Received 1 January 2024, accepted 21 January 2024, date of publication 2 February 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3361478

RESEARCH ARTICLE

Ricci Planner: Zero-Shot Transfer for Goal-Conditioned Reinforcement Learning via Geometric Flow

WONGEUN SONG AND JUNGWOO LEE, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea

Corresponding author: Jungwoo Lee (junglee@snu.ac.kr)

ABSTRACT The long-horizon problem has been a persistent challenge in the field of reinforcement learning, leading to the exploration of various solutions. Planning methods have emerged as a prominent approach, generating intermediate plans from the current state to the goal state. However, these methods often rely on the assumption of additional interaction with the environment or the availability of offline data. However, in real-world scenarios, instead of such information, only time-independent random observations of the environment can be provided. To overcome this limitation, we propose the Ricci planner, a novel algorithm capable of generating a plan from the current location to a desired goal using only a limited number of time-independent random samples from the observation space. We drew inspiration from the observation that the most efficient path is one with the minimum length and, based on this, we transformed the problem of finding an efficient path into a shortest path-finding problem. Subsequently, we formulated this as an optimization problem on a path space. However, the length functional in the path space is highly non-convex and multi-modal, which results in numerous suboptima, and makes the problem exceedingly challenging to solve. To address this issue, we employ the Ricci flow to continuously transform the target manifold into a simpler manifold. Initially, we identify the shortest path on the simpler manifold and subsequently convert it to the shortest path on the desired manifold by applying the inverse process of the Ricci flow. We conduct a experimental comparison with graph-based shortest path finding methods. We assessed both the quality of the generated plan itself and its effectiveness when applied to an agent and observed improved results from both perspectives compared to the baseline.

INDEX TERMS Deep learning, goal-conditioned reinforcement learning, planning.

I. INTRODUCTION

Reinforcement learning (RL) has gained significant attention in the field of machine learning in the past decade. Numerous studies have explored a wide range of tasks, ranging from simple Atari games [20], [39] to complex manipulation tasks [2], [18], demonstrating human-level or higher performance. Despite the extensive research efforts to date, RL methods are seldom used in practice. This can be attributed to the challenges posed by long-horizon and sparse-reward tasks encountered in the real world [26], [36], which makes it difficut to apply RL to real-life tasks. To address these challenges, various approaches have been proposed. One such approach is planning [4], [14], which involves creating a long-term plan prior to solving the task. Additionally, hierarchical reinforcement learning (HRL) [28], [34] has been suggested as a method for performing long-term tasks by combining low-level policies, which can only perform short-term tasks, with high-level policies. Another area of research concerning environments with sparse rewards involves the modification of goals used during interaction to goals achieved [3], [10], and the creation of a denser reward signal [7], [31], as well as the utilization of offline data [19], [24], [29] obtained from previous interactions for learning.

However, existing methods has assumed that the target environment is accessible for sufficient interaction [22], [37] or that offline data [19], [24], [29] obtained through past

The associate editor coordinating the review of this manuscript and approving it for publication was Abderrahmane Lakas.

interaction is available. Yet, in practical scenarios where reinforcement learning agents are used by general users, there may be limited opportunities for additional interaction or access to offline data. For instance, consider the use of a factory-manufactured robot as a household maid robot, which is the most common scenario for the use of RL technology by general users. In such cases, the robot must be able to navigate within a new indoor environment and perform tasks specific to that environment. However, it may be impractical for the robot to learn indoor navigation tasks by exploring the house or obtaining offline data through prior interactions. Instead, the observation data of the indoor environment or human demonstration data (not involving robot manipulation) may be available, and thus the RL agent should have the ability to execute the given task solely with this data.

To achieve this, one could consider employing a method that involves constructing a graph using L2 distance in the observation space and generating a plan using a shortest path algorithm to execute the given task. However, building the graph using L2 distance requires careful determination of a pruning threshold. If the threshold is too small, excessive pruning of edges can occur, resulting in isolated points or disconnected regions in the graph, even though they are actually connected. This can lead to problems where viable paths are not found even though there are accessible paths, or longer routes are taken instead of shorter ones. On the other hand, if the threshold is too large, insufficient pruning of edges can occur, resulting in connections between states that are not adjacent. In this case, the generation of unusable plan which pass through infeasible areas can occur. Therefore, it can be easily anticipated that determining an appropriate threshold level has a significant impact on performance. However, since we assume that no information about the environment is given, it is not an easy task to determine such a threshold based solely on data points. Additionally, the method of building the graph requires memory corresponding to the square of the given data points in order to store information about the edges. For example, if there are 1000 data points, this would require a quantity of 1,000,000, which may not be feasible for an embedded computing device used in a robot. Novel algorithm called the Ricci planner, which generates a plan from the current state to a goal state by utilizing only a limited number of observation data of an environment, without requiring additional interaction or offline data. With the use of the Ricci planner, even agents with basic skills or those capable of only short-term navigation can acquire the ability to perform a new task in a new environment by following the generated plan. Furthermore, the Ricci planner does not require precise determination of thresholds or space that is proportional to the square of the number of data points. The Ricci planner estimates the structure of the manifold to which the given observation data belongs, instead of building a graph based on the observation data. The problem of finding the shortest path between two points is formulated as an optimization problem, based on the fact that the shortest path is the element in the path space that minimizes a length functional. We apply gradient-based optimization method to obtain the desired plan.

However, this optimization problem exhibits highly non-convex and multimodal characteristics. Therefore, directly applying gradient-based optimization methods to this problem can result in suboptimal or infeasible plan. To address these issues, we applied a transformation process called the Ricci flow, which has been consistently used in the field of machine learning ([8], [13], [25], [48], [49]). In this work, we utilized the Ricci flow to transform the target manifold into a simpler manifold. We first get the shortest path from the simpler manifold and apply the inverse process of the Ricci flow to obtain the path from target manifold. The overall process of the Ricci planner can be observed through Figure 1. Moreover, we leverage the relationship between the Ricci flow and the diffusion process to propose an algorithm that finds the shortest paths by utilizing a time-dependent unnormalized density function. We experimentally evaluate the quality of the sampled paths using various real-world datasets and demonstrate performance improvements when our approach is applied to zero-shot reinforcement learning.

In summary, our contributions are as follows.

- We propose a method to estimate the structure of manifold given sample data points from the manifold.
- We propose an algorithm called the Ricci planner, which samples a feasible and efficient plan between the current location and the target location.
- We analyze the quality of the paths generated by the Ricci planner using qualitative and quantitative methods, and we demonstrate performance improvements when our approach is applied to zero-shot reinforcement learning.

In Section II, we introduce relevant research related to our work, including planning methods and goal-conditioned reinforcement learning. In Section III, we briefly outline the theoretical foundations of our proposed method, covering topics such as Riemannian manifolds and functional derivatives. Section IV presents an overview of our proposed algorithm and provides detailed explanations. In Section V, we experimentally validate the superiority of our approach, and in Section VI, we discuss future work and conclude the paper.

II. RELATED WORKS

A. PLANNING AND HIERARCHICAL REINFORCEMENT LEARNING

One of the fundamental challenges addressed in modern reinforcement learning literature is the long horizon problem. The long horizon problem refers to tasks that require a substantial number of interactions with the environment to be completed. Using vanilla reinforcement learning algorithms to learn such tasks might require an extensive number of interactions or might lead to the failure of training. Numerous techniques have been proposed to deal with this long horizon

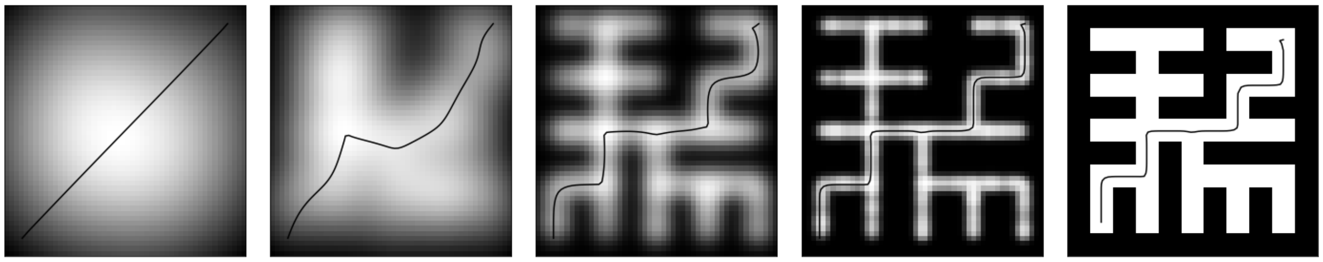

FIGURE 1. **Visualization of the path generation process in Ricci planner on a randomly generated maze. The left four figures illustrate the progressive transformation from a simple manifold to the target manifold, depicting the shortest distance on each manifold. The rightmost figure displays the actual maze environment and the final resulting path.**

problem. One such technique is model-based planning [4], [14], which involves learning a world model using additional models that can be employed for planning. Another technique is hierarchical reinforcement learning [28], [34], which involves learning low-level policies that operate only in short horizons and high-level policies that leverage low-level policies to solve long horizon tasks.

B. GOAL-CONDITIONED REINFORCEMENT LEARNING

Sparse reward signals are an important challenge in the field of reinforcement learning [33], [46]. Prior research tended to overlook this issue by formulating the reinforcement learning problem simply as a Markov decision process (MDP) [23], [43], [44]. However, MDP assumes that a continuous reward signal can be observed at each time step, providing feedback on the agent's actions. This is often not the case in real-world scenarios, where agents typically only receive a binary evaluation indicating success or failure after completing the task. Consequently, there is a need for novel formulations and methods that account for such characteristics of environments. To address this challenge, a new formulation has been proposed, called the goal augmented Markov decision process (GA-MDP) [1], and active research has been conducted under the name of goal conditioned reinforcement learning (GCRL) [1], [15], [32] in the literature.

Efforts have been undertaken to tackle the challenge of the long horizon problem in GA-MDP situations. However, GA-MDPs with long horizons create even sparser rewards, which make it difficult to directly apply existing methods due to the sparsity of the reward signal. Therefore, further research has been conducted to address this issue. Another approach proposed is graph-based planning [17], [21], [50], which is developed based on the intuition that there exist common waypoints that must be traversed in order to reach distant goals. In this approach, the first step involves identifying waypoints, referred to as landmarks. These landmarks are then represented as vertices, and a graph is constructed by using a value function to define the estimated distances between each pair of landmarks as edges. Subsequently, planning is performed using the constructed graph to guide the agent towards the goal.

C. ZERO-SHOT TRANSFER FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

To enable the deployment of a reinforcement learning (RL) agent in real-world applications, it is crucial for the agent to be able to learn and perform new tasks without interaction and offline data. Despite advancements in algorithms that can adapt to zero-shot scenarios [27], their evaluation has been confined to short horizon tasks such as moving objects to predetermined positions on an obstacle-free flat desk. Additionally, while the Ricci planner only necessitates a small number of random samples in the goal space, prior approaches require both random samples in the goal space and transition information between states to learn.

D. DIFFUSION MODEL

The sampling process in Energy based model [41] relies heavily on the accuracy of the score function estimate, which is the gradient of the energy function [42]. However, it has been observed that the score function estimate is often incorrect in low-density regions [40]. To address this issue, a sampling method called noise conditioned score network (NCSN) [40] has been proposed. NCSN obtains samples from the target distribution by first generating samples from a simple noisy distribution and then iteratively removing noise. In parallel with this work, a similar framework called the deep diffusion probabilistic model (DDPM) [16] has been proposed, and subsequent studies have interpreted NCSN and DDPM as a unified framework by utilizing mathematical tools [42] from stochastic differential equations (SDE). The concept of iteratively transforming samples from a simple distribution to obtain samples from a target distribution is similar to the framework employed by the Ricci planner.

However, significant distinctions exist between the two approaches. In terms of the inference process, the diffusion model is designed to sample data points from a target distribution and modify the sampling target using SDE. Conversely, in the Ricci planner, the emphasis lies in sampling paths that connect given points, rather than individual points, and transforming the path through the application of geometric flow. Regarding the training process, in the case of the diffusion model, data samples from the target distribution are provided. In contrast, for the Ricci planner, it is impractical to obtain data samples directly from the target distribution.

III. PRELIMINARY

Due to the limitations of space, it is beyond the scope of this section to comprehensively cover the topics of stochastic calculus, differential geometry and functional analysis with full mathematical rigor. Thus, we aim to offer a more intuitive explanation to facilitate readers' comprehension.

A. FOKKER-PLANCK EQUATION

A stochastic differential equation (SDE) is an equation that describes the infinitesimal change, denoted as dX_t, in a stochastic process X_t using the infinitesimal changes of generally well-known stochastic processes, such as Brownian motion. SDE is widely applied in modeling natural phenomena, such as the movement of particles in fluids, as well as in modeling social scientific phenomena, such as stock market dynamics. In SDE, the infinitesimal change dX_t is typically expressed as the sum of a deterministic component and a stochastic component as

$$dX_t = \mu(X_t, t)dt + D(X_t, t)dW_t. \tag{1}$$

Here, W_t represents Brownian motion, $\mu(X_t, t)$ is referred to as the drift, and $D(X_t, t)$ is the diffusion coefficient. Furthermore, the change in the distribution, denoted as $p(x, t)$, of the solution X_t over time t for the given SDE is expressed as

$$\frac{\partial}{\partial t}p(x,t) = -\frac{\partial}{\partial x}[\mu(x,t)p(x,t)] + \frac{\partial^2}{\partial x^2}[D(x,t)p(x,t)]. \tag{2}$$

This equation is commonly known as the Fokker-Planck equation.

B. METRIC TENSOR FIELD AND COMFORMALLY FLAT MANIFOLD

To begin with, a manifold is defined as a set that is locally homeomorphic to Euclidean space. This means that for any point on the manifold, its very small neighborhood can be viewed as a subset of Euclidean space. This implies that most of the geometric objects that we are familiar with, such as the plane, torus, and sphere, fall into this category.

A metric tensor field [12] on a manifold provides a way to measure the length of curves on the manifold. In Euclidean space, we can define a distance metric $d(x, y) = \sqrt{(x-y)^T A(x-y)}$, where A is a positive definite matrix defined using a basis for $\mathbb{R}^n$. Additionally, the length $L(x)$ of any differentiable curve $x : [a, b] \to \mathbb{R}^n$ can be defined as $L(x) = \int_a^b \sqrt{\dot{x}(t)^T A\dot{x}(t)}dt$, where $\dot{x}(t) := \frac{dx}{dt}$. However, for a general manifold, it is challenging to define a unified basis and, consequently, to define a metric or length of a curve using a single matrix.

However, as each point on a manifold has a local neighborhood that can be viewed as a subset of Euclidean space, for each point, a local basis can be defined, allowing the definition of a matrix-valued function $g(x)$ that provides a local metric on the neighborhood of each point on the manifold $\mathcal{M}$. The length of a curve $x : [a, b] \to \mathcal{M}$ on this manifold can be defined as

$$L(x) = \int_a^b \sqrt{\dot{x}(t)^T g(x)\dot{x}(t)}dt. \tag{3}$$

If this matrix can be expressed as a scaling of a matrix tensor η from some flat manifold, such as Euclidean space, using a smooth function $\lambda(x)^2$, the manifold is said to be conformally flat. The scaling factor $\lambda(x)$ used in this definition is called the conformal factor. In this work, we use the term "conformally flat" to mean $\eta = I$, so we have $g(x) = (\lambda(x))^2 I$.

C. GEOMETRIC FLOW AND RICCI FLOW

Geometric flow [6] is a fundamental mathematical tool employed in Riemannian geometry to model the time evolution of manifolds. It involves a partial differential equation that describes the time variation of the metric tensor $g_t(x)$ of the manifold. Among the most prominent geometric flows, the Ricci flow has been an invaluable tool in the study of geometry and topology since its introduction by Richard Hamilton in the 1980s [5]. After Grigori Perelman used the Ricci flow to prove the Poincaré conjecture in 2003 [35], its importance became even more evident. The mathematical expression for the Ricci flow is given by

$$\frac{\partial g_t(x)}{\partial t} = 2\mathrm{Ric}^g, \tag{4}$$

where Ric^g is called Ricci curvature which is a measure of how much a given point is curved or twisted. By changing the manifold in the opposite direction of this measure, Eq 10 transforms the manifold into a simpler shape, thereby reducing its irregularities. We have leveraged some of the key properties of the Ricci flow, which has been established as a fundamental tool in the development of the Ricci planner.

D. GEODESIC AND FUNCTIONAL DERIVATIVE

In Euclidean space, the shortest path between two points is a straight line. However, in a Riemannian manifold, a straight line may not be well-defined, and even if it is, it may not be the shortest path. Thus, a new definition of a straight line is required for manifolds, which is called a geodesic. A geodesic is the shortest path among all paths connecting two points.

The functional gradient is a generalization of the gradient to a function space. For a given path x, if the functional (a function that takes a function as input) is represented as $J(x) = \int L(x, \dot{x}, t)dt$, the functional derivative [11] of the functional is given by

$$\frac{\delta J}{\delta x} = \frac{\partial L}{\partial x} - \frac{d}{dt}\frac{\partial L}{\partial \dot{x}}. \tag{5}$$

Geodesics are minimizers of the length functional in path space. Therefore, if the metric tensor is given in an analytic form, the closed-form expression of the geodesic can be obtained from the condition that the functional derivative is zero. However, since the metric tensor is not given in an analytic form, in our case, we propose a numerical method to obtain the geodesic.

Algorithm 1 Ricci Planner

```
Input: current state $s$, goal state $g$, sample points $\mathcal{D}$
Output: sequence of subgoal $\{p_i\}_{i\in[N]}$
Initialize:
for $i = 0, 1, \dots, N$ do
    $p_i \leftarrow (1 - \frac{i}{N})s + \frac{i}{N}g$
Update:
for $t = T, T-1, \dots, 0$ do
    $\tilde{p}_t(\cdot) \leftarrow \mathcal{P}_t(\cdot; \mathcal{D})$          ◁ density estimation
    Set $\lambda_t(\cdot)$ by substituting $\tilde{p}_t(\cdot)$ into (7)
    for $j = 1, 2, \dots, S$ do
        Calculate $\{\frac{\delta L}{\delta x}(\frac{i}{N})\}_{i\in[N]}$ using (6 and 16)
        $\{p_i\}_{i\in[N]} \leftarrow \{p_i - \alpha\frac{\delta L}{\delta x}(\frac{i}{N})\}_{i\in[N]}$
        $\{p_i\}_{i\in[N]} \leftarrow$ Make-Regular$(\{p_i\})_{i\in[N]})$
return $\{p_i\}_{i\in[N]}$
```

Algorithm 2 Make Regular

```
Input: point sequence $\{p_i\}_{i\in[N]}$
Output: regularized point sequence
for $k = 1, 2, \dots, K$ do
    $l_0 \leftarrow 0, L_0 \leftarrow 0$
    for $i = 1, \dots, N$ do
        $l_i \leftarrow ||p_i^{(k)} - p_{i-1}^{(k)}||$
        $L_i \leftarrow L_{i-1} + l_i$
    for $i = 1, 2, \dots, N-1$ do
        $\iota \leftarrow \sum_{j=1}^{N} \mathbb{I}(\frac{L_j}{L_N} \leq \frac{i}{N})$
        $\tau \leftarrow \frac{L_N \times \frac{i}{N} - L_\iota}{l_{\iota+1}}$
        $p_i^{(k+1)} \leftarrow \tau \times p_\iota^{(k)} + (1-\tau) \times p_{\iota+1}^{(k)}$
    $p_0^{(k+1)} \leftarrow p_0^{(k)}, p_N^{(k+1)} \leftarrow p_N^{(k)}$
return $\{p_i^{(K)}\}_{i\in[N]}$
```

We apply Eq 5 to calculate the functional derivative of the length functional. When determining the geodesic, the functional corresponds to the length. For a conformally flat manifold, the length functional can be obtained through Eq 3. Additionally, the functional derivative can be computed using Eq 5. Therefore, upon substitution, the calculation can be performed as

$$\frac{\delta J}{\delta x}(t) = \nabla\lambda(x(t))\|\dot{x}(t)\| - \frac{\langle\nabla\lambda(x(t)), \dot{x}(t)\rangle\,\dot{x}(t)}{\|\dot{x}(t)\|} - \frac{\lambda(x(t))\ddot{x}(t)}{\|\dot{x}(t)\|} + \frac{\lambda(x(t))\dot{x}(t)\,\langle\dot{x}(t), \ddot{x}(t)\rangle}{\|\dot{x}(t)\|^3}, \tag{6}$$

where $\lambda(\cdot)$ is a conformal factor of the manifold.

IV. METHODOLOGY

In this section, we elucidate our newly proposed algorithm, the Ricci Planner. Firstly, in the initial subsection, we provide an overview of the problem scenario, along with the assumptions and notations employed herein. Following that, in the subsequent subsection, we expound on how the planning problem is formulated as an optimization problem. In the next section, we delve into the challenges posed by this optimization problem. Subsequently, we detail how the Ricci flow is employed to alleviate these challenges. Lastly, in the concluding section, we provide an in-depth explanation of the overall framework and the proposed algorithm.

A. PROBLEM SETTING

Our objective is to generate a feasible path from the starting point to the endpoint without any dependency on a specific application. We do not make any assumptions about the agent or environment dynamics, but instead rely on reasonable assumptions regarding the structure of the manifold that the given data belongs to and the way it is generated. We assume that the observation space of an agent is a manifold with a finite metric tensor field for all $x \in \mathbb{R}^n$. Although this may seem like a restrictive assumption, it covers most connected subsets of $\mathbb{R}^n$, including those where the manifold is a low-dimensional embedded set. The areas $M \subset \mathbb{R}^n$ that are included in the manifold are handled by assigning the desired metric tensor to them, and by assigning a high metric tensor to the areas $M^C \subset \mathbb{R}^n$ that are not included in the manifold. This ensures that any geodesic connecting two points p and q in M does not pass through M^C. Consequently, the inclusion or exclusion of M^C has no impact on the form of the geodesic if the metric tensor values in M are the same as those in the original manifold. To ensure that every geodesic on the manifold is a differentiable path, we assume that the metric tensor field is differentiable at every point x.

Regarding the data generating process, we make the assumption that the observed data is generated by observing the positions of randomly moving particles on the manifold at random time instances. This assumption encompasses a wide range of scenarios, such as the cases where the input data consists of photographic images of the environment, in which case the particle can be thought of as a human and the random time instance corresponds to the moment when the photograph was taken. If the input data corresponds to the trajectory of another agent, then the particle would correspond to the agent, and the random time instance would correspond to a sequence of fixed intervals. Additionally, we assume that the agent moves randomly on the manifold while preferring paths with lower cost.

Given the Riemann metric tensor field, we can obtain almost all the information about the a Riemannian manifold, including its curvature. Therefore, we first estimate the Riemann metric tensor field using the given distribution. The problem scenario assumes that we only have information about the observed positions of a particle at random times. However, we cannot determine when the particle passed through a certain point, in which direction, or at what speed. Therefore, we only have information about how much each point is stretched, but not the direction. Consequently, the information we can obtain from the given data is limited to the scale of the metric tensor of each point. Therefore, we only consider cases where the manifold is conformally flat, i.e., those where $g(x) = \lambda^2 I$. Thus, our goal is to estimate the conformal factor $\lambda(x)$ instead of a general metric tensor field $g(x)$. We have assumed that the metric tensor field is finite and differentiable everywhere. Therefore, it can be said that the conformal factor is also finite and differentiable everywhere.

Let us examine the impact of the conformal factor on particle dynamics. Consider two points, a and b, on a conformally flat manifold, and suppose there exist two paths between them of equal Euclidean length, but differing lengths under the Riemannian metric. One path traverses a region of high conformal factor, while the other traverses a region of low conformal factor. Under conformal geometry, the path passing through the high conformal factor region will be longer. If a particle moves from point a to point b via either path, the particle will be more likely to select the shorter path, resulting in an increased probability of detection per unit length on the path through the region with a lower conformal factor. Consequently, particles are more likely to traverse regions with smaller conformal scales, and the frequency of particle observations will be greater in those regions. Therefore, if particles are frequently detected in a specific area of an unknown manifold, the absolute value of the conformal factor is likely higher in areas with fewer particle detections. Thus, we can infer the existence of a monotonically decreasing function $f(x)$ that satisfies $|\lambda(x)| = f(\tilde{p}_{data}(x))$ for the unnormalized density $\tilde{p}_{data}(x)$, where $\lambda(x)$ is the conformal factor and $f(x)$ is finite differentiable for all regions. We now turn our attention to determining the function $f(x)$. Since $|\lambda(x)| = f(\tilde{p}_{data}(x))$ must hold for all $x, f(x) > 0$ for all x. Furthermore, since the conformal factor is finite differentiable for all regions, $f(x)$ must also be finite differentiable. Thus, we adopt the function $f(x) = e^{-Ax+B}$ $(A > 0)$ that satisfies these conditions, which results in

$$\lambda(x) = e^{-A\tilde{p}_{data}(x)+B}. \tag{7}$$

Here, A and B are arbitrary constants that satisfy $A > 0$ and $|A|, |B| < \infty$.

B. PLANNING AS OPTIMIZATION PROBLEM

First, we discuss the method to find the geodesic in the set of paths that connect two given points. On a Riemannian manifold, given arbitrary path $x(t)$, the length $L(x)$ is defined as a continuous functional on the path space. Therefore, in optimization perspective, finding the geodesic between two points is equivalent to finding the minimizer of the functional on the set of paths that connect two points. As a closed-form expression for the conformal factor is not given, we need to use a numerical optimizer to find the geodesic. In the optimization problem, the objective is the length of the path, which is given by the integral

$$L(x) = \int_0^1 \lambda(x_\theta(t))||\dot{x}_\theta(t)||dt, \tag{8}$$

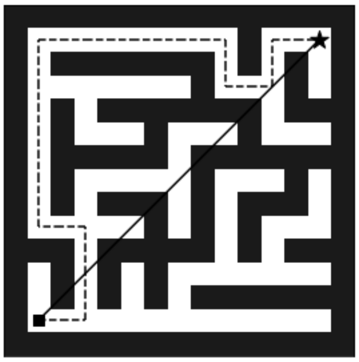
FIGURE 2. Illustration of the shortest path and initial path in a randomly generated maze.

where $\lambda(\cdot)$ is a conformal factor and θ are the parameters of the path. To find minizer in numerical way, we first determine how to update the parameters of path. One possible approach is to estimate the path length using Monte Carlo methods, compute the gradient utilizing numerical differentiation method, and update the parameters using gradient descent method. Specifically, if we use a random variable $T_U \sim U[0, 1]$ following a uniform distribution on the interval $[0, 1]$, the expected value of the length can be expressed as

$$\mathop{\mathbb{E}}_{T_U \sim U[0,1]}[\lambda(x_\theta(T_U))||\dot{x}_\theta(T_U)||]. \tag{9}$$

Therefore, we can estimate the length by sampling n_T points from the uniform distribution on $[0, 1]$ and computing the average $\frac{1}{n_T}\sum_i \lambda(x_\theta(t_i))||\dot{x}_\theta(t_i)||$. However, this gradient represents the steepest direction in the parameter space, whereas what we are interested in is the steepest direction in the space of paths connecting p and q. If we update the parameters along this direction, we can achieve a faster convergence rate. Moreover, since the length of the path $x(t)$ is a functional of the path, we can also compute its functional derivative $\frac{\delta L}{\delta x}$. Therefore, instead of updating the parameters using the gradient in the parameter space, we update them in the direction of the functional gradient. (Please refer to Eq 6 for the exact form of the functional gradient of the length functional.)

C. OPTIMIZATION WITH RICCI FLOW

Finding the shortest path between two points in an arbitrary manifold is a high-dimensional and multi-modal problem that is more challenging than optimizing a single point. This complexity is exemplified in Figure 2, which depicts a maze environment with a square-shaped starting point and a star-shaped goal. The initial path between the two points is represented by a straight line, whereas the dashed line shows the optimal path. The estimation of the conformal factor from the samples can result in infeasible areas having high values and feasible areas having low values. In Figure 2, the initial path encounters six walls, labeled 1 through 6, on its way to the goal. We can easily see that in order for the straight

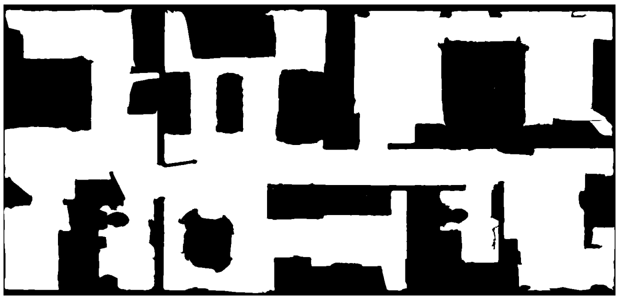

FIGURE 3. **Visualization of the Cantwell environment, where the black regions represent unfeasible areas and the white regions represent feasible areas.**

line to approach the optimal path, it should move upwards. However, for the straight line, it is difficult to determine the optimal direction of change for the walls of 3 and 5. Even if it manages to choose the correct direction, it will eventually encounter another infeasible area and face the same dilemma again. Moreover, escaping from area 5 is challenging as it requires encountering more walls, thereby increasing the path length. This situation creates difficult local optima in the path space, and each case has its own local optima depending on where and in which direction new walls are encountered. As a result, the optimization process encounters a significantly larger number of local optima than in typical optimization problems.

We provided an example of this in a 2D maze environment, but this approach can be applied to real-world data and may even become more severe. Therefore, directly performing path optimization for environments with complex topology may not yield feasible results. Thus, instead of performing optimization directly on the target manifold, we used geometric flow to gradually transform the structure in a simple direction and then transform it slowly back into the desired result on the target manifold through inverse transformation. We now consider which a geometric flow to use in our framework. To do this, we revisit the key properties of the Ricci flow. The Ricci flow updates the metric tensor in a direction opposite to the Ricci curvature, which characterizes the protrusions and indentations of the manifold. In the case of conformally flat manifolds, the Ricci flow reshapes the conformal factor landscape in such a way that previously depressed regions are elevated and previously elevated regions are depressed. This property is desirable for our purposes, and thus we adopt the Ricci flow as the basis of our framework. Furthermore, we observe that the scaling of the Ricci flow does not impact the transformation pattern of the geometric flow. As a result, we introduce a positive, time-dependent scaling function $h(t) > 0$ to scale Eq 10. The resulting geometric flow can be expressed as

$$\frac{\partial g_t}{\partial t} = -2h(t)\mathrm{Ric}^g. \tag{10}$$

The Ricci flow on conformally flat manifolds exhibits an important property. Let us define $\omega_t(x)$ as $\log \lambda_t(x)$, where $\lambda_t(\cdot)$ is the conformal factor of the manifold. Then, $\omega_t(x)$ satisfies the heat equation given by

$$\frac{\partial \omega_t(x)}{\partial t} = \Delta\omega_t(x), \tag{11}$$

where Δ denotes the Laplace operator. In our case, the conformal factor $\lambda_t(x)$ changes according to the scaled Ricci flow, starting with $\lambda_0(x) = e^{-A\tilde{p}_{\text{data}}(x)+B}$ $(A > 0)$.Therefore, if we define $\tilde{p}_t(x) = -\frac{1}{A}(\log \lambda_t(x) - B)$, we can observe that $p_t(\cdot)$ satisfies Eq 12, where $D(t)$ is a positive scaling factor that depends only on time as

$$\frac{\partial p_t(x)}{\partial t} = \Delta(D(t)p_t(x)). \tag{12}$$

According to Eq 2, this corresponds to the time evolution of the probability density function of a pure diffusion process with a zero drift coefficient. Thus, if we have knowledge of the time evolution of the unnormalized probability density function for a pure diffusion process, $\tilde{p}_t(x)$, we can deduce the time evolution of the conformal factor, $\lambda_t(x)$.

It has been established in previous research that the stochastic process X_t with dynamics given by $dX_t = \sqrt{\frac{d[\sigma^2(t)]}{dt}}dW_t$ satisfies $X_t \overset{d}{=} X_0 + \sigma(t)Z$ for $Z \sim \mathcal{N}(0, I)$. Hence, if we can determine $D(t)$, we can determine $\sigma(t)$ and draw samples from $p_t(x)$. In other words, it means that we can determine the conformal factor at any given time by choosing an arbitrary positive non-decreasing function $\sigma(t)$.

D. ALGORITHM

In this subsection, we present a concise overview of the overall algorithmic framework, followed by a detailed explanation of each step. Initially, we seek the path that minimizes the length for the simplest manifold structure. Subsequently, we make a subtle adjustment to the manifold structure in the direction of the target manifold. We then proceed with the optimization process to identify the minimum-length path on the altered manifold, utilizing the previously obtained path as the initial point for the optimization process. This iterative process continues until the manifold structure converges to the target manifold. Ultimately, we obtain the minimum-length path on the desired manifold. The pseudocode for the entire algorithm is provided in Alg 1.

We adopted a method to represent the path $\tilde{x}(\cdot)$ from the current state s to the goal state g by connecting a fixed number $N + 1$ of points $\{p_i\}_{i\in[N]}$ with straight lines ($[N] = \{0, 1, \ldots, N\}$). To uniformly represent all positions on the path, we utilized points where adjacent points in the sequence are equidistant. This results in

$$\forall i, j \in [N-1], ||p_{i+1} - p_i|| = ||p_{j+1} - p_j||. \tag{13}$$

For notation convenience, we designated the first point p_0 and the last point p_N of the sequence as s and g respectively, setting the domain of the path as $[0, 1]$. Therefore, given $\{p_i\}_{i\in[N]}$, the path $\tilde{x}(\cdot)$ can be represented as

$$x(t) = \tau(t) \times p_{\lfloor Nt \rfloor} + (1 - \tau(t)) \times p_{\lceil Nt \rceil}, \tag{14}$$

where $\lceil x \rceil = \min\{n \in \mathbb{Z} : n \geq x\}, \lfloor x \rfloor = \max\{n \in \mathbb{Z} : n \leq x\}$, and $\tau(t) = N\left(t - \frac{\lfloor Nt \rfloor}{N}\right)$ $(0 \leq t \leq 1)$ with $x(0) = s$ and $x(1) = g$.

First, we initialize points evenly distributed along a straight line connecting the starting point and the ending point, denoted as $\{p_i\}_{i \in [N]}$. When updating $\{p_i\}_{i \in [N]}$, we use the functional gradient $\frac{\delta L(x)}{\delta x}$ with respect to the length of $\tilde{x}(\cdot)$. Before providing a detailed explanation of the update process, let's consider the case of updating an arbitrary element $x(t)$ in the path space for an arbitrary functional $J(x)$ using gradient descent on path space. In this case, one step of the update is carried out as

$$\tilde{x}^{(i+1)}(\cdot) \leftarrow (\tilde{x}^{(i)} - \alpha \frac{\delta L}{\delta x})(\cdot), \tag{15}$$

where α represents the learning rate.

However, in practical situations, there are additional challenges compared to the aforementioned case. Firstly, in practical scenarios, the set on which we perform optimization is not the set of all paths but rather the set of paths that can be parameterized by a method that we define. Therefore, even if the update direction in the path space is given, we cannot directly perform the update in that direction. Instead, we need to calculate the parameter changes that bring the direction closest to the desired update. To calculate this, it is necessary to determine which changes in a parameter induce the desired direction of variation at each t, and the change in each parameter must be averaged over all t within the domain. However, calculating this influence for all t within the path's domain is infeasible. One possible approach is to sample a predefined number M of elements from the path's domain, calculate the parameter variation for each, and then average these values. This method incurs a computation cost of $O(MN)$, considering each t_i influences all parameters. However, in our case, since $x(\frac{i}{N}) = p_i$ holds true, we decide on $N = M$ and determine $t_i = \frac{i}{N}$. As a result, t_i is evenly distributed over the path's domain, and the path's variation at t_i affects only p_i, reducing the gradient calculation cost to $O(N)$.

However, an additional problem arises here. To compute the path's variation at each point, we require the first and second-order derivatives of the path at each point. Since we parameterize the path by connecting a finite number of points with straight lines, the path is not differentiable at each t_i. Therefore, we estimate the speed $\dot{x}_i(t_i)$ and acceleration $\ddot{x}_i(t_i)$ of the path at t_i as the average speed between t_{i-1} and t_{i+1} and the change in average speed between $[t_{i-1}, t_i]$ and $[t_i, t_{i+1}]$, respectively. This can be expressed mathematically as

$$\begin{aligned} \dot{x}(t_i) &= \frac{1}{2\Delta t}(p_{i+1} - p_{i-1}) \\ \ddot{x}(t_i) &= \frac{1}{\Delta t}\left(\frac{p_{i+1} - p_i}{\Delta t} - \frac{p_i - p_{i-1}}{\Delta t}\right), \end{aligned} \tag{16}$$

where $\Delta t = \frac{1}{N}$. We then use these estimated values by substituting them into Eq 6. to estimate the functional gradient $\frac{\delta L}{\delta x}$ at each point p_i.

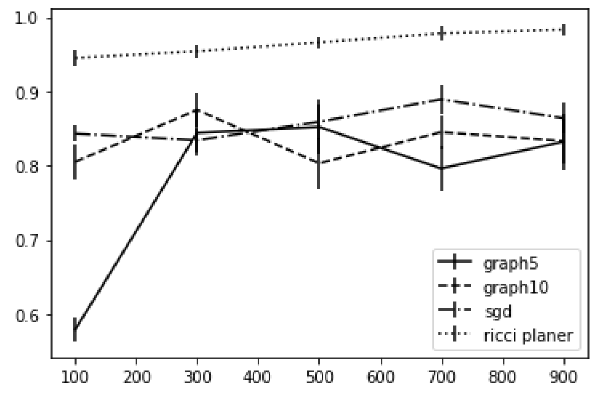

FIGURE 4. The feasible rate of the generated path based on the given number of samples.

However, after performing the update, an additional problem arises. We desire that the intervals between points remain constant. Specifically, when denoting the collection of sequences of equidistant points as P_N, we aim for the updated entity to belong to P_N. However, when updating using the aforementioned method, without considering the intervals between points, there is no guarantee that an entity that belonged to P_N before the update will still be an element of P_N afterward. Furthermore, the imbalance can worsen as updates overlap. To prevent the variation in interval length, it is necessary to project the target entity onto P_N after each update. One conceivable approach for this is to perform additional optimization at each update step, using the deviation of the intervals between each pair of points from their average as the objective. However, applying such a method would be computationally expensive, requiring additional optimization at each step.

Therefore, to transform the given point sequence into a sequence satisfying our desired conditions, we first define a curve with a constant speed using this sequence and update it with points at equal intervals on the curve's domain. We apply this update iteratively to obtain points that meet our desired conditions. To explain in more detail, let us denote the target point sequence at the k-th iteration as $\{p_i^{(k)}\}_{i \in [N]}$. We connect adjacent points in this sequence to define a curve. The domain of this curve is set to [0, 1], and it is endowed with a constant speed. This way, a piecewise linear curve $\tilde{x}_k(\cdot)$ with a constant speed defined on [0, 1] is determined.If we denote $\sum_{j=1}^{i} ||p_j^{(k)} - p_{j-1}^{(k)}||$ and $||p_j^{(k)} - p_{j-1}^{(k)}||$ as L_j and l_j respectively, then $x^{(k)}(\cdot)$ can be expressed as

$$x^{(k)}(t) = \tau(t) \times p_{\iota(t)}^{(k)} + (1 - \tau(t)) \times p_{\iota(t)+1}^{(k)}, \tag{17}$$

where $\iota(t) = \sum_{i=1}^{N} \mathbb{I}(\frac{L_i}{L_N} < t)$ and $\tau(t) = \frac{L_N \times t - L_{\iota(t)}}{l_{\iota(t)+1}}$. Furthermore, we determine $p_i^{(k+1)}$ by using the values of the curve at $\frac{i}{N}$. Therefore, the results $p_i^{(k+1)}$ after the k-th iteration can be expressed as

$$p_i^{(k+1)} = \tilde{x}_k(\frac{i}{N}). \tag{18}$$

Furthermore, we verified that applying Eq 18 repeatedly to any point sequence ultimately results in equidistant spacing between the points. This observation can be formally expressed through the proposition outlined below. For a positive integer N and a sequence of $N+1$ points in $\mathbb{R}^n$, denoted as $\{p_i^{(0)}\}_{i\in[N]}$, if $\{p_i^{(k)}\}_{i\in[N]}$ is defined through Eq 18, we have

$$\lim_{k\to\infty} \max_{i\in[N-1]} ||p_{i+1}^{(k)} - p_i^{(k)}|| = \lim_{k\to\infty} \min_{i\in[N-1]} ||p_{i+1}^{(k)} - p_i^{(k)}||.$$

The above proposition signifies that repeatedly applying Eq 18 ultimately results in equal spacing between adjacent points in the sequence. The proof for this can be found in the Appendix A The pseudocode for the entire algorithm and the projection algorithm can be found in Alg 1 and Alg 2, respectively.

V. EXPERIMENT

In this section, we empirically demonstrate the performance of the Ricci planner. We first experimentally analyze the plans generated by the Ricci planner itself and then assess the performance improvement when applying those plans. We analyze the generated plans from two perspectives: how often the generated plans deviate from the feasible area and how much they deviate on average. To do this, we utilize two metrics: the feasibility ratio and the infeasibility score. The feasibility ratio is defined as the proportion of paths sampled that belong to the feasible region. It is measured as the ratio of positions within the feasible region among equidistant points in time. The infeasibility score measures how far the deviated points are from the feasible region. Therefore, it calculates the average distance from each point to the closest point in the feasible region. The points in the feasible region have a score of 0.

Furthermore, in order to assess the effectiveness of the plan, we utilize the success rate when applying the generated plan to a very simple policy. The very simple policy refers to a policy that only moves in the direction of the goal, without utilizing any information about the environment. The plan itself consists of a sequence of points that connect the starting position to the goal. When employing this policy with the plan, a sequential achievement approach is employed, where each point is computed in a sequential manner.

A. BENCHMARKS AND BASELINES

We conducted experiments on two datasets: a synthetic maze dataset and the Indoor Navigation 2D Dataset (IN2D) [9], [47]. The maze is generated through the utilization of the randomized Prim algorithm [30] (Alg 3), and the data points were randomly sampled along the lines connecting the centers of adjacent cells. The IN2D dataset includes information about the locations of furniture and whether there are obstacles at each location in a typical indoor environment. We conducted experiments on the most complex environment in the dataset of the Canwell environment (Figure 3).

Algorithm 3 Randomized Prim

```
Input: weight $W$, height $H$
Output: cell state matrix $S$
$S \leftarrow$ cell state matrix of size of $W \times H$
$S$[odd] $\leftarrow$ "Wall"; $S$[even] $\leftarrow$ "Room"
$c \leftarrow (0, 0)$; $Q \leftarrow \{c\}$
while $|Q| \geq 1$ do
    $c \leftarrow Q$.pop()
    $c$.visited $\leftarrow$ True
    $\mathcal{N} \leftarrow$ Cells with a positional difference of 2
    Partition $\mathcal{N}$ into visited $\mathcal{V}$ and unvisited $\mathcal{U}$ cells
    if $|\mathcal{V}| \geq 1$ then
        d $\leftarrow$ sample one cell from $\mathcal{V}$
        Denote the cell between c and d as "Room"
    Q.extend(U)
return $S$
```

Regarding the baseline method, the problem scenario we address involves only time-independent random observations of the environment, making it infeasible to apply existing planning methods. Therefore, as a baseline, we used a method that builds a graph with all given points as vertices and the distance between each pair of points as the weight of the edge, and performs shortest path finding algorithm on the graph for planning. However, in Euclidean space, the shortest path is a straight line, so if there are edges between all pairs of points, the shortest path will be a path that only includes the start and end points. Therefore, we sorted the pairs by distance and set the minimum distance between them to infinity, which reduces the probability of generating a plan that connects the two points directly and provides a more meaningful baseline.

Therefore, we used the pruning cases of 95% and 90% edge removal as baselines. This means that in the case of a total of 100 points, only the closest 5 and 10 points have edges, which can be considered a reasonable choice. We used the same baseline in all experiments, which we referred to as "graph5" and "graph10" repectively. In the Ricci planner, we applied the Ricci flow method to the target manifold. To investigate the criticality of applying Ricci flow to performance improvement, we also conducted experiments without applying Ricci flow and directly optimized the path on the target manifold. We labeled this approach as "sgd". Additionally, we labeled our proposed method as "ricci planner."

B. EXPERIMENTAL DETAIL

The following subsection provides a detailed description of the experimental setup used. In the experiment evaluating the robustness with respect to the number of samples, we measured the feasible rate by incrementing the number of samples from 100 to 900 in increments of 200. For

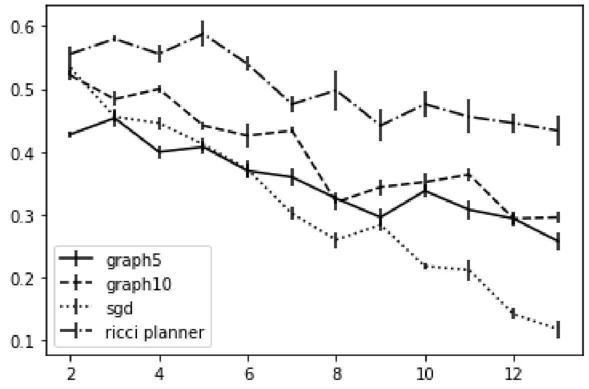

FIGURE 5. Success rate based on the distance between the starting position and the goal.

TABLE 1. Results of the feasible rate.

method	feasible rate	min
graph5	0.780 ± 0.185	0.48
graph10	0.832 ± 0.138	0.48
sgd	0.857 ± 0.0965	0.68
ricci planner	0.965 ± 0.0473	0.83

TABLE 2. Results of the unfeasible score for the IN2D dataset.

method	unfeasible score	max
graph5	4.025 ± 9.54	58.0
graph10	2.464 ± 8.13	58.0
sgd	3.008 ± 7.87	48.0
ricci planner	1.534 ± 5.35	37.0

the remaining experiments, we used 1000 observation data points. The policy used for measuring the success rate uniformly sampled step distances ranging from 0.01 to 0.1, and a criterion of 0.2 was employed to determine the success of an episode. Specifically, if the distance between the current state and the given goal during the episode was less than or equal to 0.2, the episode was immediately considered a success. To verify the usefulness of the plan based on the task's difficulty, we measured the success rate according to the distance between the starting point and the goal. Here, the starting point and the end point were sampled in the same way as the training data, and the distance was determined based on the grid distance between the cells to which each point belonged. Furthermore, we conducted 20 repetitions for all measurement used in our experiments.

For density estimation from the given data, we employed the kernel density estimation method [38], using a radial basis function (RBF) kernel [45]. To illustrate the changes in the distribution due to Ricci flow, the bandwidth of the kernel was varied from 0.005 to with uniform intervals. Furthermore, we used the same hyperparameters for all experiments, and the details of the hyperparameters used in the experiments can be found in Appendix B.

C. RESULT

To measure the sensitivity of the given data set to the number of samples, we measured the feasible rate corresponding to the number of samples. This result can be seen in Figure 4, which shows the experimental results for synthetic data, while the remaining data sets show values near 0.8. However, the proposed method shows values of 0.9 or higher, and as the number of samples increases, the feasible rate steadily improves. We also confirmed that performing gradient-based optimization without the Ricci flow produces almost the same results. Therefore, the application of the Ricci flow has significant meaning.

Table 1 provides the results of feasible rate. Through this, we found that the Ricci planner not only has the largest mean of the feasible rate compared to other methods but also has the smallest standard deviation and shows a significant difference in the case of the minimum value. Table 2 shows the measurement values of the score in IN2D, where lower values are better. Here again, the Ricci planner showed better performance than other methods. Overall, our experiments demonstrate that the Ricci planner can be an advanced form of planner compared to other methods.

Finally, Figure 5 illustrates the success rate based on the distance between the starting and ending points. By examining the graph, it can be observed that the success rates of all baseline methods exhibit a decreasing trend. This is expected because the task difficulty is roughly proportional to the distance. Graph 5 and Graph 10 show the same pattern. However, the Ricci planner shows significant improvement compared to other methods. Although there is no significant difference between Graph 10 or SGD and the Ricci planner when the distance is small, a widening discrepancy becomes evident as the distance increases, leading to a substantial difference in the end. This suggests that when the distance is small, there are fewer obstacles between the two points, where there is less dependency on the planning method. However, as the number of obstacles increases, the influence of the planning method becomes more significant. From the shape of this plot, we can conclude that the Ricci planner is more effective in long-term planning. Furthermore, it can be observed that SGD performs significantly worse with a much larger margin compared to the measurements of the feasible rate or feasible score. This can be attributed to the presence of numerous local optima in the path space as the number of obstacles increases, resulting in suboptimal paths, as mentioned in the previous section. Thus, through this analysis, we can confirm the importance of transforming the original manifold structure into a simpler manifold.

VI. CONCLUSION

In this paper, we focused on the problem of finding the shortest path between any two arbitrary points when only observation data points are given. We first identified the limitations of a simple approach that builds a graph and applies a shortest path algorithm on the graph. Therefore, instead of building a graph, we proposed a method to estimate the manifold structure and formulated the problem of finding the shortest path as an optimization problem. We also identified the challenges of this optimization problem and proposed a method called the Ricci planner, which alleviates these challenges by continuously transforming the geometry using the Ricci flow. Furthermore, we experimentally validated both the feasibility of the paths generated by our proposed algorithm and the advantages of applying these paths. However, there is a limitation that, as with gradient-based methods such as the diffusion model, it can incur a high computational cost.

Gradient computation can be bypassed through additional training of the score function. However, our research focuses on situations where only a limited number of data points are given, so utilizing a trained score function falls beyond the scope of our study, and we leave it as a future work. In this study, our focus was on sampling a path between two points, which is a function defined on a closed interval of the real line. However, the method of utilizing geometric flow to transform functions is not necessarily limited to one-dimensional functions. Therefore, we anticipate that our research can promote studies on the utilization of geometric flow in generating functions in various domains.

APPENDIX A PROOF OF PROPOSITION 1

Proof: What we intend to demonstrate is the existence of the limits for $\min_{i\in[N-1]} ||p_{i+1}^{(k)} - p_i^{(k)}||$ and $\max_{i\in[N-1]} ||p_{i+1}^{(k)} - p_i^{(k)}||$, with both limits converging to the same value. If we denote $\max_{i\in[N-1]} ||p_{i+1}^{(k)} - p_i^{(k)}||$ as L_k and $\frac{1}{N}\sum_{i\in[N-1]} ||p_{i+1}^{(k)} - p_i^{(k)}||$ as l_k, then it is evident that the proposition holds if L_k and l_k each have a limit, and these limits are equal. Therefore, we demonstrate the convergence of l_k first and subsequently establish that L_k also converges to the same value.

Firstly, we prove the convergence of l_k. Consider the case where there exists a k' such that $p_i^{(k')} = p_i^{(k'+1)}$ for all i. In this scenario, l_k remains constant, and therefore, it converges. Now, let's consider the case where no such k' exists. In this situation, let's examine the k-th update using Eq 18. In this case, the distance traveled by $x_k(\cdot)$ between $p_{i+1}^{(k+1)}$ and $p_i^{(k+1)}$ for all i becomes l_k. This distance cannot be smaller than the straight-line distance between $p_{i+1}^{(k+1)}$ and $p_i^{(k+1)}$, i.e., $||p_{i+1}^{(k+1)} - p_i^{(k+1)}||$. In other words, for all i, we have $||p_{i+1}^{(k+1)} - p_i^{(k+1)}|| \leq l_k$. However, the equality $||p_{i+1}^{(k+1)} - p_i^{(k+1)}|| = l_k$ for all i implies that $x_k(\cdot)$ between $p_{i+1}^{(k+1)}$ and $p_i^{(k+1)}$ is a straight line, indicating $p_i^{(k')} = p_i^{(k'+1)}$ for all i. Since we are considering cases where such k' does not exist, $l_k > ||p_{i+1}^{(k+1)} - p_i^{(k+1)}||$ holds for all i. Averaging over all i, we get $l_k > \frac{1}{N}\sum_{i\in[N-1]} ||p_{i+1}^{(k+1)} - p_i^{(k+1)}|| = l_{k+1}$, which implies $l_k > l_{k+1}$. Since l_k is bounded below by 0 and is a monotonically decreasing sequence, it converges.

Next, we demonstrate that when l_k converges to the limit l, L_k converges to l as well. Given that l_k is monotonically decreasing and converges to l, for any arbitrary $\epsilon > 0$, there exists a k' such that for all $k > k'$, $l_k - l < \epsilon$. Moreover, as previously established, for all i, $||p_{i+1}^{(k+1)} - p_i^{(k+1)}|| \leq l_k$. Therefore, $L_{k+1} \leq l_k$ holds. Consequently, for $k > k'$, $L_{k+1} - l < \epsilon$ is satisfied. Additionally, $L_{k+1} \geq l_k \geq l$, leading to $L_{k+1} - l \geq 0$. Thus, $|L_{k+1} - l| < \epsilon$. Consequently, for any ϵ, there exists a k' such that for all $k > k' + 1$, $|L_k - l| < \epsilon$. This implies that L_k converges to l. Therefore, we can obtain the desired conclusion. □

APPENDIX B HYPERPARAMETER SETTINGS

In this section, we describe the hyperparameter settings used in the experiment. The hyperparameters used are listed in Table 3, and we utilized the same hyperparameters for all experiments. T, N, S, and α are variables used in Alg 1. Here, T represents the number of steps used to modify the manifold, N is the number of points used to represent the path, S is the number of optimization steps in a single manifold for path optimization, and α denotes the step size used in path optimization. Additionally, K in Alg 2 represents the number of steps used for projection iterations. Furthermore, A and B represent the variables used in Eq 7, which describe the relationship between the conformal factor and the distribution of data. Since B influences only the scale of comformal factor, adjusting it is equivalent to modifying α. Therefore, for simplicity, we fixed B at 0.

TABLE 3. *T*, *N*, *S*, α refer to the variables used in Alg 1., while *K* denotes the variable used in Alg 2. Additionally, *A*, *B* represent the variables used in Eq 7.

T	N	S	α	K	A	B
400	100	2	100	100	200	0

REFERENCES

[1] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," 2022, *arXiv:2201.08299*.

[2] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation-the state of the art," 2017, *arXiv:1701.08878*.

[3] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[4] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, Feb. 2022.

[5] I. Bakas, "The algebraic structure of geometric flows in two dimensions," *J. High Energy Phys.*, vol. 2005, no. 10, p. 38, Oct. 2005.

[6] I. Bakas, "Renormalization group equations and geometric flows," 2007, *arXiv:hep-th/0702034*.

[7] T. Carta, P.-Y. Oudeyer, O. Sigaud, and S. Lamprier, "EAGER: Asking and answering questions for automatic reward shaping in language-guided RL," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 12478–12490.

[8] J. Chen, T. Huang, W. Chen, and Y. Liu, "Thoughts on the consistency between Ricci flow and neural network behavior," 2022, *arXiv:2111.08410*.

[9] M. Dobrevski and D. Skocaj, "Adaptive dynamic window approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 6930–6936.

[10] B. Eysenbach, X. Geng, S. Levine, and R. R. Salakhutdinov, "Rewriting history with inverse RL: Hindsight inference for policy improvement," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 14783–14795.

[11] B. A. Frigyik, S. Srivastava, and M. R. Gupta, ''An introduction to functional derivatives,'' Dept. Electr. Eng., Univ. Washington, Seattle, WA, USA, Tech. Rep. UWEETR-2008-0001, 2008.

[12] S. Gallot, D. Hulin, and J. Lafontaine, *Riemannian Geometry*, vol. 2. Berlin, Germany: Springer, 1990.

[13] S. Glass, S. Spasov, and P. Liò, ''RicciNets: Curvature-guided pruning of high-performance neural networks using Ricci flow,'' 2020, *arXiv:2007.04216*.

[14] C. F. Hayes, R. Radulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane, P. Mannion, A. Nowé, G. Ramos, M. Restelli, P. Vamplew, and D. M. Roijers, ''A practical guide to multi-objective reinforcement learning and planning,'' *Auto. Agents Multi-Agent Syst.*, vol. 36, no. 1, p. 26, Apr. 2022.

[15] X. He and C. Lv, ''Robotic control in adversarial and sparse reward environments: A robust goal-conditioned reinforcement learning approach,'' *IEEE Trans. Artif. Intell.*, vol. 5, no. 1, pp. 244–253, Jan. 2024.

[16] J. Ho, A. Jain, and P. Abbeel, ''Denoising diffusion probabilistic models,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6840–6851.

[17] Z. Huang, F. Liu, and H. Su, ''Mapping state space using landmarks for universal goal reaching,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[18] R. Jangir, G. Alenya, and C. Torras, ''Dynamic cloth manipulation with deep reinforcement learning,'' in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 4630–4636.

[19] M. Janner, Q. Li, and S. Levine, ''Offline reinforcement learning as one big sequence modeling problem,'' in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1273–1286.

[20] S. Kapturowski, V. Campos, R. Jiang, N. Rakicevic, H. van Hasselt, C. Blundell, and A. P. Badia, ''Human-level Atari 200x faster,'' 2022, *arXiv:2209.07550*.

[21] J. Kim, Y. Seo, S. Ahn, K. Son, and J. Shin, ''Imitating graph-based planning with goal-conditioned policies,'' 2023, *arXiv:2303.11166*.

[22] G. Kumar, N. S. Shankar, H. Didwania, R. D. Roychoudhury, B. Bhowmick, and K. M. Krishna, ''GCExp: Goal-conditioned exploration for object goal navigation,'' in *Proc. 30th IEEE Int. Conf. Robot Human Interact. Commun. (RO-MAN)*, Aug. 2021, pp. 123–130.

[23] M. Lapan, *Deep Reinforcement Learning Hands-on: Apply Modern RL Methods, With Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More*. Birmingham, U.K.: Packt, 2018.

[24] S. Levine, A. Kumar, G. Tucker, and J. Fu, ''Offline reinforcement learning: Tutorial, review, and perspectives on open problems,'' 2020, *arXiv:2005.01643*.

[25] Y. Li and R. Lu, ''Applying Ricci flow to high dimensional manifold learning,'' Sci. China Inf., Beijing, China, Tech. Rep. 192101, 2017.

[26] Y. Li, T. Gao, J. Yang, H. Xu, and Y. Wu, ''Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning,'' in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2022, pp. 12765–12781.

[27] J. Y. Ma, J. Yan, D. Jayaraman, and O. Bastani, ''Offline goal-conditioned reinforcement learning via f-advantage regression,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 310–323.

[28] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, ''Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning,'' 2019, *arXiv:1911.04936*.

[29] Y. Jason Ma, J. Yan, D. Jayaraman, and O. Bastani, ''How far I'll go: Offline goal-conditioned reinforcement learning via f-advantage regression,'' 2022, *arXiv:2206.03023*.

[30] S. Manen, M. Guillaumin, and L. V. Gool, ''Prime object proposals with randomized Prim's algorithm,'' in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2536–2543.

[31] S. Mirchandani, S. Karamcheti, and D. Sadigh, ''ELLA: Exploration through learned language abstraction,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 29529–29540.

[32] Y. Niu, S. Jin, Z. Zhang, J. Zhu, D. Zhao, and L. Zhang, ''GOATS: Goal sampling adaptation for scooping with curriculum reinforcement learning,'' 2023, *arXiv:2303.05193*.

[33] J. M. Catacora Ocana, R. Capobianco, and D. Nardi, ''An overview of environmental features that impact deep reinforcement learning in sparse-reward domains,'' *J. Artif. Intell. Res.*, vol. 76, pp. 1181–1218, Apr. 2023.

[34] S. Pateria, B. Subagdja, A.-H. Tan, and C. Quek, ''Hierarchical reinforcement learning: A comprehensive survey,'' *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–35, Jun. 2022.

[35] G. Perelman, ''Ricci flow with surgery on three-manifolds,'' 2003, *arXiv:0303109*.

[36] S. Reddy, A. D. Dragan, and S. Levine, ''SQIL: Imitation learning via reinforcement learning with sparse rewards,'' 2019, *arXiv:1905.11108*.

[37] Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng, ''Exploration via hindsight goal generation,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

[38] M. Rosenblatt, ''Remarks on some nonparametric estimates of a density function,'' *Ann. Math. Statist.*, vol. 27, no. 3, pp. 832–837, 1956.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, ''Proximal policy optimization algorithms,'' 2017, *arXiv:1707.06347*.

[40] Y. Song and S. Ermon, ''Generative modeling by estimating gradients of the data distribution,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–13.

[41] Y. Song and D. P. Kingma, ''How to train your energy-based models,'' 2021, *arXiv:2101.03288*.

[42] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, ''Score-based generative modeling through stochastic differential equations,'' 2020, *arXiv:2011.13456*.

[43] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[44] C. Szepesvári, ''Algorithms for reinforcement learning,'' *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–103, 2010.

[45] J.-P. Vert, K. Tsuda, and B. Schölkopf, ''A primer on kernel methods,'' *Kernel methods Comput. Biol.*, vol. 47, pp. 35–70, Dec. 2004.

[46] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, ''Deep reinforcement learning: A survey,'' *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 28, 2022, doi: 10.1109/TNNLS.2022.3207346.

[47] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, ''Gibson env: Real-world perception for embodied agents,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9068–9079.

[48] W. Xu, E. R. Hancock, and R. C. Wilson, ''Rectifying non-Euclidean similarity data using Ricci flow embedding,'' in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 3324–3327.

[49] X. Yu, N. Lei, Y. Wang, and X. Gu, ''Intrinsic 3D dynamic surface tracking based on dynamic Ricci flow and teichmuller map,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5390–5398.

[50] L. Zhang, G. Yang, and B. C. Stadie, ''World model as a graph: Learning latent landmarks for planning,'' in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12611–12620.

WONGEUN SONG received the B.S. degree in electrical and computer engineering from Seoul National University, in 2017, where he is currently pursuing the Ph.D. degree.

JUNGWOO LEE (Senior Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, in 1988, and the M.S.E. and Ph.D. degrees in electrical engineering from Princeton University, in 1990 and 1994, respectively. He is currently a Professor with the Department of Electrical and Computer Engineering, Seoul National University.