University of Amsterdam

Hybrid Intelligence

Equivariant Eikonal Neural Networks

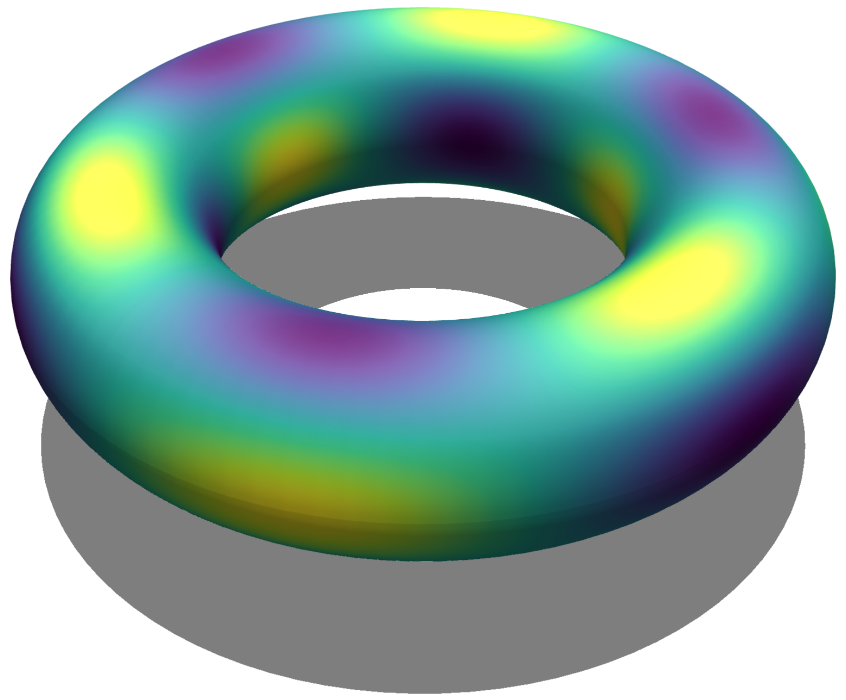
Grid-free, scalable travel-time prediction on homogeneous spaces

Alejandro García-Castellanos, David R. Wessels, Nicky J. van den Berg, Remco Duits, Daniël M. Pelt, Erik J. Bekkers

Take a picture to
download the full paper

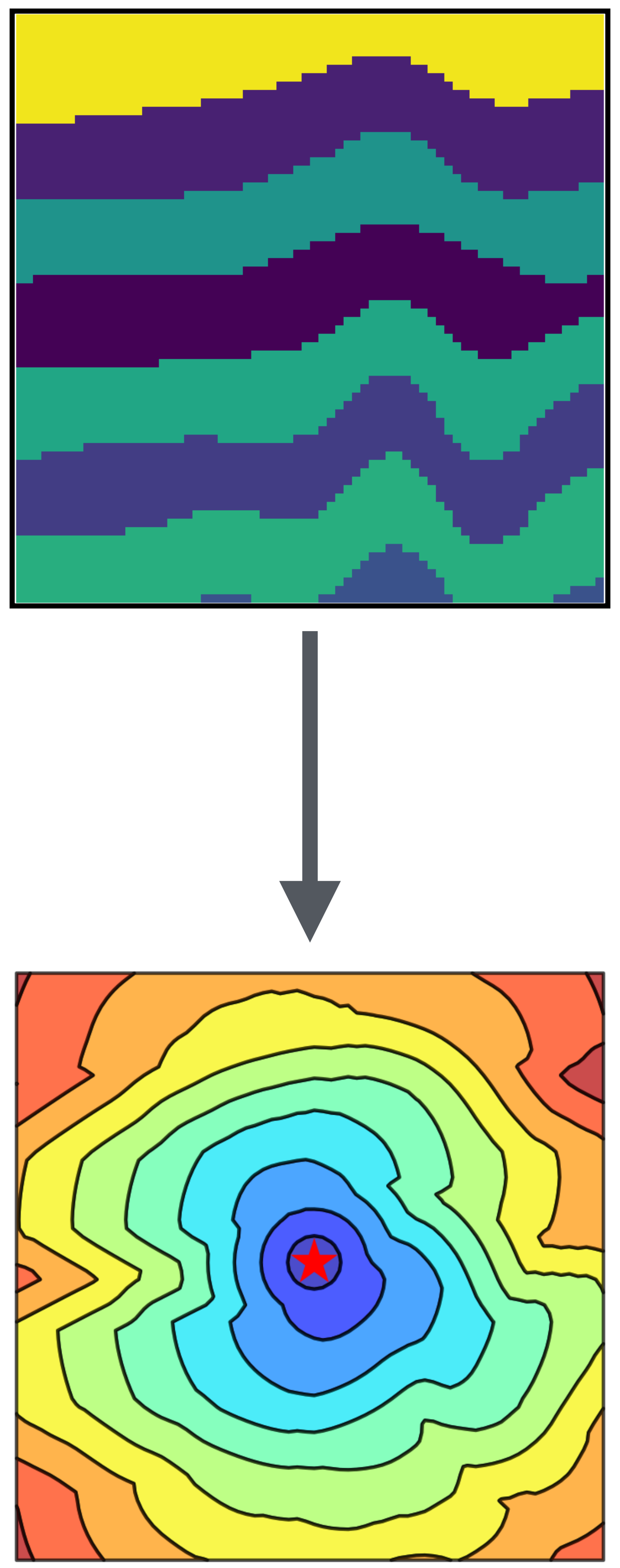
Motivation and related work

The eikonal equation



On a Riemannian manifold $(\mathscr{M}, \mathscr{G})$, the two-point Riemannian *Eikonal equation* with respect to a velocity field $v : \mathscr{M} \to [v_{\min}, v_{\max}]$ (where $0 < v_{\min} \leq v_{\max} < \infty$) is:

$$\begin{cases} \| \operatorname{grad}_s T(s, r) \|_{\mathscr{G}} = v(s)^{-1}, \\ \| \operatorname{grad}_r T(s, r) \|_{\mathscr{G}} = v(r)^{-1}, \\ T(s, r) = T(r, s), \quad T(s, s) = 0, \end{cases}$$



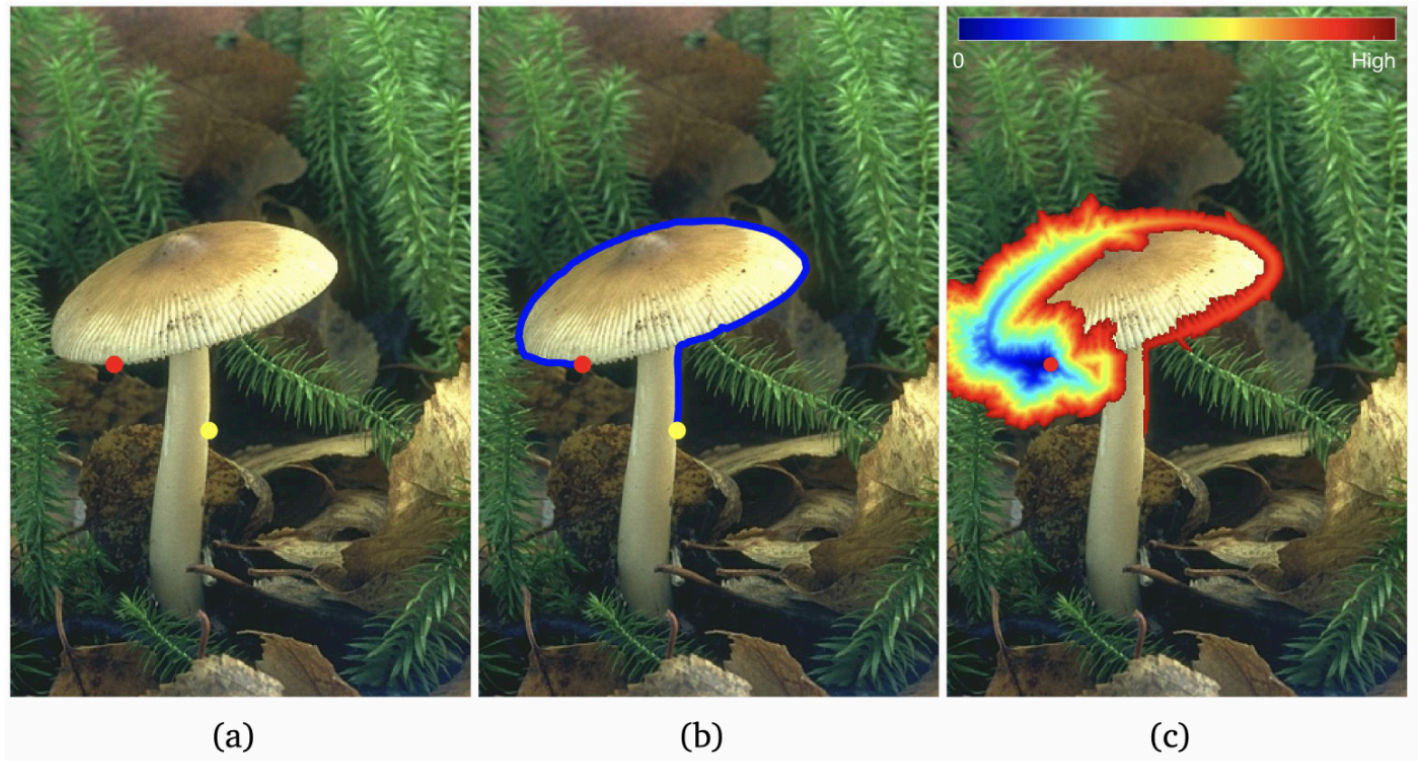
where grad_s and grad_r denote the Riemannian gradients with respect to the source $s \in \mathscr{M}$ and the receiver $r \in \mathscr{M}$, respectively.

The solution $T\colon \mathscr{M} \times \mathscr{M} \to \mathbb{R}_+$ corresponds to the travel-time function, and the interval $[v_{\min}, v_{\max}]$ specifies the minimum and maximum velocity values in the training set.

Where can I use the eikonal equation?

Computer Vision

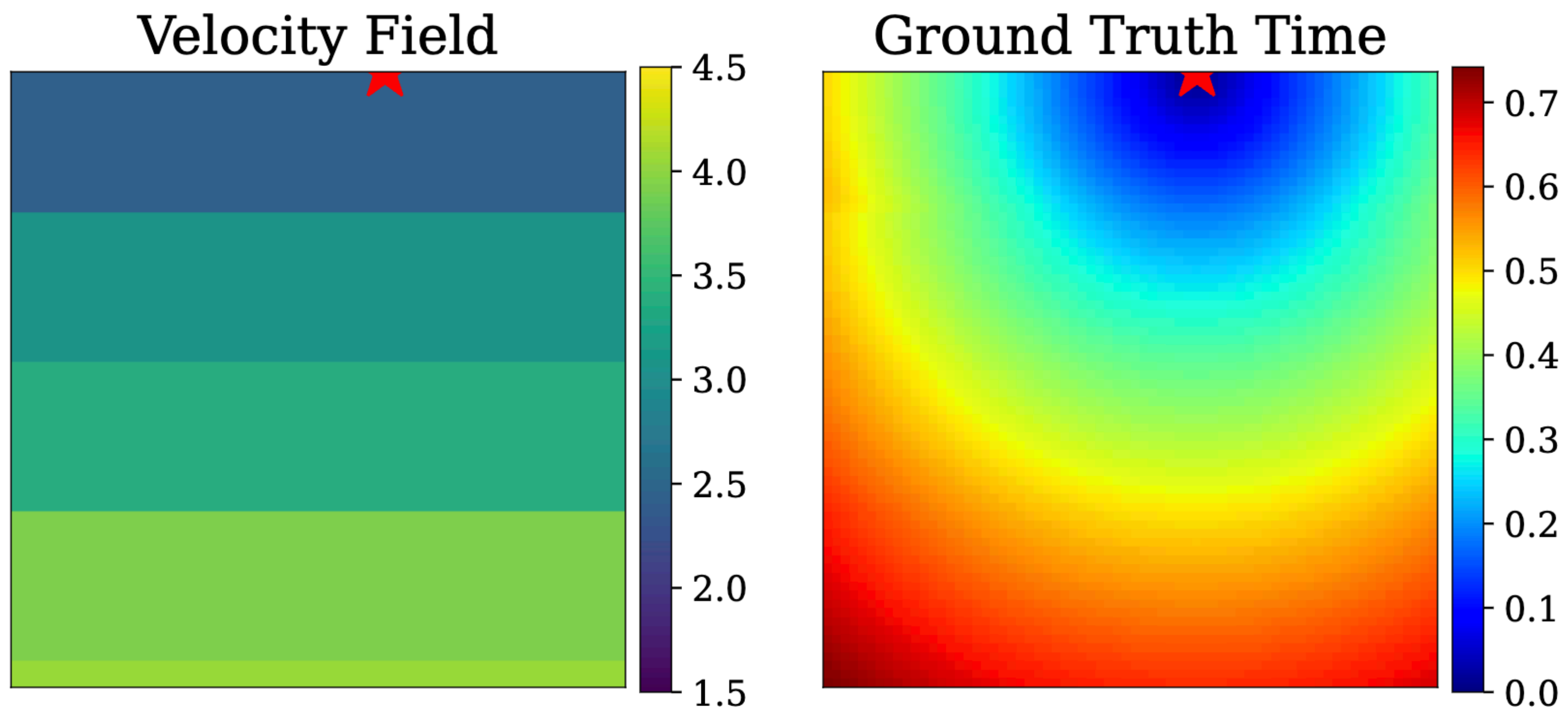
Geodesic Segmentation



Sign Distance Functions

Seismology

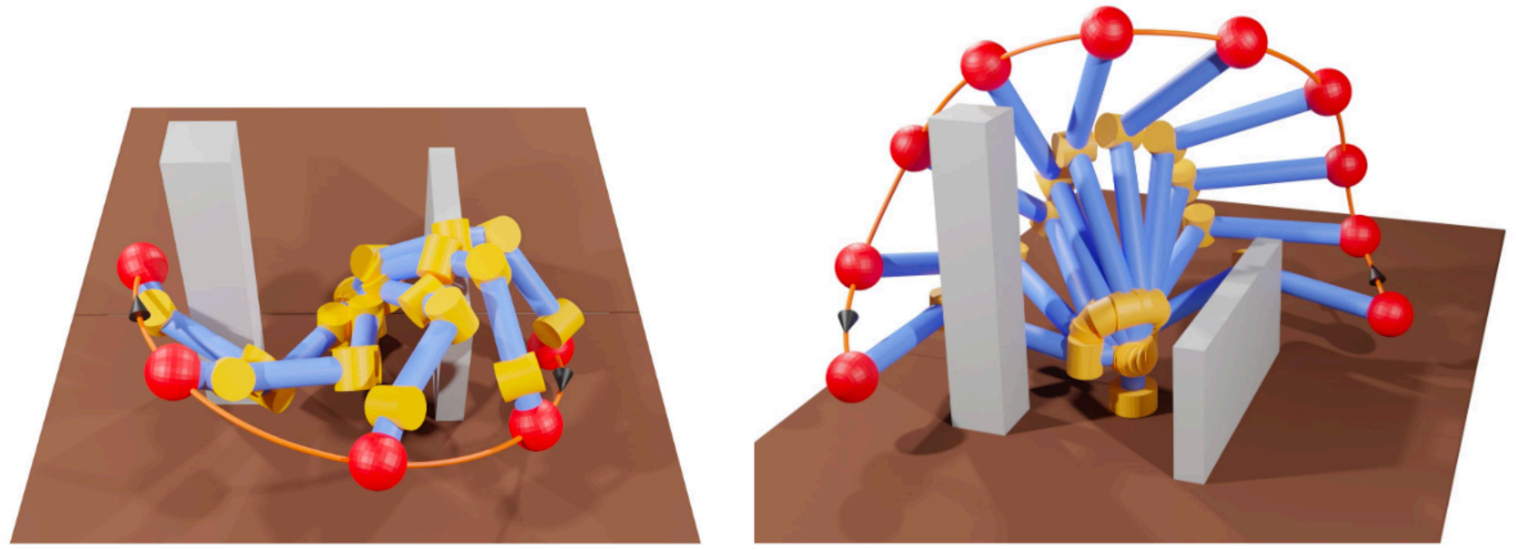
Travel-time predictions



Ray-tracing

Robotics

Motion-planing

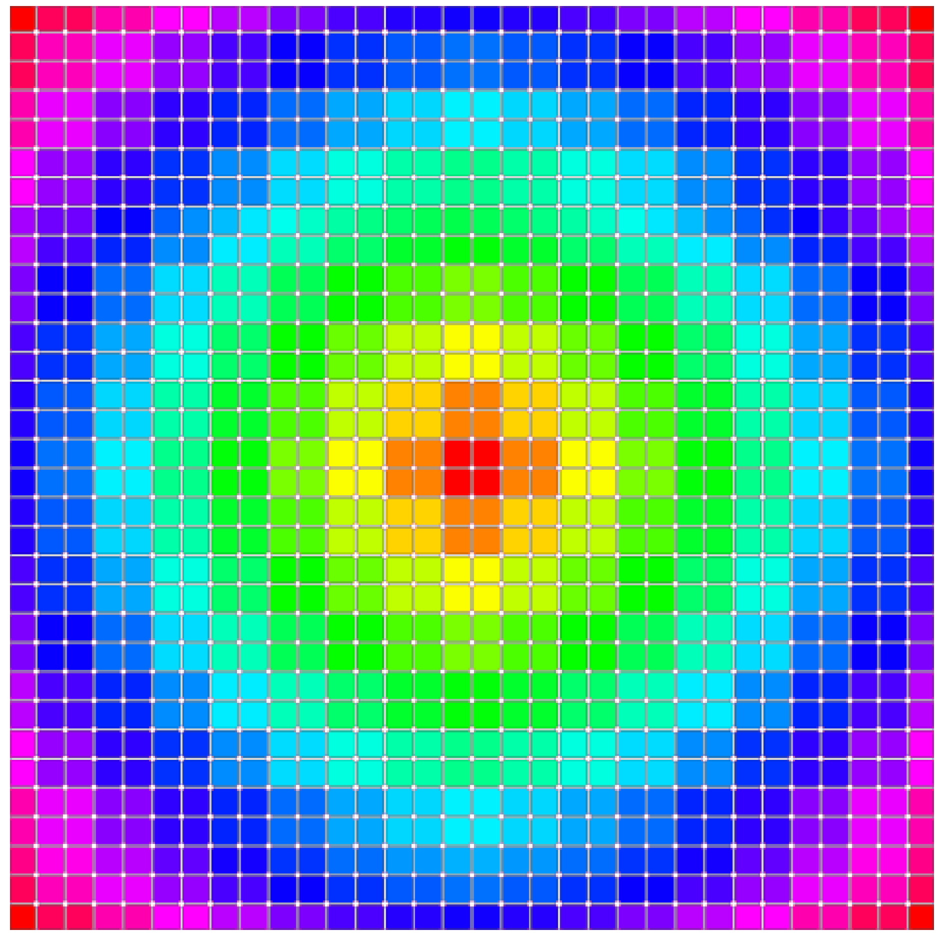


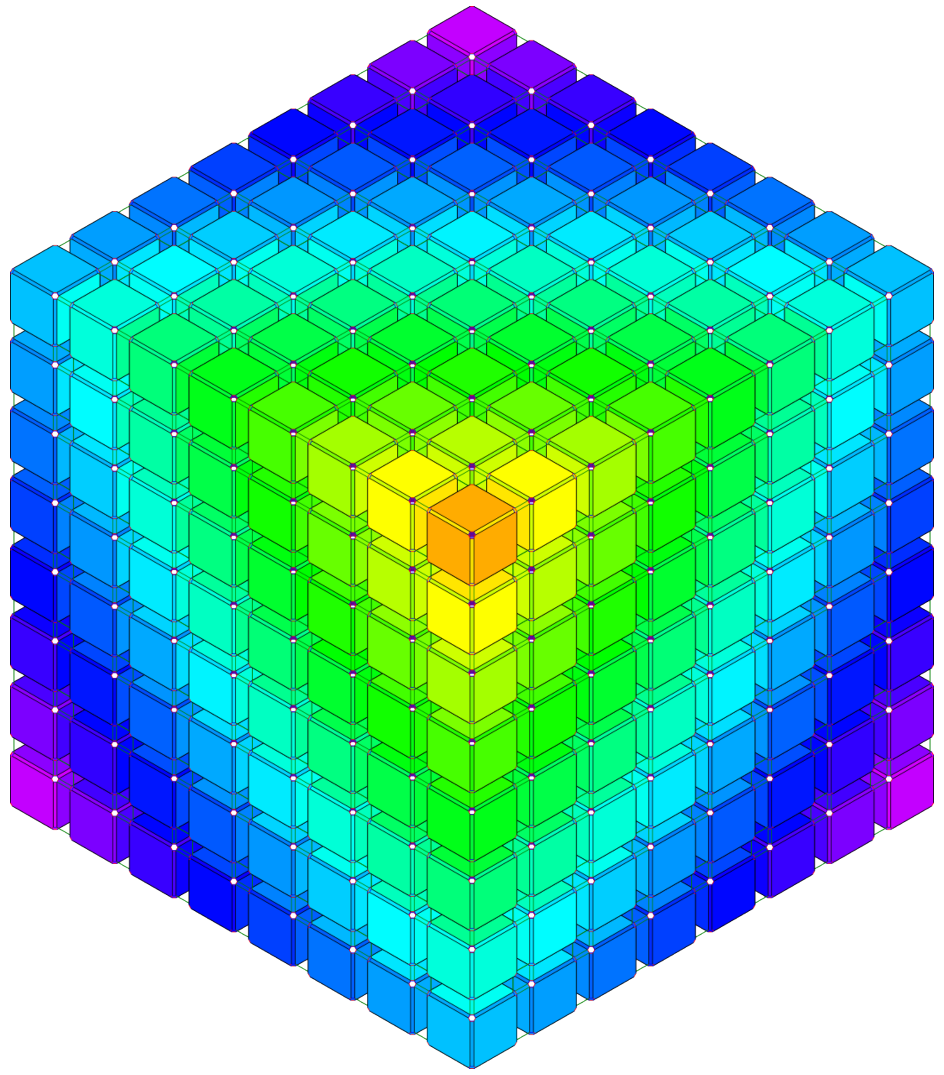
Inverse kinetics

Segmentation Figure from: Da Chen and Laurent D. Cohen. From Active Contours to Minimal Geodesic Paths: New Solutions to Active Contours Problems by Eikonal Equations, 2019.
Robotics Figure from: Ruiqi Ni and Ahmed H. Qureshi. NTFields: Neural Time Fields for Physics-Informed Robot Motion Planning, March, 2023.

Limitations of current eikonal solvers

- Classical: **Fast Marching Method (FMM)**
  - Bad memory and computation scalability
  - Requires discretization



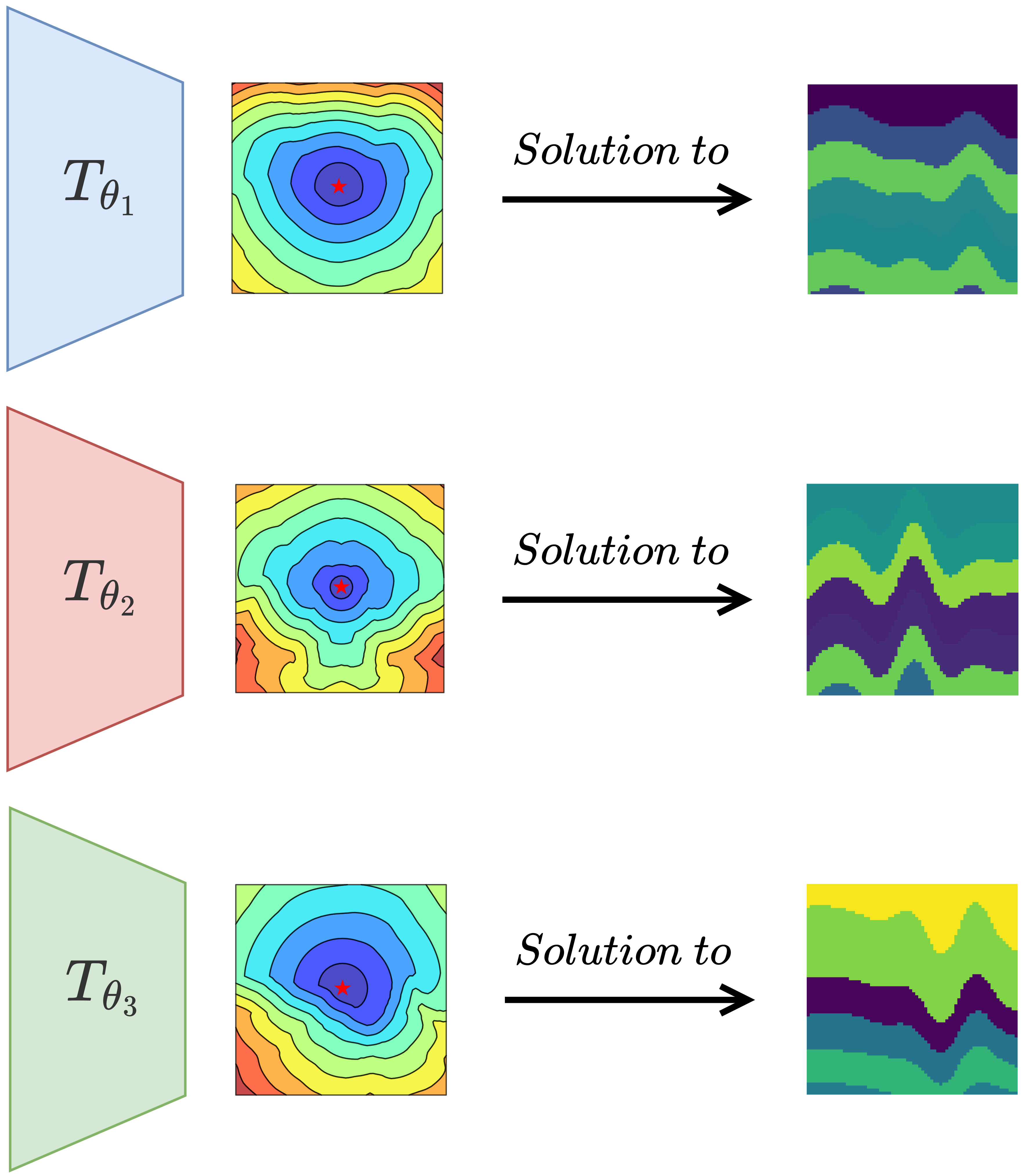


FMM Figure from: Sofian Toujja, Thierry Bay, Hakim Belhaouari, Laurent Fuchs. Topological Data Structure: The Fast Marching Example.

Limitations of current eikonal solvers

- <u>Classical:</u> **Fast Marching Method (FMM)**
  - Bad memory and computation scalability
  - Requires discretization

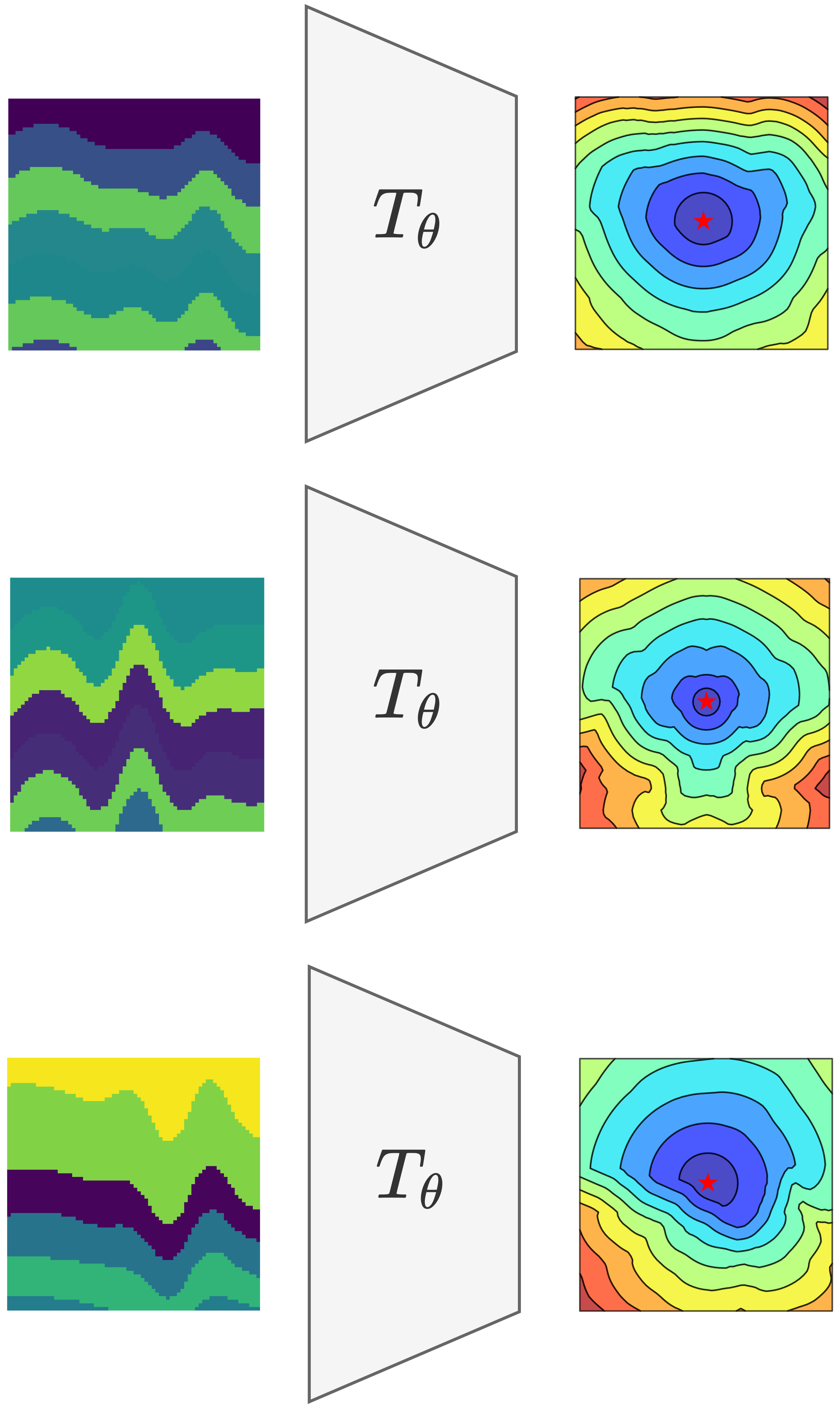
- <u>Neural solvers:</u>
  - **Physics-Informed Neural Networks (PINNs):** encode eikonal equation into loss function
    - Separate network for each solution → Memory inefficient
    - Not sharing knowledge between solutions → Training inefficient



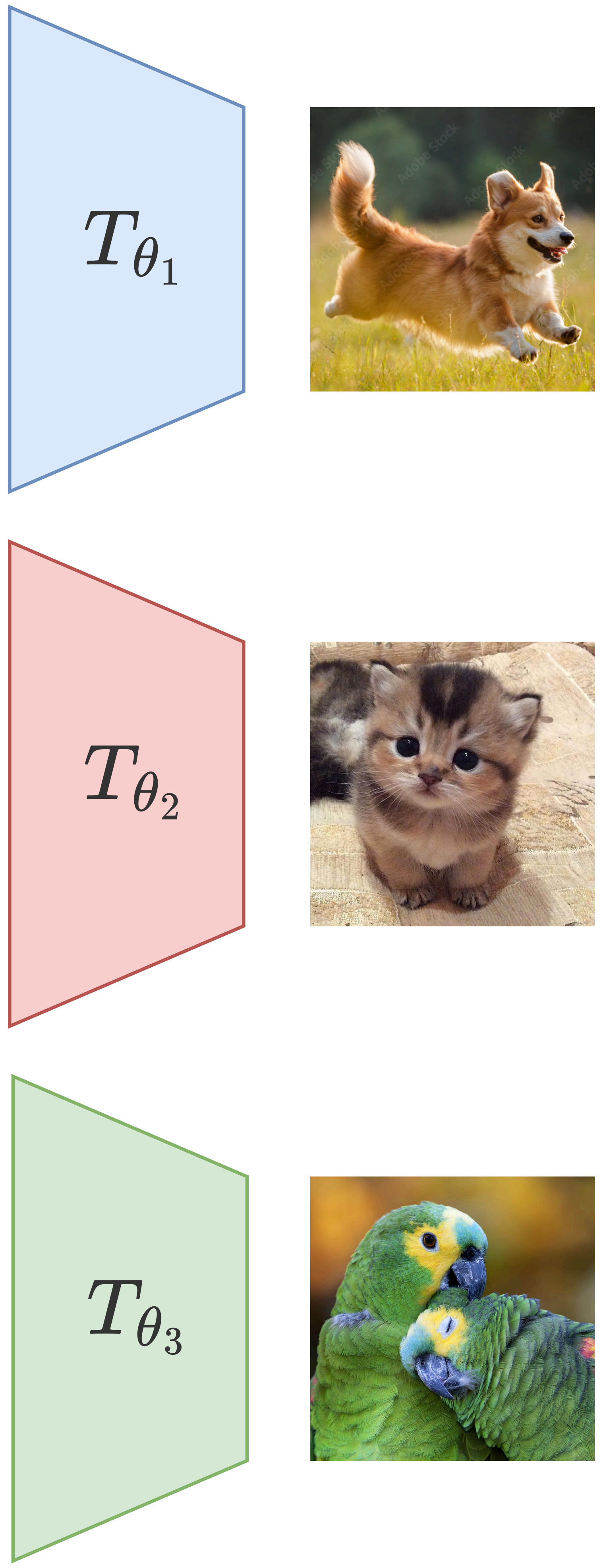
Limitations of current eikonal solvers

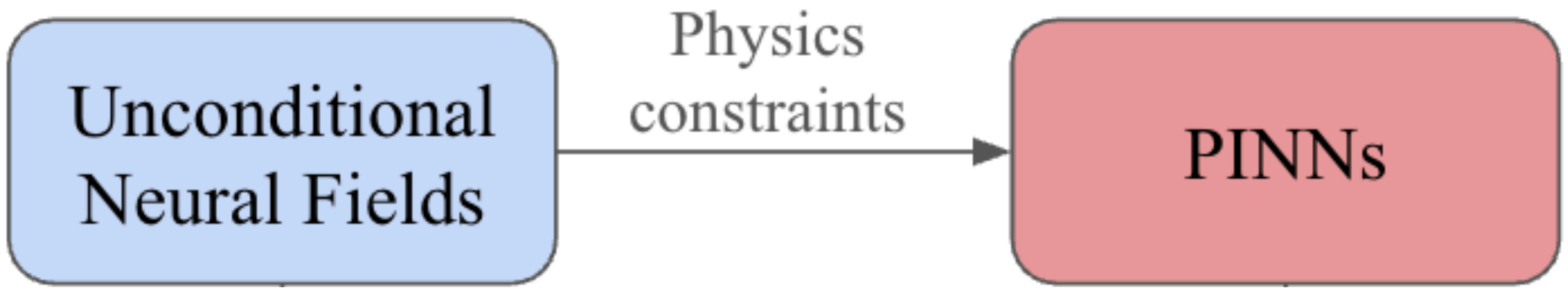
- Classical: **Fast Marching Method (FMM)**
  - Bad memory and computation scalability
  - Requires discretization

- Neural solvers:
  - **Physics-Informed Neural Networks (PINNs):** encode eikonal equation into loss function
    - Separate network for each solution → Memory inefficient
    - Not sharing knowledge between solutions → Training inefficient
  - **Neural Operators:** share a common backbone across solutions and condition on velocities profiles
    - Naive conditioning variables → Inefficient adaptability
    - Current approaches are not fully grid free



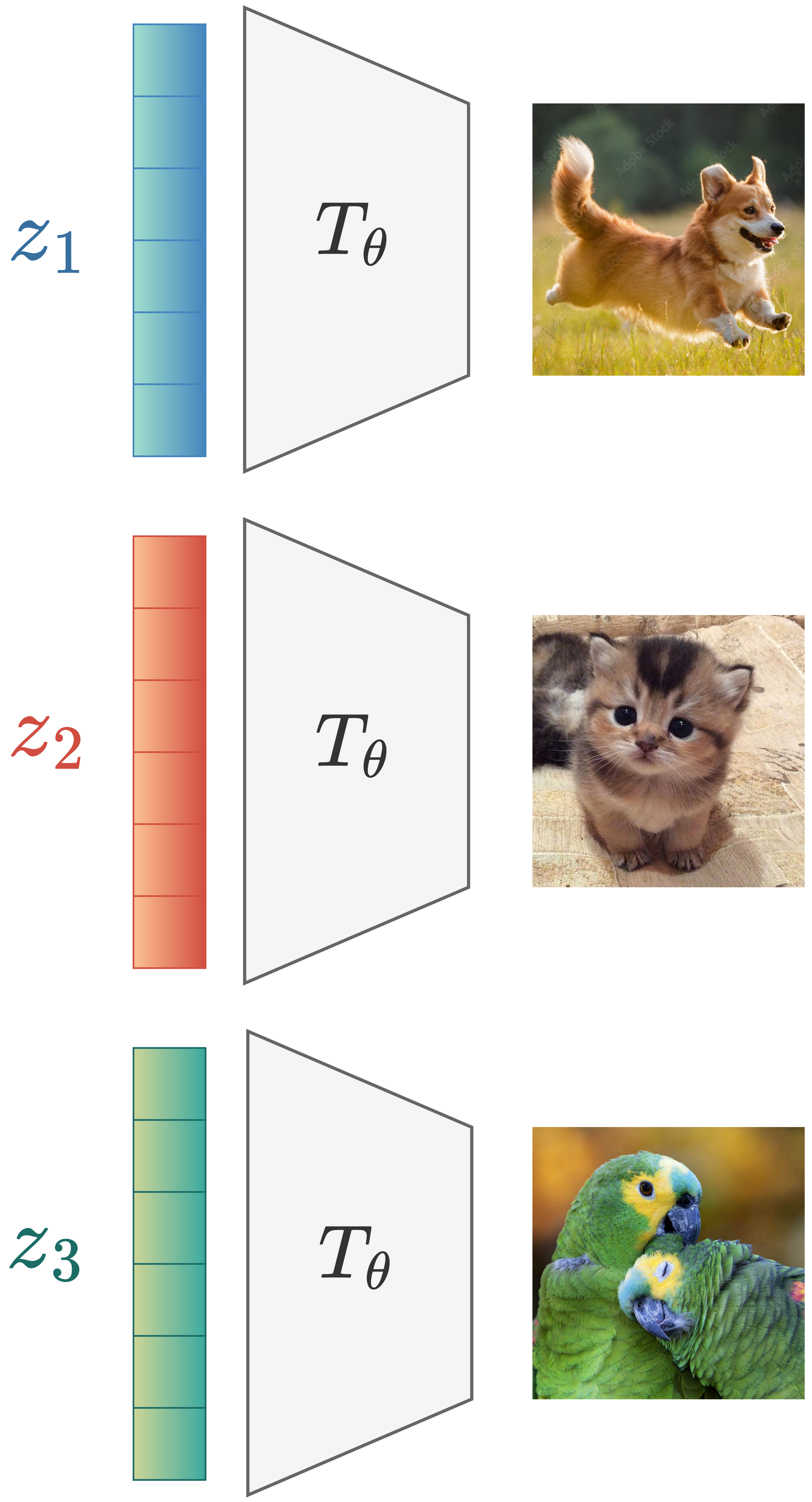
Conditional Neural Fields perspective

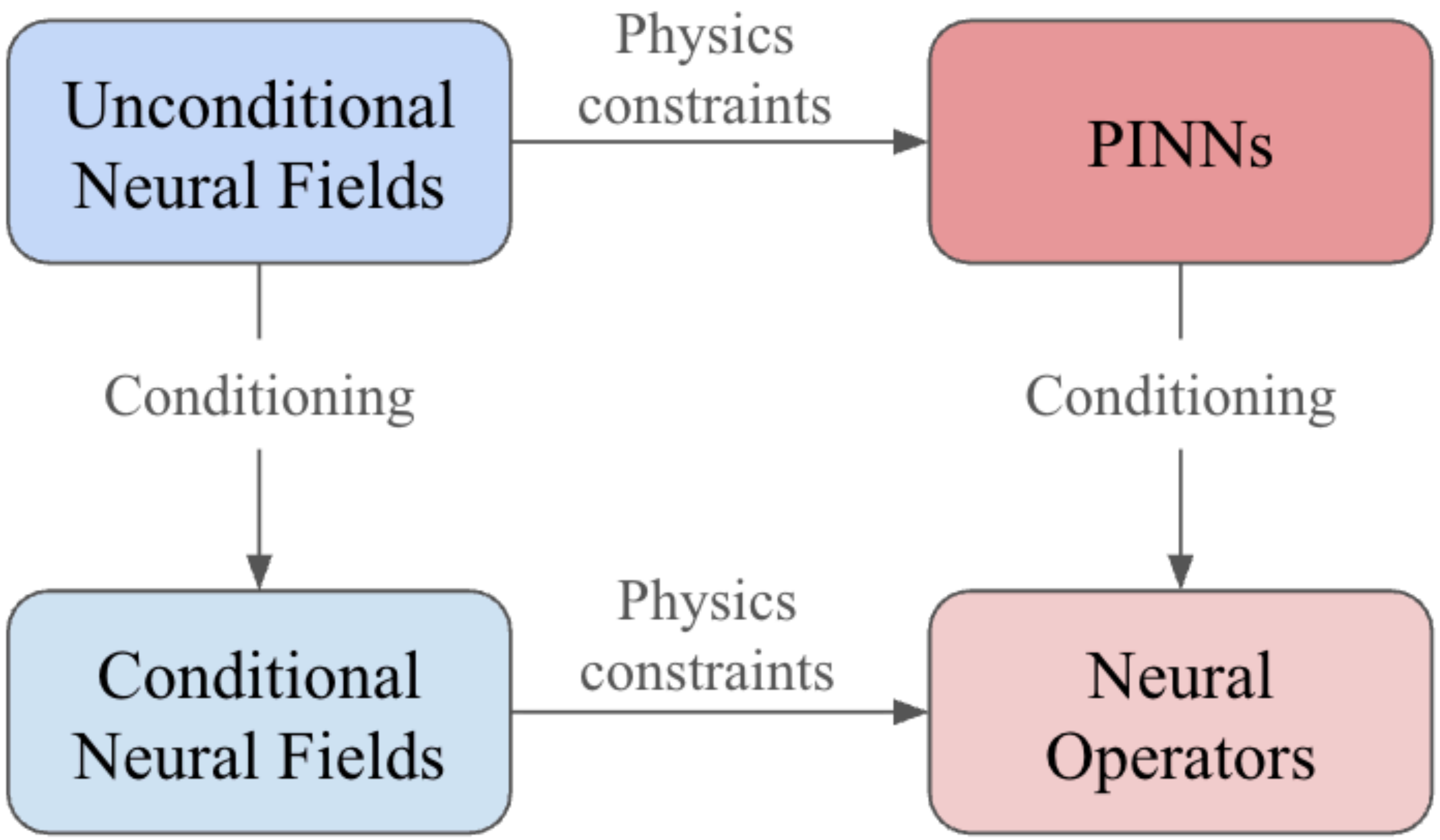




Sifan Wang, Jacob H. Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. Bridging Operator Learning and Conditioned Neural Fields: A Unifying Perspective, 2024.

Conditional Neural Fields perspective





Sifan Wang, Jacob H. Seidman, Shyam Sankaran, Hanwen Wang, George J. Pappas, and Paris Perdikaris. Bridging Operator Learning and Conditioned Neural Fields: A Unifying Perspective, 2024.

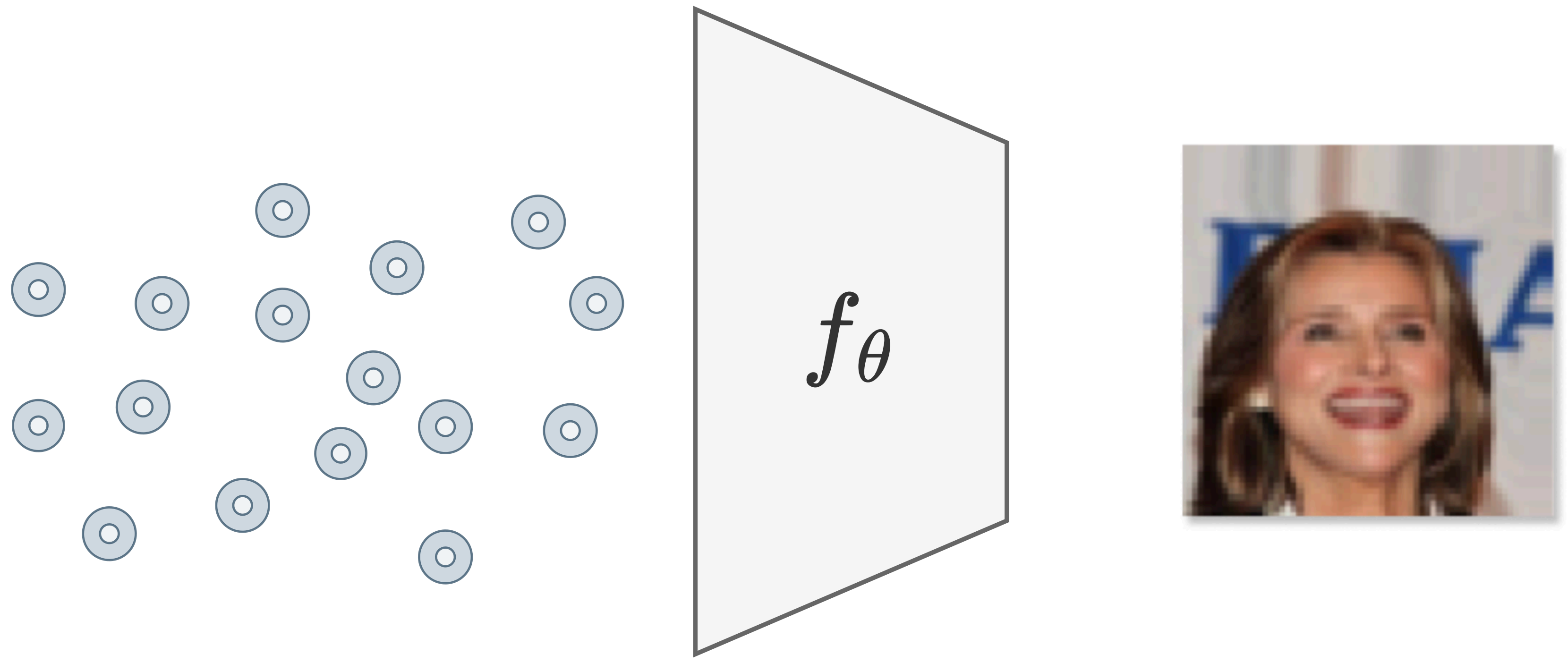
Research question

Which are the benefits of framing neural eikonal solvers as conditional neural fields?

From Conditional to *Equivariant Neural Fields*

Given a dataset $\mathcal{D} = \{f_i\}_{i=1}^{n}$ of continuous signals $f_i : \mathcal{M} \rightarrow \mathbb{R}^d$, each signal f_l can be associated with a latent code z_l such that a single network $f_\theta : \mathcal{M} \times \mathcal{Z} \rightarrow \mathbb{R}^d$, can represent the entire dataset: $f_\theta(p; z_l) \approx f_l(p)$, for all $f_l \in \mathcal{D}$

Conditioning via a learnable point cloud $\{z_i\}_{i=1}^{m} \subseteq \mathcal{Z}$ enhances expressivity and reconstruction fidelity



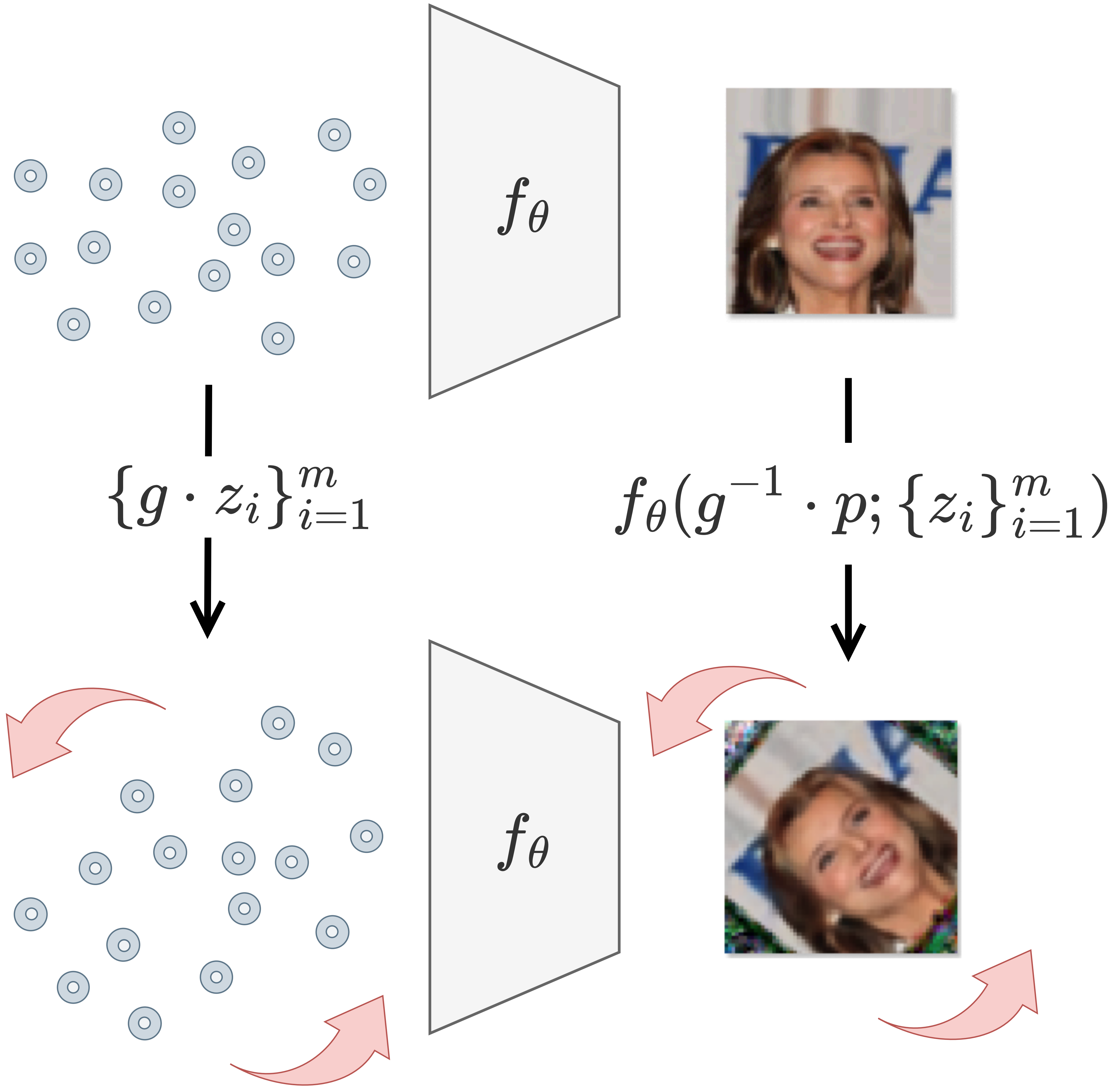
From Conditional to *Equivariant Neural Fields*

Given a dataset $\mathscr{D} = \{f_i\}_{i=1}^{n}$ of continuous signals $f_i : \mathscr{M} \to \mathbb{R}^d$, each signal f_l can be associated with a latent code z_l such that a single network $f_\theta : \mathscr{M} \times \mathscr{Z} \to \mathbb{R}^d$, can represent the entire dataset: $f_\theta(p; z_l) \approx f_l(p)$, for all $f_l \in \mathscr{D}$

Conditioning via a learnable point cloud $\{z_i\}_{i=1}^{m} \subseteq \mathscr{Z}$ enhances expressivity and reconstruction fidelity

Equivariant Neural Fields encode the **Steerability property** under a group G:

$$\boxed{f_\theta(g^{-1} \cdot p; \{z_i\}_{i=1}^{m}) = f_\theta(p; \{g \cdot z_i\}_{i=1}^{m}), \text{ for all } g \in G}$$



More on Equivariant Neural Fields

- The steerability of Equivariant Neural Fields can be achieved if and only if the function is invariant with respect to transformations of both input and latent variables:

$$\boxed{f(g \cdot p; \{g \cdot z_i\}_{i=1}^{m}) = f(p; \{z_i\}_{I=1}^{m}) \quad \forall g \in G}$$

More on Equivariant Neural Fields

- The steerability of Equivariant Neural Fields can be achieved if and only if the function is invariant with respect to transformations of both input and latent variables:

$$\boxed{f(g \cdot p; \{g \cdot z_i\}_{i=1}^{m}) = f(p; \{z_i\}_{I=1}^{m}) \quad \forall g \in G}$$

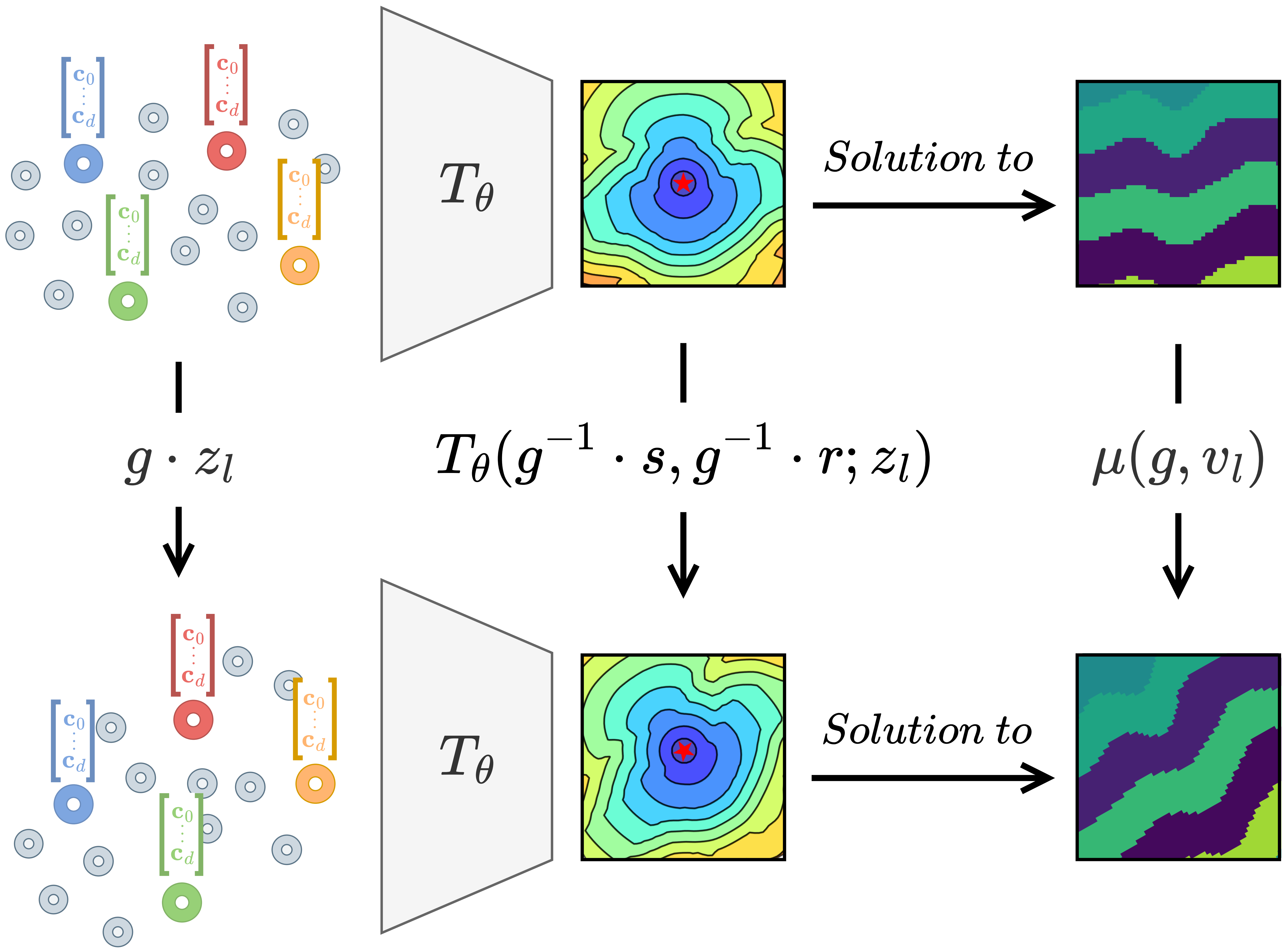
- We introduce a conditioning variable, represented as a geometric point cloud $z = \{(g_i, \mathbf{c}_i)\}_{i=1}^{N}$
  - $g_i \in G$ is referred to as a *pose*
  - $\mathbf{c}_i \in \mathbb{R}^d$ is referred to as a *context vector*

- We will denote the space of pose-context pairs as the product manifold $\mathcal{Z} = G \times \mathbb{R}^d$, so that z is an element of the power set $\mathcal{P}(\mathcal{Z})$

- This representation naturally supports a G-group action defined by $g \cdot z = \{(g \cdot g_i, \mathbf{c}_i)\}_{I=1}^{N}$

Method

Steerability on eikonal solvers



Steerability on eikonal solvers

Proposition 4.1 (Steered Eikonal Solution). *Let $T_\theta : \mathcal{M} \times \mathcal{M} \times \mathscr{P}(\mathcal{Z}) \to \mathbb{R}_+$ be a conditional neural field satisfying the steerability property (2), and let z_l be the conditioning variable representing the solution of the eikonal equation for $v_l : \mathcal{M} \to \mathbb{R}^*_+$, i.e., $T_\theta(s, r; z_l) \approx T_l(s, r)$ for T_l satisfying Equation (1) for the velocity field v_l. Let $\mathcal{G}^g$ be a g-steered metric (Definition 4.1). Then:*

1. *The map $\mu : G \times (\mathcal{M} \to \mathbb{R}^*_+) \to (\mathcal{M} \to \mathbb{R}^*_+)$ defined by*

$$\mu(g, v_l)(s) := \left\| \mathrm{grad}_{g^{-1}s}\, T_l(g^{-1} \cdot s,\ g^{-1} \cdot r) \right\|^{-1}_{\mathcal{G}^g}, \tag{3}$$

where r is an arbitrary point in $\mathcal{M}$, ***is a well-defined group action****.*

2. *For any $g \in G$, $T_\theta(s, r; g \cdot z_l)$* ***solves the eikonal equation*** *with velocity field $\mu(g, v_l)$.*

Definition 4.1 (g-steered metric). *For all $g \in G$, define the g-steered metric $\mathcal{G}^g : T\mathcal{M} \times T\mathcal{M} \to \mathbb{R}$ as:*

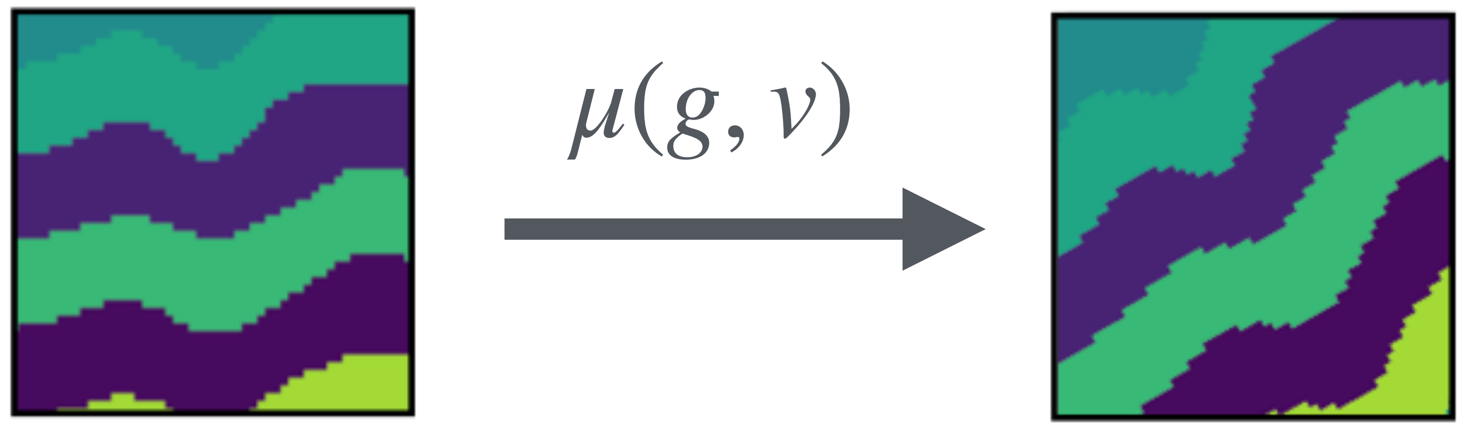
$$\mathcal{G}^g_p(\dot{u}, \dot{v}) := \mathcal{G}_{gp}\left((\mathrm{d}L_{g^{-1}}(g \cdot p))^*[\dot{u}], (\mathrm{d}L_{g^{-1}}(g \cdot p))^*[\dot{v}] \right) \quad \textit{for } p \in \mathcal{M}, \textit{ and } \dot{u}, \dot{v} \in T_p\mathcal{M},$$

where $L_{g^{-1}} : \mathcal{M} \to \mathcal{M}$ is the diffeomorphism defined by $L_{g^{-1}}(p) = g^{-1} \cdot p$.

Steerability on eikonal solvers

Isometries

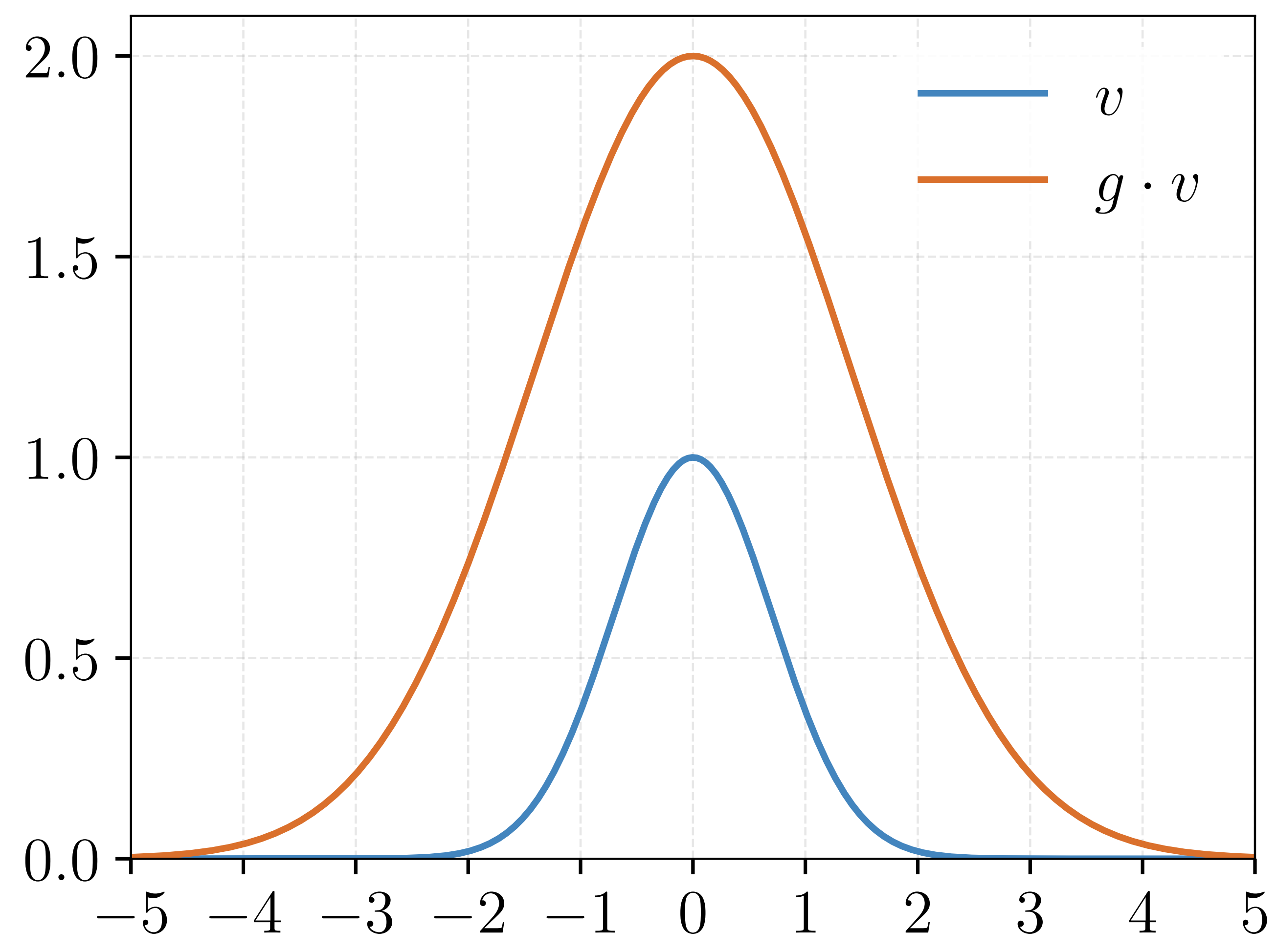
$$\boxed{\begin{array}{c}\mu(g, v_l)(s) = v_l(g^{-1} \cdot s) \text{ if } G \\ \text{acts isometrically on } \mathscr{M}\end{array}}$$



Conformal

$$\mathcal{G}_{gs}\left(\mathrm{d}L_g(s)[\dot{s}_1],\, \mathrm{d}L_g(s)[\dot{s}_2]\right) = \Omega(g,s)^2\, \mathcal{G}_s\left(\dot{s}_1, \dot{s}_2\right)$$
for all $\dot{s}_1, \dot{s}_2 \in T_s\mathscr{M}$

$$\boxed{\begin{array}{c}\mu(g, v_l)(s) = \Omega(g,s)\, v_l(g^{-1} \cdot s) \text{ if } G \\ \text{acts conformally on } \mathscr{M} \text{ with} \\ \text{conformal factor } \Omega(g,s) > 0\end{array}}$$



Equivariant Neural Eikonal Solver (E-NES)

1. **Factored eikonal equation:** $T_\theta(s, r; z) = \tilde{d}(s, r)\, \tau_\theta(s, r; z)$
   - $\tilde{d}(s, r)$ is an approximation of the Riemannian distance
   - Avoids irregular behavior as $r \to s$

Equivariant Neural Eikonal Solver (E-NES)

1. **Factored eikonal equation:** $T_\theta(s, r; z) = \tilde{d}(s, r)\, \tau_\theta(s, r; z)$
   - $\tilde{d}(s, r)$ is an approximation of the Riemannian distance
   - Avoids irregular behavior as $r \to s$

2. Parametrized as $\tau_\theta = P \circ E$
   - $E : \mathscr{M} \times \mathscr{M} \times \mathscr{P}(\mathscr{Z}) \to \mathbb{R}^L$ adapts the **invariant cross-attention encoder** of *Wessels et al. 2024*
   - $P : \mathbb{R}^L \to \mathbb{R}_+$ is the **bounded projection head** from *Grubas et al. 2023*

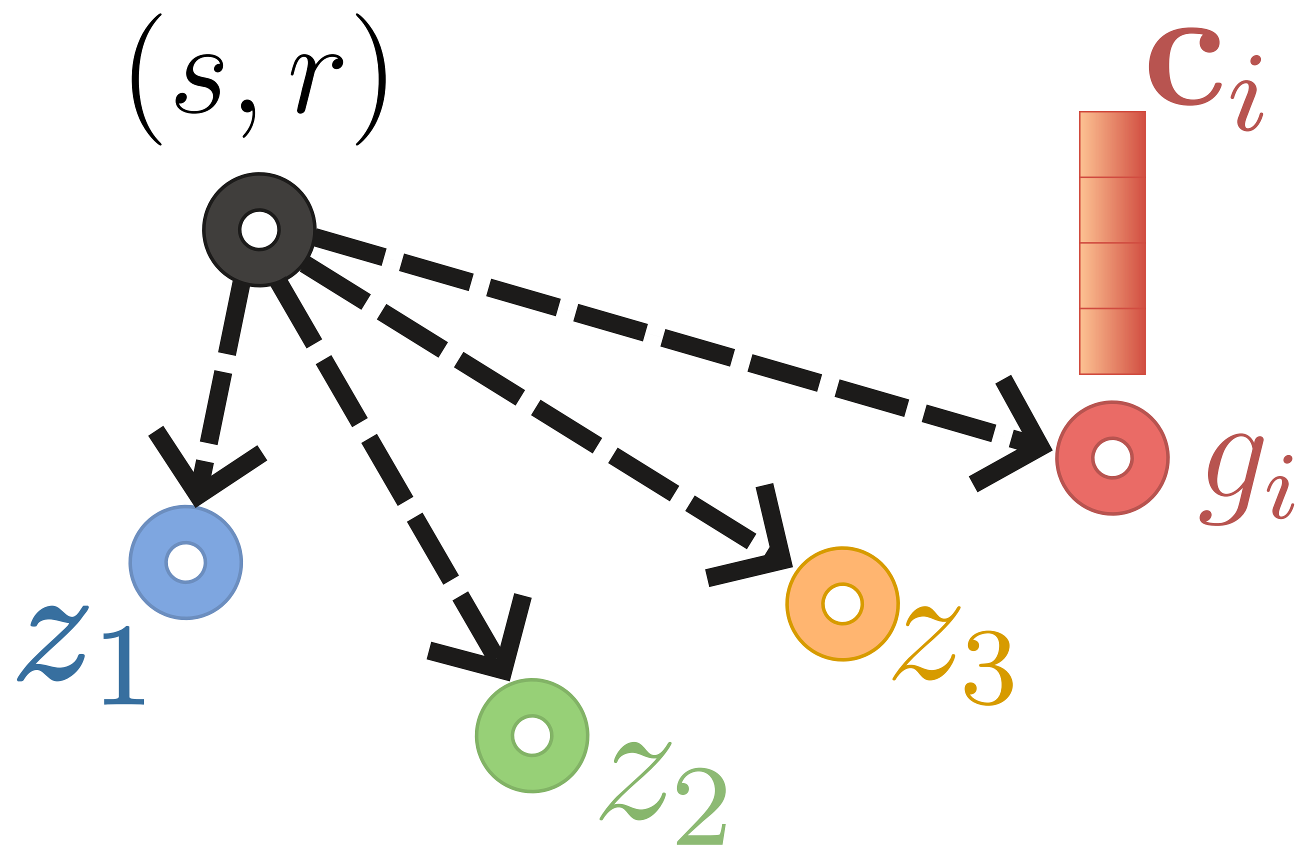
David R. Wessels, David M. Knigge, Samuele Papa, Riccardo Valperga, Sharvaree Vadgama, Efstratios Gavves, and Erik J. Bekkers. Grounding Continuous Representations in Geometry: Equivariant Neural Fields, 2024

Serafim Grubas, Anton Duchkov, and Georgy Loginov. Neural Eikonal solver: Improving accuracy of physics-informed neural networks for solving eikonal equation in case of caustics. 2023

Invariant cross-attention encoder

$$E(s,r;z) = \mathrm{FFN}_E\left(\sum_{i=1}^{N} \alpha_i \, v(\tilde{\mathbf{a}}_i, \mathbf{c}_i)\right) \quad \text{with } \alpha_i = \frac{\exp(q(\tilde{\mathbf{a}}_i)^\top k(\mathbf{c}_i)/\sqrt{d_k})}{\sum_{j=1}^{N} \exp(q(\tilde{\mathbf{a}}_j)^\top k(\mathbf{c}_j)/\sqrt{d_k})},$$

$$q(\tilde{\mathbf{a}}) = W_q \tilde{\mathbf{a}}, \quad k(\mathbf{c}) = W_k \, \mathrm{LN}(W_c \mathbf{c}),$$
$$v(\tilde{\mathbf{a}}, \mathbf{c}) = \mathrm{FFN}_v(W_v \, \mathrm{LN}(W_c \mathbf{c}) \odot (1 + \mathrm{FFN}_\gamma(\tilde{\mathbf{a}})) + \mathrm{FFN}_\beta(\tilde{\mathbf{a}})),$$



To enforce the steerability, $\mathbf{a}_i^{(s,r)} = \mathrm{RFF}\big(\mathrm{Inv}(s,r,g_i)\big), \quad \mathbf{a}_i^{(r,s)} = \mathrm{RFF}\big(\mathrm{Inv}(r,s,g_i)\big),$

To enforce $\tau_\theta(s,r;z) = \tau_\theta(r,s;z)$, we use $\tilde{\mathbf{a}}_i = (\mathbf{a}_i^{(s,r)} + \mathbf{a}_i^{(r,s)})/2$

Computation of Fundamental Joint-Invariants

Let G be a Lie group acting smoothly and regularly (but not necessarily freely) on each Riemannian manifold $\mathscr{M}_i$ via $\delta_i : G \times \mathscr{M}_i \to \mathscr{M}_i$, for $i = 1,\ldots,m$, and hence diagonally on

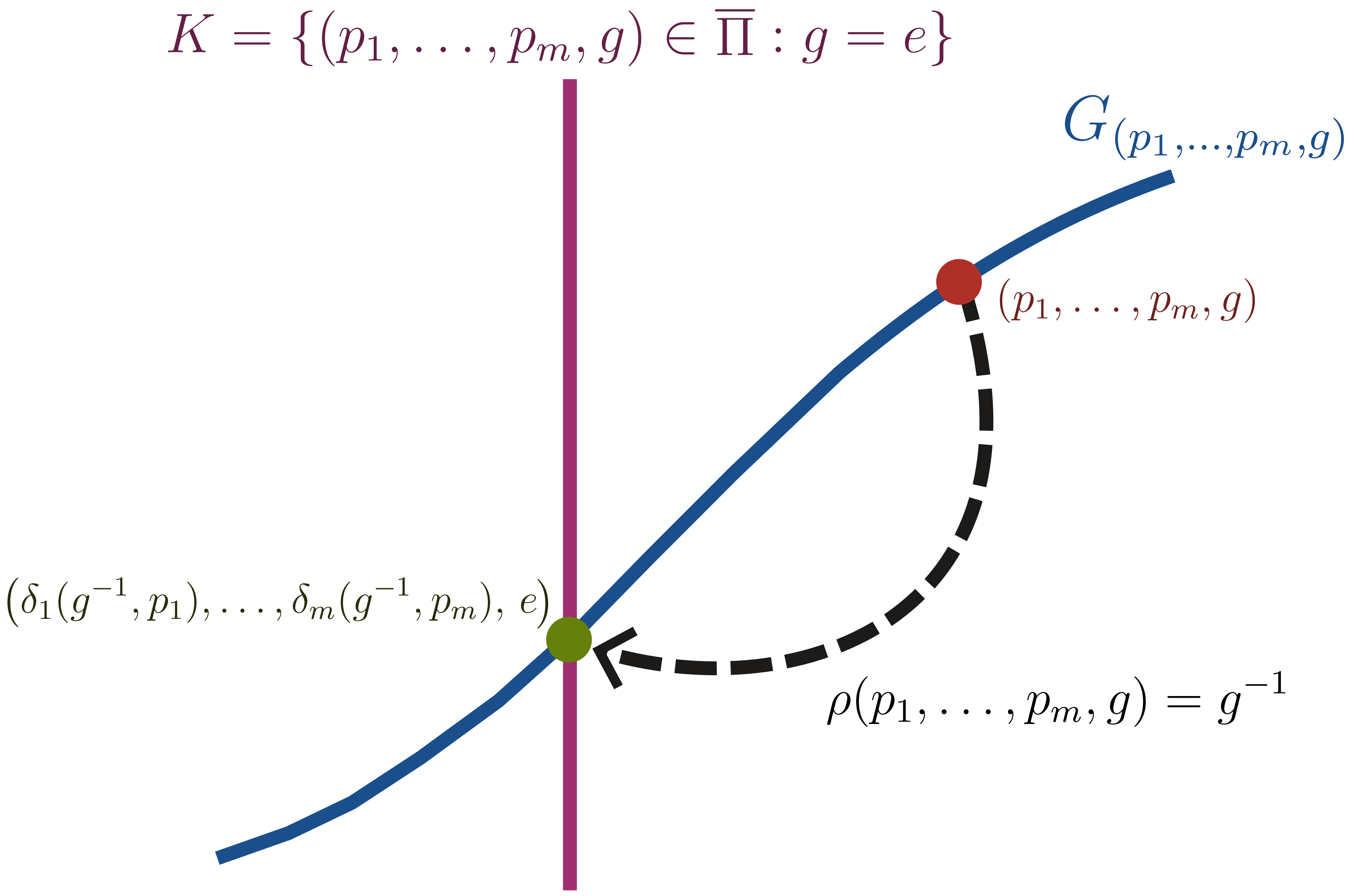
$$\Pi = \mathscr{M}_1 \times \cdots \times \mathscr{M}_m, \quad \delta\big(g,(p_1,\ldots,p_m)\big) = \big(\delta_1(g,p_1),\ldots,\delta_m(g,p_m)\big)\,.$$

On the augmented space $\overline{\Pi} = \Pi \times G$, define

$$\bar{\delta}\big(h,(p_1,\ldots,p_m,g)\big) = \big(\delta(h,(p_1,\ldots,p_m)),\, h\,g\big)\,.$$

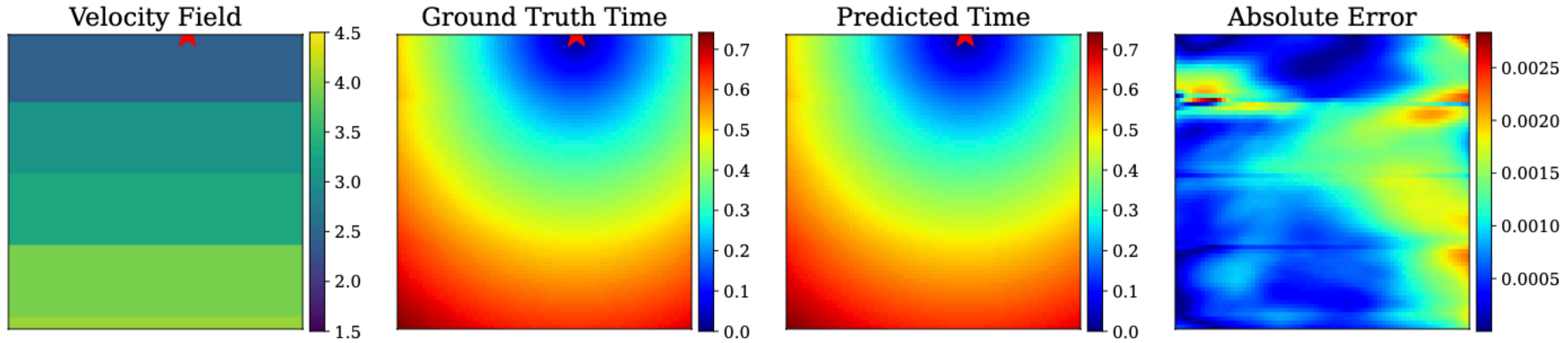
Then:

- $\bar{\delta}$ is free.
- A **moving frame** is given by $\rho : \overline{\Pi} \to G$, such that $\rho(p_1,\ldots,p_m,g) = g^{-1}$.
- **The set $\left\{\delta_i(g^{-1},p_i)\right\}_{i=1}^{m}$ forms a complete collection of functionally independent invariants of the action $\bar{\mu}$.**

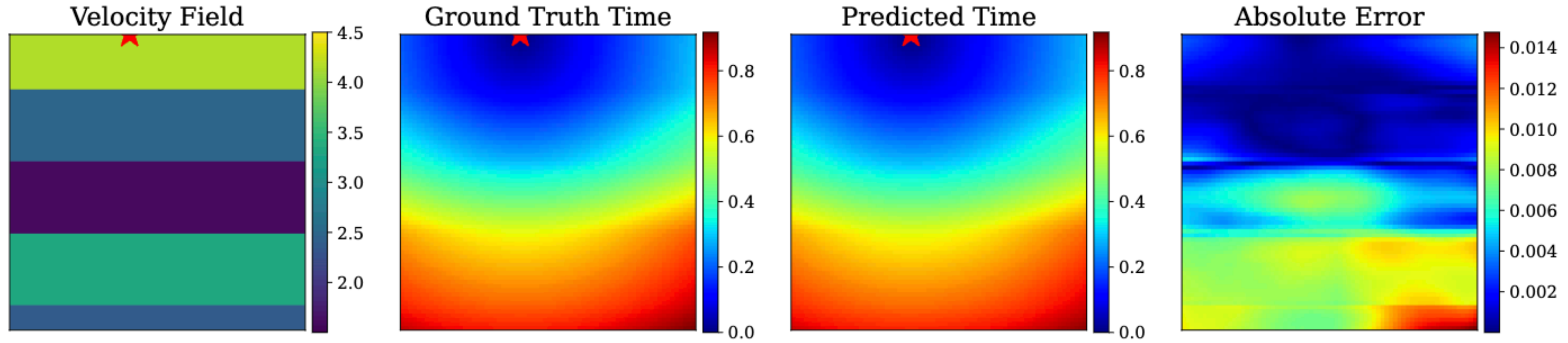


Results

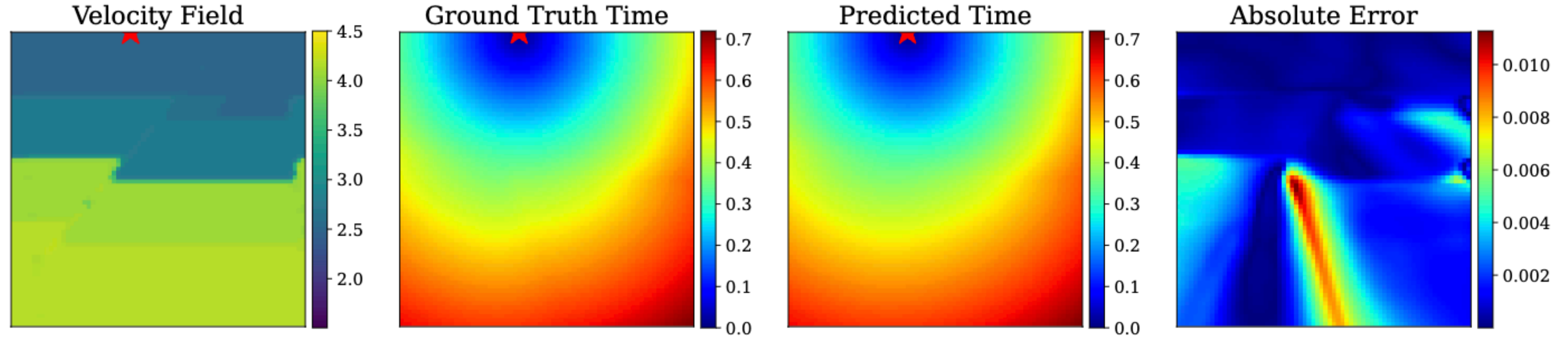
2D OpenFWI dataset: Empirical results



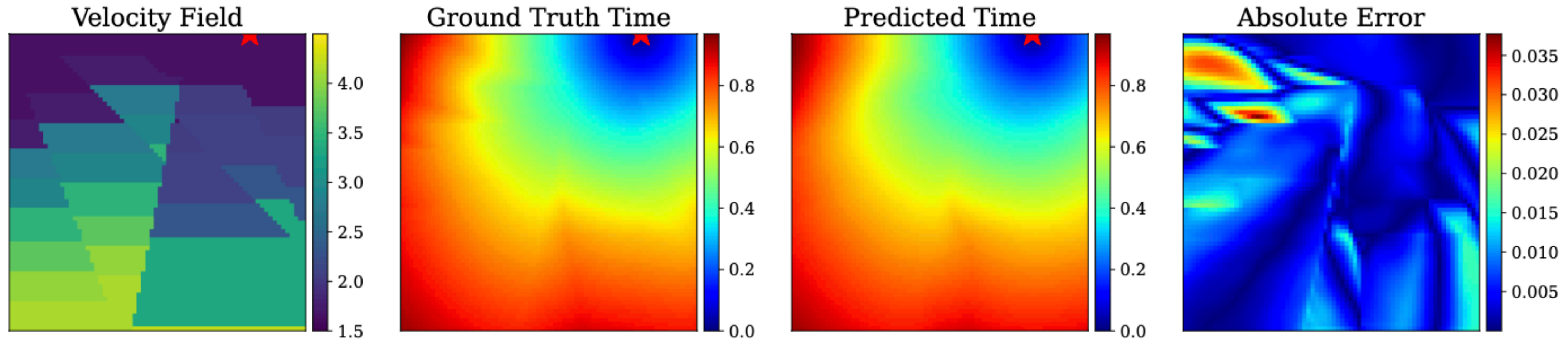
Results for FlatVel-A



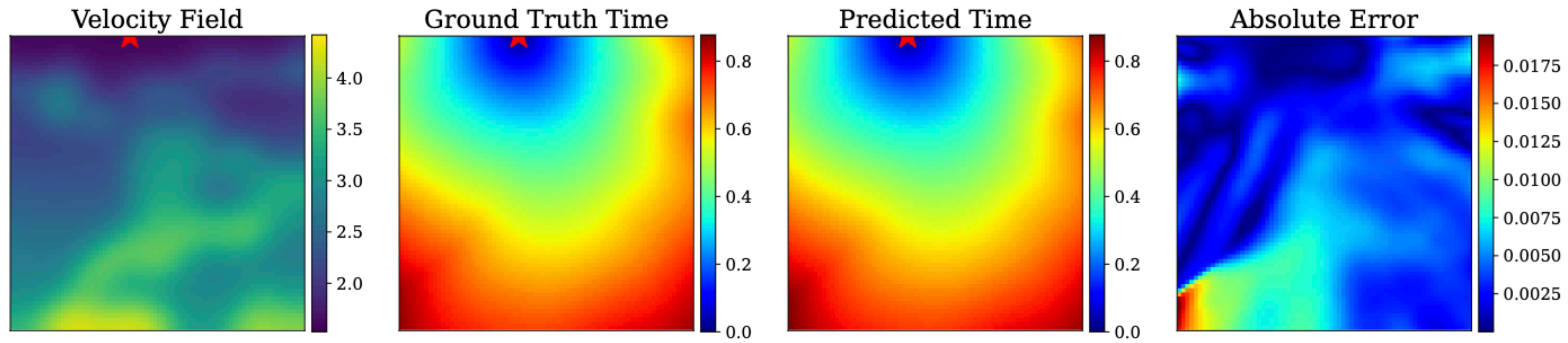
Results for FlatVel-B



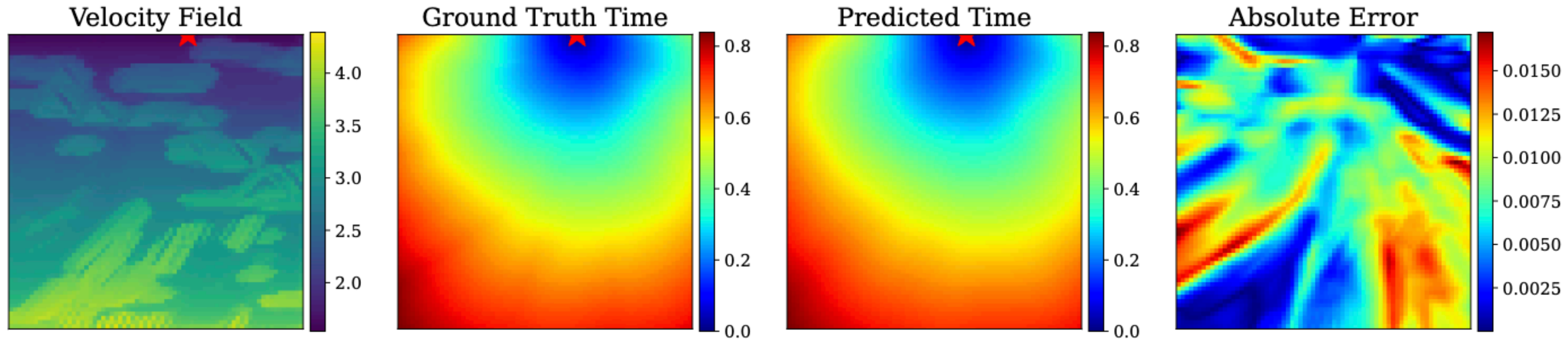
Results for FlatFault-A



Results for FlatFault-B



Results for Style-A



Results for Style-B

2D OpenFWI dataset: Comparison vs. FC-DeepONet

Table 1: Performance comparison on OpenFWI datasets. Colours denote **Best**, Second best, and Third best performing setups for each dataset.

			E-NES					
	FC-DeepONet		Autodecoding (100 epochs)		Autodecoding (convergence)		Meta-learning	
Dataset	RE (↓)	Fitting (s)	RE (↓)	Fitting (s)	RE (↓)	Fitting (s)	RE (↓)	Fitting (s)
FlatVel-A	**0.00277**	-	0.00952	223.31	0.00506	1120.25	0.01065	5.92
CurveVel-A	0.01878	-	0.01348	222.72	**0.00955**	1009.67	0.02196	5.91
FlatFault-A	**0.00514**	-	0.00857	222.61	0.00568	1014.45	0.01372	5.92
CurveFault-A	0.00963	-	0.01108	222.89	**0.00820**	1123.90	0.02086	5.92
Style-A	0,03461	-	0.01034	222.00	**0.00833**	1117.99	0.01317	5.92
FlatVel-B	**0.00711**	-	0.01581	222.74	0.00860	1010.32	0.02274	5.91
CurveVel-B	0.03410	-	0.03203	222.97	**0.02250**	1127.87	0.03583	5.90
FlatFault-B	0.04459	-	0.01989	222.70	**0.01568**	1133.98	0.03058	5.93
CurveFault-B	0.07863	-	0.02183	222.89	**0.01885**	893.84	0.03812	5.89
Style-B	0.03463	-	0.01171	221.90	**0.01069**	896.06	0.01541	5.90

Yifan Mei, Yijie Zhang, Xueyu Zhu, Rongxi Gou, and Jinghuai Gao. Fully Convolutional Network- Enhanced DeepONet-Based Surrogate of Predicting the Travel-Time Fields. 2024.

Ablation: Effect of equivariance

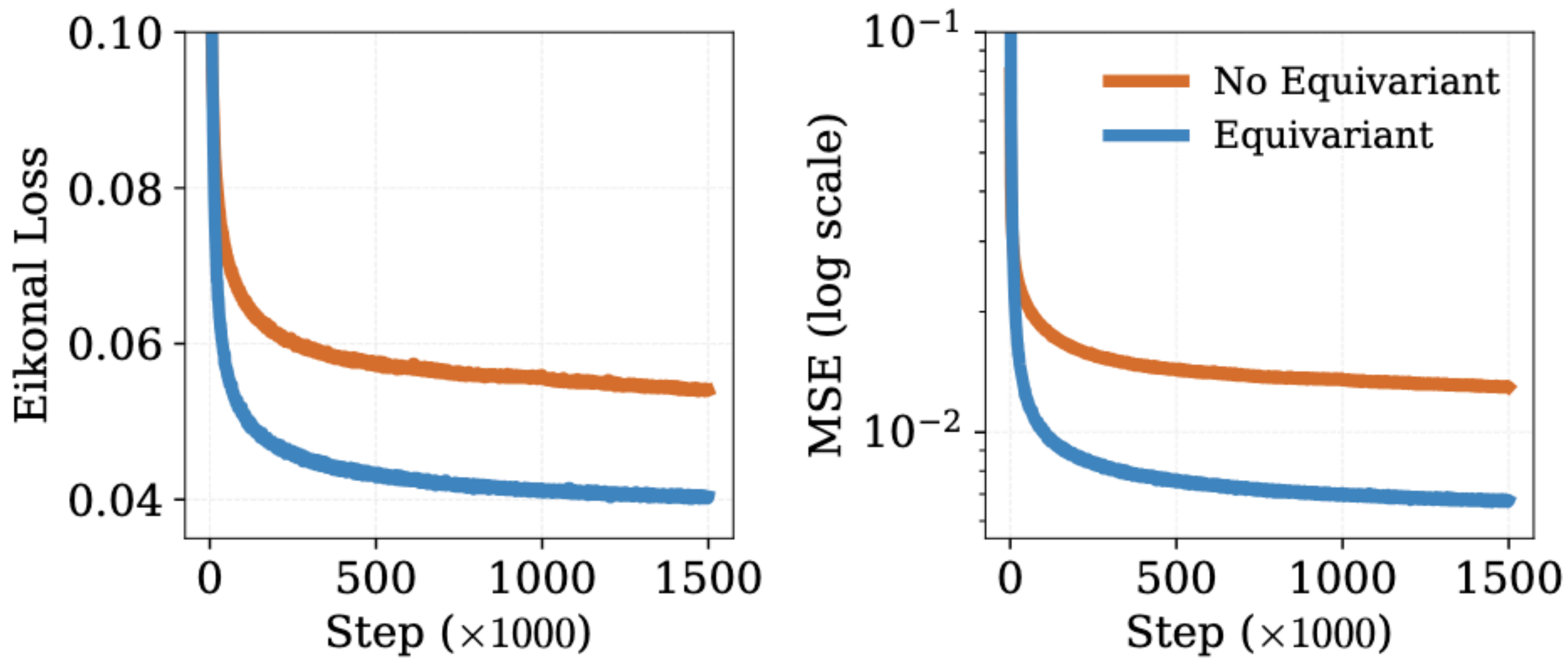
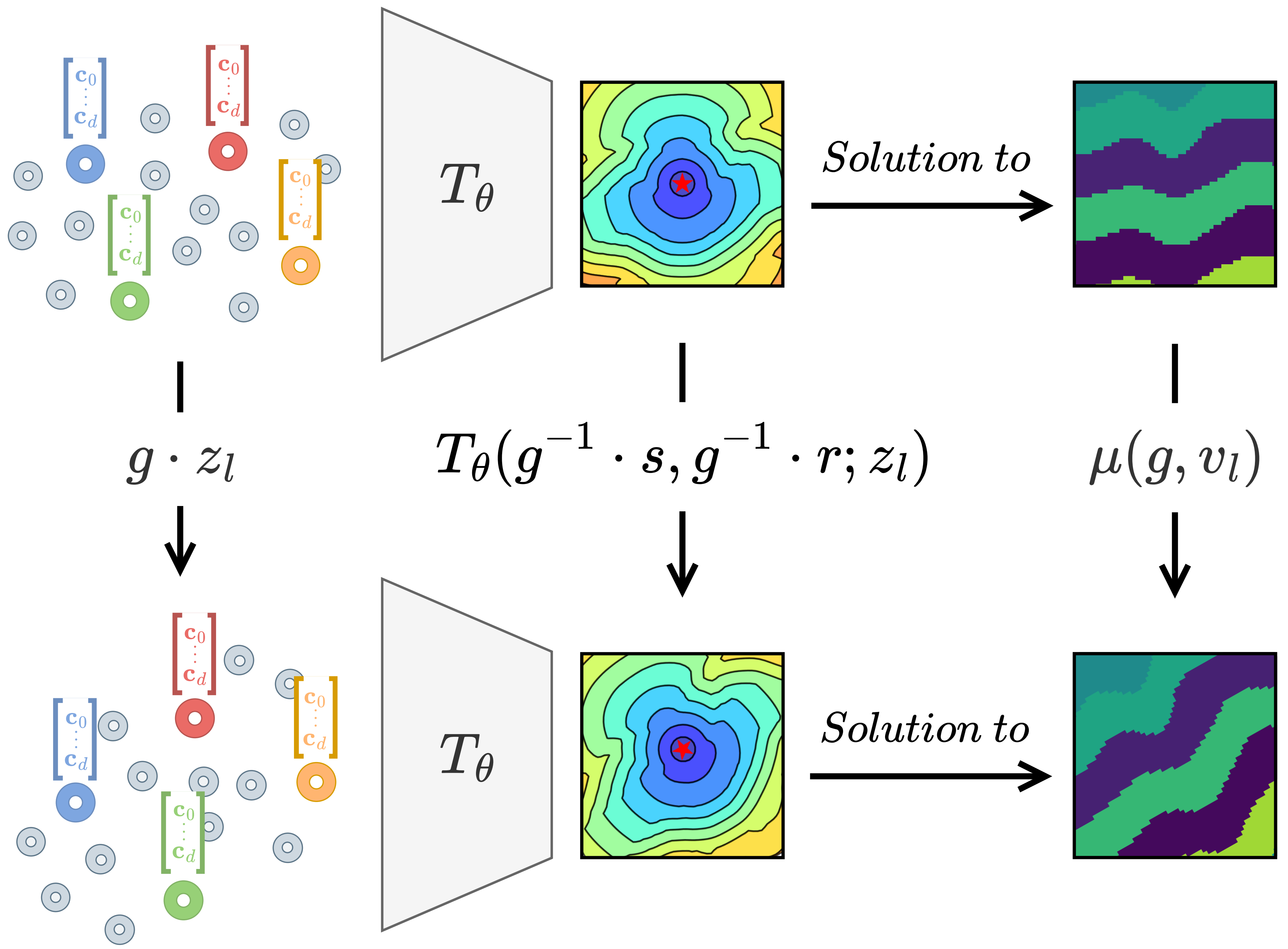


Figure 2: Comparative analysis of equivariant conditioning variables on the Style-B dataset. For non-equivariant models $\mathcal{Z} \cong \mathbb{R}^c$, while equivariant models use $\mathcal{Z} = SE(2) \times \mathbb{R}^c$.

Conclusions

- **Novel, expressive generalization of Equivariant Neural Fields** to functions defined over products of Riemannian manifolds with regular group actions.
- **Equivariant Neural Eikonal Solver** (E-NES), efficiently solve eikonal equations by leveraging geometric symmetries, **enabling generalization across group transformations without explicit data augmentation.**
- We validate our approach through comprehensive experiments on **2D and 3D seismic travel-time benchmarks**,
  - Improved **scalability**,
  - Improved **adaptability**
  - Improved user **controllability**



Thank You

Q&A

Appendix

Bounded projection head

2. Bounded Velocity Projection. The encoder output $\mathbf{h} = E(s, r; z)$ passes through a second MLP network FFN_P with AdaptiveGauss activations to model sharp wavefronts and caustics [Grubas et al., 2023]. The final output is projected into $[1/v_{\text{max}}, 1/v_{\text{min}}]$ by:

$$P(\mathbf{h}) = \left(\frac{1}{v_{\text{min}}} - \frac{1}{v_{\text{max}}}\right) \sigma(\alpha_0 \cdot \text{FFN}_P(\mathbf{h})) + \frac{1}{v_{\text{max}}},$$

where σ is the sigmoid function and $\alpha_0 \in \mathbb{R}_+$ is a learnable temperature parameter.

Serafim Grubas, Anton Duchkov, and Georgy Loginov. Neural Eikonal solver: Improving accuracy of physics-informed neural networks for solving eikonal equation in case of caustics. 2023

Training loss

Let $\mathcal{V} = \{v_l : \mathcal{M} \to [v_{\min}, v_{\max}]\}_{l=1}^{K}$ be our training set of K velocity fields over the domain $\mathcal{M}$. At each iteration, we sample a batch $\mathcal{B}$ with B velocity fields $\{v_i\}_{i=1}^{B} \subseteq \mathcal{V}$ and N_{sr} source–receiver pairs $\{(s_{i,j}, r_{i,j})\}_{j=1}^{N_{sr}} \subset \mathcal{M}^2$ for each v_i. Let $\{z_i\}_{i=1}^{B}$ be the conditioning variables associated with $\{v_i\}_{i=1}^{B}$, then we minimize a physics-informed loss that enforces the Hamilton-Jacobi equation [Grubas et al., 2023]:

$$L(\theta, \{z_i\}_{i=1}^{B}, \mathcal{B}) = \frac{1}{B\,N_{sr}} \sum_{i=1}^{B} \sum_{j=1}^{N_{sr}} \Big(\big| v_i(s_{i,j})^2 \| \operatorname{grad}_s T_\theta(s_{i,j}, r_{i,j}; z_i) \|_{\mathcal{G}}^2 - 1 \big| + \big| v_i(r_{i,j})^2 \| \operatorname{grad}_r T_\theta(s_{i,j}, r_{i,j}; z_i) \|_{\mathcal{G}}^2 - 1 \big| \Big). \tag{5}$$

Autodecoding algorithm

Algorithm 1 Autodecoding Training

Require: Velocity fields $\mathcal{V} = \{v_l\}_{l=1}^{K}$, epochs num_epochs, batch size B, pairs per field N_{sr}, learning rate η

```
Randomly initialize shared base network T_θ
Initialize latents z_l ← {(g_i, c_i)}_{i=1}^N for all velocity fields
for epochs = 1 to num_epochs do
    while dataloader not empty do
        Sample batch 𝓑 = {(s_{i,j}, r_{i,j}, v_i(s_{i,j}), v_i(r_{i,j}))}_{i=1,j=1}^{B,N_sr}
        Compute loss L(θ, {z_i}_{i=1}^B, 𝓑) (see Equation 5)
        Update θ ← θ − η∇_θ L
        Update each z_i ← z_i − η∇_{z_i} L
    end while
end for
```

Ensure: Trained θ and latents $\{z_l\}_{l=1}^{K}$

Meta-learning algorithm

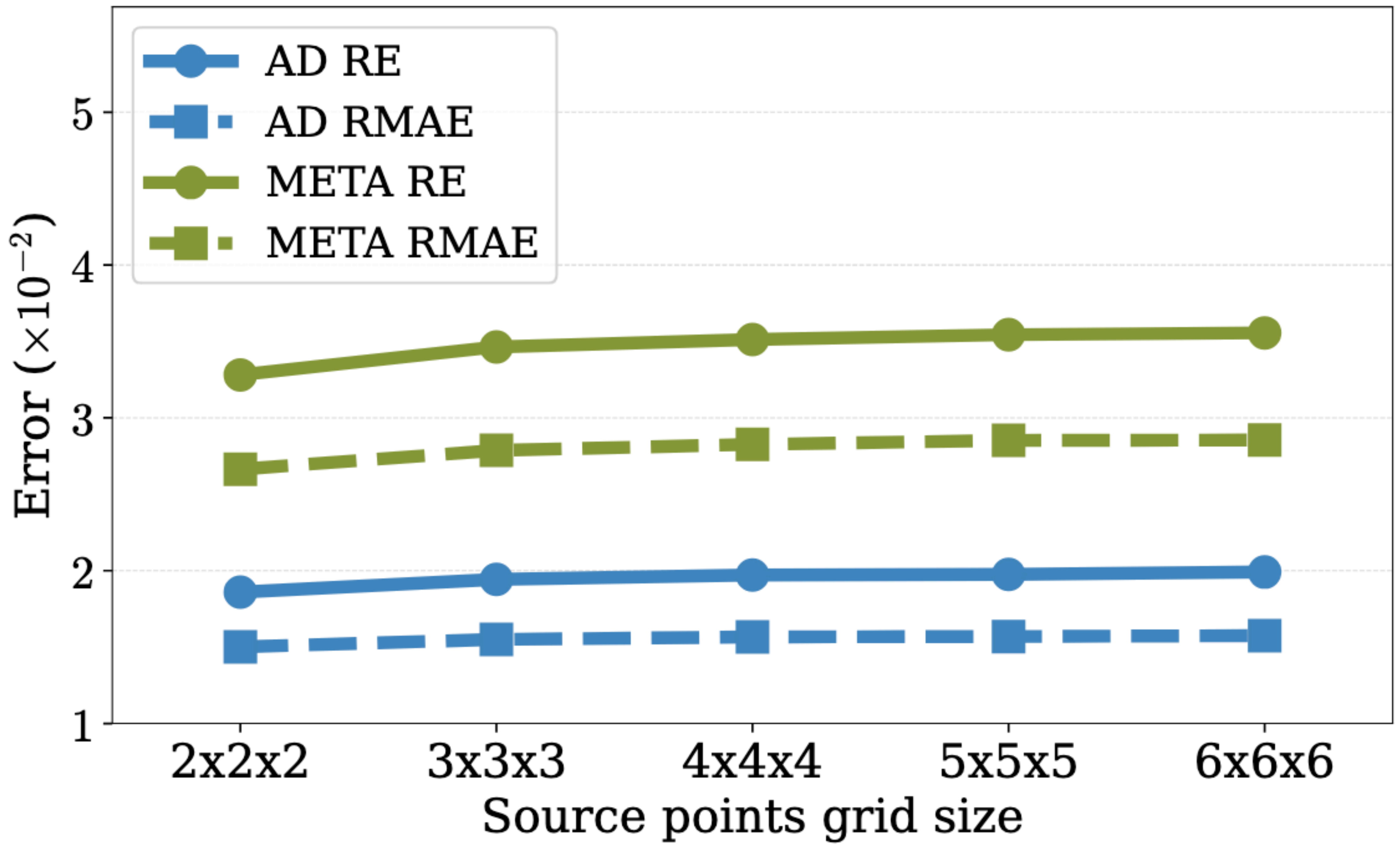
Algorithm 2 Meta-learning Training

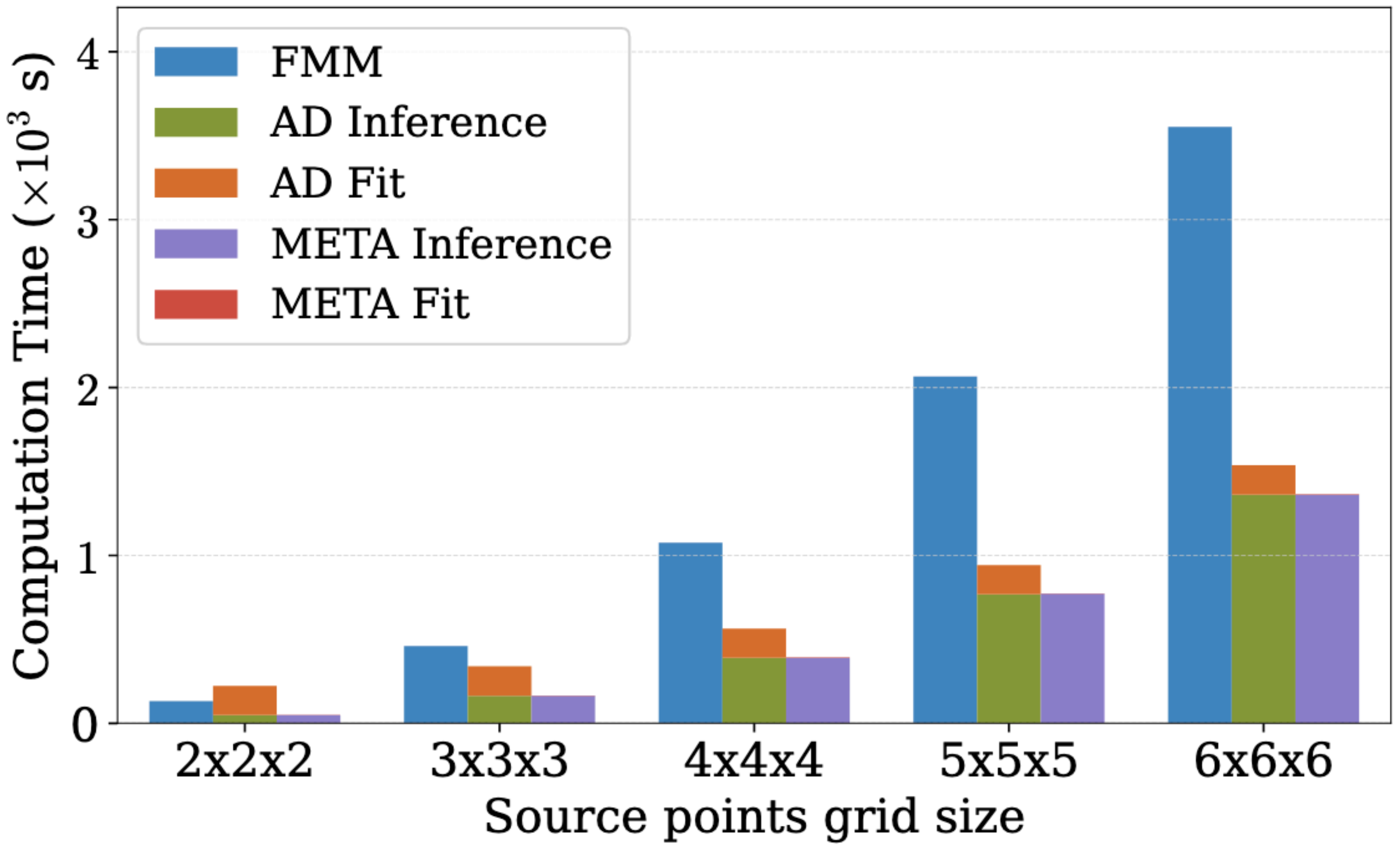
Require: Velocity fields $\mathcal{V} = \{v_l\}_{l=1}^{K}$, outer epochs num_epochs, inner steps S, batch size B, pairs per field N_{sr}, learning rates $\eta_\theta, \eta_{\text{SGD}}$

```
Initialize shared base network T_θ (optionally pretrained), and learnable learning rate η_z.
for epochs = 1 to num_epochs do
    while dataloader not empty do
        Sample batch of velocity fields {v_i}_{i=1}^B ⊆ V
        Initialize latents z_i^(0) for each v_i
        for t = 1 to S do                                          ▷ Inner loop: Update latents
            Sample N_sr source–receiver pairs {(s_{i,j}^(t-1), r_{i,j}^(t-1))}_{j=1}^{N_sr} ⊂ M^2, for each v_i
            Construct batch B^(t-1) = {(s_{i,j}^(t-1), r_{i,j}^(t-1), v_i(s_{i,j}^(t-1)), v_i(r_{i,j}^(t-1)))}_{i=1,j=1}^{B,N_sr}
            Compute L̃(θ, {z_i^(t-1)}_{i=1}^B, B^(t-1))
            Update each z_i^(t) ← z_i^(t-1) − η_z ∇_{z_i} L̃
        end for
        Sample N_sr source–receiver pairs {(s_{i,j}^(S), r_{i,j}^(S))}_{j=1}^{N_sr} ⊂ M^2, for each v_i
        Construct batch B^(S) = {(s_{i,j}^(S), r_{i,j}^(S), v_i(s_{i,j}^(S)), v_i(r_{i,j}^(S)))}_{i=1,j=1}^{B,N_sr}
        Compute L̃_meta(θ) = L̃(θ, {z_i^(S)}_{i=1}^B, B^(S))
        Update θ ← θ − η_θ, ∇_θ L̃_meta
        Update η_z ← η_z − η_SGD ∇_{η_z} L̃_meta
    end while
end for
```

Ensure: Trained θ

3D OpenFWI dataset: Scalability analysis





Ablation: Pretrained Meta-learning

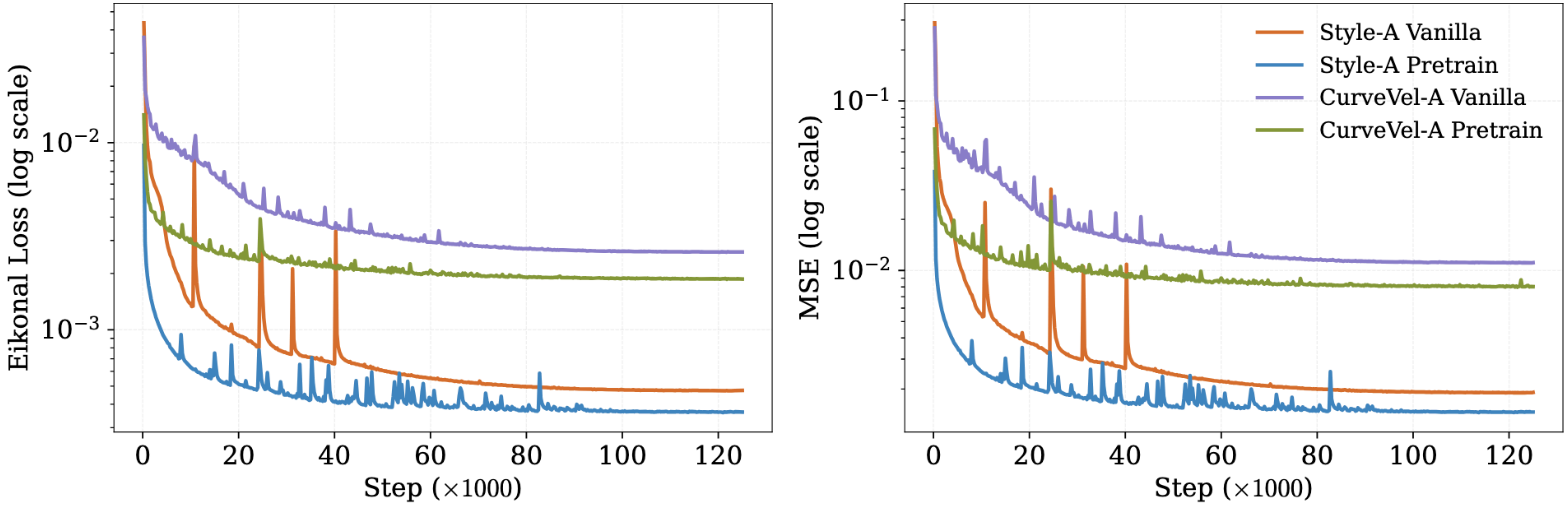


Figure 5: Comparative analysis of meta-learning convergence with pretrained versus random initialization on Style-A and CurveVel-A OpenFWI datasets.

Full-grid 2D OpenFWI results

Table 2: Performance on OpenFWI datasets on a 14×14 grid of source points.

	Autodecoding (100 epochs)			Autodecoding (convergence)			Meta-learning		
Dataset	RE (↓)	RMAE (↓)	Fitting (s)	RE (↓)	RMAE (↓)	Fitting (s)	RE (↓)	RMAE (↓)	Fitting (s)
FlatVel-A	0.01023	0.00827	223.31	0.00624	0.00509	1010.90	0.01304	0.01003	5.92
CurveVel-A	0.01438	0.01139	222.72	0.01069	0.00841	1009.67	0.02460	0.01878	5.91
FlatFault-A	0.01050	0.00751	222.61	0.00744	0.00510	1014.45	0.01749	0.01255	5.92
CurveFault-A	0.01380	0.00976	222.89	0.01088	0.00745	1007.97	0.02471	0.01807	5.92
Style-A	0.00962	0.00785	222.00	0.00795	0.00646	783.13	0.01326	0.01036	5.92
FlatVel-B	0.01988	0.01586	222.74	0.01178	0.00906	786.48	0.03077	0.02474	5.91
CurveVel-B	0.04291	0.03349	222.97	0.03297	0.02528	1010.70	0.04977	0.03930	5.90
FlatFault-B	0.01889	0.01413	222.70	0.01557	0.01147	898.28	0.02998	0.02214	5.93
CurveFault-B	0.02244	0.01728	222.89	0.01991	0.01537	561.22	0.03824	0.02945	5.89
Style-B	0.01061	0.00860	221.90	0.00984	0.00798	1120.09	0.01566	0.01227	5.90

Ablation: Autodecoding steps