LEARNING NEURAL TIME FIELDS ON RANDERS MANIFOLDS FOR DRIFT-AWARE MOTION PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-informed neural time fields provide a self-supervised learning framework for robot motion planning. Existing Eikonal and Riemannian formulations primarily model isotropic or symmetric motion costs. This limits their ability to represent directional effects arising from environmental flows, gravity, and direction dependent dynamics. We introduce *Randers Neural Time Fields (RaNTFields)*, a physics-informed neural framework for learning asymmetric anisotropic traveltime fields. RaNTFields extends the Eikonal formulation using a Randers metric that combines a configuration-dependent Riemannian metric with a directional one-form, derived from the corresponding Hamilton-Jacobi equation. We evaluate RaNTFields on 2D navigation under complex flows, terrain navigation, and 5-DoF robot arm motion generation under gravitational drift. In constant-flow environments, RaNTFields achieves a 0.997 correlation with the Ordered Upwind reference solution across start-goal pairs. In the 5-DoF arm, it follows the preferred gravity drift direction and consequently reduces the peak torque by 17.8% relative to an Eikonal planner, with a 96% success rate compared to 98% for the baseline. These results demonstrate that RaNTFields can encode directional preferences arising from environmental flows and robot dynamics.

1 INTRODUCTION

Motion planning is a fundamental aspect of robotics and it is evident in autonomous cars that navigate road networks, air taxis that plan collision-free flight paths, and humanoid robots that manipulate objects in factories and homes. Classical approaches, including sampling-based planners such as PRM* and RRT* (Karaman & Frazzoli, 2011), provide strong theoretical guaranties but can require substantial sampling and collision-checking effort in cluttered or high-dimensional configuration spaces. Fast Marching Method (FMM) (Sethian, 1999) solves for a cost to go value function over the configuration space of the robot, where its gradient provides a local direction of travel. It uses the Eikonal formulation which assumes isotropic propagation, limiting its ability to represent directional properties of robotic motion. In many robotic tasks, reversing a motion changes its cost. A boat traveling with the river currents may reach its destination faster than one traveling against it; traversing a slope uphill differs from traveling downhill; and lifting a payload can require different actuator effort from lowering it. They call for a planner that represents the cost of motion in each direction, rather than assigning the same travel cost to both directions of a route.

Ordered Upwind Methods (OUM) (Sethian & Vladimirsky, 2003) is a classical numerical method that solves static Hamilton-Jacobi (HJ) equations for anisotropic optimal-control problems, in which propagation depends on both position and direction. OUM provides an accurate numerical solution but like other grid-based PDE solvers, it requires explicit discretization of the state space which becomes intractable in high-dimensional configurational spaces.

Riemannian metrics represent configuration-dependent anisotropic costs while preserving symmetry, whereas Finsler metrics allow the local cost to depend directly on the direction of motion. In particular, Randers metrics combine a symmetric Riemannian component with a directional one-form. Recent neural formulations demonstrate that Riemannian structure can be incorporated into continuous neural distance fields (Li et al., 2026). Extending this framework to Randers geometry provides a principled mechanism for representing both anisotropy and directional asymmetry.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we introduce *RaNTFields*, a physics-informed neural formulation for asymmetric anisotropic travel-time fields. RaNTFields extends neural Eikonal motion planning to a Randers formulation consisting of a configuration-dependent metric and a directional drift. The resulting HJ equation retains the continuous and differentiable representation of neural time fields while incorporating the directional structure of asymmetric optimal-control problems.

Our formulation unifies neural time fields into a single hierarchy: an identity metric with zero drift yields the Eikonal model, a configuration-dependent metric recovers Riemannian formulation, and adding nonzero drift produces the asymmetric Randers formulation.

The main contributions of this work are:

- We formulate asymmetric anisotropic motion planning using a Randers PDE (Partial Differential Equation) and derive the corresponding Hamilton-Jacobi equation.
- We introduce RaNTFields, a physics-informed neural representation that learns continuous Randers travel-time fields without precomputed travel-time labels.
- We evaluate RaNTFields against classical OUM solutions in two-dimensional problems and demonstrate its ability to encode directional preferences in 5-dimensional motion planning.

In constant-flow environments, the learned field closely reproduces the directional asymmetry of an Ordered Upwind solution, with a correlation of 0.997 across start–goal pairs. On the 5 DoF robot arm task, it reduces peak torque by 17.8% relative to the Eikonal planner, while achieving a 96% success rate versus 98% for that baseline.

2 Related work

Imitation-based neural motion planners (Xie et al., 2023; Chi et al., 2024) amortize planning by learning from expert trajectories. However, such demonstrations can be expensive to generate in high-dimensional configuration spaces, and performance depends on their coverage of test-time problems. These limitations have motivated physics-informed neural motion planners that learn continuous travel-time fields by enforcing geometric PDE constraints without expert trajectories. We review these methods, followed by the Hamilton–Jacobi formulations and Finsler geometry that provide the foundations for direction-dependent motion costs.

2.0.1 Physics-Informed Neural Motion Planning

Neural implicit representations offer grid-free solutions to PDE-based robot motion planning. Originating with EikoNet (Smith et al., 2021) and NTFields (Ni & Qureshi, 2023a), recent physics-informed time fields improve scalability via self-supervision, temporal consistency, and Riemannian manifold extensions (Ni & Qureshi, 2023b; Ni et al., 2025; Shen et al., 2024; Ni et al., 2026; Li et al., 2026). Although these methods avoid grid discretization, their underlying Eikonal or Riemannian formulations are inherently isotropic or symmetric, preventing them from modeling direction-dependent travel costs. We extend this paradigm to asymmetric anisotropic motion through a Randers Hamilton–Jacobi formulation.

2.1 Anisotropic Hamilton-Jacobi Methods

Classical numerical methods for Hamilton-Jacobi equations provide a principled framework for anisotropic motion planning. Level-set methods (Osher & Sethian, 1988) solve HJ equations through implicit front propagation, while OUM (Sethian & Vladimirsky, 2003) extend upwind propagation to direction-dependent problems and have been used for energy-efficient terrain navigation applications (Shum et al., 2015; Sánchez-Ibáñez et al., 2023). However, their explicit spatial discretization limits scalability to low-dimensional problems.

Neural HJ solvers such as DeepReach (Bansal & Tomlin, 2020) and NeHMO (Chen et al., 2026) address this dimensionality challenge using neural representations, primarily for reachability and safety value functions. In contrast, we learn a travel-time field whose local geometry explicitly encodes configuration and direction-dependent motion costs through a Randers metric. We use OUM as the numerical reference for our two-dimensional asymmetric anisotropic experiments.

2.2 Finsler Geometry in Robotics

Finsler geometry generalizes Riemannian geometry by allowing the local cost to depend on the tangent direction, providing a natural framework for asymmetrical motion and directional preferences. Recent work has applied this idea to asymmetric anisotropic friction for geometric locomotion (Hatton et al., 2026). Geometric Fabrics (Van Wyk et al., 2022) extend the structure of Riemannian Motion Policies (Ratliff et al., 2018) to Finsler geometry, enabling velocity-dependent metrics and more expressive local motion policies. Neural Geometric Fabrics further extend this framework to high-dimensional robotic systems by learning structured Finsler-based policies from demonstrations (Xie et al., 2023), while related work uses geometric fabrics as a local motion planner (Merva et al., 2025). These approaches demonstrate the expressive directional structure of Finsler geometry, but primarily formulate it as a local policy rather than a global travel-time representation.

A useful subclass of Finsler metrics is the Randers metric, which combines a Riemannian norm with a directional one-form (Cheng & Shen, 2012). Randers geometry is closely connected to Zermelo's navigation problem (Zermelo, 1931), where background wind or drift induces direction-dependent travel cost. Bao et al. (2003) showed that a minimum-time navigation problem on a Riemannian manifold can be formulated geometrically using a Randers metric, providing a direct interpretation of its one-form as a directional preference or drift.

In contrast, RaNTFields learns an approximately global travel-time field governed by a Randers Hamilton-Jacobi equation. It combines the grid-free representation of neural time fields with the directional structure of Finsler geometry, allowing task or environment-dependent directional preferences to be encoded directly in the metric.

3 Background

3.1 Problem Definition

Let $\mathcal{Q} \subset \mathbb{R}^d$ be the robot configuration space, with free space $\mathcal{Q}_{\text{free}} = \mathcal{Q} \setminus \mathcal{Q}_{\text{obs}}$. Given $q_s, q_g \in \mathcal{Q}_{\text{free}}$, we seek a collision-free motion from q_s to q_g. Workspace obstacles $\mathcal{X}_{\text{obs}}$ are assumed known. We write $T(q, q_g)$ for the travel cost from the current configuration q to the goal configuration q_g.

3.2 Eikonal Neural Motion Planning

The Eikonal equation defines a travel-time field through

$$\|\nabla_q T(q, q_g)\|_2 = \frac{1}{S(q)}, \qquad T(q_g, q_g) = 0, \qquad (1)$$

where the positive speed field $S(q)$ determines the local traversal cost. Following prior NTFields work, we use an obstacle-aware target speed

$$S^*(q) = \text{clip}\left(\frac{d_{\text{obs}}(q, \mathcal{X}_{\text{obs}})}{d_{\max}}, \frac{d_{\min}}{d_{\max}}, 1\right). \qquad (2)$$

Here d_{obs} is the robot's minimum workspace clearance. Smaller speeds near obstacles increase their traversal cost. The Eikonal formulation is isotropic: its local cost does not distinguish among directions of motion.

3.3 Riemannian Travel-Time Fields

A symmetric positive-definite metric $G(q)$ makes the local cost configuration-dependent and anisotropic. Its travel-time field obeys:

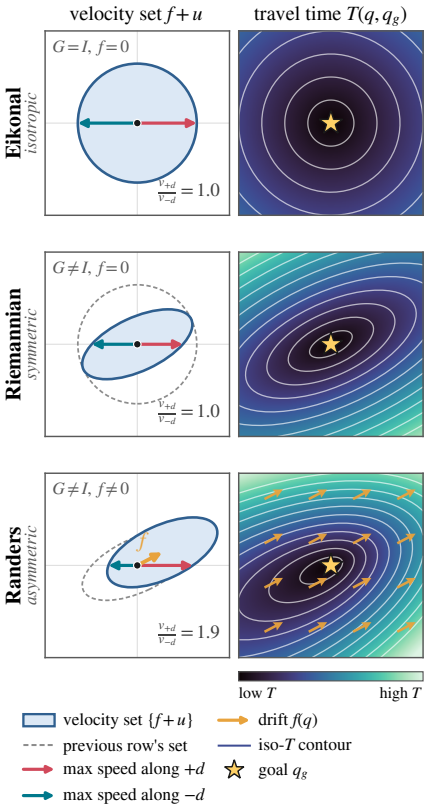

Figure 1: **Eikonal vs. Riemannian vs. Randers.** Left: set of reachable velocities $f + u$; the dashed outline is the row above. Right: travel time $T(q, q_g)$ to the goal. G makes speed depend on direction but keeps $\pm d$ equally fast; the drift f shifts the set, so a direction and its reverse differ ($v_{+d}/v_{-d} \neq 1$).

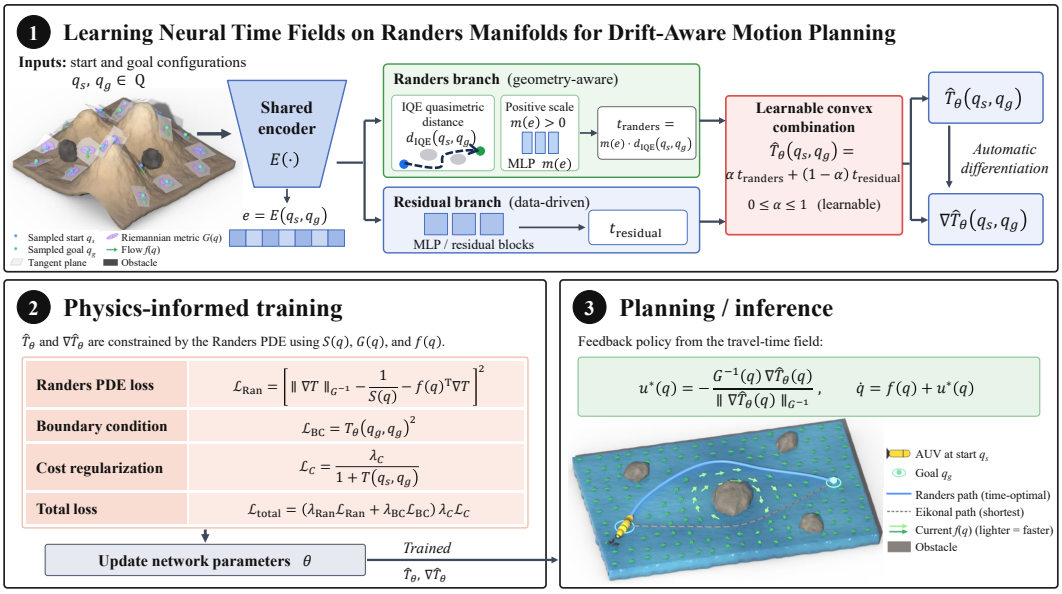

Figure 2: **Pipeline overview.** (1) A shared encoder feeds a geometry-aware Randers branch and a data-driven residual branch; their learnable convex combination gives the travel-time field $\hat{T}_\theta(q_s, q_g)$ and, by automatic differentiation, its gradient. (2) The network is trained with the Randers PDE, a boundary condition, and a cost regularization term. (3) At inference, $\nabla\hat{T}_\theta$ gives a feedback policy that exploits the drift $f(q)$, yielding time-optimal paths rather than the shortest ones.

$$\|\nabla_q T(q, q_g)\|_{G(q)^{-1}} = \frac{1}{S(q)}, \qquad T(q_g, q_g) = 0, \tag{3}$$

This metric can favor some directions over others, but reversing a path leaves its cost unchanged.

3.4 Randers geometry

A Randers metric adds a directional term to a Riemannian norm:

$$F(q, v) = \sqrt{v^\top G(q) v} + \beta(q)^\top v. \tag{4}$$

The one-form $\beta(q)$ changes sign when v is reversed, allowing forward and reverse travel costs to differ. For positivity, it must satisfy $\|\beta(q)\|_{G(q)^{-1}} < 1$. Setting $\beta = 0$ recovers the symmetric Riemannian metric. See Appendix A.2 for more information.

4 Method

We learn a goal-conditioned travel-time field from three local quantities: an obstacle-aware speed $S(q)$, a positive-definite metric $G(q)$, and a directional drift $f(q)$. The metric encodes symmetric anisotropy, while the drift makes motion costs asymmetric. Rather than supervising the field with precomputed optimal paths or travel times, we train it to satisfy the corresponding stationary Hamilton-Jacobi equation at sampled configurations. The learned field then supplies a direction of motion through its gradient.

4.1 Randers travel-time formulation

Consider the first-order model $\dot{q} = f(q) + u$, with $u^\top G(q) u \le 1$ and $\|f(q)\|_{G(q)} < 1$. Let $T(q, q_g)$ be the minimum accumulated cost, with local running cost $1/S(q)$, from the current configuration q

to the goal q_g. The Randers stationary Hamilton–Jacobi equation is

$$\|\nabla_q T(q, q_g)\|_{G(q)^{-1}} = \frac{1}{S(q)} + f(q)^\top \nabla_q T(q, q_g), \qquad T(q_g, q_g) = 0. \tag{5}$$

When $f = 0$, this reduces to the Riemannian Eikonal equation; setting $G = I$ also recovers the isotropic case. Thus G changes the cost across directions, while f distinguishes a direction from its reverse. Figure 1 illustrates the difference between Eikonal, Riemannian and Randers metrics. Appendix A.1 gives the derivation of Eq. 5 from an optimal-control perspective.

For all $q \in \mathcal{Q}_{free}$, the minimizing control is:

$$u^*(q) = -\frac{G(q)^{-1}\nabla_q T(q, q_g)}{\|\nabla_q T(q, q_g)\|_{G(q)^{-1}}}. \tag{6}$$

The resulting first-order motion follows $\dot{q} = f(q) + u^*(q)$.

4.2 Learning a Randers Neural Time Field

We describe the training objective and neural architecture used to learn a continuous Randers time field. We first introduce the physics-informed losses used to enforce the Randers Eikonal equation and boundary condition, then describe the asymmetric architecture used to learn the travel-time field.

4.2.1 Physics-informed training

We approximate $T(q, q_g)$ with a neural field $T_\theta(q, q_g)$. We apply the ***SoftPlus*** activation function to the output of the forward pass to ensure non-negativity in predicted travel time. At continuously sampled start-goal pairs, we penalize the residual of Eq. 5:

$$\mathcal{L}_{\text{Ran}} = \mathbb{E}_{q,q_g}\left[\left(\|\nabla_q T_\theta(q, q_g)\|_{G(q)^{-1}} - f(q)^\top \nabla_q T_\theta(q, q_g) - \frac{1}{S^*(q)}\right)^2\right]. \tag{7}$$

To satisfy the goal boundary condition $T(q_g, q_g) = 0$, we additionally penalize the boundary value using $\mathcal{L}_{\text{BC}} = \mathbb{E}_{q_g}[T_\theta(q_g, q_g)^2]$. This is necessary as our neural network does not directly satisfy the boundary condition. Moreover, learning smaller $T(q, q_g)$ values associated with shorter paths well before learning larger values ensures the model follows the desirable natural one-way propagation of value functions and stabilizes training when paths of varying lengths are included. We adapt the causality loss from TD-NTFields Ni et al. (2025) and define it as $\mathcal{L}_{\text{C}} = \frac{\lambda_{\text{C}}}{(1+T(q,q_g))}$, which results in a larger value for smaller $T(q, q_g)$ and vice versa.

The resulting training objective is:

$$\mathcal{L}_{\text{total}} = (\lambda_{\text{Ran}}\mathcal{L}_{\text{Ran}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}})\mathcal{L}_{\text{C}} \tag{8}$$

where the λ terms control the relative contributions of the respective losses.

The field therefore learns from evaluations of S^*, G, and f, without labeled optimal travel times or a discretized configuration-space solution. At inference, its gradient replaces $\nabla_q T$ in Eq. 6. Thus, the neural time field can be used directly as a continuous feedback policy at every state, where the gradient of the learned field determines the locally optimal direction.

4.2.2 Neural Network architecture

Fig. 2 shows the neural model's two complementary branches. A shared encoder represents the current configuration, the goal, and their displacement. One branch combines an Interval Quasimetric Embedding (Wang & Isola, 2024) of the ordered configuration pair with a positive learned scale. This branch supplies an explicit mechanism for representing forward-reverse asymmetry. A residual MLP supplies additional capacity for the travel-time function. A learned sigmoid gate forms a convex combination of their non-negative outputs. Architecture details are given in the Appendix A.3.

A shared encoder ϕ applies a two-layer MLP with layer normalization. The same weights embed the current configuration q, the goal q_g, and their displacement $q_g - q$. Because q and q_g share these weights, they land in a common embedding space, and the quasimetric branch below can compare them directly. The travel time is predicted by two branches that share this encoding.

The first Randers branch uses an Interval Quasimetric Embedding (IQE) (Wang & Isola, 2024). It splits each embedding into components, measures how far the goal's interval extends beyond the start's in each component, and aggregates the results. This yields a distance that is zero on the diagonal, satisfies the triangle inequality, and is asymmetric by construction. These properties match Randers geometry, where the flow makes $T(q, q_g) \neq T(q_g, q)$, and they rule out the symmetric start–goal operator of NTFields and TD-NTFields (Ni & Qureshi, 2023a; Ni et al., 2025). A residual MLP acts on $[\phi(q_g) - \phi(q),\ \phi(q),\ \phi(q_g)]$ and produces a configuration-dependent metric coefficient that rescales the quasimetric locally. The second Residual branch is a residual MLP on the same inputs. It adds capacity for structure the quasimetric factorization cannot capture. A single learned scalar gate α forms a convex combination of the two non-negative outputs. The IQE branch vanishes exactly at $q = q_g$, but the residual branch does not, so we enforce the boundary condition with $\mathcal{L}_{\mathrm{BC}}$.

4.3 Training domains and local fields

Training pairs (q, q_g) are sampled directly from the continuous configuration space. Half of the samples are sampled randomly and the rest are samples near the boundary of the obstacle. This ensures good sample coverage of Riemannian metrics and drift across the configuration space, as well as sufficient representation of the obstacle boundary. For each q, we evaluate the obstacle-aware speed in Eq. 2, the metric $G(q)$, and the drift $f(q)$. No ground-truth travel-time labels are required. We set $S(q) = 1$ in obstacle-free settings.

In the two-dimensional terrain domain, elevation $h(q)$ induces $G(q) = I + \nabla h(q) \nabla h(q)^\top$, such that we recover the identitity riemannian metric with there is no elevation. A calibrated downhill drift follows the direction $-\nabla h(q)$. For the UR5e robot arm, $G(q)$ is constructed from the normalized joint-space mass matrix and a regularization term. The directional field follows the gravity-related term $-M(q)^{-1}\tau_{\mathrm{gravity}}(q)$, where $\tau_{\mathrm{gravity}}(q)$ refers to the torque due to gravity compensation of the arm at static configuration q. In both domains, the drift magnitude is calibrated to satisfy $f(q)^\top G(q) f(q) < 1$. We provide the exact field construction, calibration, obstacle models, and sampling details in Appendix A.4.

4.4 Controller for path inference

Given a learned $T_\theta(q, q_g)$, we get the optimal velocity $\dot{q} = f(q) + u^*(q)$. For the 2D experiments we extract paths *open loop*: we integrate the normalized direction of $f + u^*$ from start to goal with fixed-length steps. For the UR5e arm manipulator we run a *closed-loop* geodesic torque controller. At 50 Hz, it queries T_θ at the measured joint state to get a desired joint velocity, which is a blend of u^* and a goal-directed velocity term. The purpose of the goal-directed velocity term, adapted from (Li et al., 2026), is to make convergence robust in the final approach. Computed-torque law, using inverse dynamics control, then tracks this velocity. The torques are saturated at the actuator limits. No trajectory is precomputed; the learned field itself acts as the feedback policy. See Appendix A.5 for controller details.

5 Experiments

We first evaluate how well the Randers model learns external flows on a Euclidean surface. We then demonstrate its applications to terrain and UR5e robotic arm motion planning, incorporating both Riemannian geometry and asymmetrical flows.

5.1 2-D Experiments

We evaluate RaNTFields against the ground truth OUM solution. For each method, paths are extracted open-loop from the same start-goal pairs. We report *planning time* (**PT**), *Euclidean path length* (**Length**) and *success rate* (**SR**), where success denotes reaching the goal within a prescribed tolerance without collision. Planning time measures the computational cost of path extraction.

To evaluate whether RaNTFields learns the correct directional structure of the travel-time field, we compute the mean *gradient-direction cosine similarity* (**Grad. CS**) between RaNTFields and OUM on a common grid:

$$\frac{1}{|\Omega|}\sum_{q\in\Omega}\frac{\nabla T_{\mathrm{RaNTFields}}(q)^\top \nabla T_{\mathrm{OUM}}(q)}{|\nabla T_{\mathrm{RaNTFields}}(q)||\nabla T_{\mathrm{OUM}}(q)|}. \tag{9}$$

This metric is invariant to the magnitude of the travel-time field and directly evaluates whether RaNTFields recovers an accurate gradient field, which is used for path extraction.

Finally, we evaluate the defining asymmetric property of the Randers formulation. For each start-goal pair, we compute the travel time asymmetry $T(q, q_g) - T(q_g, q)$ and report the Pearson correlation (**Asym. corr.**) of predicted and ground-truth asymmetries across pairs. Unlike the Randers formulation, the Eikonal and Riemannian baselines are symmetrical by construction and therefore provide a zero-asymmetry reference.

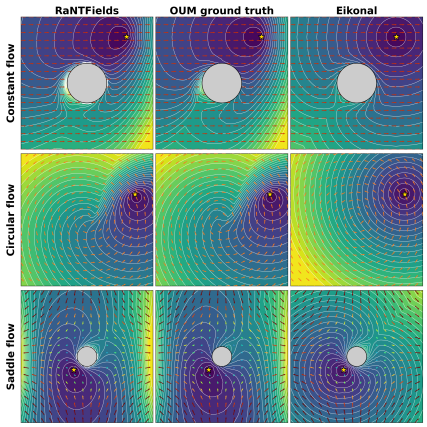

Figure 3: Travel-time fields for constant, circular, and hyperbolic flows. RaNTFields closely matches the Randers OUM reference, while the Eikonal baseline cannot represent flow-induced asymmetry.

5.1.1 Environmental flows

We evaluate RaNTFields in three flow environments with distinct directional structures: constant, circular, and hyperbolic flows. Fig. 3 compares the resulting travel-time fields against the Randers OUM ground truth and the symmetric Eikonal baseline. Across all three environments, RaNTFields closely reproduces the Randers travel-time structure, whereas the Eikonal formulation, which does not account for flow-induced asymmetry, fails to represent the directional effects of the environment.

Table 1: Open-loop evaluation of RaNTFields (RaNTF) against the Randers OUM (OUM-R) reference and symmetric Riemannian OUM (OUM-G) and Eikonal OUM (OUM-E) baselines across three flow environments. Results are averaged over 100 randomly sampled start-goal pairs.

Constant Flow

	PT (s) ↓	Length ↓	SR ↑	Grad. CS ↑	Asym. corr. ↑
RaNTF	**0.84**±0.38	1.15±0.52	**1.00**	**0.96**±0.02	**1.00**
OUM-R	7.40±0.15	**1.12**±0.51	0.99	–	–
OUM-G	2.51±0.06	1.23±1.32	0.99	0.80±0.03	0.00
OUM-E	2.50±0.04	1.23±1.32	0.99	0.80±0.03	0.00

Circular Flow

	PT (s) ↓	Length ↓	SR ↑	Grad. CS ↑	Asym. corr. ↑
RaNTF	**1.03**±0.50	1.41±0.68	1.00	**0.99**±0.00	**1.00**
OUM-R	5.09±0.08	**1.40**±0.66	1.00	–	–
OUM-G	2.43±0.04	1.08±0.49	1.00	0.78±0.09	0.00
OUM-E	2.42±0.04	1.08±0.49	1.00	0.78±0.09	0.00

Saddle Flow

	PT (s) ↓	Length ↓	SR ↑	Grad. CS ↑	Asym. corr. ↑
RaNTF	**0.88**±0.43	1.14±0.51	**1.00**	**0.98**±0.02	**0.95**
OUM-R	11.01±0.13	1.20±0.81	0.97	–	–
OUM-G	2.46±0.04	**1.07**±0.50	1.00	0.80±0.05	0.00
OUM-E	2.45±0.04	**1.07**±0.50	1.00	0.80±0.05	0.00

Table 1 compares RaNTFields with the Randers OUM reference and the symmetric Riemannian OUM and Eikonal OUM baselines over 100 randomly sampled start–goal pairs. On the constant flow environment, RaNTFields successfully extracts a path for every pair while requiring only 0.84 s per path, compared with 7.40 s for the Randers OUM solver. The resulting paths have lengths comparable to those of the reference solution.

More importantly, RaNTFields accurately recovers the structure of the Randers travel-time field. It achieves a mean gradient-direction cosine similarity of 0.964, substantially higher than the 0.803 obtained by both symmetric baselines. RaNTFields also reproduces the asymmetric travel-time structure with a Pearson correlation of 0.997 with the Randers reference across start-goal pairs. In contrast, the Riemannian and Eikonal formulations are symmetric by construction and therefore cannot represent this directional preference. These results are consistent among the three flow environments and it demonstrates that RaNTFields captures not only the spatial structure of the travel-time field, but also the directional effects induced by the underlying Randers geometry.

5.1.2 TERRAIN MOTION PLANNING

We next evaluate open-loop path planning in ridge and hill terrains, with results summarized in Table 2. RaNTFields requires substantially less planning time than the baselines in both environments, while maintaining high success rates. In particular, it achieves 1.29 s and 1.17 s per path in ridge and hill terrain, respectively, compared with 2.48–12.26 s and 2.54–11.11 s for the baselines. The resulting paths are also short, with mean normalized lengths of 1.25 and 1.18.

Table 2: Open-loop evaluation of RaNTFields (RaNTF) against the Randers OUM (OUM-R) reference and symmetric Riemannian OUM (OUM-G) and Eikonal OUM (OUM-E) baselines across ridge and hill terrain environments. Results are averaged over 100 randomly sampled start-goal pairs.

Ridge Terrain

	PT (s) ↓	Length ↓	SR ↑	Grad. CS. ↑	Asym. corr. ↑
RaNTF	$\mathbf{1.29 \pm 2.90}$	1.25 ± 0.62	0.98	$\mathbf{0.96 \pm 0.02}$	**0.99**
OUM-R	12.26 ± 0.13	1.32 ± 0.94	0.98	—	—
OUM-G	4.09 ± 0.06	1.33 ± 1.56	**0.99**	0.76 ± 0.04	0.00
OUM-E	2.48 ± 0.04	$\mathbf{1.20 \pm 0.58}$	**0.99**	0.75 ± 0.05	0.00

Hill Terrain

	PT (s) ↓	Length ↓	SR ↑	Grad. CS ↑	Asym. corr. ↑
RaNTF	$\mathbf{1.17 \pm 2.31}$	$\mathbf{1.18 \pm 0.59}$	**0.99**	$\mathbf{0.96 \pm 0.02}$	**0.99**
OUM-R	11.11 ± 0.34	1.31 ± 0.86	0.94	–	–
OUM-G	3.32 ± 0.10	1.43 ± 2.13	0.97	0.85 ± 0.03	0.00
OUM-E	2.54 ± 0.08	1.27 ± 1.51	0.98	0.85 ± 0.03	0.00

RaNTFields further achieves a gradient-direction cosine similarity of 0.96 ± 0.02 in both environments, substantially higher than the reported baselines. Its asymmetry correlation of 0.99 also indicates that the generated paths preserve the directional structure of the underlying travel-time field. Together, these results show that RaNTFields translates the learned asymmetric field into efficient and reliable paths while maintaining its geometric and directional characteristics. See Fig. 5 in Appendix A.6.1 for a graphical comparison.

5.2 UR5E ARM MOTION PLANNING

We further evaluate on a 5-DoF robot arm, an UR5e excluding the end-effector degree of freedom, to examine whether the asymmetric Randers formulation provides benefits for configuration-space motion planning under directional dynamic effects. The simulation experimental setup consists of a robot arm in the presence of a spherical obstacle with a radius of 0.2 m positioned in front it. We randomly select 100 start-goal configuration pairs such that naive interpolation between each pair results in a collision between the arm and the obstacle, requiring the planner to generate a collision-free trajectory. We compare three physics-informed neural PDE models: the Eikonal formulation, a Riemannian formulation with configuration-dependent anisotropy, and the proposed Randers formulation. We additionally compare the resulting Randers planner against sampling-based RRT and RRT* planners. All methods are evaluated using success rate, Euclidean path length, maximum torque (**Max t.**), mean (**Mean t.**), active torque (**Active t.**) and control rate (**CR**).

PDE Model Comparison

	SR ↑	Length ↓	Max t.↓	Mean t.↓	Active t.↓	CR↑
RaNTF	0.96	12.07	**74.20**	**7.46**	55.45	232.4
N-Eik.	0.98	11.47	90.22	7.73	58.67	**254.2**
N-Riem.	**0.99**	11.78	78.05	7.46	**54.28**	243.1

Sampling-Based Comparison

	SR	Length ↓	Max t. ↓	Mean t.↓	Active t. ↓
RaNTF	0.96	**12.07**	**74.57**	**7.48**	**55.91**
RRT	**1.0**	21.45	115.3	7.59	70.41
RRT*	**1.0**	13.11	138.1	7.61	56.99

Table 3: Performance comparison of RaNTFields (RaNTF) with physics-informed neural PDE baselines and sampling-based planners on a 5-DoF robot arm. The PDE comparison includes Neural Eikonal (N-Eik.), Neural Riemannian (N-Riem.), and Randers (RaNTF) formulations, while the sampling-based comparison evaluates RRT and RRT*. Results report success rate (SR), Euclidean path length, maximum torque (Max t.), mean torque (Mean t.), active torque (Active t.), and control rate (CR).

Table 3 shows the comparison between the three PDE formulations. The Randers model achieves a maximum torque of 74.20 N·m, compared with 90.22 N·m for the Eikonal model and 78.05 N·m for the Riemannian model. This corresponds to reductions of approximately 17.8% and 4.9%, respectively. The reduction in peak torque indicates that explicitly modeling directional effects allows the planner to avoid configurations and motions that require large instantaneous torques. The benefit is also reflected in the active torque, where Randers requires 55.45 compared with 58.67 for

Eikonal and 54.28 for Riemannian. While the Riemannian formulation achieves a slightly lower active torque, the Randers model provides a substantially lower maximum torque, suggesting that the asymmetric formulation is particularly effective at avoiding high-effort portions of the trajectory. Mean torque is comparable across the three formulations, with Randers achieving 7.46 N·m, matching the Riemannian model and improving upon Eikonal's 7.73 N·m.

This reduction in dynamic effort comes with a modest increase in geometric path length. The Randers trajectory has a Euclidean path length of 12.07, compared with 11.47 for Eikonal and 11.78 for Riemannian. Thus, the Randers planner accepts a slightly longer geometric trajectory in exchange for improved dynamic characteristics. This behavior is consistent with the motivation of the Randers formulation: the shortest Euclidean path is not necessarily the most desirable path when motion has direction-dependent costs.

The success rate of Randers is 0.96, comparable with 0.98 and 0.99 for Eikonal and Riemannian, respectively. The mean control rate for Randers is approximately 232.4 Hz, capable for real-time applications.

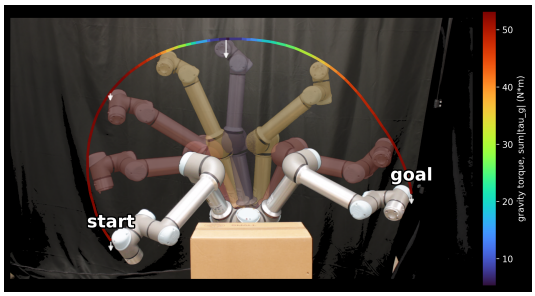

Figure 4: **Hardware execution on a UR5e.** The real robot follows the RaNTF path from *start* to *goal*, preferring a low torque configuration. Snapshots and the end-effector trace are coloured by the static gravity torque $\sum_j |\tau_{g,j}|$, and arrows show the gravity drift $f(q)$.

We next compare the Randers planner with sampling-based RRT and RRT*, as summarized in Table 3. Randers produces substantially shorter paths than both sampling-based methods, with a path length of 12.07 compared with 21.45 for RRT and 13.11 for RRT*. More importantly, the Randers trajectories require considerably lower peak torque: 74.57 N·m versus 115.3 N·m for RRT and 138.1 N·m for RRT*. This represents reductions of approximately 35% and 46%, respectively. Active torque is similarly reduced relative to RRT (55.91 versus 70.41), while remaining close to RRT* (56.99). Mean torque is comparable across all three planners.

These results highlight the distinction between geometric and physics-informed planning. RRT and RRT* optimize primarily for geometric feasibility, whereas the Randers formulation incorporates directional structure directly into the travel-time model. This allows the resulting trajectory to account for configuration-dependent directional effects while maintaining a compact path. The substantially lower maximum torque suggests that the learned Randers field can steer the manipulator away from dynamically unfavorable regions that may be selected by purely geometric planners.

The UR5e experiment therefore shows that incorporating asymmetric directional structure into the neural travel-time formulation can steer the arm through less torque configurations. While Randers does not achieve the highest success rate or shortest path, its lower peak torque demonstrates the potential of direction-dependent travel-time fields for robotic motion planning.

6 Conclusion/Limitations

We presented RaNTFields, a physics-informed neural framework for learning asymmetric, anisotropic travel-time fields for motion planning. By formulating planning as a Zermelo-type navigation problem, RaNTFields combines a configuration-dependent Riemannian metric with directional drift and learns the corresponding Randers Hamilton–Jacobi equation directly from local evaluations, without precomputed travel-time labels. In 2D flow and terrain environments, it closely matched Ordered Upwind solutions, achieving gradient-direction cosine similarities of 0.96–0.99 and asymmetry correlations of 0.95–1.00 while enabling faster path extraction. On a 5-DoF UR5e arm with gravity-induced drift, it reduced peak torque by 17.8% relative to a neural Eikonal planner and 4.9% relative to a neural Riemannian planner, with slightly longer paths and a 96% success rate. Limitations include the first-order kinematic model, the lack of guaranteed convergence to the viscosity solution, and evaluation restricted to simulated, static environments. Future work will investigate learning metric and drift fields from data, extending the framework to second-order dynamics and time-varying flows, and scaling to higher-dimensional and cluttered environments.

AI USE STATEMENT

We have not used generative AI tools to generate research ideas, write proofs, or produce experimental results, data, or reported numbers. We used generative AI tools for the following supporting tasks. **Writing:** ChatGPT was used to improve grammar, sentence structure, and the flow of author-written paragraphs; all technical content and claims are the authors' own. **Code:** Claude was used to generate auxiliary code for plotting, benchmarking, dataset generation, and bash scripts for launching experiments specified by the authors. The core software design, including the model classes, methods, and training framework, was written by the authors, and experiments were designed and run under the authors' direction. AI-generated code was verified with unit tests and by comparing generated data against numerical ground-truth solutions. **Proofs:** All proofs were written by the authors; generative AI was used only as an additional check of proofs already written. **Figures:** All plots are generated from our experimental data. AI assisted in brainstorming the layout of the architecture diagram and in writing Blender Python scripts for its illustrative renderings, which are for visual purposes only and do not depict experimental results. We have reviewed all AI-assisted work and take full responsibility for the final content of this paper, including text, claims, code, and figures produced with the aid of generative AI.

ETHICS STATEMENT

To the best of our knowledge, our work does not raise any questions regarding the Code of Ethics.

REPRODUCIBILITY STATEMENT

To make our results reproducible, the complete source code for our method, including dataset generation, model architectures, loss functions, and training and evaluation pipelines, will provided in the camera ready version. Section 4.2.2 and Appendix A.3 describe the model architectures, training objectives, and some hyperparameters. Appendix A.4 describes the environments, the obstacle and flow-field configurations, and the procedure for generating ground-truth travel times. For the camera-ready version, we will publicly release the code together with scripts and configuration files that reproduce every figure, table, and experimental result in the paper.

ACKNOWLEDGMENTS

Proper acknowledgments will be provided in the camera-ready version to prevent disclosure of author anonymity in the review process.

REFERENCES

Somil Bansal and Claire Tomlin. Deepreach: A deep learning approach to high-dimensional reachability, 2020. URL https://arxiv.org/abs/2011.02082.

David Bao, Colleen Robles, and Zhongmin Shen. Zermelo navigation on riemannian manifolds, 2003. URL https://arxiv.org/abs/math/0311233.

Qingyi Chen, Zachary Kingston, and Ahmed H. Qureshi. Nehmo: Neural hamilton-jacobi reachability learning for decentralized safe multi-arm motion planning, 2026. URL https://arxiv.org/abs/2607.00326.

Xinyue Cheng and Zhongmin Shen. *Finsler Geometry: An Approach via Randers Spaces*. Springer, 2012.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Ross L. Hatton, Yousef Salaman, and Shai Revzen. Asymmetric friction in geometric locomotion, 2026. URL https://arxiv.org/abs/2512.22484.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning, 2011. URL https://arxiv.org/abs/1105.1186.

Yiming Li, Jiacheng Qiu, and Sylvain Calinon. A riemannian take on distance fields and geodesic flows in robotics, 2026. URL https://arxiv.org/abs/2412.05197.

Tomas Merva, Saray Bakker, Max Spahn, Danning Zhao, Ivan Virgala, and Javier Alonso-Mora. Globally-guided geometric fabrics for reactive mobile manipulation in dynamic environments. *IEEE Robotics and Automation Letters*, 10(6):5553–5560, 2025. doi: 10.1109/LRA.2025.3562005.

Ruiqi Ni and Ahmed H. Qureshi. Ntfields: Neural time fields for physics-informed robot motion planning, 2023a. URL https://arxiv.org/abs/2210.00120.

Ruiqi Ni and Ahmed H. Qureshi. Progressive learning for physics-informed neural motion planning, 2023b. URL https://arxiv.org/abs/2306.00616.

Ruiqi Ni, Zherong Pan, and Ahmed H Qureshi. Physics-informed temporal difference metric learning for robot motion planning, 2025. URL https://arxiv.org/abs/2505.05691.

Ruiqi Ni, Yuchen Liu, and Ahmed H. Qureshi. Weakly-supervised learning for physics-informed neural motion planning via sparse roadmap, 2026. URL https://arxiv.org/abs/2604.13204.

Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(88)90002-2. URL https://www.sciencedirect.com/science/article/pii/0021999188900022.

Nathan D. Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies, 2018. URL https://arxiv.org/abs/1801.02854.

J. Ricardo Sánchez-Ibáñez, Carlos J. Pérez-Del-Pulgar, Javier Serón, and Alfonso García-Cerezo. Optimal path planning using a continuous anisotropic model for navigation on irregular terrains. *Intelligent Service Robotics*, 16(1):19–32, Mar 2023. ISSN 1861-2784. doi: 10.1007/s11370-022-00450-6. URL https://doi.org/10.1007/s11370-022-00450-6.

J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999. doi: 10.1137/S0036144598347059. URL https://doi.org/10.1137/S0036144598347059.

James A. Sethian and Alexander Vladimirsky. Ordered upwind methods for static hamilton–jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003. doi: 10.1137/S0036142901392742. URL https://doi.org/10.1137/S0036142901392742.

Xujie Shen, Haocheng Peng, Zesong Yang, Juzhan Xu, Hujun Bao, Ruizhen Hu, and Zhaopeng Cui. Pc-planner: Physics-constrained self-supervised learning for robust neural motion planning with shape-aware distance function. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, pp. 1–11. ACM, December 2024. doi: 10.1145/3680528.3687651. URL http://dx.doi.org/10.1145/3680528.3687651.

Alex Shum, Kirsten Morris, and Amir Khajepour. Direction-dependent optimal path planning for autonomous vehicles. *Robotics and Autonomous Systems*, 70:202–214, 2015. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2015.02.003. URL https://www.sciencedirect.com/science/article/pii/S0921889015000214.

Jonathan D. Smith, Kamyar Azizzadenesheli, and Zachary E. Ross. Eikonet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59 (12):10685–10696, December 2021. ISSN 1558-0644. doi: 10.1109/tgrs.2020.3039165. URL http://dx.doi.org/10.1109/TGRS.2020.3039165.

Karl Van Wyk, Mandy Xie, Anqi Li, Muhammad Asif Rana, Buck Babich, Bryan Peele, Qian Wan, Iretiayo Akinola, Balakumar Sundaralingam, Dieter Fox, Byron Boots, and Nathan D. Ratliff. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7(2):3202–3209, 2022. doi: 10.1109/LRA.2022.3143311.

Tongzhou Wang and Phillip Isola. Improved representation of asymmetrical distances with interval quasimetric embeddings, 2024. URL https://arxiv.org/abs/2211.15120.

Mandy Xie, Ankur Handa, Stephen Tyree, Dieter Fox, Harish Ravichandar, Nathan D. Ratliff, and Karl Van Wyk. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1355–1367. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/xie23a.html.

Ernst Zermelo. Über das navigationsproblem bei ruhender oder veränderlicher windverteilung. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 11(2):114–124, 1931. doi: 10.1002/zamm.19310110205. URL https://doi.org/10.1002/zamm.19310110205.

A Appendix

A.1 Randers PDE derivation from Optimal control

Here we show the Randers travel-time formulation from an optimal control perspective. Let $q(t) \in \mathcal{Q}$ be the robot's configuration and $u(t) \in \mathbb{R}^d$ be the controllable velocity of a fully actuated d-DOF robot. Define $f(q)$ to be the drift at a configuration q. We assume that the control satisfies the Riemannian unit ball constraint and the drift is smaller than the maximum control speed. From classical optimal control theory we have the following:

$$T(q, q_g) = \min_{u(t)} \int_t^{t_g} \frac{1}{S(q(t))}\, dt$$
$$\text{s.t.} \quad q(t) = q_s, \qquad q(t_g) = q_g, \qquad \dot{q} = f(q) + u, \tag{10}$$
$$u^\top G u \leq 1, \qquad |f(q)|_G < 1$$

Let $T(q, q_g)$ denote the optimal cost-to-go from q to q_g. The stationary Hamilton Jacobi Bellman (HJB) equation is:

$$0 = \min_{u^T G u \leq 1} \left[\frac{1}{S(q)} + \nabla T(q, q_g)^T (f(q) + u) \right] \tag{11}$$

Let

$$p = \nabla T(q, q_g) \tag{12}$$

Then

$$0 = \frac{1}{S(q)} + p^T f(q) + \min_{u^T G u \leq 1} p^T u \tag{13}$$

The minimum of a linear function over the G-unit ball is:

$$\min_{u^T G u \leq 1} p^T u = -\sqrt{p^T G^{-1} p} \tag{14}$$

Therefore, we get the following Randers PDE:

$$\sqrt{\nabla T(q, q_g)^T G^{-1} \nabla T(q, q_g)} = \frac{1}{S(q)} + f(q)^T \nabla T(q, q_g) \tag{15}$$

which we write compactly as follows:

$$|\nabla T(q, q_g)|_{G^{-1}} = \frac{1}{S(q)} + f(q)^T \nabla T(q, q_g) \tag{16}$$

We also get the optimal control u^* as follows:

$$u^*(q) = -\frac{G(q)^{-1} \nabla_q T_\theta(q, q_g)}{\sqrt{\nabla_q T_\theta(q, q_g)^\top G(q)^{-1} \nabla_q T_\theta(q, q_g)}}. \tag{17}$$

A.2 Randers form

In this section we show that the optimal-control problem of Appendix A.1 induces a Randers metric, following the Zermelo navigation construction of Bao et al. (2003). Throughout, $\langle a, b \rangle_G = a^\top G(q)\, b$ and $\|a\|_G = \sqrt{\langle a, a \rangle_G}$ denote the inner product and norm induced by $G(q)$.

A.2.1 Admissible Velocity Set

Let $y = \dot{q}$ denote the velocity of a path. Since $\dot{q} = f(q) + u$, the control is $u = y - f$, and the control constraint $u^\top G u \leq 1$ becomes

$$(y - f)^\top G(q)\, (y - f) \leq 1. \tag{18}$$

The admissible velocity set is therefore the G-unit ellipsoid centered at $f(q)$. This shift of the velocity ellipsoid by the drift is the geometric origin of the Randers structure.

A.2.2 Derivation of the Finsler Metric

Write $y = \rho\, n$ with $\|n\|_G = 1$, where $\rho > 0$ is the maximal speed attainable in direction n. On the boundary of the admissible set,

$$\|\rho\, n - f\|_G = 1. \tag{19}$$

Squaring gives

$$\rho^2 - 2\rho\ \langle f, n\rangle_G + \|f\|_G^2 = 1, \tag{20}$$

whose positive root is

$$\rho = \langle f, n\rangle_G + \sqrt{1 - \|f\|_G^2 + \langle f, n\rangle_G^2}. \tag{21}$$

The root is real and positive for every direction n because $\|f\|_G < 1$.

Let s_G denote arc length measured in G. Since $ds_G = \rho\, dt$, the elapsed time is $dt = ds_G/\rho$, and with running cost $1/S(q)$ the accumulated cost satisfies

$$dJ = \frac{1}{S(q)}\, dt = \frac{1}{S(q)\,\rho}\, ds_G. \tag{22}$$

The cost per unit G-length in direction n is therefore $F(q, n) = 1/(S\rho)$. Rationalizing Eq. equation 21 and extending by positive homogeneity to an arbitrary tangent vector y yields

$$F(q, y) = \frac{\sqrt{\left(1 - \|f\|_G^2\right)\|y\|_G^2 + \langle f, y\rangle_G^2}\ -\ \langle f, y\rangle_G}{S(q)\left(1 - \|f\|_G^2\right)}. \tag{23}$$

The optimal trajectory minimizes the Finsler length

$$J[\gamma] = \int_\gamma F\big(q, \dot{q}\big)\, dt. \tag{24}$$

A.2.3 Randers Form

A Randers metric has the form $F(q, y) = \alpha(q, y) + \beta(q, y)$, where $\alpha(q, y) = \sqrt{a_{ij}(q)\, y^i y^j}$ is a Riemannian norm and $\beta(q, y) = b_i(q)\, y^i$ is a one-form. Splitting Eq. equation 23 gives

$$\alpha(q, y) = \frac{\sqrt{\left(1 - \|f\|_G^2\right)\|y\|_G^2 + \langle f, y\rangle_G^2}}{S(q)\left(1 - \|f\|_G^2\right)}, \tag{25}$$

$$\beta(q, y) = -\,\frac{\langle f, y\rangle_G}{S(q)\left(1 - \|f\|_G^2\right)}. \tag{26}$$

Equivalently, in coordinates, with $\kappa(q) = 1 - \|f\|_G^2$,

$$A(q) = \frac{\kappa\, G + G f f^\top G}{S^2 \kappa^2}, \qquad b(q) = -\,\frac{G f}{S\,\kappa}, \tag{27}$$

so that $\alpha(q, y) = \sqrt{y^\top A\, y}$ and $\beta(q, y) = b^\top y$. Hence $F = \alpha + \beta$ is exactly of Randers form.

Positivity. A Randers metric is positive definite if and only if $\|b\|_{A^{-1}} < 1$. A direct computation from Eq. equation 27 gives $\|b\|_{A^{-1}} = \|f\|_G$, so the Randers condition is equivalent to the drift feasibility condition $\|f\|_G < 1$ enforced in Appendix A.3. The factor $1/S(q)$ scales α and β equally and does not affect this condition.

Special cases. Setting $f = 0$ gives $\beta = 0$ and $\alpha(q, y) = \|y\|_G / S(q)$, recovering the Riemannian travel-time metric of Eq. (3). Additionally setting $G = I$ recovers the isotropic Eikonal case.

A.3 Neural Network Architecture

The proposed travel-time prediction model consists of a shared coordinate encoder followed by two complementary prediction branches: a residual branch and a Randers-metric branch. The outputs of the two branches are combined using a learnable convex weighting. An overview of the architecture is illustrated in Figure 2.

A.3.1 Shared Coordinate Encoder

Given the start configuration q_s and goal configuration q_g, the model first maps the inputs into a common embedding space of dimension d_e. In particular, the encoder produces embeddings for the start, goal and displacement configurations:

$$\mathbf{e}_s = E(q_s), \qquad \mathbf{e}_g = E(q_g), \qquad \mathbf{e}\Delta = E(q_g - q_s) \tag{28}$$

The same coordinate encoder $E(\cdot)$ is shared across all three inputs. The encoder is a two-layer multilayer perceptron (MLP) with a $tanh$ nonlinear activation function. The output of the projector has dimension d_e.

A.3.2 Residual Prediction Branch

The residual branch directly learns a positive travel-time function from the encoded spatial relationship. The three coordinate embeddings are concatenated to form

$$\mathbf{z} = [\mathbf{e}_\Delta; \mathbf{e}_s; \mathbf{e}_g] \in \mathbb{R}^{3d_e}. \tag{29}$$

The concatenated representation is projected into a hidden space of dimension d_h using a linear layer, LayerNorm, and a $tanh$ nonlinear activation function.

The hidden representation is subsequently processed by K residual blocks. The final hidden representation is mapped to a scalar and passed through a Softplus activation function, a non-negative $\hat{t}_{residual}$ value.

A.3.3 Randers-Metric Prediction Branch

The Randers branch combines a learned quasimetric distance with a positive, configuration dependent scaling function. First, an MLP with the same general structure as the residual branch is applied to the encoded pair, resulting in a non-negative scalar $m(\mathbf{e})$.

In parallel, the start and goal embeddings are passed to an Interval Quasimetric Embedding (IQE) distance layer Wang & Isola (2024), which produces a quasimetric distance:

$$d_{\mathrm{IQE}} = d_{\mathrm{IQE}}(\mathbf{e}_s, \mathbf{e}_g). \tag{30}$$

The Randers branch then computes the travel-time prediction as the product of the learned quasi-distance and the positive scaling function:

$$\hat{t}_{randers} = d_{\mathrm{IQE}}(\mathbf{e}_s, \mathbf{e}_g) * m(\mathbf{e}). \tag{31}$$

This multiplicative formulation allows the model to separate the geometric component of travel time, represented by the learned quasimetric distance, from a learned spatially dependent scaling factor.

A.3.4 Learnable Combination of Prediction Branches

The final model combines the residual and Randers predictions using a learnable scalar coefficient. Specifically, let $a \in \mathbb{R}$ denote a learnable parameter. The mixing coefficient is defined as

$$\alpha = \sigma(a), \tag{32}$$

where $\sigma(\cdot)$ is the sigmoid function to transform the learnable parameter to an open interval of $(0, 1)$. The final prediction is then given by

$$\hat{t} = \alpha \hat{t}_{randers} + (1 - \alpha)\hat{t}_{residual} \tag{33}$$

Since $\alpha \in (0, 1)$, the final prediction is a convex combination of the Finsler and residual predictions. The parameter a is initialized to zero, resulting in $\alpha = 0.5$ at initialization. Consequently, both branches initially contribute equally to the prediction, while training allows the model to learn the appropriate balance between the geometric and residual components.

A.4 Data generation

We generate training samples directly in the continuous configuration space rather than constructing a discretized representation of the environment. Each configuration q is associated with an obstacle-aware speed field $S(q)$, a Riemannian metric $G(q)$, and a drift field $f(q)$, which together define the Randers PDE equation introduced in Section 4.1. The same data-generation procedure is used for all experimental domains, while the physical interpretation of $G(q)$ and $f(q)$ differs between the terrain and robot arm manipulator settings.

A.4.1 Obstacle-Aware Speed Field

We incorporate obstacle avoidance through the spatially varying speed field used by NTFields as discussed in Eq. 2.

This construction assigns low speeds to configurations near obstacles and progressively increases the speed with obstacle clearance, approaching one sufficiently far from obstacles. Since the travel cost is proportional to $1/S(q)$, configurations near obstacles incur a larger traversal cost and are consequently discouraged by the learned distance field. Within obstacle regions, $S(q)$ is assigned a small positive numerical floor of 10^{-4}. In environments without obstacles, we set $S(q) = 1$ throughout the domain.

For the terrain experiments, obstacle distances are evaluated from analytic geometric primitives. For manipulator experiments, they are determined from a spherized UR5e robot arm collision model and workspace obstacles, which can either be primitive shapes or point clouds.

A.4.2 Terrain Environment

We first consider a two-dimensional point mass moving over a spatially varying terrain. The normalized configuration of the point mass is $q = [x, y]^\top \in [-1, 1]^2$, and the terrain is represented by an elevation field $h(q)$. The elevation induces a configuration-dependent Riemannian metric through the pullback of the Euclidean metric on the terrain surface,

$$G(q) = I + \nabla h(q) \nabla h(q)^\top. \tag{34}$$

For $h_x = \partial h/\partial x$ and $h_y = \partial h/\partial y$, this metric is equivalently

$$G(q) = \begin{bmatrix} 1 + h_x^2 & h_x h_y \\ h_x h_y & 1 + h_y^2 \end{bmatrix}. \tag{35}$$

The metric therefore increases the cost of motion according to the local terrain slope and produces spatially varying anisotropy. Note that when the elevation field is 0, the identity metric used in the eikonal equation is recovered.

To introduce directional asymmetry, we model the gravitational tendency of the point mass to move downhill. The negative normalized terrain gradient gives the drift direction:

$$f(q) = c * \hat{u}(q), \qquad \hat{u}(q) = -\frac{\nabla h(q)}{|\nabla h(q)|_2}, \tag{36}$$

where $c > 0$ is a global drift magnitude. At locally flat configurations, where $|\nabla h(q)|_2 = 0$, we set $f(q) = 0$. Importantly, c is fixed globally for a given terrain rather than being tuned independently at each configuration. We choose it such that the Randers feasibility condition is satisfied throughout the domain,

$$f(q)^\top G(q) f(q) < 1, \qquad \forall q \in \mathcal{Q}. \tag{37}$$

Using Eq. equation 36, the corresponding scale is

$$c = \sqrt{\frac{1}{\max_{q \in \mathcal{Q}} \hat{u}(q)^\top G(q) \hat{u}(q)}}. \tag{38}$$

Thus, the spatial variation of the terrain drift is determined solely by the local downhill direction, while its magnitude is controlled by a single globally calibrated parameter. This provides a physically motivated source of directional asymmetry while maintaining the feasibility required by the Randers formulation.

We consider two spatially varying elevation profiles, including radial hills and ridges. In particular, we use non-planar elevation fields to expose the configuration-dependent nature of the induced metric. A planar slope produces a constant ∇h and consequently a constant metric, whereas spatially varying hills produce a genuinely configuration-dependent anisotropic geometry. The gradients of the elevation fields are evaluated analytically.

A.4.3 Manipulator Environment

We additionally consider a UR5e manipulator and only consider its first five joints, ignoring the end-effector. In this setting, the Riemannian metric is derived from the robot's configuration-dependent rigid-body dynamics. Specifically, we define

$$G(q) = \frac{M(q)}{m_{\text{ref}}} + \lambda_{\text{reg}} I, \tag{39}$$

where $M(q)$ is the mass matrix, m_{ref} is a reference scaling factor, and $\lambda_{\text{reg}} \geq 0$ is an optional regularization parameter. The mass matrix defines a configuration-dependent geometry in which motion along different joint directions has different costs, thereby providing a physically grounded source of anisotropy.

The directional drift is induced by gravity,

$$f(q) = -M(q)^{-1} \tau_{\text{gravity}}(q), \tag{40}$$

where $\tau_{\text{gravity}}(q)$ denotes the gravity torque at configuration q. Unlike a synthetic flow field, this drift is determined by the physical dynamics of the manipulator and varies with its configuration. The resulting formulation therefore combines inertial anisotropy through $G(q)$ with gravity-induced directional asymmetry through $f(q)$.

The mass matrix is normalized by m_{ref} to maintain a numerically well-scaled metric, while λ_{reg} can be used to improve the conditioning of $G(q)^{-1}$. This is important because the inverse metric appears explicitly in the Randers-HJB equation and in the corresponding optimal control law. The same regularized metric is used consistently when evaluating the drift feasibility condition and the HJB residual.

As in the terrain setting, the drift magnitude is globally calibrated to remain within the controllable velocity set. Specifically, we enforce

$$\max_{q \in \mathcal{Q}} f(q)^\top G(q) f(q) < 1 \tag{41}$$

The resulting drift therefore satisfies the Randers admissibility condition over the full configuration space while retaining the configuration-dependent direction dictated by the robot's gravitational dynamics.

A.4.4 CONTINUOUS CONFIGURATION-SPACE SAMPLING

For both domains, training samples are generated by directly sampling configurations from the continuous configuration space. Start-goal pairs (q, q_g) are subsequently formed from the sampled configurations. For each configuration, we evaluate the local quantities

$$(S(q), G(q), f(q)) \tag{42}$$

using the corresponding environment or robot model. These quantities are then used to evaluate the Randers-HJB residual during neural-network training.

Consequently, the generated dataset consists of local evaluations of the underlying geometry, drift, and obstacle-dependent speed rather than expensive precomputed distance labels. This enables the same physics-informed learning procedure to be applied to both low-dimensional terrain environments and the higher-dimensional configuration space of the manipulator.

Overall, the two environments instantiate the same asymmetric anisotropic planning problem through different physical mechanisms:

Environment	$G(q)$	$f(q)$
Terrain	$I + \nabla h(q)\nabla h(q)^\top$	$-c\,\dfrac{\nabla h(q)}{\|\nabla h(q)\|_2}$
UR5e	$\dfrac{M(q)}{m_{\text{ref}}} + \lambda_{\text{reg}} I$	$-M(q)^{-1}\tau_{\text{gravity}}(q)$

(43)

while the obstacle-aware speed field $S(q)$ and the Randers feasibility condition $f(q)^\top G(q) f(q) < 1$ are shared across both settings. This common construction allows us to evaluate whether the proposed neural Randers formulation can capture asymmetric anisotropic geometry arising from both environmental terrain and robot dynamics.

A.5 CONTROLLER DETAILS

Optimal control from the learned field. Let $T_\theta(q, q_g)$ be the predicted travel time from q to goal q_g and $p = \nabla_q T_\theta$, computed by automatic differentiation. Under the Randers metric with Riemannian part $G(q)$ and drift $f(q)$, the minimizing control and the resulting velocity are

$$u^*(q) = -\frac{G(q)^{-1}p}{\sqrt{p^\top G(q)^{-1}p}}, \qquad \dot{q} = f(q) + u^*(q). \tag{44}$$

Both controllers use the unit direction u^*. When $G = I$ and $f = 0$ this reduces to plain normalized gradient descent.

2D: open-loop gradient descent. We work in C-space coordinates normalized to $[-1, 1]^2$. Starting from $q_0 = q_s$, we iterate

$$q_{k+1} = \left(q_k + \eta_k\, u_k^*\right), \qquad \eta_k = \alpha_k \min(\eta_0,\ 0.1\,\|q_k - q_g\|), \tag{45}$$

with $\eta_0 = 0.005$. Π is the projection onto the box. Each step has a fixed length, so the path does not stall where $\|p\|$ is small. The cap $0.1\|q_k - q_g\|$ refines the step as the path approaches the goal. A fixed step can overshoot a sharp feature of T_θ and fall into a 2-cycle, so we backtrack. If consecutive directions reverse (${u_k^*}^\top {u^*}_{k-1} < -0.5$), the multiplier becomes $\alpha_k \leftarrow \max(0.5\,\alpha_{k-1}, 0.05)$. Otherwise it recovers as $\alpha_k \leftarrow \min(1, 1.15\,\alpha_{k-1})$. Descent stops when $\|q_k - q_g\| < 0.02$ or after 2000 steps. The drift $f(q_k)$ and the metric $G(q_k)$ are evaluated at every step analytically. No goal-attraction term is added, so the extracted path is the pure integral curve of $f + u^*$. This makes it directly comparable to the characteristic that the OUM ground-truth solver traces.

UR5e: closed-loop geodesic torque control. We simulate in MuJoCo with timestep $\Delta t = 2\,\text{ms}$ and joint torques as the only actuation. Every $N = 10$ substeps (50 Hz) the controller measures q and computes the desired velocity

$$\dot{q}_{\text{des}} = v_c\, \min\!\Big(1, \tfrac{\|q_g - q\|}{r}\Big)\, \frac{\lambda\, u^*(q) + (1-\lambda)\,\hat{e}(q)}{\|\lambda\, u^*(q) + (1-\lambda)\,\hat{e}(q)\|}, \qquad \hat{e} = \frac{q_g - q}{\|q_g - q\|}. \tag{46}$$

Here u^* is mapped from normalized to joint coordinates. $v_c = 1.0\,\text{rad/s}$ is the cruise speed and $r = 0.2\,\text{rad}$ is the radius over which the command slows linearly to zero. The blend weight λ mixes the field direction with a direct pull toward the goal. The pull makes convergence robust in the final approach, where small model errors in p are comparable to the remaining distance. $\dot{q}_{\text{des}}$ is held constant between queries. The torque loop runs every substep ($500\,\text{Hz}$) on the current state:

$$\ddot{q}_{\text{des}} = \text{clip}\left(\frac{\dot{q}_{\text{des}} - \dot{q}}{\tau_c},\ \pm\ddot{q}_{\max}\right), \tag{47}$$

$$\tau = \text{clip}(M(q)\,\ddot{q}_{\text{des}} + C(q,\dot{q})\,\dot{q} + g(q),\ \pm\tau_{\max}), \tag{48}$$

with time constant $\tau_c = N\Delta t = 20\,\text{ms}$ and $\ddot{q}_{\max} = 3\,\text{rad/s}^2$. $\tau_{\max}$ is the UR5e actuator limit, and M, C and g come from MuJoCo inverse dynamics.

A.6 Experiments

A.6.1 Terrain

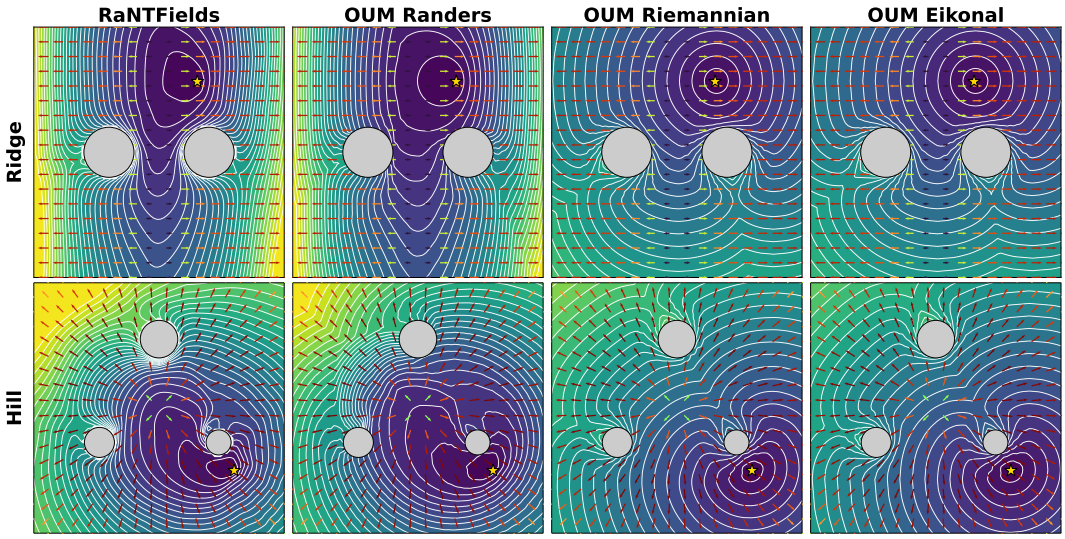

Figure 5: Comparison of RaNTFields with numerical methods on Ridge and Hill terrains. RaNTFields is able to capture accurate travel times, consistent with the reference OUM Randers.