Monte Carlo Analysis of a Physics-Based Path Planner

Sarvesh Mayilvahanan
Department of Robotics, University of Michigan

I. Introduction

Path planners are a key component for autonomous vehicles, providing a set of way-points for the vehicle to follow. Ideal path planners provide a path avoiding obstacles based on the sensor data obtained by the vehicle. For off-road autonomous vehicles, this can include lidar data of nearby obstacles such as trees and bushes.

In this paper, a novel physics-based path planner that uses a combination of lidar data as well as the vehicle's current path as inputs is presented. Additionally, a method for generating random lidar data and paths to construct large training datasets is described.

II. Random Path Generation

Training neural networks typically requires large, diverse datasets, especially for complex tasks such as path planning. However, collecting enough data from real-world situations is very time consuming and often not realistically possible. Therefore, a method to generate random lidar data and random vehicle paths was developed so that diverse situations could be generated as training scenarios for the neural network.

A. Random Environments

Generating training scenarios begins with generating environments that the vehicle may encounter. Here, the goal is not to generate the lidar data itself, but rather the post-processed version that combines the lidar data gathered as a 2-dimensional cartesian histogram. The cartesian histogram starts as an empty map, and a random number of objects are successively placed into the map. These objects have a range of shapes and sizes and are representative of commonly occurring obstacles such as bushes and trees.

An example of these random environments is shown in Figure 1. We see that objects are placed in such a manner that they do not overlap and without intersecting the center of the map, which is the vehicle's location.

B. Random Paths

A wide variety of random paths must be generated because the path planner takes into consideration the vehicle's current path when producing an updated, safe path. These paths ideally cover a range of scenarios, including straight paths between the start and end points as well as extremely curved paths that may intersect objects.

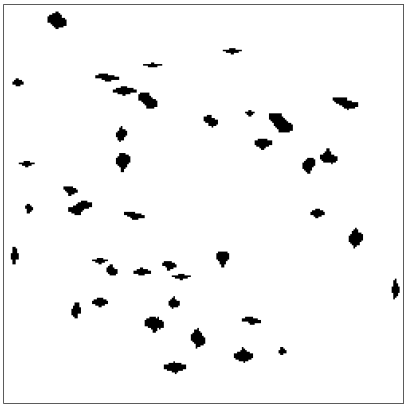

Fig. 1. Example of a randomly generated environment.

There are a number of random variables associated with the random path generation. Initially, a random endpoint must be chosen by sampling from a uniform distribution in the cartesian coordinates $U([-\frac{w}{2}, \frac{w}{2}] \times [-\frac{h}{2}, \frac{h}{2}])$ where w, h are the width and height of the cartesian histogram. Given the start (which is always $(0, 0)$ in the map frame) and end coordinates, a set of points representing the path is generated. This process begins by first inserting n linearly spaced "inflection" points between the start and end, where $n \sim U([0, 5])$. Each of the n inflection points are moved perpendicular the straight path between the start and end points by a distance $d \sim \mathcal{N}(0, \sigma_d^2)$. Using the set of inflection and start and end points, a cubic spline is fit and interpolated to provide n_p evenly spaced points. In the case of $n = 0$ inflection points, the path is simply linear.

An example of these randomly generated paths are shown in Figure 2. We see a wide variety in path lengths, curvature, and end points.

III. Physics-Based Path Planner

The local path planner presented here employs the concept of an adaptive path that changes it's shape based on applied virtual forces. These virtual forces are a function of the surrounding obstacles and the distance to those obstacles. For this implementation, the adaptive path is represented by a set of point masses connected in sequence by springs and dampers. This serves to ensure the path points move as a collective. Additionally, the path points are anchored to the original path position by another set of springs. This tethers the path points to the original path to ensure the resultant path has a similarity to the original path.

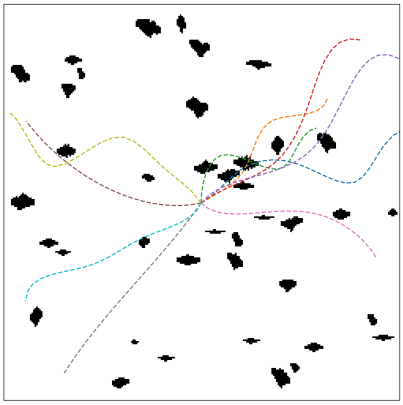

Fig. 2. Example of 10 randomly generated paths for a given environment.

The movement of the path points P_i can be calculated using an Euler approximation as follows

$$P_{i,t+\Delta t} = P_{i,t} + \Delta t \dot{P}_{i,t} \tag{1}$$

$$\dot{P}_{i,t+\Delta t} = \dot{P}_{i,t} + \Delta t \ddot{P}_{i,t} \tag{2}$$

$$\ddot{P}_{i,t+\Delta t} = -\frac{b}{m_i}\dot{P}_{i,t} + \frac{1}{m_i}\left(F_{p,i} + \sum_{r=0}^{\infty}\sum_{\theta=0}^{2\pi} F_{o,i}(r,\theta)\right) \tag{3}$$

Where b is the system damping constant and m_i is the path point mass. The acceleration of the points $\ddot{P}_i$ is affected by the damping as well as the path point force $F_{p,i}$ and the object force $F_{o,i}$.

A. Forces

There are two forces acting on the path points within the system: path point-based forces and object-based forces.

1) Path Force: The path force $F_{p,i}$ is the force acting on point P_i by the points P_{i-1} and P_{i+1} through the connecting springs. This force is calculated as follows:

$$\begin{aligned} F_{p,i} = {} & k_p(P_{i-1} - P_i - l_{i,i-1}) + k_p(P_{i+1} - P_i - l_{i,i+1}) \\ & + k_a(P_a - P_i - l_{i,a}) \end{aligned} \tag{4}$$

Where k_p is the spring constant between the path points and k_a is the spring constant from the anchor points. The spring length $l_{j,k}$ is the initial spring length between points P_j, P_k.

2) Object Force: The object force $F_{o,i}(r,\theta)$ is the force acting on the point P_i from an object at distance r and angle θ. This force is calculated according to the piece-wise function (5).

$$F_{o,i}(r,\theta) = \begin{cases} a_2 \cdot f(r,\theta) \cdot \frac{1}{r^n} & r \le a_1 \\ a_2 \cdot \exp\left(\frac{-(r-a_1)}{a_3}\right) \cdot f(r,\theta) \cdot \frac{1}{r^n} & r > a_1 \end{cases} \tag{5}$$

Where within a radius a_1, the object force is as defined, and beyond which there is an exponential decay in the force affecting by the tuning constant a_3. The object force is inversely proportional to the distance r between the path point and object to the power n. The force is also scaled by the tuning constant a_2. The function $f(r,\theta) = 0$ or 1 ensures that the force only acts on the point if there is an object at a distance r and angle θ.

The profile of this force as a function of the distance r can be seen in Figure 3.

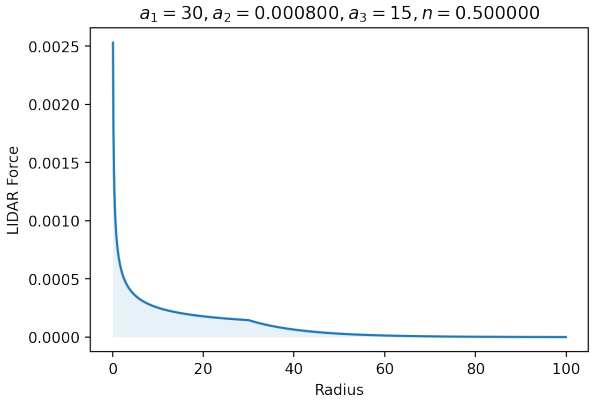

Fig. 3. Object force as a function of distance r for a specific set of tuning parameters.

The force is very high at close distances to the object and rapidly decreases due to the inverse proportionality to distance. After the cutoff distance a_1, the force decays and eventually reaches 0. This ensure that objects further away from the path point enact little to no force.

B. Resultant Path

The local path planner simulates the motion of the path points until a time t_f with a user-inputted time step Δt. If the system detects that the path points have reached a steady state prior to reaching t_f, it will automatically exit the simulation to decrease computational expense.

After reaching the final state of the path, a cubic spline is fit to the path points and is interpolated to produce a set of evenly spaced points, which is outputted from the planner.

IV. Monte Carlo Analysis of Path Generation

In order to ensure that the randomly generated paths adequately cover the distribution of paths desired for neural network training, a Monte Carlo analysis is performed. For a given number of samples N, a cartesian histogram is generated and N random paths are generated, which are the samples P. The evaluation of these samples is based on two metrics $m_A(p), m_\ell(p)$. These two metrics are defined as follows:

$$m_A(p) = \frac{A_{\text{path}}}{\ell_{\text{min}}} \tag{6}$$

$$m_\ell(p) = \frac{\ell_{\text{path}}}{\ell_{\text{min}}} \tag{7}$$

Where A_{path} is the total area between the path and a linear path between the start and end point. The metric m_A measures the curvature of the path and is normalized by the linear path length ℓ_{min}. The length metric m_ℓ measures the deviation from the linear path and is ratio of the random path length to the linear path length. When $m_\ell = 1$, the randomly generated path

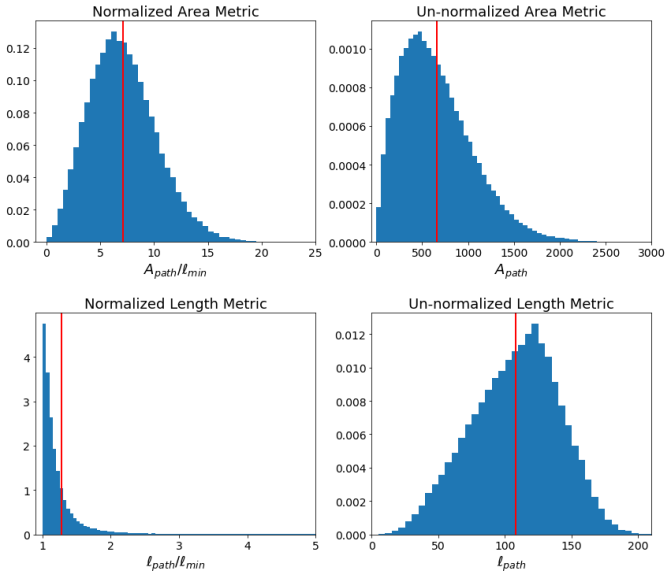

Fig. 5. Distributions of normalized and un-normalized metrics from 10^5 Monte Carlo samples. Monte Carlo estimates for each metric represented by vertical red line.

did not have any inflection and has the same properties as a straight line path.

Given the samples and the evaluation metrics, a simple Monte Carlo can be used to approximate the distribution of these metrics and their expected values.

$$\mathbb{E}[m_A] = \frac{1}{N}\sum_{i=1}^{N} m_A(P_i) \tag{8}$$

$$\mathbb{E}[m_\ell] = \frac{1}{N}\sum_{i=1}^{N} m_\ell(P_i) \tag{9}$$

For a Monte Carlo trial with 10^5 samples, the distributions of the two metrics are shown in Figure 5. The distribution of the metrics are provided both before and after normalization with respect to the straight line path length. The Monte Carlo estimates based on the samples are represented by the vertical red line. The distribution of the metric m_A appears to be Gaussian with a small tail to the right as the area A_{path} cannot be less than 0. This can be attributed to the inflection distances d being sampled from a Gaussian. The length metric m_ℓ approximately follows a Pareto distribution with $\alpha = 4.41$ where the probability density function is defined as follows:

$$f_{m_\ell}(x) = \begin{cases} \frac{\alpha}{x^{\alpha+1}} & x \geq 1 \\ 0 & x < 1 \end{cases} \tag{10}$$

The shift in the distributions of the metrics from normalization can be explained due to the inflection points and inflection lengths having more significant impact on the path length and deviation from the straight line path.

Running 10^2 Monte Carlo trials with 10^3 samples each, the convergence of the Monte Carlo estimate for each metric is observed as a function of the number of trials in Figure 4. Both metrics reasonably converge within 100 samples with low variance. Because the Monte Carlo estimate is unbiased, the distribution of estimates at each number of samples is centered about the true mean.

From the Monte Carlo analysis, the expectations of the metrics are $\mathbb{E}[m_A] = 7.11$ and $\mathbb{E}[m_\ell] = 1.28$, meaning on average, the ratio between the area between the randomly generated paths and a straight line and the straight line path length is 7.11. Additionally, the expectation of the ratio between the randomly generated path lengths and the straight line path lengths is 1.28.

A. Analysis

Based on the distribution of metrics from the Monte Carlo samples and the expectations of the metrics, the random path generator sufficiently covers a wide range of situations that a vehicle may encounter and provides the ability to develop a large, diverse training dataset for a neural network. The range of path lengths shown in Figure 5 covers paths with a variety of end points and curvature. The normalized length metric m_ℓ shows that the majority of paths have a small amount of curvature $1 \leq m_\ell < 1.5$, with a small amount of samples having large curvature and deviation from the straight line path $1.5 < m_\ell$. This accurately reflects the targeted diversity of paths for dataset generation.

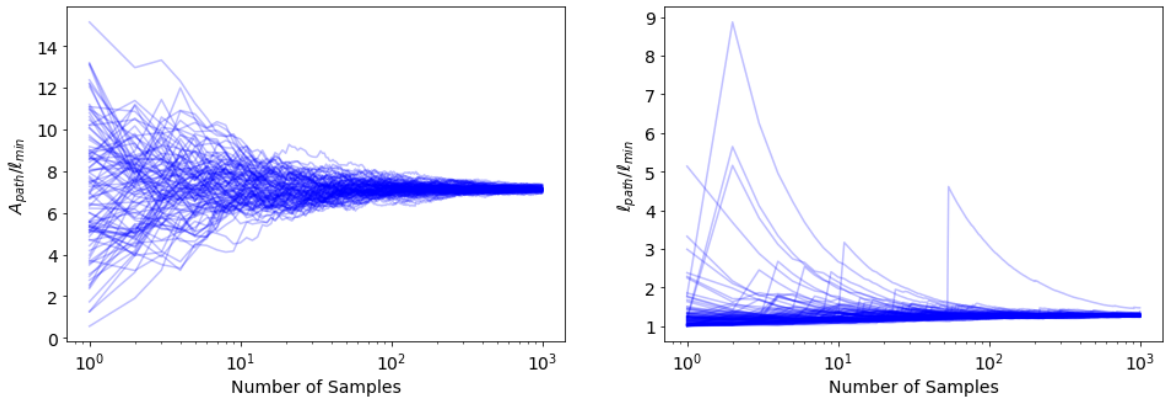

Fig. 4. Convergence of 100 Monte Carlo estimates of path metrics as a function of number of samples.

V. Randomized MLMC for Path Planner

In order to estimate the outputted path of the local path planner described above, a Monte Carlo estimate for a path is desired. Here, the original path P_0 is known, but with some defined uncertainty, which takes the form of a noise η. The Monte Carlo samples are therefore defined as

$$P^{(i)} = P_0 + \eta \qquad \eta \sim \mathcal{N}(0, \sigma^2) \tag{11}$$

However, the path planner is computationally expensive and evaluating a large number of Monte Carlo samples to obtain an estimate with reasonable variance would not be feasible. Therefore, multilevel Monte Carlo (MLMC) is employed to take advantage of lower fidelity versions of the planner with large Δt to obtain an estimate with low variance and lower cost.

Multilevel Monte Carlo uses an ensemble of models with an output random variance $Z_0, Z_1, \dots Z_L$, which are ordered according to computational cost such that $C_i < C_j$ if $i < j$ where C_i is the cost of the model i. Z_L is the output of the highest fidelity model, which is the estimate of interest. This estimate can be written as the sum of the lowest fidelity estimate Z_0 and the difference between successive estimates as shown in Equation (13).

$$Z_L = Z_0 + \sum_{i=1}^{L} Z_i - Z_{i-1} \tag{12}$$

Taking the expectation of the random variable Z_L, we find that estimator is unbiased.

$$\mathbb{E}[Z_L] = \mathbb{E}[Z_0] + \sum_{i=1}^{L} \mathbb{E}[Z_i - Z_{i-1}] \tag{13}$$

The standard MLMC estimator can therefore be defined as

$$\hat{Y} = \sum_{\ell=0}^{L} \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} Y_\ell^{(i)} \tag{14}$$

Where the estimator for each level ℓ is defined as $Y_\ell^{(i)} = Z_\ell^{(i)} - Z_{\ell-1}^{(i)}$ for $\ell \geq 1$ and $Y_0^{(i)} = Z_0^{(i)}$ for $\ell = 0$.

A. rMLMC

Randomized multilevel Monte Carlo (rMLMC) is an extension of the standard MLMC method presented above, initially introduced by [1]. Instead of using the cost and estimate variance of each level of the MLMC estimator to find an optimal sample allocation for each level for a given target variance, rMLMC only considers N total samples, where for each sample, there is a probability p_ℓ that the sample is evaluated using level ℓ.

The estimator is defined as

$$\hat{Y} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{p_{\ell^{(i)}}} Y_{\ell^{(i)}}^{(i)} \tag{15}$$

where the level for each sample $\ell^{(n)}$ is selected randomly with given probabilities for each level.

The estimator in Equation (15) can be re-expressed to follow the format of the standard MLMC estimator in Equation (14).

$$Y = \sum_{\ell=0}^{L} \left(\frac{1}{N_\ell} \sum_{i=1}^{N_\ell} Y_{\ell^{(i)}}^{(i)} \right) \tag{16}$$

The choice of probability for each level p_ℓ is critical in defining the performance of the rMLMC estimator. According to [2], the optimal choice for p_ℓ is

$$p_\ell = \sqrt{V_\ell / C_\ell} \left(\sum_{\ell'=0}^{L} \sqrt{V_{\ell'} / C_{\ell'}} \right)^{-1} \tag{17}$$

where V_ℓ and C_ℓ are the variance and cost of each level respectively.

Additionally, for a given target variance ε^2, the optimal number of samples N_ε is

$$N \approx \varepsilon^{-2} \sum_{\ell=0}^{L} V_\ell / p_\ell \approx \varepsilon^{-2} \left(\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell} \right) \left(\sum_{\ell'=0}^{L} \sqrt{V_{\ell'} / C_{\ell'}} \right) \tag{18}$$

and the total cost becomes

$$C_{rMLMC} = N \sum_{\ell=0}^{L} p_\ell C_\ell \approx \varepsilon^{-2} \left(\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell} \right)^2 \tag{19}$$

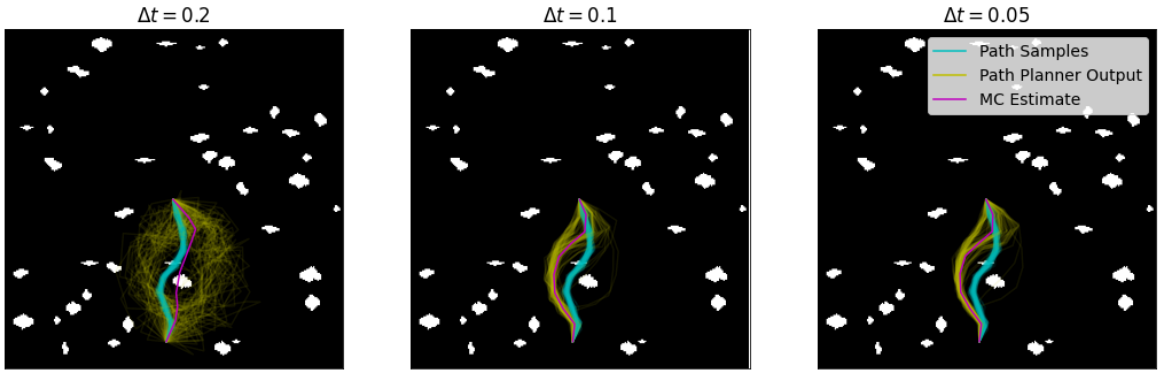

Fig. 6. Comparison of Monte Carlo samples and estimates of different fidelity path planners.

B. Implementation

For this implementation, the varying fidelity models were chosen as $\Delta t = 0.1, 0.05, 0.02, 0.01$ because models with lower fidelity ($\Delta t > 0.1$) did not produce consistent results. The cost of each model is directly proportional to the timestep Δt, and therefore, the costs of each level are $1, 3, 7, 15$ respectively, where the model with $\Delta t = 0.1$ is considered to have a cost of 1.

The Monte Carlo samples and estimates of the path planner with select Δt values are shown in Figure 6.

By initially using 20 samples for each level, the variances of the levels were found to be $3.24, 2.61, 2.09, 0.85$ for each level where the lowest fidelity level has the highest variance, and the following delta levels $\ell \geq 1$ have progressively lower variance due to the models being more accurate.

Given the variances and costs of each level, the optimal probabilities p_ℓ for each level are found according to (17) to be $0.51, 0.27, 0.16, 0.07$ for each level. For a target variance of $\varepsilon^2 = 0.25$, the optimal number of samples was found to be 168, resulting in a cost of 575.7.

For each sample, a random variable $u^{(i)} \sim U(0, 1)$ was generated and based on the probabilities p_ℓ a level ℓ was chosen for the sample $P^{(i)}$.

The resulting estimate is shown in Figure 7 where the rMLMC estimate is shown in magenta and the original path is shown in cyan. Clearly, the rMLMC estimate is able to estimate the path planner's output while meeting the target variance and taking advantage of the lower fidelity models.

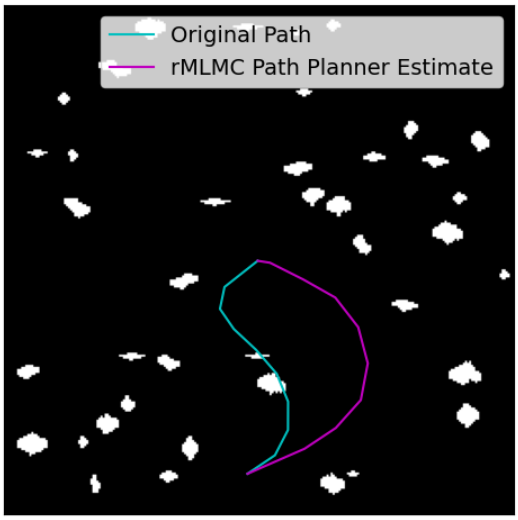

Fig. 7. Resulting rMLMC estimate for one particular original path.

VI. Conclusion and Discussion

In this paper, a method for random path generation and physics-based local path planning were presented. The random path generator was validated by conducting a Monte Carlo analysis of the paths using several metrics that measure the curvature and other distinct features of the paths. This ultimately concluded that a sufficiently large range of paths was being generated, with the distribution of paths being centered about paths with a small amount of curvature and deviation from the straight line path. Additionally, the randomized multi-level Monte Carlo method was presented and implemented for the physics-based local path planner and was used to estimate the output of the high fidelity model for a given original path. The rMLMC analysis found that given varying fidelity models, the output of the highest fidelity model can be approximated with a specified target variance with a lower cost than simply performing Monte Carlo with the high fidelity model.

References

[1] Rhee, Chang-han, and Peter W. Glynn. "Unbiased estimation with square root convergence for SDE models." Operations Research 63.5 (2015): 1026-1043.

[2] Giles, Michael B. "Multilevel monte carlo methods." Acta numerica 24 (2015): 259-328.