Generation of Geodesics with Actor-Critic Reinforcement Learning to Predict Midpoints

Kazumi Kasaura   KAZUMI.KASAURA@SINICX.COM

OMRON SINIC X Corporation
Nagase Hongo Building 3F 5-24-5 Hongo, Bunkyo-ku Tokyo-to, Japan 113-0033

Editor: Shiqian Ma

Abstract

To find the shortest paths for all pairs on manifolds with infinitesimally defined metrics, we introduce a framework to generate them by predicting midpoints recursively. To learn midpoint prediction, we propose an actor-critic approach. We prove the soundness of our approach and show experimentally that the proposed method outperforms existing methods on several planning tasks, including path planning for agents with complex kinematics and motion planning for multi-degree-of-freedom robot arms.

Keywords: Path Planning, Finsler Manifold, Riemannian Manifold, All-Pairs Shortest Paths, Sub-Goal
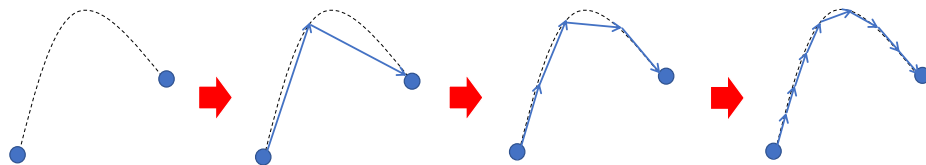
1. Introduction

Figure 1: Midpoint tree generation of a geodesic (dotted curve).

On manifolds with metrics, minimizing geodesics, or shortest paths, are minimum-length curves connecting points. Various real-world tasks can be reduced to the generation of geodesics on manifolds. Examples include time-optimal path planning on sloping ground (Matsumoto, 1989), robot motion planning under various constraints (LaValle, 2006; Ratliff et al., 2015), physical systems (Pfeifer, 2019), the Wasserstein distance (Agueh, 2012), and image morphing (Michelis and Becker, 2021; Effland et al., 2021). Typically, metrics are only known infinitesimally (a form of a Riemannian or Finsler metric), and their distance functions are not known beforehand. Computation of geodesics by solving optimization problems or differential equations is generally computationally costly and requires an explicit form of the metric, or at least, values of its differentials. To find geodesics for various pairs of endpoints in a fixed manifold, it is more efficient to use a policy that has learned to generate geodesics for arbitrary pairs. In addition, we aim to learn only from infinitesimal values of the metric.

	GCRL	SGT	MT (Ours)
Prior Metric Knowledge	global approximation	global upper bound	local values
Horizon	linear	logarithmic	logarithmic
Step Length	fixed	variable	variable

Table 1: Comparison of Planning Frameworks

Goal-conditioned reinforcement learning (GCRL) (Schaul et al., 2015) to generate waypoints of geodesics sequentially from start to goal has two issues. First, since it suffers from sparseness of the reward if an agent gets a reward only when it reaches a goal, it is necessary to give the agent appropriate rewards when it gets near its goal. However, to define reward values, we must have a global approximation of the distance function between two points beforehand. When manifolds have complex metrics, it may be difficult to find approximations. Second, in trying to generate numerous waypoints, the long horizon makes learning difficult (Wang et al., 2020).

Instead of generating waypoints in sequence, in the sub-goal tree (SGT) framework (Jurgenson et al., 2020), paths are generated by recursively applying a policy that have learned to predict intermediate points. Since the recursion depth is the logarithm of the number of waypoints, the issue of long horizons is overcome. This framework also has the advantage that parallelization allows for fast generation. In addition, the density of waypoints can be varied by changing the depth of recurrence, whereas, in GCRL, the length of a single step is fixed in advance. On the other hand, the issue of prior knowledge of the metric remains. In this framework, the length of the "segment" between any two points must be given, which is an upper bound for the distance. However, it is not always easy to calculate them. For example, in motion planning for a robotic arm, the segment between two states is not clearly defined, and verifying the validity of the corresponding motion is computationally expensive.

To overcome this difficulty, we propose a framework called *midpoint tree* (MT), which is a modification of the sub-goal tree framework. In this framework, a policy learns to predict midpoints of given pairs of points instead of arbitrary intermediate points, and paths are generated by inserting predicted midpoints recursively, as illustrated in Figure 1. Since adjacent pairs in the generated waypoints are close, the path length can be calculated even if the metric is known only locally, and the policy can be trained to generate shorter paths.

Table 1 summarizes the characteristics of the frameworks discussed in this introduction.

In addition to the aforementioned issue, the original learning method for sub-goal prediction via a policy gradient in Jurgenson et al. (2020) has poor sample efficiency when recursion is deep. To improve on this, we propose an actor-critic learning approach for midpoint prediction, which is similar to the actor-critic method (Konda and Tsitsiklis, 1999) for conventional reinforcement learning.

We prove theoretically that, under mild assumptions, if the training by our approach converges in the limit of infinite recursion depth, the resulting policy can generate true geodesics. This result does not hold for generation by arbitrary intermediate points.

We experimentally compared our proposed method, on five path (or motion) planning tasks, to sequential generation with goal-conditioned reinforcement learning and midpoint tree generation trained by a policy gradient method without a critic. Two tasks involved continuous metrics or constraints (local planning), while the other three involved collision

avoidance (global planning). In both local and global planning, our proposed method outperformed baseline methods for the difficult tasks.

2. Related Works

2.1 Path Planning with Reinforcement Learning

One of the most popular approaches for path planning via reinforcement learning is to use a Q-table (Haghzad Klidbary et al., 2017; Low et al., 2022). However, that approach depends on the finiteness of the state spaces, and the computational costs grow with the sizes of those spaces.

Several studies have been conducted on path planning in continuous spaces via deep reinforcement learning (Zhou et al., 2019; Wang et al., 2019; Kulathunga, 2022; Qi et al., 2022). In those works, the methods depend on custom rewards.

2.2 Goal-Conditioned Reinforcement Learning and Sub-Goals

Goal-conditioned reinforcement learning (Kaelbling, 1993; Schaul et al., 2015) trains a universal policy for various goals. It learns a value function whose inputs are both the current and goal states. Kaelbling (1993) and Dhiman et al. (2018) pointed out that goal-conditioned value functions are related to the Floyd-Warshall algorithm for the all-pairs shortest-path problem (Floyd, 1962), as this function can be updated by finding intermediate states. They proposed methods that use brute force to search for intermediate states, which depend on the finiteness of the state spaces. The idea of using sub-goals for reinforcement learning as options was suggested by Sutton et al. (1999), and Jurgenson et al. (2020) linked this notion to the aforementioned intermediate states. Wang et al. (2023) drew attention to the quasi-metric structure of goal-conditioned value functions and suggested using quasi-metric learning (Wang and Isola, 2022) to learn those functions.

The idea of generating paths by predicting sub-goals recursively has been proposed in three papers with different problem settings and methods. The problem setting for goal-conditioned hierarchical predictors (Pertsch et al., 2020) differs from ours because they use an approximate distance function learned from given training data, whereas no training data are given in our setting. Divide-and-conquer Monte Carlo tree search (Parascandolo et al., 2020) is similar to our learning method because it trains both the policy prior and the approximate value function, which respectively correspond to the actor and critic in our method. However, their algorithm depends on the finiteness of the state spaces.

The problem setting for sub-goal tree framework (Jurgenson et al., 2020) is the most similar to ours, but it remains different, as mentioned in the introduction. In their main experiment on robot arm planning, local motion is executed by a controller, while only the global path is generated by a policy. Consequently, the cost of the segment between two states is determined by the controller, and the recursion is not as deep as in our setting.

3. Preliminaries and Notation

In § 3.1, we describe general notions for quasi-metric spaces that are necessary to formulate our framework. In § 3.2, we describe Finsler manifolds, which serve as important examples

of quasi-metric spaces and include formulations of various planning tasks. On the other hand, formalizing our framework in a general form enables it to handle cases with hard collision avoidance constraints (§ 4.5) that cannot be treated as Finsler manifolds.

3.1 Quasi-Metric Space

We follow the notation in Kim (1968). Let X be a space. A *pseudo-quasi-metric* on X is a function $d : X \times X \to \mathbb{R}$ such that $d(x,x) = 0$, $d(x,y) \geq 0$, and $d(x,z) \leq d(x,y) + d(y,z)$ for any $x, y, z \in X$. For $x \in X$ and $r > 0$, the open ball $\{x' \in X | d(x,x') < r\}$ is denoted by $B_r(x)$. A topology on X is induced by d, which has the collection of all open balls as a base. A pseudo-quasi-metric d is called a *quasi-metric* if $d(x,y) > 0$ for any $x, y \in X$ with $x \neq y$.

A pseudo-quasi-metric d is called *weakly symmetric* if $d(y_i, x) \to 0$ for any $x \in X$ and sequence $y_0, y_1, \ldots \in X$ with $d(x, y_i) \to 0$ (Arutyunov et al., 2017). When d is weakly symmetric, d is continuous as a function with respect to the topology it induces.

For two points $x, y \in X$, a point $z \in X$ is called a *midpoint* between x and y if $d(x,z) = d(z,y) = d(x,y)/2$. The space (X,d) is said to have the *midpoint property* if at least one midpoint exists for every pair of points. The space (X,d) is said to have the *continuous midpoint property* if there exists a continuous map $m : X \times X \to X$ such that $m(x,y)$ is a midpoint between x and y for any $x, y \in X$ (Horvath, 2009).

Let (X, d_X) and (Y, d_Y) be pseudo-quasi-metric spaces. A pseudo-quasi-metric $d_{X \times Y}$ can be defined on $X \times Y$ by

$$d_{X \times Y}((x_1, x_2), (y_1, y_2)) := d_X(x_1, y_1) + d_Y(x_2, y_2)$$

and the induced topology coincides with the topology as a direct product.

A function $f : (X, d_X) \to (Y, d_Y)$ is called *uniformly continuous* if the following holds: For any $\varepsilon > 0$, there exists $\delta > 0$ such that, for any $x_1, x_2 \in X$ with $d_X(x_1, x_2) < \delta$, we have $d_Y(f(x_1), f(y_2)) < \varepsilon$.

A series $(f_0, f_1, \ldots)$ of functions from (X, d_X) to $\mathbb{R}$ is called *eventually equicontinuous* if the following holds (Yang, 1969): For any $x \in X$ and $\varepsilon > 0$, there exists $I \in \mathbb{N}$ and $\delta > 0$ such that, for any $i \geq I$ and $x' \in B_\delta(x)$, we have $|f_i(x) - f_i(x')| < \varepsilon$. Note that each f_i is not necessarily continuous.

3.2 Finsler Geometry

An important family of quasi-metric spaces is Finsler manifolds, including Riemannian manifolds as special cases. Just like Riemannian manifolds, Finsler manifolds are also equipped with an infinitesimal metric, from which the distance is derived. While Riemannian manifolds have symmetric metrics, Finsler manifolds allow for asymmetric ones, enabling the formulation of planning tasks with directional costs.

A *Finsler manifold* is a differential manifold M equipped with a function $F : TM \to [0, \infty)$, where $TM = \bigcup_{x \in M} T_x M$ is the tangent bundle of M, and F satisfies the following conditions (Bao et al., 2000).

1. F is smooth on $TM \setminus 0$.

2. $F(x, \lambda v) = \lambda F(x, v)$ for all $\lambda > 0$ and $(x, v) \in TM$.

3. The Hessian matrix of $F(x, -)^2$ is positive definite at every point of $TM_x \setminus 0$ for all $x \in M$.

Let $\gamma : [0, 1] \to M$ be a piecewise smooth curve on M. We define the length of γ as

$$L(\gamma) := \int_0^1 F\left(\gamma(t), \frac{d\gamma}{dt}(t)\right) dt.$$

For two points $x, y \in M$, we define the distance $d(x, y)$ as

$$d(x, y) := \inf \{L(\gamma) | \gamma : [0, 1] \to M, \gamma(0) = x, \gamma(1) = y\}.$$

Then, d is a weakly symmetric quasi-metric (Bao et al., 2000). A curve $\gamma : [0, 1] \to M$ is called a *minimizing geodesic* if $L(\gamma) = d(\gamma(0), \gamma(1))$.

Clearly, (M, d) has the midpoint property. It is known (Bao et al., 2000; Amici and Casciaro, 2010) that, for any point of M, there exists a neighborhood $U \subseteq M$ such that any two points $p, q \in U$ can be uniquely connected by a minimizing geodesic inside U. Note that (U, d) has the continuous midpoint property. That is, Finsler manifolds always have the continuous midpoint property locally (but not always globally). See also Remark 10.

Example 1 *The Matsumoto metric is an asymmetric Finsler metric that considers times to move on inclined planes (Matsumoto, 1989). Let $M \subseteq \mathbb{R}^2$ be a region on the plane with the standard coordinates x, y, and let $h : M \to \mathbb{R}$ be a differentiable function that indicates heights of the field. The Matsumoto metric $F : TM \to [0, \infty)$ is then defined as follows:*

$$F(p, (v_x, v_y)) := \frac{\alpha^2}{\alpha - \beta}$$

where,

$$\beta := v_x \frac{\partial h}{\partial x}(p) + v_y \frac{\partial h}{\partial y}(p), \; \alpha := \sqrt{v_x^2 + v_y^2 + \beta^2}.$$

This metric takes larger values on uphill slopes and smaller values on downhill slopes.

Example 2 *For unidirectional car-like agents, the cost of trajectories considering kinematics as in Rösmann et al. (2017) can be formulated as a Finsler metric. See § 5.4.2 for the details.*

4. Theoretical Results

We briefly explain our main idea to generate the shortest path between any two points in § 4.1. We prove propositions justifying our idea in § 4.2 and § 4.3. This result relies on the assumption that a function matching the true distance in the infinitesimal limit is known. We present a construction of this assumed function for Finsler manifolds in § 4.4. In § 4.5, we also show that we can handle the case where paths to be generated are restricted to a subset of a Finsler manifold, which enable our approach to be applied to tasks with hard collision avoidance constraints.

4.1 Midpoint Tree

We propose the *midpoint tree* framework to solve the all-pairs shortest path problem in a given pseudo-quasi-metric space. In this framework, a policy learns to predict a midpoint between any two points. The shortest path between any start and goal points can be generated by recursively applying this prediction to adjacent pairs of previously generated waypoints, as illustrated in Figure 1. We train the policy (actor) to predict midpoints by an actor-critic approach. In other words, we simultaneously train a function (critic) to predict distances.

In the following two subsections, we describe the theoretical considerations of this actor-critic learning approach. In § 4.2, we prove that, if certain functional equations hold and the critic coincide with the true distance in the infinitesimal limit, then the actor and the critic coincide with the true midpoint and distance, respectively. This result is important as it justifies restricting actor predictions to midpoints (Remark 3). In § 4.3, we prove that these conditions hold if the actor and critic converge through iterative improvement.

4.2 Functional Equation

Let (X, d) be a pseudo-quasi-metric space. When we do not know d globally, we want to train a function (actor) $\pi : X \times X \to X$ to predict midpoints. We also train a function (critic) $V : X \times X \to \mathbb{R}$ to predict distances.

If π and V coincide with the true midpoints and distances, respectively, the following functional equations hold:

$$V(x, y) = V(x, \pi(x, y)) + V(\pi(x, y), y), \tag{1}$$

$$\pi(x, y) \in \arg\min_z \left(V(x, z)^2 + V(z, y)^2 \right), \tag{2}$$

$$d(x, \pi(x, x)) = d(\pi(x, x), x) = 0. \tag{3}$$

Note that $d(x, z)^2 + d(z, y)^2$ takes its minimum value when z is a midpoint between x and y, because

$$d(x, z)^2 + d(z, y)^2 = \frac{1}{2}(d(x, z) + d(z, y))^2 + \frac{1}{2}(d(x, z) - d(z, y))^2.$$

The first term takes the minimum value $d(x, y)^2/2$ when z lies on some shortest path from x to y and the second term takes the minimum value 0 when z is equidistant from x and y.

Remark 1 *The above argument is generalized as follows: A point dividing x and y internally in the ratio $\alpha : 1$ minimizes $d(x, -)^2 + \alpha d(-, y)^2$, because*

$$d(x, z)^2 + \alpha d(z, y)^2 = \frac{\alpha}{1 + \alpha}(d(x, z) + d(z, y))^2 + \frac{1}{1 + \alpha}(d(x, z) - \alpha d(z, y))^2.$$

The following proposition states that, under mild assumptions, the conjunction of these functional equations and the condition that V and d are equal in the infinitesimal limit constitutes a sufficient (and obviously necessary) condition for π and V to coincide with the midpoints and distances, respectively.

Proposition 2 *Assume that (X, d) has the midpoint property and π and V satisfy (1), (2), and (3). Assume also that π is uniformly continuous.*

Let $\varepsilon \in (0, 1/9)$. If there exists $\delta > 0$ such that, V approximates d locally for pairs with distances less than δ, which means that

$$(1 - \varepsilon)d(x, x') \leq V(x, x') \leq (1 + \varepsilon)d(x, x') \text{ for any } x, x' \in X \text{ with } d(x, x') < \delta, \quad (4)$$

then V approximates d globally, which means that

$$(1 - \varepsilon)d(x, y) \leq V(x, y) \leq (1 + \varepsilon)d(x, y) \text{ for any } x, y \in X.$$

In particular, if such $\delta > 0$ exists for any $\varepsilon > 0$, then $V = d$ and $\pi(x, y)$ is a midpoint between x and y for any $x, y \in X$.

Proof Before proof, we explicitly rewrite the assumption of uniform continuity of π: For any $\alpha > 0$, there exists $\beta > 0$ such that, for any $x, y, z \in X$ with $d(y, z) < \beta$, we have

$$\max\{d(\pi(x, y), \pi(x, z)), d(\pi(y, x), \pi(z, x))\} < \alpha.$$

We assume that $\delta > 0$ satisfies the condition for a sufficiently small $\varepsilon > 0$.

We first prove that $|V(x, y)| \leq (1 + \varepsilon)d(x, y)$ for any $x, y \in X$. We prove that $|V(x, y)| \leq (1 + \varepsilon)d(x, y)$ when $d(x, y) < 2^n\delta$ by induction for $n \in \mathbb{N}$.

For the case $n = 0$, this is a direct consequence of (4) since $-(1 + \varepsilon)d(x, y) \leq (1 - \varepsilon)d(x, y)$.

We assume the induction hypothesis is true for n and take $x, y \in X$ such that $d(x, y) < 2^{n+1}\delta$. Let m be the midpoint between x and y. Then,

$$\begin{aligned}
V(x, y)^2 &= (V(x, \pi(x, y)) + V(\pi(x, y), y))^2 \\
&\leq 2V(x, \pi(x, y))^2 + 2V(\pi(x, y), y)^2 \\
&\leq 2V(x, m)^2 + 2V(m, y)^2 \\
&\leq 2(1 + \varepsilon)^2 \left(d(x, m)^2 + d(m, y)^2 \right) \\
&= (1 + \varepsilon)^2 d(x, y)^2
\end{aligned} \quad (5)$$

where the first equality comes from (1), the first inequality comes from $(a + b)^2 \leq 2a^2 + 2b^2$, the second inequality comes from (2), and the third inequality comes from the induction hypothesis and $d(x, m) = d(m, y) < 2^n\delta$. Thus, $|V(x, y)| \leq (1 + \varepsilon)d(x, y)$.

By induction, for all $x, y \in X$,

$$|V(x, y)| \leq (1 + \varepsilon)d(x, y). \quad (6)$$

Next, we prove that $(1 - \varepsilon)d(x, y) \leq V(x, y)$ for any $x, y \in X$ with the following strategy: To prove $(1 - \varepsilon)d(x, y) \leq V(x, y)$, it is enough to show that this inequality holds for the pairs $(x, \pi(x, y))$ and $(\pi(x, y), y)$. Thus, we use induction and apply the induction hypothesis to these pairs. However, it is not straightforward to bound $d(x, \pi(x, y))$ and $d(\pi(x, y), y)$. On the other hand, conversely, if the values of V are close to those of d, π is close to the midpoint. Using this and the uniform continuity of π, we gradually extend the range in which the inequality holds, starting from the case where $\pi(x, y)$ is sufficiently close to both x and y.

From the assumption of uniform continuity of π, we can take $\delta' > 0$ such that, for any $x, y, z \in X$ with $d(y, z) < \delta'$,

$$\max\{d(\pi(x, y), \pi(x, z)), d(\pi(y, x), \pi(z, x))\} < \delta.$$

Especially, by considering the cases where (x, y, z) is (x, x, y) or (y, x, y), when $d(x, y) < \delta'$,

$$\max\{d(\pi(x, x), \pi(x, y)), d(\pi(x, y), \pi(y, y))\} < \delta.$$

From (3) and the triangular inequality,

$$\max\{d(x, \pi(x, y)), d(\pi(x, y), y)\} < \delta.$$

Let $c_\varepsilon := (1 + 2\sqrt{\varepsilon} + \varepsilon)/2$, which appears later in (10). Since $\varepsilon < 1/9$, we have $(1 - \varepsilon)^{-1} c_\varepsilon < 1$. Using the uniform continuity of π again, we can take $\eta' > 0$ such that, for any $x, y, z \in X$ with $d(y, z) < \eta'$,

$$\max\left\{d(\pi(x, y), \pi(x, z)), d(\pi(y, x), \pi(z, x))\right\} < \left(1 - \frac{c_\varepsilon}{1 - \varepsilon}\right)\delta'.$$

Let $\eta := \min\{\eta'/2, \delta'\}$.

We prove by induction for $n \in \mathbb{N}$ that, when $d(x, y) < \delta' + n\eta$,

$$(1 - \varepsilon)d(x, y) \leq V(x, y), \tag{7}$$

$$(1 - \varepsilon)d(x, \pi(x, y)) \leq V(x, \pi(x, y)), \tag{8}$$

$$(1 - \varepsilon)d(\pi(x, y), y) \leq V(\pi(x, y), y). \tag{9}$$

Note that (7) follows from (8) and (9) because

$$
\begin{aligned}
(1 - \varepsilon)d(x, y) &\leq (1 - \varepsilon)(d(x, \pi(x, y)) + d(\pi(x, y), y)) \\
&\leq V(x, \pi(x, y)) + V(\pi(x, y), y) \\
&= V(x, y).
\end{aligned}
$$

For the case $n = 0$, if $d(x, y) < \delta'$, by the condition of δ', $d(x, \pi(x, y))$ and $d(\pi(x, y), y)$ are smaller than δ. Thus, by the condition of δ, (8) and (9) hold.
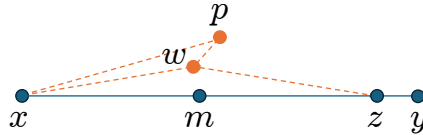

Figure 2: Positions of x, y, z, w, m, p.

We assume the induction hypothesis is true for n and take x, y such that $d(x, y) < \delta' + (n+1)\eta$. If $d(x, y) < \delta' + n\eta$, the conclusions hold by the induction hypothesis. Thus, we

can assume that $\delta' + n\eta \leq d(x,y)$. By taking midpoints recursively, we can take $z \in X$ such that $d(x,z) + d(z,y) = d(x,y)$ and $\eta \leq d(z,y) < 2\eta$ as shown in Figure 2. Let $w := \pi(x,z)$, $a := V(x,w)$, $b := V(w,z)$, and $l := d(x,z)$. By (1) and (6), $a + b = V(x,z) \leq (1+\varepsilon)l$. On the other hand, as $l < \delta' + n\eta$, by (7) in the induction hypothesis, $a + b = V(x,z) \geq (1-\varepsilon)l$. Let m be a midpoint between x and z. Then, by (2) and (6),

$$a^2 + b^2 \leq V(x,m)^2 + V(m,y)^2 \leq (1+\varepsilon)^2 \left(d(x,m)^2 + d(m,z)^2\right) = \frac{(1+\varepsilon)^2 l^2}{2}.$$

Thus,

$$
\begin{aligned}
a &\leq \frac{1}{2}\left(a + b + |a - b|\right) \\
&= \frac{1}{2}\left(a + b + \sqrt{2(a^2 + b^2) - (a+b)^2}\right) \\
&\leq \frac{1}{2}\left((1+\varepsilon)l + \sqrt{(1+\varepsilon)^2 l^2 - (1-\varepsilon)^2 l^2}\right) \\
&= c_\varepsilon l.
\end{aligned}
\tag{10}
$$

By (8) in the induction hypothesis, $d(x,w) \leq (1-\varepsilon)^{-1} a \leq (1-\varepsilon)^{-1} c_\varepsilon l$. Let $p := \pi(x,y)$. Then,

$$d(x,p) \leq d(x,w) + d(w,p) < \frac{c_\varepsilon}{1-\varepsilon} l + \left(1 - \frac{c_\varepsilon}{1-\varepsilon}\right)\delta' < \delta' + n\eta,$$

where the second inequality comes from $d(z,y) < 2\eta$ and the way η is given. Thus, by (7) in the induction hypothesis, $(1-\varepsilon)d(x,p) \leq V(x,p)$. By a symmetrical argument, we can also prove $(1-\varepsilon)d(p,y) \leq V(p,y)$.

Therefore, $(1-\varepsilon)d(x,y) \leq V(x,y)$ for any $x, y \in X$.

Thus, if the assumption holds for any $\varepsilon > 0$, $V = d$. Then, by (2) and the midpoint property, $\pi(x,y)$ is a midpoint between x and y for any $x, y \in X$. ∎

Remark 3 *The conclusion does not follow if (2) is replaced with*

$$\pi(x,y) \in \arg\min_z \left(V(x,z) + V(z,y)\right). \tag{11}$$

Let $f : [0,\infty) \to [0,\infty)$ be a non-decreasing subadditive function such that $\lim_{h\to+0} f(h)/h = 1$ (for example, $f(h) := 2\sqrt{1+h} - 2$) and let $V := f \circ d$. We consider the case $\pi(x,y) = x$. Then, for any $x, y, z \in X$,

$$V(x, \pi(x,y)) + V(\pi(x,y), y) = V(x,y) \leq f(d(x,z) + d(z,y)) \leq V(x,z) + V(z,y).$$

Thus, (11) and all the conditions of Proposition 2 except (2) are satisfied. However, $V \neq d$ generally. Note that $V \leq d$ follows even from these conditions, which supports the following insights. If the upper bounds of distances are given, then the approach to predict arbitrary intermediate points can work as in Jurgenson et al. (2020). However, if distances can be approximated only for two close points, then it is necessary to avoid generating points near endpoints.

9

Remark 4 *In Proposition 2, it is necessary to assume that π is uniformly continuous. Let (X, d) be a pseudo-quasi-metric space with the midpoint property, and let $m(x, y)$ be the midpoint between $x, y \in X$. We consider the case where*

$$V(x, y) = \begin{cases} d(x, y) & d(x, y) \leq \sqrt{2}, \\ 1 & otherwise \end{cases}$$

and

$$\pi(x, y) = \begin{cases} m(x, y) & d(x, y) \leq \sqrt{2}, \\ x & otherwise. \end{cases}$$

Then, for any $x, y, z \in X$,

$$V(x, z)^2 + V(z, y)^2 \geq \min \left\{ d(x, y)^2 / 2, 1 \right\}$$
$$= V(x, \pi(x, y))^2 + V(\pi(x, y), y)^2.$$

Thus, except for the assumption that π is uniformly continuous, the conditions of Proposition 2 are satisfied.

4.3 Iteration

Although we do not know the metric d globally, we assume that we know it infinitesimally, that is, we have a continuous function $C : X \times X \to \mathbb{R}$ that coincides with d in the limit $d(x, y) \to 0$ or $C(x, y) \to 0$. Formally, we assume the following conditions:

Assumption 5

1. For $x \in X$ and a series $y_0, y_1, \ldots \in X$, if $C(x, y_i) \to 0$, then $d(x, y_i) \to 0$.

2. For $x \in X$ and $\varepsilon > 0$, there exists $\delta > 0$ such that, for any $y, z \in B_\delta(x)$,

$$(1 - \varepsilon)d(y, z) \leq C(y, z) \leq (1 + \varepsilon)d(y, z).$$

Starting with C, we consider iterative construction of actors and critics to satisfy (1) and (2). Formally, we assume series of functions, $V_i : X \times X \to \mathbb{R}$ and $\pi_i : X \times X \to X$, indexed by $i = 0, 1, \ldots$, that satisfy the following conditions:

$$V_0(x, y) = C(x, y), \tag{12}$$

$$\pi_i(x, y) \in \arg\min_z \left(V_i(x, z)^2 + V_i(z, y)^2 \right), \tag{13}$$

$$V_{i+1}(x, y) = V_i(x, \pi_i(x, y)) + V_i(\pi_i(x, y), y). \tag{14}$$

The following proposition states that, under mild assumptions, the limits of $\pi_0, \pi_1, \ldots$ and $V_0, V_1, \ldots$, if they exist, coincide with our desired outcome.

Proposition 6 *Assume that (X, d) is a compact, weakly symmetric pseudo-quasi-metric space with the midpoint property, and that series of functions $\pi_i : X \times X \to X$ and $V_i : X \times X \to \mathbb{R}$ satisfy (12), (13), and (14). Assume also that the series $(V_0, V_1, \dots)$ is eventually equicontinuous. For functions $\pi : X \times X \to X$ and $V : X \times X \to \mathbb{R}$, we assume that $\pi_i(x, y) \to \pi(x, y)$ and $V_i(x, y) \to V(x, y)$ when $i \to \infty$ for any $x, y \in X$. Then, if π is continuous, V and π satisfy all conditions in Proposition 2 and thus coincide with the exact distances and midpoints.*

See Appendix A.1 for the proof.

Example 3 *We assume that (X, d) has the midpoint property. We consider the case in Remark 3 where $f(h) := 2\sqrt{1 + h} - 2$ and $C = f \circ d$. Then, π_0 coincide with the true midpoints. Indeed, by the triangle inequality and the monotonicity of f,*

$$C(x, z)^2 + C(z, y)^2 \le f\left(d(x, z)\right)^2 + f\left(d(x, y) - d(x, z)\right)^2$$
$$= 16 + 4d(x, y) - 8\left(\sqrt{1 + d(x, z)} + \sqrt{1 + d(x, y) - d(x, z)}\right),$$

and the last term takes the maximum value when $d(x, z) = d(x, y)/2$. Thus, $C(x, z)^2 + C(z, y)^2$ takes the minimum value when z is the midpoint between x and y. Therefore,

$$V_1(x, y) = 2f\left(\frac{d(x, y)}{2}\right) = 4\sqrt{1 + \frac{d(x, y)}{2}} - 4.$$

As before, $V_1(x, z)^2 + V_1(z, y)^2$ takes the minimum value when z is the midpoint between x and y, and π_1 coincide with the true midpoints. Since this continues,

$$V_i(x, y) = 2^{i+1}\sqrt{1 + \frac{d(x, y)}{2^i}} - 2^{i+1}$$

and π_n always coincide with the true midpoints. By the Taylor expansion of the square root function at 1, V_i converges to d. Note that, as shown in Remark 3, the iteration does not work if (13) is replaced by

$$\pi_i(x, y) \in \arg\min_z \left(V_i(x, z) + V_i(z, y)\right). \tag{15}$$

Example 4 *We assume that (X, d) has the midpoint property and the distance of any two points does not exceed a value $c \in \mathbb{R}$. Let $m(x, y)$ be the midpoint between x and y. We consider the case where*

$$C(x, y) = \begin{cases} d(x, y) & d(x, y) \le 1, \\ c & \text{otherwise.} \end{cases}$$

Then, if

$$\pi_0(x, y) = \begin{cases} m(x, y) & d(x, y) \le 2, \\ x & \text{otherwise,} \end{cases}$$

(13) is satisfied. Thus,

$$V_1(x, y) = \begin{cases} d(x, y) & d(x, y) \le 2, \\ c & \text{otherwise.} \end{cases}$$

Since this continues,

$$V_i(x, y) = \begin{cases} d(x, y) & d(x, y) \le 2^i, \\ c & \text{otherwise} \end{cases}$$

and

$$\pi_i(x, y) = \begin{cases} m(x, y) & d(x, y) \le 2^{i+1}, \\ x & \text{otherwise}. \end{cases}$$

*Therefore, V_i converges to d and π_i converges to m. Obviously, neither V_i nor π_i is continuous for small i. Unlike the previous example, since C give an upper bound of distance, this iteration works even if (13) is replaced by (15). However, practically, this discontinuous C is not suitable for learning. See the result of **Cut** in our experiments.*

4.4 Finsler Case

We consider the case where (X, d) is a Finsler manifold (M, F). We assume that there exists a global coordinate system (a diffeomorphism to a subset) $f : M \hookrightarrow \mathbb{R}^d$. We consider the case where C is defined as

$$C(x, y) := F\left(x, df_x^{-1}(f(y) - f(x))\right), \tag{16}$$

where $df_x : T_xM \to T_{f(x)}\mathbb{R}^d = \mathbb{R}^d$ is the differential of f at x.

The following proposition states that Proposition 6 is applicable to this case.

Proposition 7 *C satisfies Assumption 5.*

See Appendix A.2 for the proof.

Remark 8 *The Assumption 5 is preserved under weighted averaging. Thus, even if a Finsler manifold does not have a global coordinate system, we can construct C satisfying Assumption 5 by using local coordinate systems and a partition of unity. Note that values of C outside a neighborhood of the diagonal can be anything that does not converge to zero.*

Remark 9 *When M is compact, once the desired midpoint function $\pi : M \times M \to M$ is found, we can construct minimizing geodesics for all pairs of points. Let $A := \{N/2^n | n \ge 0, 0 \le N \le 2^n\}$. For any $x, y \in M$, by applying π recursively, we can construct $\gamma : A \to M$ such that $d(x, \gamma(a)) = ad(x, y)$ and $d(\gamma(a), y) = (1 - a)d(x, y)$ for any $a \in A$. For any $r \in [0, 1]$, we can take a non-decreasing sequence $a_1, a_2, \ldots \in A$ such that $\lim_i a_i = r$. Then, because $\gamma(a_i)$ is a forward Cauchy sequence with respect to d, it converges (Bao et al., 2000). Therefore, we can extend the domain of γ to $[0, 1]$.*

Remark 10 *Since we assume that π is continuous in Proposition 6, it is appliable only for spaces with the continuous midpoint property. In Finsler case, this property always holds true locally, as mentioned in § 3.2. On the other hand, our method is practically feasible even if this property does not hold. Indeed, the environment in § 5.4.1 does not have this property (Remark 17).*

Remark 11 *Instead of using (16), if we set*

$$C(x,y) := \int_0^1 F\left(l(t), df_{l(t)}^{-1}(f(y) - f(x))\right) dt, \tag{17}$$

where

$$l(t) := f^{-1}((1-t)f(x) + tf(y)),$$

then C gives the length of the curve connecting points as a segment in the coordinate space, which is an upper bound of the distance. In that case, we can use the setting of Jurgenson et al. (2020). However, the integration in (17) is not always efficiently computable.

Remark 12 *Instead of using (16), we could define C as*

$$C(x,y) := \frac{1}{2}\left(F\left(x, df_x^{-1}(f(y) - f(x))\right) + F\left(y, df_y^{-1}(f(y) - f(x))\right)\right).$$

Because this definition is not biased toward the x side, it might be more appropriate for approximating distances. The main reason we do not adopt this definition in this paper is the difficulty of using it in sequential reinforcement learning.

Remark 13 *For pseudo-Finsler manifolds, which are similar to Finsler manifolds but do not necessarily satisfy the condition of positive definiteness, the distance function d can be defined and is a weakly symmetric quasi-metric (Javaloyes and Sánchez, 2011). However, we do not expect that Proposition 7 generally holds for pseudo-Finsler manifolds.*

4.5 Free Space

Here, we consider the situation where there exist a *free space* (a path-connected closed subset) $M_{\text{free}} \subseteq M$, and we want to generate paths inside M_{free}. Examples include motion planning with obstacles in environments and multi-agent motion planning where collisions between agents are to be avoided.

We modify d to
$$d'(x,y) := d(x,y) + P(x,y)$$
where
$$P(x,y) := \begin{cases} 0 & x, y \in M_{\text{free}} \text{ or } x, y \notin M_{\text{free}}, \\ c_P & \text{otherwise}, \end{cases}$$

and $c_P \gg 0$ is a constant. Clearly, d' is also a quasi-metric. When M_{free} is compact and c_P is large enough, midpoints of any pair of points in M_{free} with respect to d' lie in M_{free}.

Let
$$C'(x,y) := C(x,y) + P(x,y). \tag{18}$$

Obviously, C' satisfies Assumption 5 with respect to d'.

Remark 14 *Whereas a procedure to determine whether direct edges cause collisions is assumed in Jurgenson et al. (2020), we use only a procedure to determine whether states are safe and aim to generate dense waypoints in the free space. By introducing a margin between the free space and the obstacle region, one can ensure that the entire path lies outside the obstacle region, provided that consecutive waypoints are sufficiently close.*

5. Experiments

In § 5.1, by building on the previous section's results, we describe our proposed learning algorithm for midpoint prediction. In the following subsections, we compared our method, which generates geodesics via policies trained by our algorithm, with baseline methods on several path planning tasks. The first two environments deal with asymmetric metrics that change continuously (local planning tasks), while the other three environments have simple and symmetric metrics but involve obstacles (global planning tasks).

5.1 Learning Algorithm

Let (X, d) be a pseudo-quasi-metric space, and let C be a function satisfying Assumption 5.

We simultaneously train two networks with parameters θ and ϕ: the *actor* π_θ, which predicts the midpoints between two given points, and the *critic* V_ϕ, which predicts the distances between two given points. The critic learns to predict distances from the lengths of the sequences generated by the actor recursively predicting midpoints, where the length of a sequence is calculated as the sum of the values of C for consecutive waypoints. On the other hand, the actor learns to predict midpoints from the critic's predictions.

We train only one actor and one critic, unlike in § 4.3. See also Remark 15.

The network for actor π_θ has a form for which the reparameterization trick (Haarnoja et al., 2018) can be applied, that is, a sample is drawn by computing a deterministic function of the input, parameters θ, and an independent noise. Henceforth, we abuse notations and denote a sampled prediction from $\pi_\theta(\cdot|s, g)$ by $\pi_\theta(s, g)$ even if it is not deterministic.

Algorithm 1 gives the pseudocode for our method.

We define the critic loss L_c for $s, g \in X$ with an estimated distance c as

$$L_\mathrm{c} := L_\mathrm{SLE} + L_\mathrm{symm}^\mathrm{c}, \tag{19}$$

where

$$L_\mathrm{SLE} := \left(\log\left(V_\phi(s, g) + 1\right) - \log\left(c + 1\right)\right)^2.$$

We take logarithms to reduce influence of large values. If the distance d is known to be symmetric, then setting the second term to

$$L_\mathrm{symm}^\mathrm{c} := \left(\log\left(V_\phi(s, g) + 1\right) - \log\left(V_\phi(g, s) + 1\right)\right)^2$$

ensures that V_ϕ is also symmetric. Otherwise, $L_\mathrm{symm}^\mathrm{c} := 0$.

The actor loss L_a for $s, g \in X$ is defined as

$$L_\mathrm{a} := L_\mathrm{mid} + L_\mathrm{sm} + L_\mathrm{symm}^\mathrm{a}, \tag{20}$$

where the first term,

$$L_\mathrm{mid} := V_\phi(s, \pi_\theta(s, g))^2 + V(\pi_\theta(s, g), g)^2,$$

is intended to make $\pi_\theta(s, g)$ a midpoint between s and g. The second term,

$$L_\mathrm{sm} := V_\phi(\pi_\theta(s, g), \pi_\theta(\pi_\theta(s, \pi_\theta(s, g)), \pi_\theta(\pi_\theta(s, g), g)))^2$$

Algorithm 1 Actor-Critic Midpoint Learning

1: Initialize θ, ϕ
2: **while** learning is not done **do**
3:     $data \leftarrow \emptyset$
4:     **while** $data$ is not enough **do**
5:         $data \leftarrow data \cup \text{CollectData}(\pi_\theta)$
6:     **end while**
7:     Split $data$ into $batches$
8:     **for** $epoch = 1, \ldots, N_{\text{epochs}}$ **do**
9:         **for all** $b \in batches$ **do**
10:             Update ϕ with $\nabla_\phi \sum_{(s,g,c)\in b} L_{\text{c}}(s, g, c)$, which is defined in (19)
11:             Update θ with $\nabla_\theta \sum_{(s,g,c)\in b} L_{\text{a}}(s, g)$, which is defined in (20)
12:         **end for**
13:     **end for**
14: **end while**
15:
16: **procedure** COLLECTDATA(π)
17:     Decide depth D
18:     Sample two points p_0, p_{2^D}
19:     **for** $i = 0, \ldots, D - 1$ and $j = 0, \ldots, 2^i - 1$ **do**
20:         $p_{2^{D-i-1}(2j+1)} \leftarrow \pi_\theta(p_{2^{D-i}j}, p_{2^{D-i}(j+1)})$
21:     **end for**
22:     $data \leftarrow \{(p_0, p_0, 0), (p_1, p_1, 0), \ldots, (p_{2^D}, p_{2^D}, 0)\}$
23:     $c_{D,0}, c_{D,1} \ldots, c_{D,2^D-1} \leftarrow C(p_0, p_1), C(p_1, p_2), \ldots, C(p_{2^D-1}, p_{D,2^D})$
24:     $data \leftarrow data \cup \{(p_0, p_1, c_{D,0}), (p_1, p_2, c_{D,1}), \ldots, (p_{2^D-1}, p_{2^D}, c_{D,2^D-1})\}$
25:     **for** $i = D - 1, \ldots, 0$ and $j = 0, \ldots, 2^i - 1$ **do**
26:         $c_{i,j} \leftarrow c_{i+1,2j} + c_{i+1,2j+1}$
27:         $data \leftarrow data \cup \{(p_{2^{D-i}j}, p_{2^{D-i}(j+1)}, c_{i,j})\}$
28:     **end for**
29:     **return** $data$
30: **end procedure**

---

ensures that the midpoint between the point of $1 : 3$ division and the point of $3 : 1$ division equals the midpoint between the original pair. This term aims to smooth the generated paths. If the distance d is known to be symmetric, then the third term

$$L_{\text{symm}}^{\text{a}} := V_\phi(\pi_\theta(s, g), \pi_\theta(g, s))^2$$

ensures that π_θ is also symmetric. Otherwise, $L_{\text{symm}}^{\text{a}} := 0$.

The data for training is collected using the actor π_θ with the current parameters. We sample two points from X and generate a sequence of points by repeatedly inserting points between adjacent points via π_θ. Adjacent pairs of points at each iteration are collected as data. The estimated distance between a collected pair of points is simply calculated as the sum of values of C for adjacent pairs between the points in the final sequence. In other

words, we use a Monte Carlo method. The number of recursion, which is denoted by D and called *depth*, is gradually increased during the learning process.

After collecting enough data, we update the parameters of the actor and critic according to the gradients of the sum of the aforementioned losses, via an optimization algorithm. We repeat this process of data collection and optimization a sufficient number of times.

Remark 15 *For learning efficiency, we train the single actor and the single critic in this algorithm, while we theoretically consider series of actors and critics in § 4.3. Instead, we gradually increase the depth for data collection to train the critic, so that C is evaluated for closer pairs of points. In this way, V and π start by learning values of V_0 and π_0 and gradually learn the values of V_i and π_i for higher i. Note that, since C is closer to the true distance d for pairs of closer points, V_i and π_i converge more quickly in the region of pairs of closer points.*

Remark 16 *While we use the Monte Carlo method to calculate the critic's target values, we could use TD(λ) (Sutton and Barto, 2018) for $0 \leq \lambda \leq 1$ instead, as $c_{D,j} := C(p_j, p_{j+1})$ and, for $i = D - 1, \ldots, 0$,*

$$c_{i,j} := (1-\lambda)(V_\phi(p_{2^{D-i}j}, p_{2^{D-i-1}(2j+1)}) + V_\phi(p_{2^{D-i-1}(2j+1)}, p_{2^{D-i}(j+1)})) + \lambda(c_{i+1,2j} + c_{i+1,2j+1}).$$

5.2 Tasks and Evaluation Method

We compared the methods in terms of their success rate on the following task. A Finsler manifold (M, F) with a global coordinate system $f : M \hookrightarrow \mathbb{R}^d$ (and a free space $M_{\text{free}} \subseteq M$) is given as an environment. The number n of segments to approximate paths and a proximity threshold $\varepsilon > 0$ are also fixed. For our method, n must be a power of two: $n = 2^{D_{\max}}$, where $D_{\max}$ is the midpoint trees' depth for evaluation. When two points $s, g \in M_{\text{free}}$ are given, we want to generate a sequence $s = p_0, p_1, \ldots, p_n = g$ of points such that no value of C for two consecutive points is greater than ε, where C is defined by (16) and, for cases with obstacles, (18). If the points generated by a method satisfy this condition, then it is considered successful in solving the task; otherwise, it is considered to have failed. Note that, for cases with obstacles, when $c_P > \varepsilon$, successes imply that all waypoints are lying on the free space.

For each environment, we randomly generated 100 pairs of points from the free space, before the experiment. During training, we evaluated the models regularly by solving the tasks for the prepared pairs and recorded the success rates. We ran each method with different random seeds ten times for the environment described in § 5.4.1 and five times for the other environments.

For each environment, the total number T of timesteps was fixed for all methods. Therefore, at Line 2 of our method, we continue learning until the number of timesteps reaches the defined value. Timesteps were measured by counting the evaluation of C during training. In other words, for sequential reinforcement learning (see the next subsection), the timesteps have their conventional meaning. For the other methods, one generation of a path (one cycle) with depth D is counted as 2^D timesteps.

We mainly used success rate, not path length, for evaluation because metrics are only be defined locally and lengths thus cannot be calculated unless the success condition is satisfied.

Furthermore, the evaluation by success rate allows for fair comparison with the baseline method using sequential generation. However, we also compared lengths of generated paths in Appendix C.

5.3 Compared Methods

The baseline methods were as follows:

- **Sequential Reinforcement Learning (Seq)**: We formulated sequential generation of waypoints as a conventional, goal-conditioned reinforcement learning environment. The agent moves to the goal step by step in M. If the agent is at p, it can move to q such that $F\left(p, df_p^{-1}(f(q) - f(p))\right) = \varepsilon$. If q is outside the free space M_{free}, the reward R is set to $-c_P$ and the episode ends as a failure. Otherwise, R is defined as

$$R := -\varepsilon + F\left(g, df_g^{-1}(f(g) - f(p))\right) - F\left(g, df_g^{-1}(f(g) - f(q))\right), \qquad (21)$$

  where g is the goal.[1] The discount factor is set to 1. An episode ends and is considered a success when the agent reaches a point p that satisfies $F\left(p, df_p^{-1}(f(g) - f(p))\right) < \varepsilon$. When the episode duration reaches n steps without satisfying this condition, the episode ends and is considered a failure.

  We used Proximal Policy Optimization (PPO) (Schulman et al., 2017) to solve reinforcement learning problems with this formulation, which is a widely used sophisticated algorithm and has been used for path planning in recent studies (Kulathunga, 2022; Qi et al., 2022).

- **Policy Gradient (PG)**: We modified the method in Jurgenson et al. (2020) to predict midpoints. For each $D = 1, \ldots, D_{\max}$, a stochastic policy π_D is trained to predict midpoints with depth D. To generate waypoints, we apply the policies in descending order of the index. We train the policies in ascending order of the index by the policy gradient.

  To predict midpoints, the value to be minimized in training is changed. Let $\rho(\pi_1, \ldots, \pi_D)$ be the distribution of $\tau := (p_0, \ldots, p_{2^D})$, where p_0 and p_{2^D} are sampled from the predefined distribution on M and $p_{2^{i-1}(2j+1)}$ is sampled from the distribution $\pi_i\left(\cdot \middle| p_{2^i j}, p_{2^i(j+1)}\right)$. Let θ_D denote the parameters of π_D. Instead of minimizing the expected value of $\sum_{i=0}^{2^D-1} C(p_i, p_{i+1})$ as in the original method, we train π_D to minimize the expected value of

$$c_\tau := \left(\sum_{i=0}^{2^{D-1}-1} C(p_i, p_{i+1})\right)^2 + \left(\sum_{i=2^{D-1}}^{2^D-1} C(p_i, p_{i+1})\right)^2.$$

  Here, we use

$$\nabla_{\theta_D} \mathbb{E}_{\rho(\pi_1,\ldots,\pi_D)}\left[c_\tau\right] = \mathbb{E}_{\rho(\pi_1,\ldots,\pi_D)}\left[(c_\tau - b\left(p_0, p_{2^D}\right)) \nabla_{\theta_D} \log \pi_D\left(p_{2^{D-1}} | p_0, p_{2^D}\right)\right], \quad (22)$$

  where b is a baseline function.

---

1. We do not define reward by using C because $C(p, g)$ does not necessarily decrease when p gets closer to g.

Note that, when the model is evaluated during training, if the current trained policy is π_D ($1 \leq D \leq D_{\max}$), then evaluation is performed with depth D (2^D segments). (The other methods are always evaluated with $n = 2^{D_{\max}}$ segments.)

For our proposed method described in §5.1, we tried two scheduling strategies to increase the trees' depth for training from zero to $D_{\max}$, under the condition of fixed total timesteps.

- **Timestep-Based Depth Scheduling (Our-T)**: For each depth, training lasts the same number of timesteps.

- **Cycle-Based Depth Scheduling (Our-C)**: For each depth, training lasts the same number of calls to the data collection procedure (cycles).

More precisely, at Line 17 in Algorithm 1, for timestep-based depth scheduling, the depth is $\lfloor t/t_d \rfloor$, where t is the number of timesteps at the current time and $t_d := \lfloor T/D_{\max} \rfloor + 1$. For cycle-based depth scheduling, the depth for the c-th call to the data collection procedure is $\lfloor c/c_d \rfloor$, where $c_d := \lfloor T/(2^{D_{\max}+1} - 1) \rfloor + 1$. Note that **Our-T** provides more training for low depths than **Our-C** does.

In addition, we ran the following variants of our method.

- **Intermediate Point (Inter)**: Instead of (20), we use the following actor loss:

$$L_{\mathrm{a}} := V_\phi(s, \pi_\theta(s,g)) + V_\phi(\pi_\theta(s,g), g) + L^{\mathrm{a}}_{\mathrm{symm}},$$

where, if d is known to be symmetric,

$$L^{\mathrm{a}}_{\mathrm{symm}} := V_\phi(\pi_\theta(s,g), \pi_\theta(g,s)).$$

Otherwise, $L^{\mathrm{a}}_{\mathrm{symm}} := 0$. This means that π_θ learns to predict intermediate points that are not necessarily midpoints.

- **2:1 Point (2:1)**: Instead of (20), we use the following actor loss:

$$L_{\mathrm{a}} := V_\phi(s, \pi_\theta(s,g))^2 + 2V_\phi(\pi_\theta(s,g), g)^2.$$

By Remark 1, this means that π_θ learns to predict $2:1$ points instead of midpoints.

- **Intermediate Point with Cut (Cut)**: To overcome the flaw of **Inter** in Remark 3, we use a modified version C_ε of the function C as

$$C_\varepsilon(x, y) := \begin{cases} C(x,y) & \text{if } C(x,y) < \varepsilon, \\ c_{\mathrm{cut}} & \text{otherwise,} \end{cases}$$

where $c_{\mathrm{cut}} \gg 0$ is a constant. We set $c_{\mathrm{cut}} := 30$.

For each environment, the depth scheduling method with better results in our proposed methods was adopted for these variants. Specifically, we used cycle-based depth scheduling for the environments in § 5.4.1 and § 5.4.2, and timestep-based depth scheduling for the environments in § 5.4.3, § 5.4.4, and § 5.4.5.

Note that **Seq** and **Cut** have a slight advantage because they use values of ε while others do not.

5.4 Environments

We experimented in the following five environments.

5.4.1 MATSUMOTO METRIC

We consider the Matsumoto metric (Example 1) where the region M is the unit disk and the height function is $h(p) := -\|p\|^2$. Intuitively, this corresponds to a mountainous field with a peak at the center.
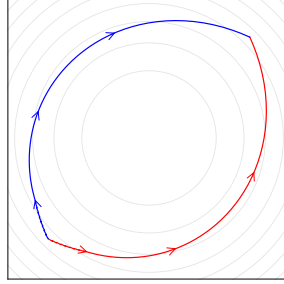

Figure 3: Example of switching geodesics.

Remark 17 *This environment does not have the continuous midpoint property, because minimizing geodesics connecting two points on the opposite sides of the peak can switch between counterclockwise and clockwise routes, as illustrated in Figure 3.*

For this environment, we set the number of segments $n = 64$ ($D_{\max} = 6$), the proximity threshold $\varepsilon = 0.1$, and the total number of timesteps $T = 2 \times 10^7$.

5.4.2 UNIDIRECTIONAL CAR-LIKE CONSTRAINTS

Inspired by the cost function for trajectories of car-like non-holonomic vehicles (Rösmann et al., 2017), we define an asymmetric Finsler metric for unidirectional car-like vehicles.

Let $M := U \times S^1 \subseteq \mathbb{R}^3$ be a configuration space for car-like vehicles, where $U \subseteq \mathbb{R}^2$ is a region with the standard coordinate system x, y and S^1 is the unit circle with the coordinate θ. We define $F : TM \to [0, \infty)$ as follows:

$$F((p, \theta), (v_x, v_y, v_\theta)) := \sqrt{v_x^2 + v_y^2 + c_c(h^2 + \xi^2)},$$

where

$$h := -v_x \sin(\theta) + v_y \cos(\theta),$$

$$\xi := \max\left\{r_{\min}|v_\theta| - v_x \cos(\theta) - v_y \sin(\theta), 0\right\}, \tag{23}$$

c_c is a penalty coefficient, and $r_{\min}$ is a lower bound of radius of curvature. The term h penalizes moving sideways, while the term ξ penalizes moving backward and turning sharply.

Strictly speaking, this metric is not a Finsler metric, as F is not smooth on the entire $TM \setminus 0$. However, we define the distance d and function C by using F in the same way

for a Finsler metric. In the formula (16) for C, we take a value in $[-\pi, \pi]$ as the difference between two angles.

Remark 18 $F((p, \theta), -)^2$ *is convex, and its Hessian matrix is positive definite where it is smooth. In terms of Fukuoka and Setti (2021), F is a C^0-Finsler structure, which can be approximated by Finsler structures.*

Remark 19 *While the vehicle model in Rösmann et al. (2017) is bidirectional, that is, it can move backward, our model is unidirectional, that is, it can only move forward. Unidirectionality seems essential for modeling by a Finsler metric. If we replace (23) with $\xi := \max\{r_{\min}|v_\theta| - |v_x \cos(\theta) + v_y \sin(\theta)|, 0\}$, then F cannot be approximated by Finsler structures because it does not satisfy subadditivity $(F(x, v + w) \leq F(x, v) + F(x, w))$.*

We take the unit disk as U and set $c_c = 100$, $r_{\min} = 0.2$, $n = 64$ $(D_{\max} = 6)$, $\varepsilon = 0.2$, and $T = 8 \times 10^7$.

5.4.3 2D DOMAIN WITH OBSTACLES

We consider a simple 2D domain with rectangular obstacles, as shown in Figure 5c, where the white area is the free space, which is taken from an experiment in Jurgenson et al. (2020). The metric is simply Euclidean. Note that we also consider the outside boundary unsafe.

We set $n = 64$ $(D_{\max} = 6)$, $\varepsilon = 0.1$, $T = 4 \times 10^7$, and the collision penalty $c_P = 10$.

5.4.4 7-DoF ROBOTIC ARM WITH AN OBSTACLE

This environment is defined for motion planning of the Franka Panda robotic arm, which has seven degrees of freedom (DoFs), in the presence of a wall obstacle. The space is the robot's configuration space, whose axes correspond to joint angles. The metric is simply Euclidean in the configuration space. The obstacle is defined as $\{x > 0.1, -0.1 < y < 0.1\}$, and a state is considered unsafe if at least one of the segments connecting adjacent joints intersects the obstacle. We ignored self-collision.

We set $n = 64$ $(D_{\max} = 6)$, $\varepsilon = 0.2$, $T = 4 \times 10^7$, and $c_P = 10$.

5.4.5 THREE AGENTS IN THE PLANE

We consider three agents in $U := [-1, 1] \times [-1, 1] \subseteq \mathbb{R}^2$. The configuration space is $M := U^3$ and the metric is defined as the sum of Euclidean metrics of three agents. A state is considered safe if all distances of pairs of agents are not smaller than $d_{\text{thres}} := 0.5$, that is,

$$M_{\text{free}} := \{(p_0, p_1, p_2) \in M \mid \|p_i - p_j\| \geq d_{\text{thres}} \text{ for } i \neq j\}.$$

We set $n = 64$ $(D_{\max} = 6)$, $\varepsilon = 0.2$, $T = 8 \times 10^7$, and $c_P = 10$.

5.5 Results and Discussion

Figure 4 shows the learning curves of the success rates for all methods averaged over random seeds. The error bars represent the standard errors. While **Seq** achieved the best success rate in the Matsumoto and 2D obstacles environments, its success rates were low and our proposed method had the best in the other environments. It may be much more difficult
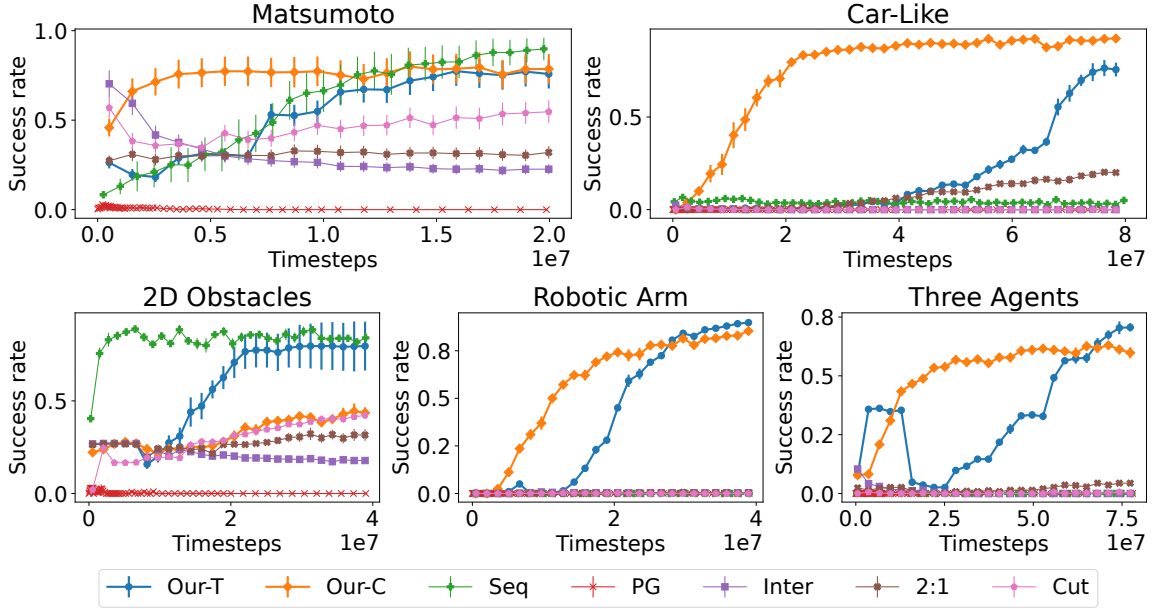
Figure 4: Success rate plots.

to determine directions of the agent sequentially in higher dimensional environments than in two-dimensional ones. Also, the approximation of distances in (21) may be close to the true values in the Matsumoto environment, but not in the car-like environment. Note that, while it may be possible to improve learning success rate for **Seq** by engineering rewards, our method was successful without adjusting rewards.

The success rates for **Inter** decreased as training progressed in the Matsumoto and 2D obstacle environments, which may have resulted from convergence to biased generation, as mentioned in Remark 3.

While the success rates for **Cut** did not decrease as for **Inter**, it was less successful than our method and failed to learn in the environments with dimensions greater than two. The discontinuity of prediction functions during training, as suggested by Example 4, may hinder learning.

The low success rates for **2:1** were, at least partially, due to its intentional uneven generation of waypoints. On the other hand, it was the only method, aside from our proposed ones, that showed an improvement in the car-like environment, indicating that learning is possible even when generation points are fixed to a ratio other than 1:1. See also Appendix C.

Figure 5a shows examples of paths in the Matsumoto environment that were generated by policies trained by compared methods. Each eighth point is marked, and the circles represent contours. The **Truth** curve represents the ground truth of the minimizing geodesic with points dividing eight equal-length parts. In this example, all methods except **PG** could generate curves close to the ground truth. While **Our-T** and **Our-C** generated waypoints near the points dividing the true curve equally, both **Inter** and **2:1** produced nonuniform waypoints.

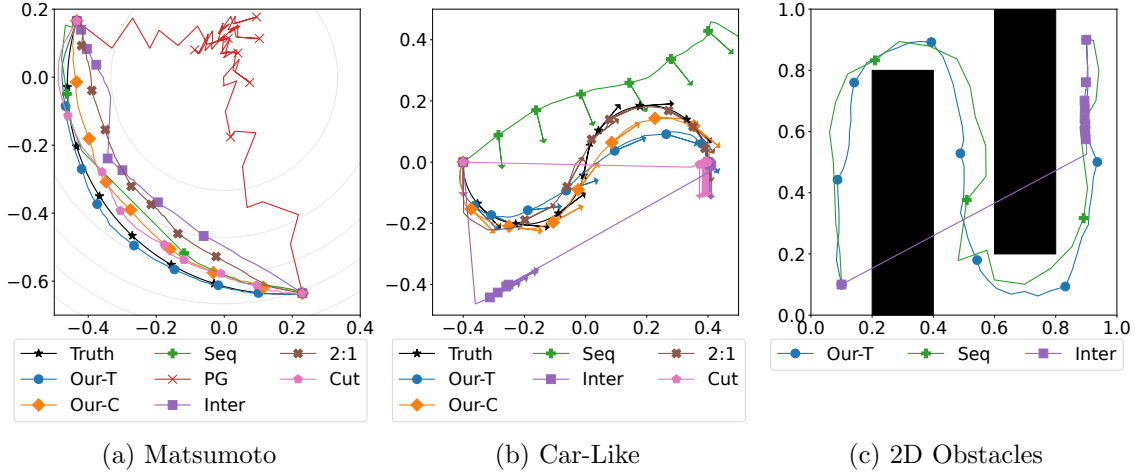(a) Matsumoto   (b) Car-Like   (c) 2D Obstacles

Figure 5: Examples of generated paths.

PG was unable to solve most tasks in all environments and failed to even generate a smooth curve in the Matsumoto environment, which may have been simply due to insufficient training, or due to instability of learning without critic for deep trees. Note that **PG** gets only one tuple of training data per path generation, whereas our method gets several, on the order of 2^D where D is the depth, per path generation. Note also that, compared with the experiments in Jurgenson et al. (2020), trees were deeper and the success condition was stricter in ours.

Figure 5b shows examples of paths generated by policies trained by methods except **PG** in the car-like environment. Orientations of the agent are represented by arrows. **Our-C** and **2:1** succeeded in generating paths close to the ground truth, while the waypoints for **2:1** were uneven. The path for **Our-T** was slightly straighter. Other methods failed to generate valid paths.

Figure 5c shows examples of paths generated by policies trained by **Our-T**, **Seq**, and **Inter** in the 2D domain with obstacles. While the waypoints for **Our-T** and **Seq** avoided the obstacles, those for **Inter** skipped the obstacles. This is natural because, in **Inter**, it is not required to generate waypoints densely.
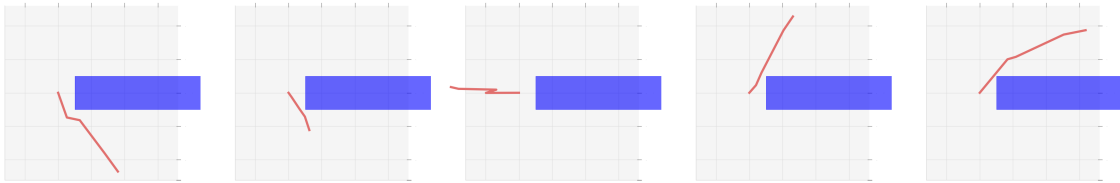

Figure 6: Generated motion for the robotic arm.

Figure 6 shows a top view visualization of an example of a robotic arm motion generated by a policy learned by **Our-T**. The obstacle is illustrated in blue. Despite the initial and

final positions of the arm tips being on opposite sides of the obstacle, the motion succeeded in avoiding the obstacle.
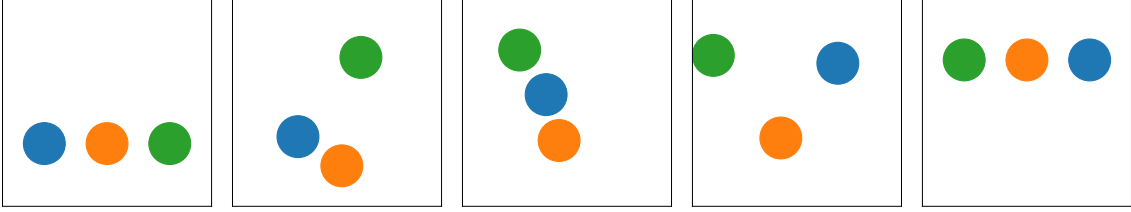

Figure 7: Generated motion for the three agents.

Figure 7 shows an example of a motion in the three agents environments generated by a policy learned by **Our-T**. Agents succeeded to swap their positions while moving upward without collisions.

See Appendix C for comparisons of lengths of generated paths.

6. Conclusion and Future Work

In this paper, we proposed a framework, called midpoint trees, to generate geodesics by recursively predicting midpoints. We also proposed an actor-critic method for learning to predict midpoints, and we theoretically proved its soundness. Experimentally, we showed that our method can solve path planning tasks that existing reinforcement learning methods fail to solve.

While we assumed continuity of the policy function to prove our approach's theoretical soundness, the continuous midpoint property may only be satisfied locally. Further research is needed on the conditions under which our method does not converge to wrong functions. In addition, we were not able to discuss the conditions under which iterations converge in this paper, which is a topic for future work.

Details of the learning algorithm in our proposed approach have not been fully investigated and may have room for improvement. For example, the followings could be considered.

- Our experiments showed that effective depth scheduling depends on the task. Exploration of an efficient depth scheduling algorithm that considers learning progress is a future challenge.

- While we tried only a straightforward actor-critic learning algorithm in this paper, algorithms for actor-critic reinforcement learning have been intensively researched. Investigation of more efficient algorithms thus also remains for future work. Especially, while we used an on-policy algorithm, off-policy algorithms (Degris et al., 2012) may be useful in our framework.

- While the architectures we used for both actors and critics were also simple, the quasi-metric learning method (Wang and Isola, 2022; Wang et al., 2023) may be useful for critics in our approach.

In our method, a policy must be learned for each environment. By modifying our method so that the actor and critic input information on environments as in Sartori et al. (2021), it may be possible to learn a policy that is applicable to different environments.

Acknowledgments

Appendix A. Proofs

We describe proofs omitted from the main text.

A.1 Proof of Proposition 6

We use the following lemma.

Lemma 20 *Let (X, d) be a pseudo-quasi-metric space. Let $\pi_i : X \times X \to X$ and $V_i : X \times X \to \mathbb{R}$ be two series of functions indexed by $i \in \mathbb{N}$ satisfying (13) and (14). Let $\pi : X \times X \to X$ and $V : X \times X \to \mathbb{R}$ be two functions such that $\pi_i(x, y) \to \pi(x, y)$ and $V_i(x, y) \to V(x, y)$ when $i \to \infty$ for any $x, y \in X$.*

1. *Assume that the series of functions $(V_0, V_1, \ldots)$ is eventually equicontinuous. Then, for any $x, y \in X$,*
$$\pi(x, y) \in \arg\min_z \left(V(x, z)^2 + V(z, y)^2 \right), \tag{24}$$
and
$$V(x, y) = V(x, \pi(x, y)) + V(\pi(x, y), y). \tag{25}$$

2. *Assume that (X, d) is weakly symmetric and that $V_0(x, x) = 0$, $V_0(x, y) \geq 0$, and $V_0(x, y) = 0 \implies d(x, y) = 0$ for any $x, y \in X$. Then, for any $x \in X$,*
$$d(x, \pi(x, x)) = d(\pi(x, x), x) = 0. \tag{26}$$

3. *For any $\varepsilon, \delta > 0$, if for any $x, y \in X$,*
$$V_0(x, y) < \delta \implies d(x, y) \leq (1 + \varepsilon)V_0(x, y),$$
then for any $x, y \in X$,
$$V(x, y) < \delta \implies d(x, y) \leq (1 + \varepsilon)V(x, y), \tag{27}$$

4. *Assume that (X, d) has the midpoint property. For any $\varepsilon, \delta > 0$, if for any $x, y \in X$,*
$$d(x, y) < \delta \implies V_0(x, y) \leq (1 + \varepsilon)d(x, y),$$
then for any $x, y \in X$,
$$d(x, y) < \delta \implies V(x, y) \leq (1 + \varepsilon)d(x, y), \tag{28}$$

Proof We first prove 1. For any sequences x_i and y_i that converge x and y, respectively,

$$|V(x, y) - V_i(x_i, y_i)| \leq |V(x, y) - V_i(x, y)| + |V_i(x, y) - V_i(x_i, y_i)|.$$

The first term can be bound by the convergence $V_i \to V$ at (x, y) and the second term can be bound by the eventual equicontinuity of $(V_0, V_1, \ldots)$ at (x, y) and convergence $x_i \to x$ and $y_i \to y$. Thus, $\lim_{i \to \infty} V_i(x_i, y_i) = V(x, y)$. In particular,

$$\lim_{i \to \infty} V_i(x, \pi_i(x, y)) = V(x, \pi(x, y)),$$

$$\lim_{i \to \infty} V_i(\pi_i(x, y), y) = V(\pi(x, y), y).$$

By (13), for any $x, y, z \in X$,

$$V_i(x, \pi_i(x, y))^2 + V_i(\pi_i(x, y), y)^2 \leq V_i(x, z)^2 + V_i(z, y)^2.$$

In the limit $i \to \infty$,

$$V(x, \pi(x, y))^2 + V(\pi(x, y), y)^2 \leq V(x, z)^2 + V(z, y)^2.$$

Thus, (24) follows. We can also show (25) by taking the limit of both sides in (14).

Next, for 2, when $V_i(x, x) = 0$, because $V_i(x, \pi_i(x, x)) = V_i(\pi_i(x, x), x) = 0$ by (13), $V_{i+1}(x, x) = 0$ by (14). Thus, $V_i(x, x) = V_i(x, \pi_i(x, x)) = V_i(\pi_i(x, x), x) = 0$ for all i by induction. As we can also show that $V_i(x, y) \geq 0$ for all i by (14) and induction, $V_{i+1}(x, y) = 0$ implies $V_i(x, \pi_i(x, y)) = V_i(\pi_i(x, y), y) = 0$ by (14). Using this, $V_i(x, y) = 0 \implies d(x, y) = 0$ for all i is proven by induction. Therefore, $d(x, \pi_i(x, x)) = d(\pi_i(x, x), x) = 0$ for all i. Finally, by the assumption that d is weakly symmetric, (26) is proven.

As for 3, because this assumption implies $V_0(x, y) \geq 0$, by (14) and induction, $V_i(x, y) \geq 0$ for all i. Thus, $V_{i+1}(x, y) < \delta$ implies $V_i(x, \pi_i(x, y)) < \delta$ and $V_i(\pi_i(x, y), y) < \delta$ by (14). Assume that $d(x, y) \leq (1 + \varepsilon) V_i(x, y)$ for all $x, y \in X$ with $V_i(x, y) < \delta$. Then, when $V_{i+1}(x, y) < \delta$, because

$$d(x, \pi_i(x, y)) \leq (1 + \varepsilon) V_i(x, \pi_i(x, y))$$

and

$$d(\pi_i(x, y), y) \leq (1 + \varepsilon) V_i(\pi_i(x, y), y),$$

we can conclude that

$$d(x, y) \leq d(x, \pi_i(x, y)) + d(\pi_i(x, y), y) \leq (1 + \varepsilon) V_{i+1}(x, y).$$

Therefore, $d(x, y) \leq (1 + \varepsilon) V_i(x, y)$ when $V_i(x, y) < \delta$ for all i by induction, which yields (27).

Lastly, we prove 4. Assume that $d(x, y) < \delta \implies V_i(x, y) \leq (1 + \varepsilon) d(x, y)$ for all $x, y \in X$. Let $x, y \in X$ be a pair such that $d(x, y) < \delta$. By assumption, their midpoint m exists. Then, by a similar calculation with (5),

$$\begin{aligned}
V_{i+1}(x, y)^2 &= (V_i(x, \pi_i(x, y)) + V_i(\pi_i(x, y), y))^2 \\
&\leq 2V_i(x, \pi_i(x, y))^2 + 2V_i(\pi_i(x, y), y)^2 \\
&\leq 2V_i(x, m)^2 + 2V_i(m, y)^2 \\
&\leq 2(1 + \varepsilon)^2 \left(d(x, m)^2 + d(m, y)^2 \right) \\
&= (1 + \varepsilon)^2 d(x, y)^2,
\end{aligned}$$

where the first equality comes from (14), the second inequality comes from (13), and the third comes from the induction hypothesis and $d(x, m) = d(m, y) < \delta$. Thus, $d(x, y) < \delta \implies V_i(x, y) \leq (1 + \varepsilon) d(x, y)$ for all i by induction, which yields (28). ∎

By the above lemma, it is sufficient to show the following to prove Proposition 6:

1. π is uniformly continuous.

2. For any $\varepsilon > 0$, there exists $\delta > 0$ such that the assumptions of 3 and 4 in Lemma 20 are satisfied for $V_0 = C$.

To prove them, we use the following lemma, which is a generalization of a well-known fact to weakly symmetric pseudo-quasi-metric spaces.

Lemma 21 *Let (X, d_X) be a compact pseudo-quasi-metric space and (Y, d_Y) be a weakly symmetric pseudo-quasi-metric space. If a function $f : X \to Y$ is continuous, then f is uniformly continuous, that is, for any $\varepsilon > 0$, there exists $\delta > 0$ such that $d_Y(f(x), f(x')) < \varepsilon$ for any $x, x' \in X$ with $d_X(x, x') < \delta$.*

Proof We take $\varepsilon > 0$ arbitrarily. Because Y is weakly symmetric, for any $x \in X$, we can take $\varepsilon(x) \leq \varepsilon/2$ such that for any $y \in Y$, $d_Y(f(x), y) < \varepsilon(x)$ implies $d_Y(y, f(x)) < \varepsilon/2$. As f is continuous, we can take $\delta(x) > 0$ such that for any $x' \in X$, $d_X(x, x') < 2\delta(x)$ implies $d_Y(f(x), f(x')) < \varepsilon(x)$. Let $B(x) := B_{\delta(x)}(x)$. As X is compact, we can take finite $x_1, \ldots, x_N$ such that $B(x_1), \ldots, B(x_N)$ covers X. Let $\delta := \min_i \delta(x_i)$.

We take $x, x' \in X$ such that $d_X(x, x') < \delta$. We can then take x_i such that $d_X(x_i, x) < \delta(x_i)$. As $d_Y(f(x_i), f(x)) < \varepsilon(x_i)$ follows from this, $d_Y(f(x), f(x_i)) < \varepsilon/2$. Because $d_X(x_i, x') < 2\delta(x_i)$, $d_Y(f(x_i), f(x')) < \varepsilon(x_i) \leq \varepsilon/2$. Therefore, $d_Y(f(x), f(x')) < \varepsilon$. ■

The uniform continuity of π is a direct consequence of Lemma 21.

Next, we prove the existence of δ in the assumption of 4 of Lemma 20 by reducing it to uniform continuity of the ratio of C to d. We define a function $r : X \times X \to \mathbb{R}$ as

$$r(x, y) = \begin{cases} \frac{C(x,y)}{d(x,y)} & d(x, y) \neq 0, \\ 1 & d(x, y) = 0. \end{cases}$$

We show r is continuous. We take $x, y \in X$ and series $x_0, x_1, \ldots \in X$ and $y_0, y_1, \ldots \in X$ that converge to x and y, respectively. If $d(x, y) \neq 0$, because $d(x_i, y_i) \neq 0$ for sufficiently large i, $r(x_i, y_i) \to r(x, y)$ by the continuity of C and d. Otherwise, because $x_i \to x$ and $y_i \to x$, $r(x_i, y_i) \to 1$ by Assumption 5. By Lemma 21, r is uniformly continuous, and the existence of δ in the assumption of 4 of Lemma 20 follows from this. Note that $d(x, y) = 0 \iff C(x, y) = 0$ from Assumption 5.

The existence of δ in the assumption of 3 of Lemma 20 follows from the uniform continuity of r and the following lemma.

Lemma 22 *If (X, d) is a compact, weakly symmetric pseudo-quasi-metric space and a continuous function $C : X \times X \to \mathbb{R}$ satisfies the first condition of Assumption 5, for any $\varepsilon > 0$, there exists $\delta > 0$ such that, for any $x, y \in X$, $C(x, y) < \delta$ implies $d(x, y) < \varepsilon$.*

Proof Take arbitrary $\varepsilon > 0$. For any $x \in X$, we can take $\delta(x) > 0$ such that for any $y \in X$, $C(x, y) < \delta(x)$ implies $d(x, y) < \varepsilon/2$. As C is uniformly continuous by Lemma 21, we can take $\eta(x) > 0$ such that for any $y, z \in X$, $d(x, z) < \eta(x)$ implies $|C(x, y) - C(z, y)| < \delta(x)/2$. Because X is weakly symmetric, after making $\eta(x)$ smaller if necessary, we can also assume that, for any $z \in X$, $d(x, z) < \eta(x)$ implies $d(z, x) < \varepsilon/2$. As X is compact, we can take

27

finite $x_1, \ldots, x_N$ such that for any $x \in X$, there exists x_i such that $d(x_i, x) < \eta(x_i)$. Let $\delta := \min_i \delta(x_i)/2$.

We prove that $C(x, y) < \delta$ implies $d(x, y) < \varepsilon$ for any $x, y \in X$. We can take x_i such that $d(x_i, x) < \eta(x_i)$, which implies $d(x, x_i) < \varepsilon/2$ and $|C(x_i, y) - C(x, y)| < \delta(x_i)/2$. When $C(x, y) < \delta \leq \delta(x_i)/2$, because $C(x_i, y) < \delta(x_i)$, $d(x_i, y) < \varepsilon/2$. Thus, $d(x, y) < \varepsilon$. ∎

A.2 Proof of Proposition 7

To prove that the first condition of Assumption 5 is satisfied, it is sufficient to show that $f(y_i) \to f(x)$ when $C(x, y_i) \to 0$, because the topology induced by d coincides with the topology of the underlying manifold (Bao et al., 2000). We can take the minimum value $c > 0$ of $F\left(x, df_x^{-1}(\mathbf{v})\right)$ for $\mathbf{v} \in S^{d-1} := \{\mathbf{v} \in \mathbb{R}^d | \|\mathbf{v}\| = 1\}$ because S^{d-1} is compact. Then, $C(x, y) \geq c\|f(y) - f(x)\|$ for any $y \in X$. Therefore, $f(y_i) \to f(x)$ when $C(x, y_i) \to 0$.

Next, we prove that the second condition of Assumption 5 is satisfied. We fix $x \in X$ and $\varepsilon > 0$. For a curve $\gamma : [0, 1] \to M$, let

$$L'(\gamma) := \int_0^1 F\left(x, df_x^{-1}\left(\frac{d(f \circ \gamma)}{dt}(t)\right)\right) dt,$$

which is an approximation of $L(\gamma)$ and uses values of F at x instead of $\gamma(t)$. Note that, although $\frac{d(f \circ \gamma)}{dt}(t)$ inherently belongs $T_{f(\gamma(t))}$, it can be naturally identified with $T_{f(x)}$ since there exists a natural basis in the tangent space of $\mathbb{R}^d$. Let $E(\gamma)$ be the Euclidean length of $f \circ \gamma$.

We can take $\eta > 0$ such that there exists $c' > 0$ such that $L(\gamma) \geq c'E(\gamma)$ holds for any curve γ contained in $B_\eta(x)$ (Busemann and Mayer, 1941). Let $\varepsilon' := \varepsilon c'$.

As F is continuous and S^{d-1} is compact, after making η smaller if necessary, we can assume that, for any $y \in B_\eta(x)$ and $\mathbf{v} \in S^{d-1}$,

$$\left|F\left(y, df_y^{-1}(\mathbf{v})\right) - F\left(x, df_x^{-1}(\mathbf{v})\right)\right| \leq \varepsilon'.$$

From this and the condition of F, for any $y \in B_\eta(x)$ and $\mathbf{v} \in \mathbb{R}^d$,

$$\left|F\left(y, df_y^{-1}(\mathbf{v})\right) - F\left(x, df_x^{-1}(\mathbf{v})\right)\right| \leq \varepsilon' \|\mathbf{v}\|.$$

Then, for any curve γ contained in $B_\eta(x)$,

$$\begin{aligned}
\left|L(\gamma) - L'(\gamma)\right| &\leq \int_0^1 \left|F\left(\gamma(t), \frac{d\gamma}{dt}(t)\right) - F\left(x, df_x^{-1}\left(\frac{d(f \circ \gamma)}{dt}(t)\right)\right)\right| dt \\
&\leq \varepsilon' \int_0^1 \left\|\frac{d(f \circ \gamma)}{dt}(t)\right\| dt \\
&= \varepsilon' E(\gamma).
\end{aligned} \tag{29}$$

We can take δ such that for any $y, z \in B_\delta(x)$, there exists a minimizing geodesic from y to z contained in $B_\eta(x)$ (Bao et al., 2000), which is denoted by $\gamma(y, z)$. Note that $d(y, z) = L(\gamma(y, z))$. After making δ smaller if necessary, we can assume that $B_\eta(x)$ also

contains the inverse image of the straight line segment from $f(y)$ to $f(z)$ by f, which is denoted by $l(y, z)$. Note that $\|f(y) - f(z)\| = E(l(y, z))$. Also note that and $C(y, z) = L'(l(y, z))$ because the derivative of $f \circ l(y, z)$ is the constant function to $f(z) - f(y)$.

As L' is a Minkowskian metric, $L'(l(y, z)) \leq L'(\gamma(y, z))$ (Busemann and Mayer, 1941). Therefore, for any $y, z \in B_\delta(x)$,

$$
\begin{aligned}
d(y, z) - \varepsilon'\|f(y) - f(z)\| \leq L(l(y, z)) &- \varepsilon'\|f(y) - f(z)\| \\
&\leq C(y, z) \\
&\leq L'(\gamma(y, z)) \\
&\leq d(y, z) + \varepsilon' E(\gamma(y, z)),
\end{aligned}
$$

where the first inequality comes from the fact that $\gamma(y, z)$ is the minimizing geodesic, the second comes from (29) with $\gamma = l(y, z)$, and the fourth comes from (29) with $\gamma = \gamma(y, z)$. Because $\|f(y) - f(z)\| \leq E(\gamma(y, z)) \leq d(y, z)/c'$, the condition is satisfied.

Appendix B. Implementation Details

	Ours	Seq	PG
Learning rate for § 5.4.1	3×10^{-5}	3×10^{-3}	5×10^{-3}
Learning rate for others	10^{-6}	3×10^{-4}	5×10^{-3}
Batch size	256	128	300
Number of epochs	10	10	1
Hidden layer sizes for § 5.4.1	[64, 64]	[64, 64]	[64, 64]
Hidden layer sizes for others	[400, 300, 300]	[400, 300, 300]	[400, 300, 300]
Activation function	ReLU	tanh	tanh
λ for GAE	-	0.95	-
Clipping parameter	-	0.2	0.2
Entropy coefficient	-	0	1
VF coefficient	-	0.5	-
Max gradient norm	-	0.5	-
Base standard derivation	-	-	0.05
Number of samples per episode	-	-	10
Number of episodes per cycle	-	-	30

Table 2: Hyperparameter values.

The codes used in the experiments for this paper are available at `https://github.com/omron-sinicx/midpoint_learning`. Table 2 lists the hyperparameter values that we used for our method and the baseline methods.

The GPUs we used were NVIDIA RTX A5500 for experiments in § 5.4.3 and § 5.4.4 and NVIDIA RTX A6000 for experiments in § 5.4.1, § 5.4.2, and § 5.4.5.

B.1 Environments

The environments were implemented by NumPy and PyTorch (Paszke et al., 2019). To implement of the robotic arm environment, we used PyTorch Kinematics (Zhong et al., 2023). We also used Robotics Toolbox for Python (Corke and Haviland, 2021) for visualization of the robotic arm motion.

For the coordinates in all environments except the angular component in the car-like environment (§ 5.4.2), if a generated value is outside the valid range, it is clamped. The angular component S^1 is represented by the unit circle $\{x^2 + y^2 = 1\}$ in $\mathbb{R}^2$. A point in $\mathbb{R}^2$ generated by a policy for this component is projected to S^1 by normalization after clamping to $[-1, 1] \times [-1, 1]$. Note that, in this environment, the dimension of the space for state representation is different from the manifold dimension.

B.2 Our Method

At Line 4 in Algorithm 1, if the data size returned from one call to the data collection procedure is larger than the batch size, that procedure is called only once for one loop. Otherwise, it is called until one mini-batch is filled. At Line 18, two points are sampled from the uniform distribution over the free space. We set the number of epochs N_{epochs} to 10 and the batch size to 256.

The actor network outputs a Gaussian distribution with a diagonal covariant matrix on the state representation space. During data collecting or training, a prediction by the actor is sampled from the distribution with the mean and deviation output by the network. During evaluating, the mean is returned as a prediction. We use the reparameterization trick to train the actor as in SAC (Haarnoja et al., 2018).

Both the actor and critic networks are multilayer perceptrons. The hidden layers were two of size 64 for § 5.4.1 and three of sizes $400, 300, 300$ for the other environments. ReLU was selected as the activation function after trying tanh function, as well. The output-layer size in the actor network is twice the dimension of the state representation space, where one half represents the mean and the other half represents logarithms of the standard deviations. The critic network outputs a single value, whose exponential minus one is returned as a prediction of distance. Adam (Kingma and Ba, 2014) was used as the optimizer. The learning rate was tuned to 3×10^{-5} for § 5.4.1 and to 10^{-6} for other environments. PyTorch was used for implementation.

B.3 Sequential Reinforcement Learning

In the conventional reinforcement learning environment, observations are pairs of current and goal states. Whenever an episode starts, start and goal points are sampled from the uniform distribution over the free space. The action space is $[-1, 1]^d$, where d is the manifold dimension. If the agent outputs v for an observation $(f(p), f(g))$, the coordinate of the next state, $f(q)$, is calculated as

$$f(q) := f(p) + \frac{\varepsilon}{F(p, f_p^{-1}(v))} v. \tag{30}$$

If $f(q)$ is outside the coordinate space, it is clamped. Because q cannot be calculated when exactly $v = 0$, in such a case, q is set to p and the agent receives reward $R = -100$ as a

penalty. Otherwise, the reward is calculated by (21). For the angular component in the car-like environment, the addition in (30) is calculated in $\mathbb{R}/2\pi\mathbb{Z}$.

We used PPO implemented in Stable Baselines3 (Raffin et al., 2019), which uses PyTorch. The discount factor was set to 1. The learning rate was tuned to 3×10^{-3} for § 5.4.1, and to 3×10^{-4} for the other environments. The batch size was tuned to 128. The network architectures were the same as those of the proposed method. Other hyperparameters were set to the default values in the library. The tanh function was selected as the activation function after trying ReLU, as well.

B.4 Policy Gradient

We modified the implementation of subgoal-tree policy gradient (SGT-PG) by the authors, available at `https://github.com/tomjur/SGT-PG`, which uses TensorFlow (Abadi et al., 2016). The hyperparameter values except the hidden-layers sizes were the same as in the original paper. We changed the network architectures to those of our proposed method.

The tanh function is used as the activation function. The policy network outputs a Gaussian distribution with a diagonal covariant matrix on the state representation space. The output-layers size is $2d$, where d is the dimension of the state representation space. Let $m_1, \ldots, m_d, \sigma_1, \ldots, \sigma_d$ be the output for input s, g. The distribution mean is $(s + g)/2 + (m_1, \ldots, m_d)^\mathsf{T}$, and the standard deviation for the i-th coordinate is $\mathrm{Softplus}(\sigma_i) + (0.05 + \mathrm{Softplus}(c_i))\|s - g\|$, where c_i is a learnable parameter. While predictions are sampled from distributions during the policy training, we take the means as predictions during evaluation or training of other policies with higher indexes.

When we trained π_D, we sampled 30 values of (p_0, p_{2^D}), the start and goal points, from the uniform distribution over the free space, in each training cycle. For each sampled pair, we sampled 10 values of $p_{2^{D-1}}$ from the distribution outputted by π_D, and we generate other waypoints deterministically by $\pi_{D-1}, \ldots, \pi_1$ for each sampled value. The average of c_τ was used as the baseline in (22). The objective was that of PPO with an entropy coefficient of 1 and clipping parameter of 0.2. The optimizer was Adam, with the learning rate set to 5×10^{-3}.

In the environments for § 5.4.1, we trained π_1 for 1000 cycles and the other π_D for 538 cycles. In the environment for § 5.4.2 and § 5.4.5, we trained each policy for 2117 cycles. In the environment for § 5.4.3 and § 5.4.4, we trained each policy for 1059 cycles. The total number of timesteps for § 5.4.1 was $2 \times 300 \times 1000 + (4 + 8 + 16 + 32 + 64) \times 300 \times 538 = 20613600 \approx 2 \times 10^7$, that for § 5.4.2 and § 5.4.5 was $(2 + 4 + 8 + 16 + 32 + 64) \times 300 \times 2117 = 80022600 \approx 8 \times 10^7$, and that for § 5.4.3 and § 5.4.4 was $(2 + 4 + 8 + 16 + 32 + 64) \times 300 \times 1059 = 40030200 \approx 4 \times 10^7$.

Appendix C. Further Analysis of Experimental Results

In addition to success rate, we compared methods in the following way. For each environment, we generated paths for all start and goal pairs in the evaluation data by the policies learned by all methods and, for all succeeded generations, calculated the path lengths, which are defined as the sum of C values for consecutive waypoints. For each pair of methods, in all start and goal pairs where both policies succeeded in generation, we calculated the percentage of pairs where the first policy generated a shorter path than the second policy.

	Our-T	**Our-C**	Seq	Inter	2:1
Our-C	$37 \pm 3\,(\mathbf{70})$				
Seq	$66 \pm 2\,(67)$	$\mathbf{73 \pm 3}\,(\mathbf{70})$			
Inter	$49 \pm 5\,(22)$	$\mathbf{57 \pm 7}\,(\mathbf{22})$	$62 \pm 8\,(20)$		
2:1	$58 \pm 5\,(31)$	$\mathbf{70 \pm 4}\,(\mathbf{32})$	$63 \pm 5\,(29)$	$57 \pm 6\,(19)$	
Cut	$77 \pm 3\,(52)$	$\mathbf{85 \pm 2}\,(\mathbf{53})$	$69 \pm 5\,(49)$	$78 \pm 4\,(22)$	$74 \pm 4\,(30)$

(a) Matsumoto

	Our-T	**Our-C**
Our-C	$18 \pm 3\,(\mathbf{77})$	
2:1	$44 \pm 2\,(21)$	$\mathbf{73 \pm 3}\,(\mathbf{21})$

(b) Car-Like

	Our-T	Our-C	**Seq**	Inter	2:1
Our-C	$73 \pm 10\,(39)$				
Seq	$\mathbf{32 \pm 10}\,(\mathbf{70})$	$\mathbf{4 \pm 2}\,(\mathbf{42})$			
Inter	$42 \pm 15\,(16)$	$9 \pm 3\,(14)$	$\mathbf{77 \pm 5}\,(\mathbf{16})$		
2:1	$52 \pm 15\,(30)$	$31 \pm 5\,(26)$	$\mathbf{84 \pm 5}\,(\mathbf{30})$	$70 \pm 8\,(15)$	
Cut	$69 \pm 10\,(36)$	$41 \pm 6\,(30)$	$\mathbf{89 \pm 2}\,(\mathbf{38})$	$80 \pm 6\,(16)$	$61 \pm 7\,(27)$

(c) 2D Obstacles

	Our-T
Our-C	$\mathbf{81 \pm 2}\,(\mathbf{81})$

(d) Robotic Arm

	Our-T
Our-C	$\mathbf{39 \pm 2}\,(\mathbf{52})$

(e) Three Agents

Table 3: Winning Rate Tables

Table 3 shows the percentages of instances where the methods in the top row outperform the methods in the left column. The values were averaged over random seeds and their standard errors are also displayed. The numbers in parentheses represent the percentages of instances where both methods succeeded. Methods with low success rates are omitted. The methods which outperformed all other methods and their results are highlighted in bold.

While **Seq** had the highest success rate in the Matsumoto environment, **Our-T** and **Our-C** outperformed it in this comparison. This may be because our methods generates denser waypoints and therefore smoother paths. In contract, in the 2D obstacle environment, **Seq** outperformed **Our-T**, despite the small difference between their success rates. This is probably because, in this environment, straight lines are the shortest where there are no obstacles. While the success rates for **Our-T** are slightly higher than those for **Our-C** in both the robotic arm and three agents environments, **Our-T** outperformed **Our-C** in the former and vice versa in the latter.

It is interesting that **2:1** loses to our proposed method in this comparison, even though the non-uniformity of waypoints does not directly affect length. A biased ratio in waypoints may make it difficult to generate smooth paths, or have a negative effect on learning.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Martial Agueh. Finsler structure in the p-wasserstein space and gradient flows. *Comptes Rendus. Mathématique*, 350(1-2):35–40, 2012.

OM Amici and BC Casciaro. Convex neighbourhoods and complete finsler spaces. *arXiv preprint arXiv:1006.0851*, 2010.

AV Arutyunov, AV Greshnov, LV Lokutsievskii, and KV Storozhuk. Topological and geometrical properties of spaces with symmetric and nonsymmetric f-quasimetrics. *Topology and its Applications*, 221:178–194, 2017.

David Bao, S-S Chern, and Zhongmin Shen. *An introduction to Riemann-Finsler geometry*, volume 200. Springer Science & Business Media, 2000.

Herbert Busemann and Walther Mayer. On the foundations of calculus of variations. *Transactions of the American Mathematical Society*, 49(2):173–198, 1941.

Peter Corke and Jesse Haviland. Not your grandmother's toolbox–the robotics toolbox reinvented for python. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11357–11363. IEEE, 2021.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

Vikas Dhiman, Shurjo Banerjee, Jeffrey M Siskind, and Jason J Corso. Floyd-warshall reinforcement learning: Learning from past experiences to reach new goals. *arXiv preprint arXiv:1809.09318*, 2018.

Alexander Effland, Erich Kobler, Thomas Pock, Marko Rajković, and Martin Rumpf. Image morphing in deep feature spaces: Theory and applications. *Journal of mathematical imaging and vision*, 63:309–327, 2021.

Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

Ryuichi Fukuoka and Anderson Macedo Setti. Mollifier smoothing of c 0-finsler structures. *Annali di Matematica Pura ed Applicata (1923-)*, 200(2):595–639, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Sajad Haghzad Klidbary, Saeed Bagheri Shouraki, and Soroush Sheikhpour Kourabbaslou. Path planning of modular robots on various terrains using q-learning versus optimization algorithms. *Intelligent Service Robotics*, 10:121–136, 2017.

Charles Horvath. A note on metric spaces with continuous midpoints. *Annals of the academy of Romanian Scientists, Series on mathematics and its applications*, 1(2), 2009.

Miguel Angel Javaloyes and Miguel Sánchez. On the definition and examples of finsler metrics. *arXiv preprint arXiv:1111.5066*, 2011.

Tom Jurgenson, Or Avner, Edward Groshev, and Aviv Tamar. Sub-goal trees a framework for goal-based reinforcement learning. In *International Conference on Machine Learning*, pages 5020–5030. PMLR, 2020.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer, 1993.

Yong-Woon Kim. Pseudo quasi metric spaces. *Proceedings of the Japan Academy*, 44(10): 1009–1012, 1968.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Geesara Kulathunga. A reinforcement learning based path planning approach in 3d environment. *Procedia Computer Science*, 212:152–160, 2022.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Ee Soong Low, Pauline Ong, Cheng Yee Low, and Rosli Omar. Modified q-learning with distance metric and virtual target on path planning of mobile robot. *Expert Systems with Applications*, 199:117191, 2022.

Makoto Matsumoto. A slope of a mountain is a finsler surface with respect to a time measure. *Journal of Mathematics of Kyoto University*, 29(1):17–25, 1989.

Mike Yan Michelis and Quentin Becker. On linear interpolation in the latent space of deep generative models. In *Proceedings of ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021.

Giambattista Parascandolo, Lars Buesing, Josh Merel, Leonard Hasenclever, John Aslanides, Jessica B Hamrick, Nicolas Heess, Alexander Neitz, and Theophane Weber. Divide-and-conquer monte carlo tree search for goal-directed planning. *arXiv preprint arXiv:2004.11410*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

Christian Pfeifer. Finsler spacetime geometry in physics. *International Journal of Geometric Methods in Modern Physics*, 16(supp02):1941004, 2019.

Chenyang Qi, Chengfu Wu, Lei Lei, Xiaolu Li, and Peiyan Cong. Uav path planning based on the improved ppo algorithm. In *2022 Asia Conference on Advanced Robotics, Automation, and Control Engineering (ARACE)*, pages 193–199. IEEE, 2022.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

Nathan Ratliff, Marc Toussaint, and Stefan Schaal. Understanding the geometry of workspace obstacles in motion optimization. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4202–4209. IEEE, 2015.

Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Kinodynamic trajectory optimization and control for car-like robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686. IEEE, 2017.

Daniele Sartori, Danping Zou, Ling Pei, and Wenxian Yu. Cnn-based path planning on a map. In *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1331–1338. IEEE, 2021.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999.

Chao Wang, Jian Wang, Yuan Shen, and Xudong Zhang. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 68(3):2124–2136, 2019.

Ruosong Wang, Simon S Du, Lin F Yang, and Sham M Kakade. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning? *arXiv preprint arXiv:2005.00527*, 2020.

Tongzhou Wang and Phillip Isola. On the learning and learnablity of quasimetrics. *arXiv preprint arXiv:2206.15478*, 2022.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pages 36411–36430. PMLR, 2023.

Jeong Sheng Yang. Net characterizations of equicontinuity. *Mathematica Scandinavica*, 25 (1):113–120, 1969.

Sheng Zhong, Thomas Power, and Ashwin Gupta. PyTorch Kinematics, 3 2023.

Xinyuan Zhou, Peng Wu, Haifeng Zhang, Weihong Guo, and Yuanchang Liu. Learn to navigate: cooperative path planning for unmanned surface vehicles using deep reinforcement learning. *Ieee Access*, 7:165262–165278, 2019.