wd =1e-4. A warmup phase of 2000 steps is employed, with the warmup factor of 1e-3, to ensure stable convergence during training.

4.3. Loss function

In the training phase, three different losses are chosen, forming a compound loss to guide the network's update: MSELoss and point cloud match loss are used to guide the network to imitate the behavior of the oracle planner. We sample a total of 1152 points on the robot arm's surface and implement point cloud match loss by computing the $L2$ and $L1$ distance between the point cloud corresponding to the predicted configuration and that for the ground truth configuration. We further apply auxiliary loss with the MSELoss between the predicted configuration and the ground truth configuration to address ambiguities specifically related to differences of 2π in configuration angles, where the point clouds are otherwise identical.

$$\mathcal{L}_{l2} = \sum_{t=1}^{L} \delta_t (c_t - \hat{c}_t)^2 / \sum_{t=1}^{L} \delta_t \tag{1}$$

$$\mathcal{L}_{pc} = \sum_{t=1}^{L} \delta_t \sum_{i=1}^{n} \left\| \phi^i(\hat{c}_t) - \phi^i(c_t) \right\|_2 + \left\| \phi^i(\hat{c}_t) - \phi^i(c_t) \right\|_1 \tag{2}$$

where δ is set to 1 for non-padding values and 0 for padding values to omit the influence of padded elements in the calculations. Function ϕ maps the robot configuration to the corresponding point cloud. L represents the sequence length.

To further strengthen the module's reasoning ability to the environmental constraints and generate collision-free paths, collision loss is added to ensure the feasibility of the predictions. Inspired by [9, 33], we utilize the generated robot arm's point cloud $\phi(\hat{c}_t)$ and formulate a penalty for those having the negative return of the signed-distance function (SDF) generated for each scene:

$$\mathcal{L}_{\text{collision}} = \sum_t \delta_t \sum_i \left\| h\left(\phi^i(\hat{c}_t)\right) \right\|_2 \tag{3}$$

$$h\left(\hat{x}_t^i\right) = \begin{cases} |sdf\left(\phi^i(\hat{c}_t)\right)|, & \text{if } sdf\left(\phi^i(\hat{c}_t)\right) \le d_s \\ 0, & \text{if } sdf\left(\phi^i(\hat{c}_t)\right) > d_s \end{cases}$$

where safety distance d_s is set as 0.02 to penalize the predicted configuration that collides or is too close to the environment surface. This setting prevents predictions from catastrophic collision failures.

The overall loss function can be formulated as the weighted combination of the three losses. We empirically set the weights, α, β, γ to be 1, 20, and 1, respectively.

$$\mathcal{L} = \alpha\mathcal{L}_{l2} + \beta\mathcal{L}_{pc} + \gamma\mathcal{L}_{collicion} \tag{4}$$

5. EVALUATION RESULTS

In this section, we evaluate our proposed method against traditional sampling-based planners and other neural motion planners. The tests are conducted on a computer with an Intel i9-10980XE CPU and an Nvidia RTX 3090 GPU. The evaluation metrics are as follows:

- **Success Rate**. The ratio of successful planning attempts that return a feasible path connecting the start and goal configurations.
- **Solution Time**. The average time taken by the algorithm to generate a valid path. For traditional sampling-based methods, which are asymptotically optimal with infinite planning time, we evaluate their performance at designated time intervals.
- **Path Cost Ratio**. The criterion for measuring the robot arm's feasible path cost is the cumulative joint differences

TABLE 1: COMPARISON WITH SAMPLING-BASED PLANNERS, RRT* AND AIT* UNDER LIMIT TIME (2S) AND ABUNDANT TIME (10S) CASE.

	Soln. Time [s]	**Succ. Rate** [%]	**Path Cost Ratio** [%]
MP-Former	**0.65 ± 0.26**	93.90	**109.8** ± **13.0**
RRT* [4] †			
Limit time	2.05 ± 0.03	79.54	140.2 ± 41.3
Abundant time	10.10 ± 0.04	89.31	124.2±23.6
AIT* [35] †			
Limit time	2.04 ± 0.03	83.17	126.4 ± 33.4
Abundant time	10.09 ± 0.03	**95.52**	110.3 ± 14.4

† Sampling-based planners use full environment info, while MP-Former only relies on partial point cloud data.

[34], expressed as:

$$\text{Path Cost} = \sum_{i=1}^{n}\sum_{J=1}^{m} \left\| c_{i,J} - c_{i-1,J} \right\| \tag{5}$$

where c_0 represents c_{start} and c_n for c_{goal}. Since path cost varies across different planning tasks, we use the path cost ratio to indicate the relative optimality of each method. The quasi-optimal path from the RRT* planner, generated with a planning time of over 30 seconds, is used as the reference. Thus, the lower Path Cost Ratio, the better.

5.1. Comparison with traditional planners

We first compare MP-former with the classic sampling-based method of RRT* planner and state-of-the-art AIT* planner, which are implemented with OMPL [36] CPP bindings. Since both sampling-based methods are global planners with asymptotic optimality, we apply these methods to complete environmental states, using two solution time limits: 2 seconds for the limited-time case and 10 seconds for the abundant-time case, to ensure completeness and fairness in the comparison. For our method, the maximum replanning time is set to 2 seconds. The results are presented in Table 1.

The evaluation results highlight the strengths of MP-Former in comparison to RRT* and AIT* planners. Though planned under incomplete environmental information, MP-Former demonstrates notably high efficiency in gaining a solution while keeping a fairly competitive success rate. This shows its capability to handle scenarios with partial environmental data efficiently and reliably. Additionally, MP-Former outperforms other trials in path cost ratio, indicating that our proposed structure can effectively utilize environmental features and prior configuration sequences to generate high-quality paths. However, given abundant time and complete state inputs, AIT* achieves higher success rates than MP-Former. This discrepancy may arise because our framework operates under partial observations, making it challenging to recover certain missing environmental details with the feature extractors. Consequently, this can lead to errors compared to advanced traditional planners that employ full environmental data.

5.2. Comparison with neural motion planners

In addition to comparing with traditional planners, we also demonstrate the comparison between our proposed method with other neural motion planners like MPNet [13] and MπNet [9]. Each planner is trained on our generic dataset, and evaluated using other generic data with similar environment settings. Further evaluation is conducted under novel production scenes to validate the generalizability of each algorithm. To ensure a fair comparison, we choose the MPNet with hybrid replanning settings and also attach an Oracle replanner to the MπNet pipeline. The maximum number of iterations is set to 100, with a replanning time limit of 0.5 seconds. The results of this experiment are summarized in Table 2.

All three neural motion planners demonstrate efficient motion planning capabilities, solving problems within 1 second. Under the seen scenario, MP-Former exhibits notable performance, achieving a high success rate of approximately 86% with near-optimal paths of cost 109%. This highlights MP-Former's proficiency in navigating complex configurations within the familiar generic environment, underpinned by robust learning and adaptation from the training data. In contrast, MPNet achieves a high success rate, but the generated paths are often sub-optimal. MπNets can generate high-quality paths in certain cases, but they often get stuck at specific steps when trained with limited datasets, leading to a high failure rate.

Moving to the evaluation under novel scene environments, MP-Former maintains its effectiveness with an 83% success rate and competitive path cost. This underscores its ability to generalize learned behaviors beyond the training scenarios and the long-term path patterns, accommodating variations and uncertainties typical of real-world applications. In comparison, MPNet achieves much worse performance in novel scene, indicating a potential over-fitting to seen dataset with low generalizability. MπNets exhibit same stuck problem in novel scene, resulting in low success rate.

5.3. Long term performance study

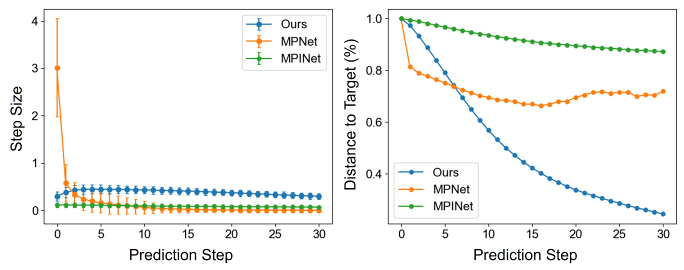

FIGURE 4: THE NORMALIZED DISTANCE TO THE TARGET AND THE MEAN STEP SIZE FOR THE FIRST 30 PREDICTIONS. OUR PROPOSED METHOD EARNS A FAST CONVERGENCE TO THE TARGET WHILE KEEPING A STABLE PACE.

In this section, the long-term performance of neural motion planners is evaluated with the mean step-wise convergence performance and step size. The left part of Fig.4 demonstrates the step-wise average distance to target ratio, which is the ratio between the current prediction's distance to target and the original distance between the start and goal. From the figure we can

TABLE 2: PERFORMANCE COMPARISON WITH OTHER NEURAL MOTION PLANNERS UNDER SEEN AND NOVEL SCENES.

		Generic Scene (Seen)		Production Scene (Novel)	
	Soln. Time (s)	**Succ. Rate (%)**	**PathCostRatio (%)**	**Succ. Rate (%)**	**PathCostRatio (%)**
MP-Former	0.61 ± 0.21	85.8	**109.0 ± 12.6**	**83.5**	**113.1 ± 14.2**
MPNet [13]	**0.39 ± 0.10**	**85.9**	154.5 ± 55.1	54.9	139.2 ± 43.9
MπNets [9]	0.93 ± 0.13	75.6	113.0 ± 19.0	72.7	115.6 ± 18.9

find that our proposed method enables a steady convergence with steps, whereas others tend to get stuck in local regions due to encountering equilibrium points. The right part of Fig.4 shows the average step size for each prediction. As illustrated, our proposed method remains steady pace with a reasonable step size, while others often run into step vanishes during the long-term prediction (MPNet) or keep a slow pace (MπNet) along the iterations. These results exhibit the ability of our proposed method to learn the long-term dependencies and generate robust and high-quality paths in long-term planning tasks.

To formally interpret such long-term behavior difference between the previous one-step-ahead methods and our proposed algorithm, we abstract all diverse neural network structures as a general function $\mathbf{f}_\theta$ with learned parameters θ. In previous algorithms, the input is a combination of current configuration c_i, planning goal configuration c_{goal} and obstacles information E, and the predicting target is either the next configuration c_{i+1} or the increment δc_i. During the inference phase, we take the previous prediction as the next step's input, namely, we replace the input c_i with $\hat{c}_i$. So we represent the algorithms' step-wise prediction with the following equations during the inference phase:

$$\text{MPNets:}\quad \hat{c}_{n+1} = f_\theta(\hat{c}_n, E, c_{goal}) \tag{6}$$
$$\text{M}\pi\text{Nets:}\quad \hat{c}_{n+1} = f_\theta(\hat{c}_n, E, c_{goal}) + \hat{c}_n \tag{7}$$

where E and c_{goal} are unchanged along iterations. It can be seen that updates manner of Eq. 6, 7 lead to the appearance of equilibrium points, which is, for all c_i that makes the next prediction c_{i+1} equals the input c_i. This implies that when a prediction approaches such a point, the algorithm could easily halt prematurely, leading to potential issues in further configuration generation (stability discussion of these points is omitted, as we cannot ensure all are unstable). This problem aggravates especially in long-term tasks, as successive prediction rounds increase the likelihood of the algorithm becoming stuck at these fixed points. Whereas, instead of using solely previous predictions $\hat{c}_i$, we take the whole sequence $\{c_i\}$ as input and insert the prediction into the sequence to form the next input.

$$\text{Ours:}\quad \hat{c}_{n+1} = f_\theta(\{\hat{c}_i\}_{i\le n}, E, c_{goal}) + \hat{c}_n \tag{8}$$

This design prevents the algorithm from easily falling into these equilibrium points, making the structure more robust for long-term planning.

5.4. Real world Evaluation

We further evaluate MP-Former in the Lab scene with a 7-DOF Jaka-zu3 robot. The environment point cloud is captured with one Realsense D435 depth camera. A body filter processes the point cloud to filter out the points related to the robot before sending it to the robot, and MP-Former generates a feasible path with a maximum of 100 steps. Details can be found in the video MP-Former. As shown, despite being trained solely on simulated data, MP-Former successfully achieves sim-to-real performance under noisy and occluded real-scene point clouds.

6. CONCLUSION

MP-Former is a transformer-based neural motion planner, which learns to generate the configuration sequence to navigate robots from start to goal configuration while avoiding collisions with the environment. Experiments indicate that this planning framework has shown high stability and generalizability, especially in long-term planning. Utilizing a transformer architecture, our proposed framework also has a high potential to integrate multi-modal perceptions or textual instructions to guide the motion planning process. Future research will focus on integrating multi-modal constraints into MP-Former's framework and applying it to complex real-world applications.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support from the National Key R&D Program of China (2022YFB4702400) and National Natural Science Foundation of China (52005328).

REFERENCES

[1] Gammell, Jonathan D., Srinivasa, Siddhartha S. and Barfoot, Timothy D. "Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs." *2015 IEEE International Conference on Robotics and Automation (ICRA)*: pp. 3067–3074. 2015. IEEE.

[2] Gammell, Jonathan D. and Srinivasa, Siddhartha S. and Barfoot, Timothy D. "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*: pp. 2997–3004. 2014. DOI 10.1109/IROS.2014.6942976.

[3] Feng, Bohan, Jiang, Xinting, Li, Boyan, Zhou, Qi and Bi, Youyi. "An Adaptive Multi-RRT Approach for Robot Motion Planning." *Expert Systems with Applications* Vol. 252 (2024): p. 124281. DOI 10.1016/j.eswa.2024.124281.

[4] Karaman, Sertac and Frazzoli, Emilio. "Sampling-based algorithms for optimal motion planning." *The international*

journal of robotics research Vol. 30 No. 7 (2011): pp. 846–894.

[5] Ichter, Brian and Pavone, Marco. "Robot Motion Planning in Learned Latent Spaces." *IEEE Robotics and Automation Letters* Vol. 4 No. 3 (2019): pp. 2407–2414. DOI 10.1109/LRA.2019.2901898.

[6] Elbanhawi, Mohamed and Simic, Milan. "Sampling-based robot motion planning: A review." *IEEE Access* Vol. 2 (2014): pp. 56–77.

[7] Hu, Chuanhui and Jin, Yan. "Path planning for autonomous systems design: A focus genetic algorithm for complex environments." *Journal of Autonomous Vehicles and Systems* Vol. 2 No. 4 (2022).

[8] Qureshi, Ahmed H., Dong, Jiangeng, Choe, Austin and Yip, Michael C. "Neural Manipulation Planning on Constraint Manifolds." *IEEE Robotics and Automation Letters* Vol. 5 No. 4 (2020): pp. 6089–6096.

[9] Fishman, Adam, Murali, Adithyavairavan, Eppner, Clemens, Peele, Bryan, Boots, Byron and Fox, Dieter. "Motion policy networks." *Conference on Robot Learning*: pp. 967–977. 2023. PMLR.

[10] Huang, Zhe, Chen, Hongyu, Pohovey, John and Driggs-Campbell, Katherine. "Neural Informed RRT*: Learning-based Path Planning with Point Cloud State Representations under Admissible Ellipsoidal Constraints." *2024 IEEE International Conference on Robotics and Automation (ICRA)* No. arXiv:2309.14595 (2024). DOI 10.48550/arXiv.2309.14595. Accessed 2024-06-15, URL 2309.14595.

[11] Tran, Tuan, Dutta, Sourav, Rekabdar, Banafsheh and Ekenna, Chinwe. "Transformer Based Approach for Sample Generation in Motion Planning." *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*: pp. 1–7. 2023. IEEE, Auckland, New Zealand. DOI 10.1109/CASE56687.2023.10260470.

[12] Bency, Mayur J., Qureshi, Ahmed H. and Yip, Michael C. "Neural Path Planning: Fixed Time, near-Optimal Path Generation via Oracle Imitation." *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*: pp. 3965–3972. 2019. IEEE.

[13] Qureshi, Ahmed Hussain, Miao, Yinglong, Simeonov, Anthony and Yip, Michael C. "Motion Planning Networks: Bridging the Gap between Learning-Based and Classical Motion Planners." Vol. 37 No. 1 (2020): pp. 48–66.

[14] Ji, Junjie and Zhao, Jing-Shan. "Multi-Point Path Planning Algorithm for a Mobile Robot With Composite Moving Costs." *Journal of Autonomous Vehicles and Systems* Vol. 2 No. 3 (2023): p. 031002. DOI 10.1115/1.4056759. URL https://asmedigitalcollection.asme.org/autonomousvehicles/article-pdf/2/3/031002/6982898/javs_2_3_031002.pdf, URL https://doi.org/10.1115/1.4056759.

[15] Wang, Weitian, Chen, Yi and Jia, Yunyi. "Evaluation and Optimization of Dual-Arm Robot Path Planning for Human–Robot Collaborative Tasks in Smart Manufacturing Contexts." *ASME Letters in Dynamic Systems and Control* Vol. 1 No. 1 (2020): p. 011012. DOI 10.1115/1.4046577. URL https://asmedigitalcollection.asme.org/lettersdynsys/article-pdf/1/1/011012/6519869/aldsc_1_1_011012.pdf, URL https://doi.org/10.1115/1.4046577.

[16] Rajendran, Pradeep, Thakar, Shantanu, Bhatt, Prahar M, Kabir, Ariyan M and Gupta, Satyandra K. "Strategies for speeding up manipulator path planning to find high quality paths in cluttered environments." *Journal of Computing and Information Science in Engineering* Vol. 21 No. 1 (2021): p. 011009.

[17] Hart, Peter E., Nilsson, Nils J. and Raphael, Bertram. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *IEEE Transactions on Systems Science and Cybernetics* Vol. 4 No. 2 (1968): pp. 100–107. DOI 10.1109/TSSC.1968.300136.

[18] LaValle, Steven. "Rapidly-exploring random trees: A new tool for path planning." *Research Report 9811* (1998).

[19] Li, Xiang, Cao, Qixin, Sun, Mingjing and Yang, Ganggang. "Fast motion planning via free c-space estimation based on deep neural network." *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*: pp. 3542–3548. 2019. IEEE.

[20] Cheng, Richard, Shankar, Krishna and Burdick, Joel W. "Learning an optimal sampling distribution for efficient motion planning." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*: pp. 7485–7492. 2020. IEEE.

[21] Acar, Cihan and Tee, Keng Peng. "Approximating constraint manifolds using generative models for sampling-based constrained motion planning." *2021 IEEE International Conference on Robotics and Automation (ICRA)*: pp. 8451–8457. 2021. IEEE.

[22] Qureshi, Ahmed Hussain, Dong, Jiangeng, Baig, Asfiya and Yip, Michael C. "Constrained Motion Planning Networks X." *IEEE Transactions on Robotics* Vol. 38 No. 2 (2022): pp. 868–886. DOI 10.1109/TRO.2021.3096070.

[23] Prakash, Aditya, Chitta, Kashyap and Geiger, Andreas. "Multi-Modal Fusion Transformer for End-to-End Autonomous Driving." *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*: pp. 7073–7083. 2021. IEEE. DOI 10.1109/CVPR46437.2021.00700.

[24] Chen, Shizhe, Guhur, Pierre-Louis, Schmid, Cordelia and Laptev, Ivan. "History Aware Multimodal Transformer for Vision-and-Language Navigation." *Advances in neural information processing systems* Vol. 34 (2021): pp. 5834–5847.

[25] Nayakanti, Nigamaa, Al-Rfou, Rami, Zhou, Aurick, Goel, Kratarth, Refaat, Khaled S. and Sapp, Benjamin. "Wayformer: Motion Forecasting via Simple & Efficient Attention Networks." *2023 IEEE International Conference on Robotics and Automation (ICRA)*: pp. 2980–2987. IEEE.

[26] Huang, Zhiyu, Mo, Xiaoyu and Lv, Chen. "Multi-Modal Motion Prediction with Transformer-based Neural Network for Autonomous Driving." *2022 International Conference on Robotics and Automation (ICRA)*: pp. 2605–2611. 2022. DOI 10.1109/ICRA46639.2022.9812060.

[27] Yu, Xumin, Tang, Lulu, Rao, Yongming, Huang, Tiejun, Zhou, Jie and Lu, Jiwen. "Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling." *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*: pp. 19291–19300. 2022. IEEE. DOI 10.1109/CVPR52688.2022.01871.

[28] Qi, Charles Ruizhongtai, Yi, Li, Su, Hao and Guibas, Leonidas J. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." *Advances in neural information processing systems* Vol. 30 (2017).

[29] Qi, Charles R., Su, Hao, Mo, Kaichun and Guibas, Leonidas J. "Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*: pp. 652–660. 2017.

[30] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Łukasz and Polosukhin, Illia. "Attention is all you need." *Proceedings of the 31st International Conference on Neural Information Processing Systems*: p. 6000–6010. 2017. Curran Associates Inc., Red Hook, NY, USA.

[31] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton and Toutanova, Kristina. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Burstein, Jill, Doran, Christy and Solorio, Thamar (eds.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019. Association for Computational Linguistics, Minneapolis, Minnesota.

[32] Coleman, David, Sucan, Ioan, Chitta, Sachin and Correll, Nikolaus. "Reducing the barrier to entry of complex robotic software: a moveit! case study." *Journal of Software Engineering for Robotics* (2014)DOI 10.6092/JOSER_2014_05_01_p3.

[33] Fishman, Adam, Paxton, Chris, Yang, Wei, Fox, Dieter, Boots, Byron and Ratliff, Nathan. "Collaborative Interaction Models for Optimized Human-Robot Teamwork." *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*: pp. 11221–11228. 2020. IEEE.

[34] Dobiš, Michal, Dekan, Martin, Beňo, Peter, Duchoň, František and Babinec, Andrej. "Evaluation criteria for trajectories of robotic arms." *Robotics* Vol. 11 No. 1 (2022): p. 29.

[35] Strub, Marlin P and Gammell, Jonathan D. "Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics." *2020 IEEE International Conference on Robotics and Automation (ICRA)*: pp. 3191–3198. 2020. IEEE.

[36] Şucan, Ioan A., Moll, Mark and Kavraki, Lydia E. "The Open Motion Planning Library." *IEEE Robotics & Automation Magazine* Vol. 19 No. 4 (2012): pp. 72–82. DOI 10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.