Motion Planning Transformers: One Model to Plan Them All

Jacob J. Johnson, Linjun Li, Ahmed H. Qureshi, and Michael C. Yip
University of California San Diego
La Jolla, CA 92093
{jjj025,lili,a1qureshi,yip}@ucsd.edu

Abstract

Transformers have become the powerhouse of natural language processing and recently found use in computer vision tasks. Their effective use of attention can be used in other contexts as well, and in this paper, we propose a transformer-based approach for efficiently solving the complex motion planning problems. Traditional neural network-based motion planning uses convolutional networks to encode the planning space, but these methods are limited to fixed map sizes, which is often not realistic in the real-world. Our approach first identifies regions on the map using transformers to provide attention to map areas likely to include the best path, and then applies local planners to generate the final collision-free path. We validate our method on a variety of randomly generated environments with different map sizes, demonstrating reduction in planning complexity and achieving comparable accuracy to traditional planners.

1 Introduction

Motion planning is an integral component of any navigation stack. It is an age-old problem with more than a few decades of innovative solutions. By far the most popular recent methods involve constructing trees or graphs in the planning space by randomly sampling points [LaValle and James J. Kuffner, 2001, Kavraki et al., 1996] or discretizing the space into grids [Hart et al., 1968]. Although these methods come with theoretical guarantees, they do not scale well to planning problems in larger planning spaces where there may be significant obstacles. The problems come up frequently in the real world, where map sizes may vary significantly, and obstacles and impedances may vary across tasks and domains. Hence, in our ever-expanding pursuit of automation, we need planning algorithms that can scale up while effectively keeping the planning problem’s complexity and computation time in check.

In the robotics community, learning-based methods to solve planning problems have been proposed to offer speed improvements to planning and typically revolve around using convolution neural networks (CNN) and multi-layer perceptrons (MLP) to predict where samples should be generated to construct trees [Qureshi et al., 2019, Johnson et al., 2020]. Other works employ different forms of latent representation such as grid cells [Banino et al., 2018] and Sparse Graphical Memory (SGM)
Emmons et al. [2020] to generate a plan. Chaplot et al. [2021] proposed one of the few works that use transformers to learn a value function similar to Value Iteration Networks [Tamar et al., 2016], but the extension of this method to larger maps still need to be explored. Overall, the utilization of Transformers for motion planning remains unexplored at large.

In this work, we propose a hierarchical approach using transformers. Our work is motivated by the recent progress in expanding the language transformer models for vision tasks and taking advantage of the power of self-attention [Dosovitskiy et al., 2021, Liu et al., 2021]. Our model plans in a two-stage approach (Fig. 1), first a transformer-based region proposal network recommends a region where a potential path can exist. We refer to this module as the Motion Planning Transformer (MPT). Next, a sampling-based planner tries to plan a path given the proposed region. Unlike previous neural motion planning methods, the objective of the MPT is to reduce the search space for the underlying motion planners. Hence MPT can be integrated with any traditional path planner.

We assess the performance of MPT augmented Sampling-based Motion Planners (SMPs) [Karaman and Frazzoli, 2011, Gammell et al., 2014] on a synthetic and real-world dataset comprising different map sizes in cluttered environments for point-mass and non-holonomic car models. Our results indicate that MPT augmented planners achieve sufficiently high accuracy in planning while substantially reducing planning time and the vertices on the planning tree compared to traditional methods. We also show that MPT maintains the same qualities to maps of different sizes without fine-tuning or additional training and generalizes to real-world environments. The diversity of environments represented, from block obstacles to narrow passage mazes speak to the potential general application of the MPT approach across multiple environment types and domains.

2 Related Work

The most relevant work to our transformers-based region proposal network for motion planning is perhaps the guided sampling-based motion planning methods. They analytically or through learned heuristics determine a subset in robot space that probably contains a path solution. For instance, Qureshi and Ayaz [2016, 2018] employ Artificial Potential Fields (APF) within sampling-based methods such as RRT* [Karaman and Frazzoli, 2011] and Bidirectional RRT* [Qureshi and Ayaz, 2015] to guide a subset of random samples towards promising regions that possibly contain an optimal path solution. In contrast, Informed-RRT* (IRRT*) [Gammell et al., 2014] depends on an initial path from an RRT* algorithm to compute an ellipsoidal region probably containing an optimal path solution. However, in most planning problems finding an initial path solution is itself challenging. In a similar vein, Batch Informed Trees (BIT*) [Gammell et al., 2015] begins from an elliptical region formed by a straight line path ignoring all obstacles and incrementally expand it until an initial path solution is found. Once an initial path is determined, it is further optimized by adapting the precomputed ellipsoid and generating new samples within that space.

Another approach for region selection is by Zucker et al. [2008], which uses REINFORCE algorithm [Williams, 1992] on discretized workspaces to guide samples for underlying SMP methods. Similarly, Value Iteration Networks (VIN) [Tamar et al., 2016] also discretizes the space and learns a value map to guide path planning. Universal Planning Networks (UPN) [Srinivas et al., 2018] extends VIN to continuous control spaces. Despite all advancements from APF-based directed trees to learning value maps, these methods only consider geometric planning, i.e., path planning under collision-avoidance constraints. They are yet to be evaluated in practical problems with complex constraints such as kinodynamic or non-holonomic constraints in robot manipulation and autonomous car navigation tasks.

Although there exist random sampling-based approaches such as RRT* [Karaman and Frazzoli, 2011, Arslan et al., 2017] and SST [Li et al., 2016] that randomly explore the robot state space and satisfy advance constraints, they suffer from the curse of dimensionality and take a considerable computational time in cluttered spaces. Neural Motion Planning [Qureshi et al., 2020, Ichter et al., 2018] has recently emerged as a promising tool for solving a wide range of planning problems under various task constraints, ranging from non-holonomic [Johnson et al., 2020, Li et al., 2021] to advanced manifold kinematic constraints [Qureshi et al., 2020], with high computational speed. These scalable methods learn sampling distributions from expert demonstrations and, on execution, generate samples for an underlying planner forming a subset that potentially contains a path solution. However, these approaches assume a fixed size input environment map and often require redefining
network architectures and retraining for different map sizes. However, recent developments in deep learning, primarily through Transformers [Dosovitskiy et al., 2021, Liu et al., 2021], have provided us with ways to relax such assumptions. Our proposed approach leverages these developments and introduces a region proposal framework that can work with variable map sizes and enhances underlying motion planners to solve complex problems in cluttered environments.

3 Method

This section will go over our planning pipeline, which includes the MPT module and the traditional SMPs used to generate the final plan. Our framework proposes promising regions from large maps to the underlying planners for quickly computing path solutions.

3.1 Motion Planning Transformer (MPT)

The MPT module, a region proposal network, is similar to that proposed in [Ren et al., 2015], but apart from having CNN and linear layers, we also include a Transformer model. The Transformer block we use is similar to [Dosovitskiy et al., 2021], which is derived from [Vaswani et al., 2017]. An overview of the model is shown in Figure 3. For a given input costmap \( m_c \in \mathbb{R}^{H \times W} \), start position \( x_s \in \mathbb{R}^2 \), and goal position \( x_g \in \mathbb{R}^2 \) we encode the start and goal position onto an array of size \( H \times W \). The goal and start location are represented by patches of size \( p \times p \) with values -1 and 1 respectively. The encoded map, \( m_e \in \mathbb{R}^{H \times W} \), is concatenated with \( m_c \) to form the input, \( m \in \mathbb{R}^{H \times W \times 2} \), to the feature extractor consisting of fully convolutional networks (FCN).

To extract the features from the encoder, we use a series of convolution, ReLU, and MaxPool layers, which have a combined receptive area of \( p \times p \). We refer to this module as the feature extractor. The feature extractor is similar to the CNN used by the hybrid architecture proposed in [Dosovitskiy et al., 2021], but instead of using a pre-trained network, we learn the filter parameters from scratch. The feature extractor is similar to the convolutional layers used in Faster R-CNN’s [Ren et al., 2015].

The output of the feature extractor is of size \( H_l \times W_l \times d \), where \( H_l \) and \( W_l \) is determined by the size of \( m_e \) and the convolution layers, and \( d \) is the latent dimension of the Transformer. The output is reshaped to size \( (H_l \cdot W_l) \times d \) in order to be compatible with the Transformer model. Each row vector is analogous to the word token’s in NLP.

Each row of the flattened output of the feature extractor can be mapped to a patch of size \( p \times p \) on \( m_e \) centered at position \( \hat{x} \in \mathbb{R}^2 \). We call all such points as anchor points. There are \( H_l \times W_l \) anchor points for a given input map. To encode the positions of each patch, we add fixed sinusoidal position
embeddings to the 1D features similar to [Dosovitskiy et al., 2021, Vaswani et al., 2017]. Since the positional encoding used is of a fixed size, there is an upper bound on the size of the input map.

The modified patch embedding is passed through a series of multi-headed self-attention (MSA), and multi-layer perceptron (MLP) blocks, identical to the architecture used by [Vaswani et al., 2017]. The MLP block consists of two linear layers with RELU non-linearity. We also apply Dropout, Layer Norm and residual connections after every MSA and MLP block. To save on memory, we add gradient checkpoint’s [Chen et al., 2016] to all MSA blocks. Since the feature extractor and the Transformer block are both invariant to the size of the input map, this architecture is scalable to maps of any size.

Each of the output token from the transformer encoder is passed through a classifier to identify regions that might represent a path. Similar to [Ren et al., 2015], this is implemented using a $1 \times 1$ convolution layer. Once all the tokens have been identified, a mask of dimension $H \times W$ is generated which highlights the region with a potential path.

### 3.2 Path Planning

The masked map is used as the reference map for finding a path and checking its feasibility. Any traditional or learning-based planners can be used to solve the given problem. In this work, we use variations of the Rapidly Exploring Random Trees (RRT) algorithm that guarantee optimality [Karaman and Frazzoli, 2011, Li et al., 2016, Gammell et al., 2014] to find the path under different settings. Note that MPT can enhance any existing path planner. It is robust to different map sizes, making it ideal for real-world scenarios, especially for robot navigation indoors (e.g., hospitals and homes) and outdoors (e.g., streets, highways) environments, where there are areas of variable sizes. MPT selects a subset in a larger map with fixed local receptive fields. The selected local patches become the new input map for the path planner, which would otherwise explore the entire space for pathfinding.

### 4 Experiments

We evaluate the planning capabilities of SMPs and Motion Planning Transformers (MPT) aided SMPs. To test the efficiency of the planner, we train the model on different types of environments of fixed size and evaluate on unseen maps of various sizes. To evaluate the applicability of the planner to other robotic systems, we train MPT on different types of robot. Lastly, we evaluate our algorithm on real world map without training on any additional data.

#### 4.1 Setup

#### 4.1.1 Environments

To test the planning capabilities of our method, we evaluated the model on two different kinds of synthetic environments. The first environment is called the Random Forest, where 100 circular and square objects are randomly placed in different positions and orientations on the map. The second environment is called the Maze environment. A perfect maze is generated using the randomized depth-first search. One characteristic of a perfect maze is that any start and goal pairs on this map are reachable by a collision-free path. The distance resolution of each map was set at 5cm per pixel.

#### 4.1.2 Robot Models and SMP

We test our algorithm on two robotic systems. The first is a simple Point Robot Model that can move in any direction in $\mathbb{R}^2$. Planning for this robot is done in the $\mathbb{R}^2$ space using the RRT* [Arslan et al., 2017] and Informed-RRT* (IRRT*) [Gammell et al., 2014] algorithms. The other robot system is a Dubins Car Model, controlled using wheel velocity and steering angle. The planning for this robot is done in the control space using the Stable Sparse RRT (SST) algorithm [Li et al., 2016]. We used the implementation from the Open Motion Planning Library (OMPL) [Sucan et al., 2012] for all SMPs.
4.1.3 Dataset

We created a simulated dataset for the environments and robot models mentioned above. We collected 25 paths for random start and goal pairs for 1750 environments using RRT* for both the Random Forest and Maze environments for the point robot. For the Dubins Car Model, we collected 50 paths for random start and goal pairs for 1000 environments using SST. All maps used for training were of size 480×480.

4.1.4 Training

The anchor points are classified as positive if they represent a path patch and negative otherwise. All anchor points near the given path up to 0.7 meters are considered positive, and the rest are considered background. For the Dubins Car Model, we used the robot’s position alone to pick the positive anchor points. Similar to Ren et al. [2015], RPN was trained with each mini-batch containing a single planning problem, with a 1:2 ratio of positive and negative samples. The negative samples are randomly sampled from the set of negative anchor points.

We trained the network by minimizing the cross-entropy loss for the select few anchor points using the Adam optimizer [Kingma and Ba, 2015] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e^{-9}$. We varied the learning rate as proposed in Vaswani et al. [2017] with warm-up steps of 3200. Each model was trained for 70k epochs with a batch size of 400. The models were trained on one machine with 4 NVIDIA 2080GTX graphics card and took 30hrs to train.

4.1.5 Metrics

To evaluate our planning algorithm, we tested it across different environments for random start and goal pairs. Since all the planners have some form of optimality criteria, it is unclear when to stop searching. A common practice in the motion planning community is to set a cost threshold for each path. The planners stopped searching when the solution path length is below this threshold or a time limit is reached. The cost threshold was the path length from an RRT* planner that searched the space till a solution was found and was collected as part of the validation set. The metrics we are interested in are the planning accuracy, the time it takes to find a solution, and the number of vertices on the planning tree. All the statistics we report, apart from accuracy, consider successful solutions which terminated before the time limit.

4.2 Point Robot Model

To evaluate the performance of our method, we compared it with RRT* and IRRT* without the aid of MPT. The aided versions of these planners are called MPT-RRT* and MPT-IRRT*. The first set of experiments examined the network’s ability to generalize to environments of the same size as the training dataset. Hence we evaluated it across 2500 different environments with a single start and goal pair for both the Maze and Random Forest environment. The summary statistics of the experiment are reported in Table 1. The MPT planners reduce the planning time and vertex count of the planning tree substantially. In Fig. 3, we show two examples of a planned path from this experiment.

As the start and goal points are sampled randomly, the statistic’s median values only provide a partial picture of the benefits of MPT. To better understand the advantages of MPT, we visualize the distribution of the planning time and vertices on the planning tree in Fig. 4 for the Random Forest and Maze environment using Letter-value plots [Hofmann et al., 2011]. These plots help to observe the tail of the distribution of the metrics. Naive RRT* has a heavier tail distribution than MPT-RRT*, which is as expected because, for start and goal pairs further away, the planner needs to generate a denser graph, requiring more time. The MPT-RRT* only searches on the region proposed by the MPT, and as a result, needs fewer tree vertices to reach the goal.

We also observe that IRRT* also performs considerably better than RRT* in the Random Forest environment and achieves similar planning time and planning tree vertices compared to the aided planners. This is because IRRT*, like MPT’s, reduces the planning search space once an initial solution is found by bounding the initial path with an Ellipse. Although this works for the Random Forest environments, for more complex maps such as the Maze, we see that MPT-RRT* and MPT-IRRT* perform considerably better than IRRT*.
Figure 3: Planned paths for two different planning problems. Top Row: Random Forest environment. Bottom Row: Maze environment. Left Column: The ground truth path from the validation data. Center Column: The prediction probability from the MPT. Right Column: The masked map and the planned trajectory using RRT*.

Figure 4: Planning statistics for the Point Robot Model. Left: The planning time for different SMP and MPT aided SMPs. Right: Number of vertices in the planning tree for SMP and MPT aided SMPs. MPT-RRT* and MPT-IRRT* consistently reduce the planning time and the vertices in the planning tree, resulting in a lower variance of these statistics for these planners.

The next experiment we conducted evaluated the planner’s ability to generalize to map sizes that were not part of the training data. We tested the model on four different map sizes, 240×240, 360×240, 480×240, and 560×560, of the Random Forest environment on 1000 randomly generated maps with an increasing number of obstacles. The same MPT model as before was used without any retraining or fine-tuning. Three successful planned paths are shown in Fig. 5. The planning statistics are summarized in Table 2.

We notice that the MPT-RRT* and MPT-IRRT* can achieve lower planning time and tree vertices count compared to RRT* without any additional training or fine-tuning. Thus MPT generalizes well to maps of different sizes. Similar to the previous experiment with fixed map sizes, IRRT* achieves similar performance to MPT-RRT*, although MPT-IRRT* achieves the best performance of the bunch. Better performance of MPT-IRRT* could be attributed to the fact that unlike IRRT*, which has to search the entire space to find an initial solution, MPT-IRRT* samples only within the proposed region, thus arriving at an initial solution faster with fewer vertices on the tree.
Table 1: Comparing planning accuracy, median planning time, and median number of vertices in the planning tree for the Point Robot Model on environments of the same size as the training data.

| Environment | Random Forest | Maze |
|-------------|---------------|------|
|             | Accuracy      | Time (sec) | Vertices | Accuracy | Time (sec) | Vertices |
| RRT*        | 99.88%        | 5.44      | 3227.5   | 100%     | 5.36      | 2042     |
| IRRT*       | 99.88%        | 0.42      | 267      | 100%     | 3.13      | 1393.5   |
| MPT-RRT*    | 97.68%        | 0.20      | 251      | 98.96%   | 0.83      | 615      |
| MPT-IRRT*   | 97.68%        | 0.07      | 133      | 98.96%   | 0.74      | 557      |

Figure 5: Plot of paths for Random Forest environments of different size. The architecture of the MPT Model allows flexibility in planning for environments of different sizes. Left: 240×240. Center: 360×240. Right: 480×240.

For the map size of 560×560, we do see a significant drop in accuracy. This drop in performance might be because the current depth of MSA cannot capture the global information for tokens that are far away. With larger map sizes, the number of input tokens increases quadratically, and as noted in Vaswani et al. [2017], a deeper MSA model performs better for longer input sequences. The problem requires further investigation and is an interesting future question to look at.

4.3 Dubins Car Model

Table 3: Planning accuracy and time for Dubins Car Model for the Random Forest Environment.

| Planner          | SST | MPT-SST |
|------------------|-----|---------|
| Accuracy         | 92.2% | 84.9%   |
| Time (sec)       | 28   | 15.8    |

To examine if the MPT can aid in the planning of non-holonomic systems, we trained a new MPT model to plan for a Dubins Car Model. We tested the trained model on 1000 Random Forest environments and compared the metrics with those from the naive SST planner. The SST aided by the MPT is called MPT-SST. Two trajectories from this experiment are plotted in Fig. 6 (Left and Center). We can observe that the MPT highlights regions so that SST can generate a kinematically feasible path.

Planners such as SST, which sample in the control space, do not scale well to larger planning spaces, and MPT can help with reducing the search space. The timing results reported in Table 3 support this
Table 2: Comparing planning accuracy, planning time, and number of vertices in the tree for Point Robot on maps of the different sizes of the Random Forest environment.

| Map Sizes  | Number of Obstacles | RRT*  | IRRT* | MPT-RRT* | MPT-IRRT* |
|------------|---------------------|-------|-------|----------|-----------|
| 240×240    | 20                  | Accuracy 100% | 100% | 95.4% | 95.4% |
|            |                     | Time (sec) 4.862 | 0.25 | 0.27 | 0.015 |
|            |                     | Vertices 3199 | 229 | 402.5 | 52 |
| 360×240    | 35                  | Accuracy 98.9% | 98.9% | 97.4% | 97.4% |
|            |                     | Time (sec) 5.93 | 0.29 | 0.27 | 0.06 |
|            |                     | Vertices 3660 | 257 | 377 | 118 |
| 480×240    | 50                  | Accuracy 99.4% | 99.4% | 96.3% | 96.3% |
|            |                     | Time (sec) 6.31 | 0.39 | 0.27 | 0.07 |
|            |                     | Vertices 3480 | 291 | 348 | 129.5 |
| 560×560    | 100                 | Accuracy 100% | 100% | 75.6% | 75.6% |
|            |                     | Time (sec) 6.71 | 0.28 | 0.22 | 0.051 |
|            |                     | Vertices 3854 | 203 | 262 | 101 |

Figure 6: MPT can also be trained to aid SMP planners for non-holonomic robots. Left and Center: Planned paths on Random Forest environment using MPT-SST. Right: The Letter-value plot comparing the planning time between SST and MPT-SST. MPT is able to reduce the planning time for SST.

4.4 Real World Maps

The final experiment we performed was to test the model trained on Point Robot Model on a real-world map. We obtained a map of building 079 University of Freiburg from publicly available 2D lidar scan data. We used the MPT model, trained on the synthetic Random Forest and Maze environment for the Point Robot, to plan on 20 randomly sampled start and goal pairs. MPT-RRT* was able to solve 17 of the 20 planning problems, and we plot two of these trajectories in Fig. 7. This result is encouraging since it shows MPT’s can solve planning problems in real-world maps by training on synthetic maps. Robots that have such generalization capabilities will be able to adapt to dynamic environments easily.

5 Conclusion

In this work, we have shown an application of the transformer model for motion planning tasks. Unlike prior methods that need to retrain models for maps of different sizes, we leverage the ability of transformers to handle sequences of different lengths and parallelize long-term dependencies without
recursions for planning problems with different map sizes. By combing MPT with SMP, we could generate paths faster, with fewer tree nodes for different environments and robot models.

While the initial results are impressive, there are lots of places to improve. One being, the performance of the planner with maps that are larger than training data size. In NLP and vision tasks, larger transformer models seem to improve performance. Similarly, larger MPT models may result in better accuracy for these maps. Extending the planner to higher dimensional robots such as robotic arms or drones is also an interesting problem with many real-world applications. Because of the higher dimensions of these spaces, sparse Transformer models would be an ideal choice for these problems.

References

Oktay Arslan, Karl Berntorp, and Panagiotis Tsiotras. Sampling-based algorithms for optimal motion planning using closed-loop prediction. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 4991–4996, 2017. doi: 10.1109/ICRA.2017.7989581.

Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. Nature, 557, 05 2018. doi: 10.1038/s41586-018-0102-6.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

Devendra Singh Chaplot, Deepak Pathak, and Jitendra Malik. Differentiable spatial planning using transformers, 2021. URL https://openreview.net/forum?id=n4IMHNb8_f

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=YicbFdNTTy

Scott Emmons, Ajay Jain, Misha Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 5251–5262. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/385822e359afa26d52b5b296226f2ceaa-Paper.pdf
Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2997–3004, 2014. doi: 10.1109/IROS.2014.6942976.

Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In 2015 IEEE international conference on robotics and automation (ICRA), pages 3067–3074. IEEE, 2015.

Peter E. Hart, Nil s J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

Heike Hofmann, Karen Kafadar, and Hadley Wickham. Letter-value plots: Boxplots for large data. Technical report, had.co.nz, 2011.

Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 7087–7094. IEEE, 2018.

Jacob J. Johnson, Linjun Li, Fei Liu, Ahmed H. Qureshi, and Michael C. Yip. Dynamically constrained motion planning networks for non-holonomic robots. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6937–6943, 2020. doi: 10.1109/IROS45743.2020.9341283.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7):846–894, 2011.

L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

Steven M. LaValle and Jr. James J. Kuffner. Randomized kinodynamic planning. The International Journal of Robotics Research, 20(5):378–400, 2001. doi: 10.1177/02783640122067453. URL https://doi.org/10.1177/02783640122067453.

Linjun Li, Yinglong Miao, Ahmed H Qureshi, and Michael C Yip. Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints. IEEE Robotics and Automation Letters, 6(3):4496–4503, 2021.

Yanbo Li, Zakary Littlefield, and Kostas E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. The International Journal of Robotics Research, 35(5):528–564, 2016. doi: 10.1177/0278364915614386. URL https://doi.org/10.1177/0278364915614386.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip. Neural manipulation planning on constraint manifolds. IEEE Robotics and Automation Letters, 5(4):6089–6096, 2020.

Ahmed H. Qureshi, Anthony Simeonov, Mayur Bency, and Michael C. Yip. Motion planning networks. In 2019 International Conference on Robotics and Automation (ICRA), pages 2118–2124, 2019. doi: 10.1109/ICRA.2019.8793889.

Ahmed Hussain Qureshi and Yasar Ayaz. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. Robotics and Autonomous Systems, 68:1–11, 2015.
Ahmed Hussain Qureshi and Yasar Ayaz. Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, 40(6):1079–1093, 2016.

Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e455ba028a21ed38046-Paper.pdf.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 02 2021. doi: 10.1162/tacl_a_00353.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4732–4741. PMLR, 2018.

Ioan A. Sucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi: 10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

Zaid Tahir, Ahmed H Qureshi, Yasar Ayaz, and Raheel Nawaz. Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, 108:13–27, 2018.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a545aa-Paper.pdf.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Matt Zucker, James Kuffner, and J Andrew Bagnell. Adaptive workspace biasing for sampling-based planners. In *2008 IEEE International Conference on Robotics and Automation*, pages 3757–3762. IEEE, 2008.

**Appendix**

**A Network Architecture**

In this section, we detail the network architecture used in our experiments. The Transformer architecture is similar to the ones proposed in [Vaswani et al., 2017]. We used 6 layers of encoder block, each consisting of 3 heads. The dimension of the keys and queries was set at 512, and the dimension of the value was set at 256. The architecture of our feature extractor is given in Table A. For the convolution layer, the dimensions in the brackets represent [Input Channel Size, Output Channel Size, Kernel Size, Stride], and for the Maxpool layer, it represents the Kernel Size.

| Layer       | Dimension               |
|-------------|-------------------------|
| 2D Convolution  | [2, 6, 5, 0]             |
| 2D Maxpool      | [2]                     |
| ReLU            |                         |
| 2D Convolution  | [6, 16, 5, 0]            |
| 2D Maxpool      | [2]                     |
| ReLU            |                         |
| Convolution     | [16, 512, 5, 5]          |

Table A: Network architecture of the Feature Extractor
B Successful Trajectories

In this section we plot few more successful trajectories of fixed map sizes for the Random Forest and Maze environments in Fig. 8 and 9 respectively.

Figure 8: Three different trajectories planned successfully using MPT on the Random Forest environment. Left Column: The ground truth path in the validation dataset. Middle Column: The probability estimate made by the MPT. Right Column: The planned path using RRT* on the region proposed by the MPT.
Figure 9: Three different trajectories planned successfully using MPT on the Maze environment. Left Column: The ground truth path in the validation dataset. Middle Column: The probability estimate made by the MPT. Right Column: The planned path using RRT* on the region proposed by the MPT.
C Distribution of Metrics for Maps of Different Sizes

The distribution of metrics reported in Table 2 is given in Fig. 10.

Figure 10: The distribution of metrics for maps of different sizes. Top: The distribution of planning time for different algorithms for maps of different sizes. Bottom: The distribution of the number of vertices in the planning tree for different algorithms for maps of different sizes. MPT aided planners can achieve faster planning times and lower variance in the metrics across maps of different sizes, which were not part of the training data.
D Failed Trajectories

In Fig. 11 and 12, we visualize some of the trajectories that MPT could not solve for the Random Forest and Maze environment, respectively. These failures were mainly caused by a single patch that was not classified as part of the path, resulting in a map with no valid paths.

Figure 11: Three different trajectories planned unsuccessfully using MPT on the Random Forest environment. Left Column: The ground truth path in the validation dataset. Middle Column: The probability estimate made by the MPT. Right Column: The planned path using RRT* on the region proposed by the MPT. We observe that in all three cases the failure to predict a few patches resulted in a scenario with no valid paths.
Figure 12: Three different trajectories planned unsuccessfully using MPT on the Maze environment. Left Column: The ground truth path in the validation dataset. Middle Column: The probability estimate made by the MPT. Right Column: The planned path using RRT* on the region proposed by the MPT. We observe that in all three cases the failure to predict a few patches resulted in a scenario with no valid paths.