

MTG: Mapless Trajectory Generator with Traversability Coverage for Outdoor Navigation

Jing Liang¹, Peng Gao¹, Xuesu Xiao², Adarsh Jagan Sathyamoorthy¹,
Mohamed Elnoor¹, Ming Lin¹ and Dinesh Manocha¹

Abstract—We present a novel learning algorithm for trajectory generation for outdoor robot navigation. Our goal is to compute collision-free paths that also satisfy the environment-specific traversability constraints. Our approach is designed for global planning using limited onboard robot perception in mapless environments and ensures comprehensive coverage of all traversable directions. Our formulation uses a Conditional Variational Autoencoder (CVAE) generative model that is enhanced with traversability constraints and an optimization formulation used for the coverage. We highlight the benefits of our approach over state-of-the-art trajectory generation approaches and demonstrate its performance in challenging and large outdoor environments, including around buildings, across intersections, along trails, and off-road terrain, using a Clearpath Husky and a Boston Dynamics Spot robot. In practice, our approach results in a 6% improvement in coverage of traversable areas and an 89% reduction in trajectory portions residing in non-traversable regions.

I. INTRODUCTION

Global planning for autonomous mobile robots has evolved significantly over the years. While many methods rely on map-based planning [1]–[3], accurate maps are not always available for many scenarios. These scenarios include rural areas without complex RGB or geometric features in the environment [4], areas under construction that undergo significant change over time, satellite-inaccessible areas, etc. In these cases, mapless global planning [4] or real-time map analysis [5], [6] are used for navigation.

Most work in map-based planning has focused on computing a collision-free optimal trajectory [8]–[10]. In contrast, a key issue in mapless global planning is to compute traversable directions for long-range navigation and combine that with collision-avoidance capabilities of a local planner [4], [11]. These traversable directions can be calculated in a low frequency (0.1 Hz) to provide high-level trajectory for robots to follow. The difference between outdoor map-based and mapless navigation arises from the fact that the lack of an accurate global map does not allow a robot to conduct very accurate collision checking with occlusions. Therefore, the global planner only needs to compute the rough navigation directions such navigation directions at different locations for outdoor navigation. This can be a challenging task, where the robot needs to estimate the possible trajectories based on the limited onboard observations on complex, outdoor terrains.

The trajectory generation task for global planning presents several challenges, especially in the absence of a map. As shown in Figure 1, on one hand, the generated trajectories

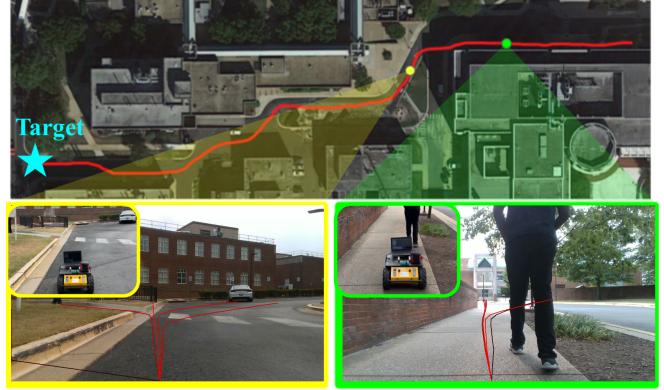


Fig. 1: Trajectory generation in a campus environment. The top view is the robot’s trajectory (around 350m). The bottom row shows generated trajectories at the two locations corresponding to green and yellow boxes, respectively. The trajectory closest to the global target, shown at the top, will be chosen to provide waypoints for the local planner. The local planner drives the robot and avoids collisions [7]. In this complex outdoor scenario, our MTG global navigation method can efficiently and safely generate trajectories for global navigation and cover most of the traversable areas, including the narrow pedestrian sidewalks.

should avoid non-traversable areas, such as bushes, trees, etc; On the other hand, the generated trajectories should be able to cover all these traversable directions in the view of the robot, and account for partial occlusion caused by static and dynamic obstacles and resolving any ambiguities in path computation, as bottom right frame of Figure 1.

Current solutions for off-road global planning typically separate trajectory generation and environmental traversability analysis [12]–[14]. Moreover, they often rely on intuitive human maps to handle the occlusion issues and analyze the traversability of the environment. However, robots could potentially leverage encoded information, which can be generated by encoding the observations of the robot with some learning models, for direct trajectory generation without necessitating pre-conceptual map construction. Furthermore, map construction and planning are also computationally intensive [15], which hinders real-time navigation. Therefore, our work aims to develop an end-to-end model that seamlessly handles occlusion, traversability analysis, and trajectory generation together. To this end, generators [16], [17] provide a solution for empirical trajectory generation. For example, Dlow [18] and Ma et al. [19] utilize historical trajectories to generate new ones, incorporating likelihood

¹University of Maryland, College Park. ²George Mason University

loss to encourage the diversity of the generated trajectories. However, they often overlook inter-trajectory influences and traversability constraints.

Main Results: We present a novel approach for trajectory generation in mapless global planning, focusing on creating trajectories that cover the most viable directions within a robot’s limited (120 degrees) field of view, while respecting environmental traversability constraints. We present an **end-to-end** mapless trajectory generator, MTG, which can efficiently compute viable trajectories that are traversable for both wheeled and legged robots. Our model is trained and demonstrated in the campus scenarios, where the non-traversable areas include bushes, trees, buildings, streets with traffic, etc. The MTG approach achieves 72% coverage of the traversable areas in the robot’s field of view. The coverage is calculated by the distance between generated trajectories and ground truth trajectories (see Section III). The major contributions include:

- 1) **Novel End-to-End Learning Model:** We propose a novel end-to-end architecture for global trajectory generation, which can efficiently and effectively generate viable trajectories. Based on the architecture of Conditional Variational Autoencoder (CVAE), we extend the method to the task of using multiple trajectories distributions to cover possible traversable areas. Different distributions of the embedded information, generated by the encoder, are applied with the attention mechanism to provide the conditions to the Decoders. The attention model informs the decoder about other trajectories, guiding the generation of more effective trajectories, which is closer to the ground truth.
- 2) **New Loss Functions:** We design an innovative loss function that accounts for multiple constraints inherent in mapless trajectory generation. These include traversability, diversity, and coverage constraints. Our approach ensures trajectories account for the occlusions caused by dynamic obstacles and does not travel into non-traversable regions, while performing comprehensive exploration of open areas.
- 3) **Performance Improvement:** Empirical evaluations reveal that our model outperforms existing methods, such as DLOW [18] and CVAE [16]. We demonstrate the performance by conducting tests on various robots, including Clearpath Husky and Boston Dynamics Spot robots, in challenging environments with occlusions by dynamic obstacles and different non-traversable areas, such as trees, buildings, bushes, etc. The testing environment for the global planning has the scale of several hundred meters. Our method consistently shows at least a 6% improvement in coverage of traversable areas and up to 89% reduction in trajectory portions belonging to non-traversable regions.

II. BACKGROUND

Path generation in traversable regions in outdoor navigation has attracted significant attention from the robotics community. In this section, we review different approaches.

Traversability Analysis: The simplest approach of path generation is to separate the task into traversability analysis and planning [12], [20], [21]. For traversability analysis, multiple sensors can be used as input. Frey et al. [21] use an RGB camera and ViT [22] with self-supervised learning to segment traversable areas; WayFAST [20] uses an RGBD sensor and a Resenet backbone to segment the traversable areas from the input observations. Step [12] uses both a Lidar and an RGB mono camera to extract an elevation map and analyze multiple risks on the map, including collisions and steep elevation. These methods assume that with the perfect traversability map the trajectory generation problem can be handled by regular path/motion planning algorithms. However, in the real-world scenarios, building an accurate traversability map in real-time is non-trivial due to occlusions in observations and the computational constraints in building a map [13]. In addition, the map is for human perception, but a robot does not require a human-friendly map for navigation. We don’t really need to generate any intermediate map in the process. Therefore, an end-to-end learning strategy could be efficient to handle the problem of trajectory generation in traversable areas with occluding obstacles.

End-to-end Approaches: Generating trajectories on traversable areas can be totally based on the empirical information of either other agents or the robot itself. TridentNetV2 [23] uses a raw path planned by a global planner as a guide to help the trajectory generation to reach a target specified by a GPS coordinate. However, this method still requires a high-level map for planning before trajectory generation. ViKING [24] and LaND [25] train their models to generate trajectories approximating a collected dataset, but these methods are trained with a specific optimization targets and cannot provide multiple possible directions for different navigation purposes. FlowMap [26] assesses the agents in front of the robot to estimate the traffic flow and to generate the trajectories. However, all these methods require some guidance beforehand and cannot fully explore the traversable areas by generating multiple trajectories. Our approach, on the other hand, generates diverse trajectories that satisfy traversability constraints and can cover most traversable areas in front of the robot.

Trajectory predictions: Trajectory prediction estimates the future trajectories of agents based on their historic paths. SocialGAN [17] and Flomo [27] use generative models to forecast the trajectories of the agent, but they cannot generate diverse trajectories to cover all traversable areas. Dlow [18] and Ma et al. [19] use the likelihood loss to learn diverse trajectories. But they do not constrain the trajectories to traversable areas. For terrains with limited traversable areas, such methods could experience convergence issues since some trajectories always lie in the non-traversable regions. Vern [28], TerraPN [14], GrASPE [29] and AdaptiveON [7] handle the navigation in off-road navigation scenarios. Although, these methods only solve short-distance motion planning problems, they are complimentary to our approach for a complete global navigation strategy.

III. APPROACH

In this section, we formulate our problem and describe our approach to solve it. For global navigation, we need the local trajectory generator to provide enough trajectory candidates to cover the traversable areas in the available free space. In this method, we only use map in the training stage and in the experiment stage, the method directly takes robots' perception as input and output trajectories. During the training stage, with a traversability map shown in Appendix Section VI-E, we use the A* algorithm to generate trajectories to different distributed targets that are a certain distance D from the robot. We assume that these trajectories mostly cover the directions to all open spaces D -meters away from the robot. We define these trajectories as the ground truth trajectories. The coverage of the generated trajectories is measured by the distances to the ground-truth trajectories.

A. Problem Formulation

Given the observation $o \in \mathcal{O}$, the our end-to-end model, $m_\theta(\cdot)$ with parameter θ , generates trajectories $\tau_k \in \mathcal{T}$, where $\tau_k = m_\theta(o)_k$ is the k th output of the model $m_\theta(o)$. K is the number of generated trajectories. We formulate trajectory generation as an optimization problem under the constraints of traversability as follows:

$$\hat{\theta} = \arg \max_{\theta} \left(\frac{\bigcup_{k=1}^K f(m_\theta(o)_k)}{|\mathcal{A}|} - \beta \sum_{k=1}^K g(m_\theta(o)_k, \tilde{\mathcal{A}}) \right), \quad (1)$$

where $\hat{\theta}$ is final well-trained parameter of our model. $f(\cdot)$ represents the area covered by the generated trajectories τ_k in the traversable area, \mathcal{A} . $g(\cdot)$ calculates the portion of the path τ_k in non-traversable area, $\tilde{\mathcal{A}}$. β is a hyper-parameter, a weight of the penalty of trajectories lying in non-traversable area. Intuitively, the first term is used to ensure the coverage of generated trajectories and the second term is used to constrain their traversability. In this approach, we use the negative exponential distance to the ground truth to substitute the first half of Equation 1.

In our formulation, we consider the input as two observations $\mathcal{O} = \{\mathcal{L}_i, \mathcal{V}\}$, where \mathcal{L}_i represents the past N_l frames of 3D Lidar point clouds and \mathcal{V} composes N_v consecutive frames of the robot's velocities. The encoder, as shown in Figure 2, processes the observation into an embedded vector \mathbf{z} , where the randomness is introduced for the trajectory generation. After processing the embedded vector to several different Gaussian distributions, each decoder takes one distribution as input and generates one trajectory. The model's formulation is represented as,

$$p(\tau|\mathbf{c}) = \int_{\mathbf{z}} p(\tau|\mathbf{z}, \mathbf{c}) p(\mathbf{z}) d\mathbf{z}; \quad (2)$$

$$p(\tau|\mathbf{c}) \approx \frac{1}{S} \sum_{s=1}^S p(\tau|\mathbf{z}^{(s)}, \mathbf{c}), \mathbf{z}^{(s)} \sim p_\theta(\mathbf{z}|\mathbf{x}). \quad (3)$$

Here, $\tau \in \mathcal{T}$ denotes the generated trajectories and its distribution form is denoted as $p(\tau|\mathbf{x})$. τ is composed of S waypoints in the forward (X-axis) and the left (Y-axis)

directions of the current robot frame. $\mathbf{x} \in \mathcal{O}$ denotes the input observation. In addition, $\mathbf{l} \in \mathcal{L}$, $\mathbf{v} \in \mathcal{V}$, and $\mathbf{z} \in \mathcal{Z}$ denote random sampling data from a Gaussian distribution $p(\mathbf{z})$. \mathbf{c} denotes a function of input \mathbf{x} , and we use a network to represent the function $\mathbf{c} = f_d(\mathbf{x})$. f_d represents the encoder.

B. Attention-based CVAE

We present a new model to enhance the condition values of CVAE(Conditional Variational Autoencode) [16]. We introduce the attention mechanism to provide the information of other generated trajectories as condition to the decoder to make the generated trajectories more diverse and accurate compared with original CVAE [16] and DLOW [18].

Our goal is to generate diverse trajectories to cover all the traversable directions in the field of view of the robot. Therefore, given a certain number of trajectories, under the constraint of traversability, we need to make the trajectories as diverse as possible. We define the trajectories closest to the ground truth trajectories as *effective* and other trajectories as *redundant*. In addition to the constraint of traversability, we require the effective trajectories to be as diverse as possible to cover all the ground truth trajectories and the redundant trajectories to be also in traversable areas and close to the effective trajectories. Dlow [18] provides the idea of generating diverse trajectories, but it does not handle redundant trajectories or constrain the traversability of the trajectories. We utilize Dlow's idea of processing and computing embedded information \mathbf{z} , shown in Figure 2 as an embedded vector from the encoder. In this work, we use an attention model to provide the information about other trajectories to support the constraints of the traversability and different sorts of trajectories.

The output of our decoder is the positional difference $(\Delta x_n, \Delta y_n)$, which can be accumulated to future waypoints $\tau = \{p_1, \dots, p_n, \dots, p_N\}$, where each position $p_n = (x_n, y_n)$. We also use multiple invertible linear transformations $\mathbf{z}_k = A_i(c)\mathbf{z} + b_i(c) = h_\psi(\mathbf{z})$ to transform certain Gaussian distributions $\mathcal{N}(\mu, \nu)$ to different Gaussian distributions $\mathcal{N}(\mu_k, \nu_k)$, where $A_i(c)$ and $b_i(c)$ are all functions of condition \mathbf{c} . ψ is the trainable parameters of the linear transformation layers. \mathbf{z} is the output of encoder $q(\mathbf{x}, \epsilon)$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For K trajectories, each trajectory τ_k can be generated by the decoder, $p(\tau_k|\mathbf{z}_k, \mathbf{c})$, where $\mathbf{z}_k \in \mathcal{Z}$ and \mathcal{Z} represents the set of Gaussian distributions.

Given the embeddings \mathcal{Z} , we use a self-attention model [30] $g(\mathcal{Z})$ to calculate the relationship among all $\mathbf{z}_k \in \mathcal{Z}$. Therefore, each embedding \mathbf{z}_k is enhanced by other embeddings' information $\bar{\mathcal{Z}}$, where $\bar{\mathcal{Z}}_k = \mathcal{Z} \setminus \mathbf{z}_k$. The enhanced embedding \mathbf{z}_k will be put into a decoder to generate a single trajectory. Now we have the CVAE formulation as

$$p(\tau_k|\mathbf{c}) = \int_{\mathbf{z}_k} p(\tau_k|\mathbf{z}_k, \mathbf{c}, \bar{\mathcal{Z}}_k) d\mathbf{z}_k. \quad (4)$$

This formulation is to constrain the effective trajectories τ_c and redundant trajectories τ_o to be as close as possible, and τ_c are as diverse as possible to help cover all ground truth trajectories. There are two targets: $\min d(\tau_c, \tau_o)$ and

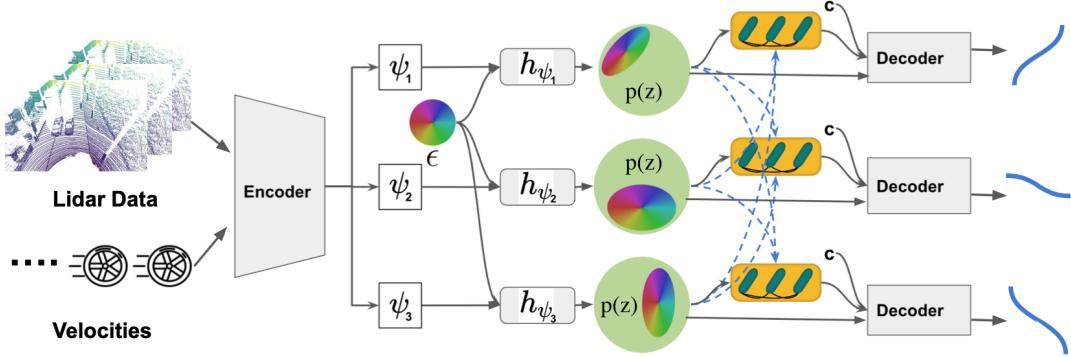


Fig. 2: **Overall Pipeline of MTG:** The inputs are several consecutive frames of Lidar point clouds and velocities of the robot, ϵ is a Normal distribution, and c is the condition value. Green circles represent distribution of \mathbf{z} , $p(\mathbf{z})$. The last part represents the decoders, and each decoder generates one trajectory.

$\max d(\tau_{c,i}, \tau_{c,j})$, where $d(\cdot, \cdot)$ is the distance function between two trajectories. Dlow [18] solves the second target, but it's not able to solve the first target effectively due to the lack of information of other trajectories. Our formulation provides the information $\bar{\mathcal{Z}}$ to the decoders and helps trajectories to achieve the first target.

C. Constraints on Traversability, Diversity, and Coverage

The generated trajectories should satisfy different constraints in the outdoor environment. The generated trajectories should cover different traversable directions; Trajectories should not lie in non-traversable areas; and if there are redundant trajectories after covering all the traversable areas, these trajectories should still lie in traversable areas. These constraints are not trivial challenges, so in the following we present the methods to apply the constraints in our model.

The CVAE's lower bound loss function: The lower bound loss function contains two parts: KL-divergence and reconstruction loss, as in Equation 5

$$\mathcal{L}_C = -\mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) + \frac{1}{L} \sum_{l=1}^L \log p_\theta(\tau|\mathbf{x}, \mathbf{z}). \quad (5)$$

The reconstruction loss can be formulated in two parts: 1. The difference between the final positions from both the ground truth and the generated trajectory; 2. The average Hausdorff function [31] to make the generated trajectories $\hat{\tau}$ cover all the target trajectories τ , as in Equation 6, where s and \hat{s} are waypoints in the trajectories.

$$d_h(\hat{\tau}, \tau) = \frac{1}{2} \left(\frac{1}{|\tau|} \sum_{s \in \tau} \min_{\hat{s} \in \hat{\tau}} d(s, \hat{s}) + \frac{1}{|\hat{\tau}|} \sum_{\hat{s} \in \hat{\tau}} \min_{s \in \tau} d(\hat{s}, s) \right). \quad (6)$$

In contrast to the traditional CVAE reconstruction loss, we want the generative model to cover all the ground truth instead of directly training all the trajectories with the same ground truth trajectories. Therefore, we change the reconstruction loss to the coverage loss, where in each time step we only back propagate the information of the nearest trajectory to each of the ground truth trajectories:

$$\mathcal{L}_{CVAE} = -\mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) + d_h(\hat{\tau}_c, \tau_k), \quad (7)$$

where the $\hat{\tau}_c$ is the closest trajectory to the k^{th} ground truth trajectory τ_k .

Diversity loss function: In this problem, we generate T trajectories at each step that are diverse enough. If T is more than the ground truth trajectories, we want the redundant generated trajectories as close as possible to other ground truth trajectories. We have the diversity loss function:

$$\mathcal{L}_D = \exp(-d(\hat{\tau}_c, \hat{\tau}_c)) + \exp(d(\hat{\tau}_o, \hat{\tau}_c)), \quad (8)$$

where $\hat{\tau}$ is the output trajectory, τ_c represents the trajectories closest to one of the ground truth trajectories, and τ_o represents the other trajectories. The redundant τ_o always appear when generated trajectories are more than ground truth trajectories. $d(\cdot)$ is the Euclidean distance between two trajectories, where each τ_o only compares with the closest τ_c . The attention model contributes to telling the information of other trajectories and enhances the generated trajectories either close to the ground truth or close to other effective trajectories, τ_c .

Traversability loss function: Trajectory generation requires the generated trajectories to be without collision with nearby big obstacles (bushes, trees, buildings, etc) or incursions into non-traversable areas. We use the distance to the non-traversable areas as the loss to train the generated trajectories away from the regions:

$$\mathcal{L}_c = \exp \left(1 - \min(\max(\frac{1}{N} \sum_{n=1}^N \min d(\mathcal{C}, \mathbf{p}_n), 1), 0) \right), \quad (9)$$

where \mathcal{C} represents the obstacles set near the robot and distance function $d(\cdot)$ represents the Euclidean distance between the generated waypoint \mathbf{p}_n and the obstacles or non-traversable areas.

The total loss function can be written as,

$$\mathcal{L} = -\beta_1 \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) + \beta_2 (d_h(\hat{\tau}_c, \tau_k) + \mathcal{L}_D) + \beta_3 \mathcal{L}_c, \quad (10)$$

where the first part is the CVAE KL-divergence, and the second part contributes to the exploration to achieve the maximum open space by covering the ground truth paths. The third part constrains the paths to the traversable areas.

IV. EXPERIMENTS

In this section, we detail the implementation of our method and describe the results compare with other state-of-the-art methods. For the real-world experiment we combine MTG and a low-level motion planner AdaptiveON [7] for global navigation as Figure 1. Details of the real-world experiment can be found in Appendix VI-A. We also analysis the confidences of the generated trajectories and the generalizability of our approach in the Appendix VI-B.

A. Implementation

During experimentation, we deployed the model on different robots, a Clearpath Husky and a Boston Dynamics Spot. The major perceptive sensor is a 16-channel Lidar, Velodyne VLP-16, with 3Hz frequency. we choose $N_l = 3$ frames in the experiment, base on the prior research [32], [33] that 3 previous frames are enough to encode the information of dynamic obstacles for collision avoidance. To keep the observation in the same time period, we choose $N_v = 10$ for velocities based on a 10Hz frequency measured by the robots' odometers. Our training and testing datasets are collected in a campus environment, as shown in Appendix VI-G; the two datasets are in very different areas. The training dataset contains three parts: 1. The original observation data, including Lidar and velocities. 2. Based on the perception, we build a traversability map, which is only used as the ground truth to guide the training. We briefly described the generation of the traversability map in Appendix VI-C. 3. Based on the map, we sample multiple diverse targets and apply an A* planner to generate raw ground truth trajectories. The targets are sampled 15m away, considering robot's speed is 1m/s and the trajectories are generated for the next 15 seconds. Details are in Appendix VI-D

As Figure 2 shows, the encoder encodes perception information to a hidden vector \mathbf{c} , which is also the condition value to the decoder. It contains two sub-models. One is the Lidar model that composes multiple 3D convolutions layers to process the stacked Lidar data. In this work, we use PointCNN [34] to process the Lidar data with a 0.08m voxelization radius. The other model is a velocity model, including three consecutive fully connected layers, to process the velocity data. Then these two encoded perception data are concatenated and processed by several fully connected layers to generate the hidden vector \mathbf{c} . The $A_i = l_a(\mathbf{c})$ and $b_i = l_b(\mathbf{c})$ are generated by linear procession of \mathbf{c} , and l_a and l_b are fully connected layers. The $g(\mathcal{Z})$, is processed by an attention model where the inputs are the processed Gaussian distribution vectors and the outputs are \mathbf{c}_s . Concatenated with the encoded condition \mathbf{c} , we have $\mathbf{c}_i = \{\mathbf{c}_s, \mathbf{c}\}$ as the input of the condition to the decoder. The decoder is composed of a sequence of GRU cells, and outputs a sequence of $\{\Delta x_n, \Delta y_n\}$, which are accumulated to waypoints $p_n = (x_n, y_n)$ with the initial position of $\{0, 0\}$. The details of the architecture can be found in Appendix VI-F. In each generated trajectory, there are 16 waypoints, based on the furthest distance of 15m. The training is processed by an NVIDIA RTX A5000 GPU and an Intel Xeon(R) W-2255

CPU, and we use this machine for evaluation. The qualitative evaluation results between the ground truth A* and generated trajectories are in Appendix VI-E.

B. Comparing Results

In this section, we qualitatively and quantitatively evaluate the performance of our method with other approaches.

1) *Quantitative Results:* The evaluation metrics include the following:

Non-traversable rate: Ratio of the generated trajectories lying on non-traversable areas. This metric is calculated by

$$r_n = \frac{1}{K} \sum_{k=1}^K \frac{g(m_\theta(o)_k, \tilde{\mathcal{A}})}{|\tau_k|}, \quad (11)$$

where $g(\cdot)$ is defined in Equation 1 as the segments of trajectory τ_k lying on non-traversable areas.

Coverage rate: This metric measures the coverage of the traversable areas in the robot's perception by generated trajectories. In this project, we set the robot's perception to a 120 degree field of view in front of the robot. The assumption is that the ground truth trajectories covered all the traversable areas, which is true of most of the cases based on the traversable map we built. We measure the smallest Hausdorff distance between ground truth trajectories and the generated trajectories.

$$r_c = \frac{1}{G} \sum_{i=1}^G \exp - \min d_h(\tau_i, \hat{\mathcal{Y}}), \quad (12)$$

where G is the number of all ground truth trajectories in the map and $\hat{\mathcal{Y}}$ is the set of generated trajectories. τ_i is the i^{th} of the ground truth trajectory.

Diversity rate: Diversity of the generated trajectories. This metric measures the Euclidean distance among the generated trajectories. This metric is a complement to the coverage metric.

$$r_d = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_h(\hat{\tau}_i, \hat{\tau}_j), \quad i \neq j. \quad (13)$$

Methods	Non-traversable	Coverage	Diversity (10^9)	Running Time (ms)
Ground Truth	0	1	28	N/A
CVAE [16]	0.14	0.63	7.0	5
Dlow [18]	0.15	0.68	79.0	3
MTG ¹ (Ours)	0.017	0.70	10.0	4
MTG(Ours)	0.013	0.72	14.0	5

TABLE I: Comparison of CVAE [16], Dlow [18], vanilla MTG, and complete MTG. The ground truth values are also provided as reference. Our method has 6% improvement in coverage of traversable areas as Equation 12, and improves the traversability of the trajectories by Equation 11.

Running time: t in each step and the unit is seconds.

We compared our method with ground truth A* paths, vanilla CVAE method [16], and Dlow [18]. As shown in Table I, CVAE doesn't have a diversity function or coverage constraints, so the output centers on very similar trajectories. Therefore, the coverage and diversity of CVAE are very

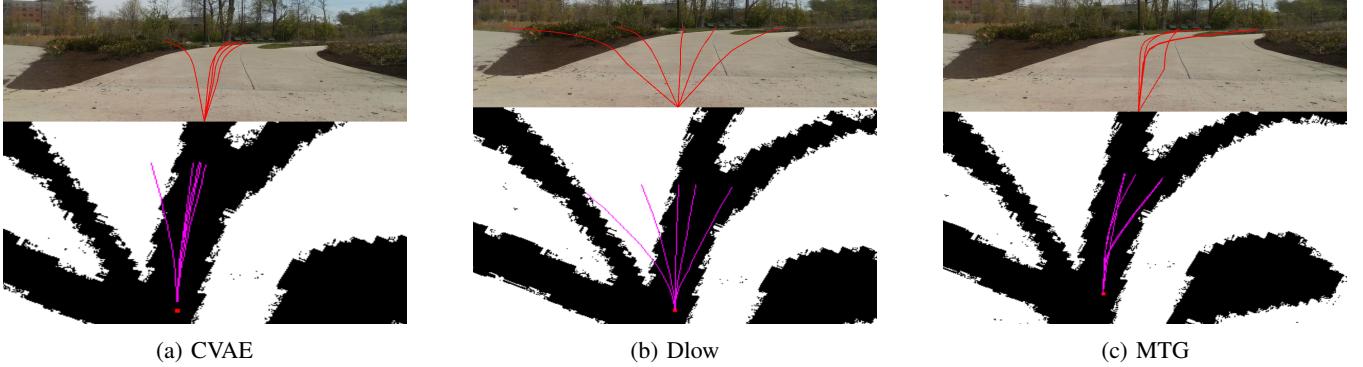


Fig. 3: **The quality results** of CVAE [16], Dlow [18], and MTG. The top row is from the robot’s view, and the bottom row is the birds-eye-view. White areas are non-traversable areas. The CVAE generates trajectories very similar to each other, while Dlow has large diversity but not good in terms of traversability. traversability. Our approach generates trajectories that mostly cover all traversable directions and lie on only traversable areas.

low. The Dlow method evenly implies the diversity loss to separate the generated trajectories, so the diversity value is high. However, neither CVAE or the Dlow method provide the hard constraints on the non-traversable areas, leading to trajectories with large segments lying on non-traversable areas. MTG¹ is our method without global information of other trajectories. With the traversability loss, the generated trajectories mostly lie on the traversable areas. Because our diversity loss only applies to the trajectories nearest to the most effective trajectories (closest to the ground truth trajectories) and drive redundant trajectories close to effective trajectories, this model has better performance in coverage but lower diversity than Dlow. The final model, MTG, implies the information of other trajectories in each decoder; with the references from other trajectories, the model is easier to train with better coverage.

2) *Qualitative Results*: In this section, we qualitatively show the generation performance of our approach and compare trajectory quality with the other approaches.

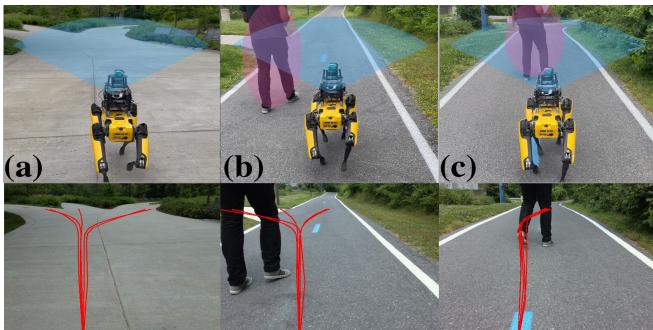


Fig. 4: Trajectory generation with different traversable areas; the blue area corresponds to the region visible to the robot, and the purple areas denote people.

As shown in Figure 4, the bottom rows of the figures are from the robot’s view and the top rows of the figures are captured from behind the robot. The generated trajectories are the red curves on the bottom row. Figure (a) to (c) shows that, with different perception occlusions, the trajectory generator is still able to cover all diverse directions the robot can

achieve on traversable areas. Figure (a) shows the generated trajectories covering diverse future directions the robot can achieve in traversable areas. Figure (b) shows the trajectory generation with partially occluded perception and (c) shows the trajectory generation when the only path in front of the robot is blocked. We can observe that the trajectories cover the traversable areas well and can also account for the influence of the dynamic obstacles.

The trajectories generated by different methods are compared in Figure 3. Each figure in Figure 3 has two vertical parts. The top of each figure is the robot view, where we transform the trajectories in the camera’s view. The bottom part is the local map; the white areas are the non-traversable areas, which are manually segmented in the local maps for evaluation. The purple and red curves are generated trajectories that start from the bottom center of each image.

As shown in Figure 3a, the trajectories generated by CVAE are mostly similar, and some of the trajectories lie on non-traversable areas. The Dlow method can generate very diverse trajectories, but, similar to CVAE and as shown in Figure 3b, the trajectories cannot avoid non-traversable areas well. Figure 3c shows the trajectories from the MTG method, which perform better in both at covering most of the traversable areas and not lying in the non-traversable areas.

V. CONCLUSION AND FUTURE WORK

We present MTG, a novel method in mapless trajectory generation for autonomous mobile robots. We introduce an innovative trajectory generation architecture that incorporates Conditional Variational Autoencoder (CVAE) with an attention mechanism. We also propose novel loss functions that account for traversability, diversity, and coverage constraints. We demonstrate superior performance in terms of coverage of traversable areas and feasibility of the trajectories compared to SOTA methods. This work can be used in future for extensive global planning tasks. This work also has some limitations. Current dataset only provides front view of the the Lidar and that makes the robo not able to generate trajectories behind it. The traversability map also requires much manual labor before training. We plan the address the issues in future work.

REFERENCES

- [1] S. Ganesan, S. K. Natarajan, and J. Srinivasan, “A global path planning algorithm for mobile robot in cluttered environments with an improved initial cost solution and convergence rate,” *Arabian Journal for Science and Engineering*, vol. 47, no. 3, pp. 3633–3647, 2022.
- [2] P. Gao, Z. Liu, Z. Wu, and D. Wang, “A global path planning algorithm for robots using reinforcement learning,” in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 1693–1698.
- [3] M. Psotka, F. Duchon, M. Roman, T. Michal, and D. Michal, “Global path planning method based on a modification of the wavefront algorithm for ground mobile robots,” *Robotics*, vol. 12, no. 1, p. 25, 2023.
- [4] T. Ort, L. Paull, and D. Rus, “Autonomous vehicle navigation in rural environments without detailed prior maps,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2040–2047.
- [5] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, “An efficient sampling-based method for online informative path planning in unknown environments,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.
- [6] I. Iluvia, E. Lazcano, and A. Ansuategi, “Active mapping and robot exploration: A survey,” *Sensors*, vol. 21, no. 7, p. 2445, 2021.
- [7] J. Liang, K. Weerakoon, T. Guan, N. Karapetyan, and D. Manocha, “Adaptiveon: Adaptive outdoor local navigation method for stable and reliable actions,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 648–655, 2022.
- [8] A. Gasparetto, P. Boscaroli, A. Lanzutti, and R. Vidoni, “Path planning and trajectory planning algorithms: A general overview,” *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, pp. 3–27, 2015.
- [9] U. Ozturk, M. Akdaug, and T. Ayabakan, “A review of path planning algorithms in maritime autonomous surface ships: Navigation safety perspective,” *Ocean Engineering*, vol. 251, p. 111010, 2022.
- [10] T. Guo and J. Yu, “Efficient heuristics for multi-robot path planning in crowded environments,” *arXiv preprint arXiv:2306.14409*, 2023.
- [11] M. Dobrevski and D. Skovcav, “Deep reinforcement learning for mapless goal-driven robot navigation,” *International Journal of Advanced Robotic Systems*, vol. 18, no. 1, p. 1729881421992621, 2021.
- [12] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Aghamohammadi, “Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation,” *arXiv preprint arXiv:2103.02828*, 2021.
- [13] P. Fankhauser, M. Bloesch, and M. Hutter, “Probabilistic terrain mapping for mobile robots with uncertain localization,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.
- [14] A. J. Sathyamoorthy, K. Weerakoon, T. Guan, J. Liang, and D. Manocha, “Terrapn: Unstructured terrain navigation using online self-supervised learning,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7197–7204.
- [15] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.
- [16] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” *Advances in neural information processing systems*, vol. 28, 2015.
- [17] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.
- [18] Y. Yuan and K. Kitani, “Dlow: Diversifying latent flows for diverse human motion prediction,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*. Springer, 2020, pp. 346–364.
- [19] Y. J. Ma, J. P. Inala, D. Jayaraman, and O. Bastani, “Likelihood-based diverse sampling for trajectory forecasting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 279–13 288.
- [20] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. Velasquez, V. A. Higuti, J. Rogers, H. Tran, and G. Chowdhary, “Wayfast: Navigation with predictive traversability in the field,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 651–10 658, 2022.
- [21] F. Jonas, M. Matias, C. Nived, C. Cesar, F. Maurice, and H. Marco, “Fast traversability estimation for wild visual navigation,” in *Proceedings of Robotics: Science and Systems*, 2023. [Online]. Available: <https://arxiv.org/pdf/2305.08510.pdf>
- [22] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.
- [23] D. Paz, H. Xiang, A. Liang, and H. I. Christensen, “Tridentnetv2: Lightweight graphical global plan representations for dynamic trajectory generation,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9265–9271.
- [24] D. Shah and S. Levine, “Viking: Vision-based kilometer-scale navigation with geographic hints,” *arXiv preprint arXiv:2202.11271*, 2022.
- [25] G. Kahn, P. Abbeel, and S. Levine, “Land: Learning to navigate from disengagements,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1872–1879, 2021.
- [26] W. Ding, J. Zhao, Y. Chu, H. Huang, T. Qin, C. Xu, Y. Guan, and Z. Gan, “Flowmap: Path generation for automated vehicles in open space using traffic flow,” in *2023 International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.
- [27] C. Schöller and A. Knoll, “Flomo: Tractable motion prediction with normalizing flows,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7977–7984.
- [28] A. J. Sathyamoorthy, K. Weerakoon, T. Guan, M. Russell, D. Conover, J. Pusey, and D. Manocha, “Vern: Vegetation-aware robot navigation in dense unstructured outdoor environments,” *arXiv preprint arXiv:2303.14502*, 2023.
- [29] K. Weerakoon, A. J. Sathyamoorthy, J. Liang, T. Guan, U. Patel, and D. Manocha, “Graspe: Graph based multimodal fusion for robot navigation in unstructured outdoor environments,” *arXiv preprint arXiv:2209.05722*, 2022.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [31] O. U. Aydin, A. A. Taha, A. Hilbert, A. A. Khalil, I. Galinovic, J. B. Fiebach, D. Frey, and V. I. Madai, “On the usage of average hausdorff distance for segmentation performance assessment: Hidden bias when used for ranking,” *arXiv preprint arXiv:2009.00215*, 2020.
- [32] J. Liang, U. Patel, A. J. Sathyamoorthy, and D. Manocha, “Crowdsteer: Realtime smooth and collision-free robot navigation in densely crowded scenarios trained using high-fidelity simulation,” in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 4221–4228.
- [33] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, “Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 345–11 352.
- [34] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, “Pointwise convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 984–993.

VI. APPENDIX

A. Real-world Experiment

For testing, a real-world experiment is done by combining MTG with a low-level AdaptiveON [7] motion planner for global navigation, as in Figure 1. We choose the generated trajectory with the last waypoint closest to the target as the following trajectory for the local planner. In each step, the robot takes the nearest next waypoint in the trajectory as the next goal. The model runs in the timestep around 0.01s on an onboard machine with an Intel i7 CPU and one Nvidia GTX 1080 GPU.

B. Analysis

Confidence of Trajectories: The trajectories are generated by the embedding Gaussian distributions, which are inputs of the decoder, and we train each distribution to cover each distribution to cover one traversable direction so the standard deviation can tell the confidence of the trajectory. When the standard deviation is large, the confidence is low because the distribution is not certain to the mean value. In our formulation, the variances for each trajectory can be calculated as $\mathbf{v}_c = \mathbf{A}\mathbf{v}\mathbf{A}^T$, where \mathbf{v} is the variance of the embedding. $\mathbf{v}_c = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, where \mathbf{v}_k and \mathbf{A} are defined in Section III-B. The generated trajectories are shown in Figure 5; the higher the variance, the lower the confidence and the darker the trajectory is. As the following figures show, when the trajectories are near the wall or other non-traversable areas, the confidence is lower. In addition, the last row shows when the person is very close to the robot. Although the model can still ignore the person, the confidence of the trajectories (passing through the person) gets lower.

Generalization: As shown in the above results, our model is well generalizable to the campus environment. However, for completely different scenarios like cities or woods, we don't expect our method to generalize very well to those out-of-distribution scenarios, which we believe is reasonable for most learning-based approaches. In addition to the out-of-distribution scenarios, because our model is not trained in the non-traversable areas, the model cannot perform well in fully non-traversable areas. In Figure 6, we demonstrate that when the robot operates within non-traversable zones, its trajectories exhibit significant randomness. However, as the robot exits these zones, its paths realign within the traversable areas.

C. Traversability Map

Before building 2D maps, we use Lio-sam [15] to build 3D maps by running robots on the campus. Then we remove the noisy points and calculate the surfaces of the area. Then large cliffs, bushes, and buildings are detected and marked as non-traversable areas. Next, we press the 3D maps into 2D maps, creating traversable maps with a resolution of 0.1m. Finally, we manually label some missing non-traversable areas by combining satellite maps.

D. Trajectory Selection

The trajectory length is empirically determined by our robot's speed and the perceptive range of the Lidar sensor. On the one hand, we would like to generate long trajectories to provide the robot with proper guidance for future directions. On the other hand, unnecessarily long trajectories may lead to occluded areas or out of the perception of the Lidar sensor. Our two robots drive around 1-2m/s, and 10-20 meters are enough for robots to drive for 10-20 seconds before the next trajectory generation. Therefore, we choose 15 meters as our trajectory length.

E. Qualitative Evaluation

Figure 7 shows the birds-eye-view of the generated trajectories (purple) and ground truth trajectories (yellow and generated by A* algorithm). Cyan represents Lidar points from the middle channel (8th channel) of a 16-channel Lidar scan for reference. The white areas are all non-traversable areas. We can see the purple trajectories generated by our model are very smooth and cover the traversable areas (black areas), similar to the yellow A* paths.

F. Architectural Details

Perception models in Figure 8: The output of PointCNN has dimension 512, and it concatenates with the velocity embeddings, with dimension 256 as input of the Encoder. The output of the encoder has dimension 512. The function $h_\psi(\mathbf{z})$ contains two linear layers, which output dimension 256. The $A(\cdot)$ and $b(\cdot)$ functions are all single linear layers, and the outputs have the dimension of (512 x trajectory number).

Self-Attention Model: As in Figure 9, the trajectory embedding keeps the dimension 512.

G. Dataset

The dataset is collected on a university campus. As shown in the satellite map 10, the blue trajectories are for the training dataset and the red trajectories are for the testing dataset.



Fig. 5: Trajectory Confidence: The top row shows the camera view of the generated trajectories and the bottom row shows the bird-eye-view of the trajectories. The Cyan color represents the obstacles detected from the middle channel of the 3D Lidar.



Fig. 6: The out-of-distribution cases: The top figure shows the robot fully in a non-traversable area with no traversable area around it. The bottom figure shows the robot leaving the non-traversable area, where the robot can still generate good trajectories.

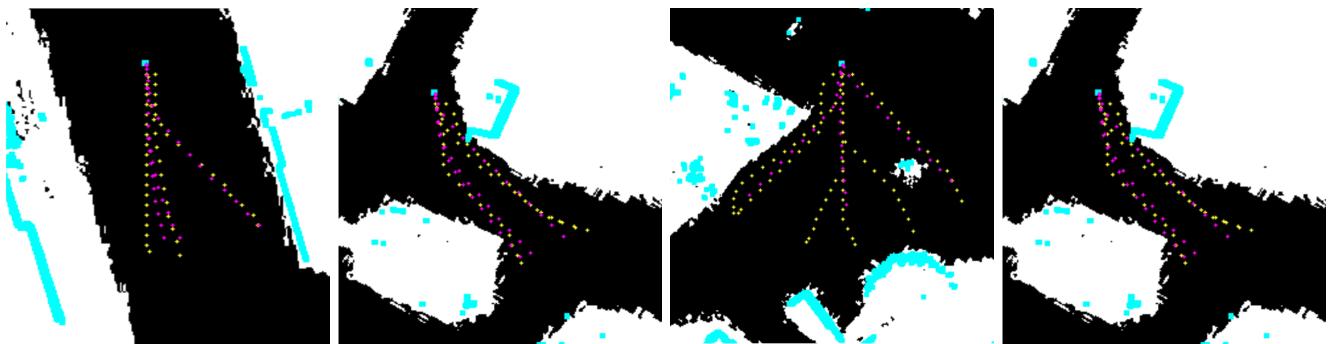


Fig. 7: Evaluation: A* ground truth paths are yellow trajectories and generated paths from MTG are purple trajectories. The white are non-traversable areas and cyan is the obstacle detected by the middle channel of the 16-channel Velodyne Lidar.

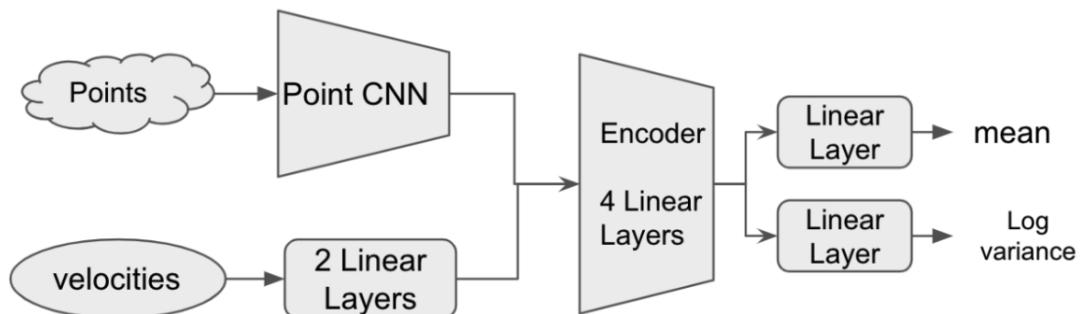


Fig. 8: Perception Models

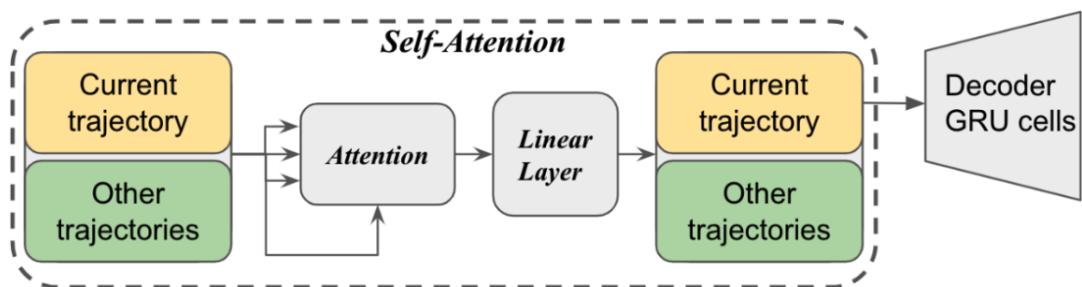


Fig. 9: Self-Attention Model

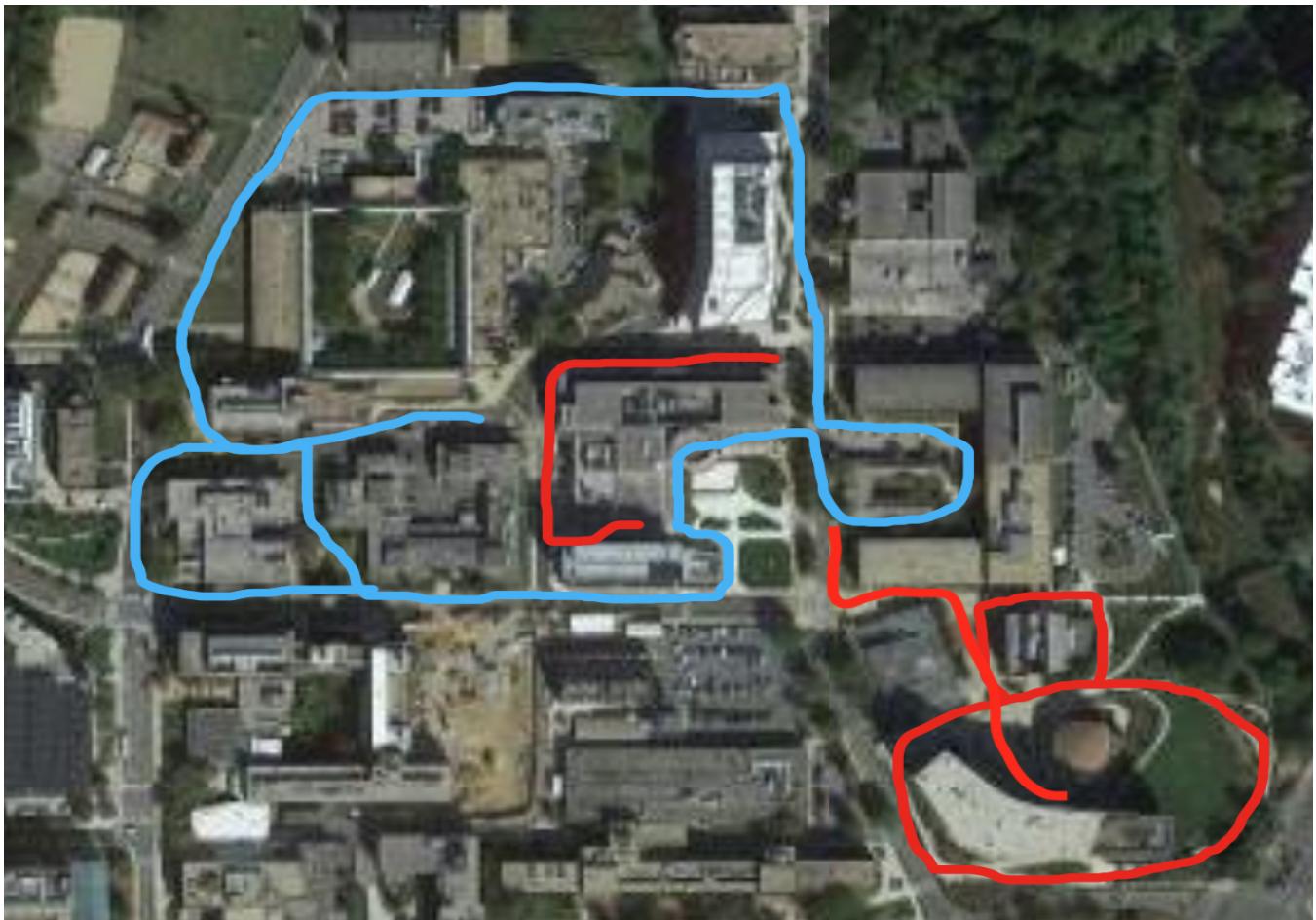


Fig. 10: The data is collected at a university campus. The blue trajectories are for training and red are for testing.