A Implementation details

A.1 Aether

Here we present the full Aether architecture. We first describe the details of the neural field used for field discovery, and then we describe our graph network formulated as a variational autoencoder [21, 37].

A.1.1 Neural field

In its general form, the neural field takes as inputs query states $\mathbf{v}$ that comprise positions $\mathbf{p} \in \mathbb{R}^d$, orientations $\boldsymbol{\omega} \in \mathrm{SO}(d)$, and velocities $\mathbf{u} \in \mathbb{R}^d$, as well as a latent code $\mathbf{z} \in \mathbb{R}^{D_\mathrm{z}}$ used to condition the field, and predicts latent forces at the query states. Thus, it is defined as $\mathbf{f} : \mathbb{R}^d \times \mathrm{SO}(d) \times \mathbb{R}^d \times \mathbb{R}^{D_\mathrm{z}} \to \mathbb{R}^d$, where $d \in \{2, 3\}$. Depending on the task at hand, we can omit the latent code $\mathbf{z}$, *e.g.* if we are modelling a static field, or the orientations $\boldsymbol{\omega}$, if we have prior knowledge that the field is independent to them.

Encoding positions We encode the query positions using Gaussian random Fourier features [46], $\gamma(\mathbf{p}) = [\cos(2\pi\mathbf{B}\mathbf{p}), \sin(2\pi\mathbf{B}\mathbf{p})]^\top$, where $\mathbf{B} \in \mathbb{R}^{\frac{D_c}{2} \times d}$ is a matrix with entries sampled from a Gaussian distribution, $\mathbf{B}_{kl} \sim \mathcal{N}(0, \sigma^2)$. Throughout the experiments, and unless otherwise specified, we use a unit variance $\sigma^2 = 1$, and $\frac{D_c}{2} = 256$. Thus, the encoded positions have a dimension of 512.

Encoding orientations In 3 dimensions, for the orientations $\boldsymbol{\omega} = (\theta, \phi, \psi)^\top$, we follow [24], and use the angles of the velocity vectors as a proxy. We represent each angle in $\boldsymbol{\omega}$ as unit vector, $\hat{\boldsymbol{\omega}} = [\cos\boldsymbol{\omega}, \sin\boldsymbol{\omega}]^\top$, and encode them with a linear layer $\delta(\hat{\boldsymbol{\omega}}) = \mathbf{W}_\omega \hat{\boldsymbol{\omega}}$, where $\mathbf{W}_\omega \in \mathbb{R}^{D_c \times |\hat{\boldsymbol{\omega}}|}$. In 2 dimensions, we use the same encoding, except that we now have a single angle $\boldsymbol{\omega} = \theta$.

Encoding velocities For velocities, we simply encode them using a linear layer $\zeta(\mathbf{u}) = \mathbf{W}_u \mathbf{u}$, where $\mathbf{W}_u \in \mathbb{R}^{D_c \times d}$. Finally, we concatenate the encoded positions, orientations, and velocities in a single vector before we feed them as input to the neural field.

Latent code The latent code $\mathbf{z}$ "summarizes" the input graph such that it isolates global field effects. We employ a simple global spatio-temporal attention mechanism, similar to Li et al. [25], that aggregates the input system in a latent vector representation. First, we define object embeddings $\mathbf{o}_i = \mathrm{GRU}\big(\mathbf{W}_g \mathbf{x}_i^{1:T}\big)$, where $\mathbf{W}_g \in \mathbb{R}^{D_\mathrm{o} \times d}$ is a matrix used to linearly transform the inputs, and GRU is the Gated Recurrent Unit [7]. We also define temporal embeddings $\mathbf{t} = \mathrm{PE}(t)$, where PE are positional encodings [49], defined as:

$$\mathrm{PE}(t)_{2i} = \sin\Big(t/10000^{2i/D_\mathrm{s}}\Big), \tag{11}$$

$$\mathrm{PE}(t)_{2i+1} = \cos\Big(t/10000^{2i/D_\mathrm{s}}\Big), \tag{12}$$

where i is the i-th dimension.

The aggregation is then defined as follows:

$$\mathbf{z} = \sum_{i,t} \mathrm{softmax}\big(f_a\big(\mathbf{s}_i^t\big)\big) \cdot f_b\big(\mathbf{s}_i^t\big), \quad \text{with} \quad \mathbf{s}_i^t = \big[\mathbf{x}_i^t, \mathbf{o}_i\big] + \mathbf{t}, \tag{13}$$

where $f_a : \mathbb{R}^{D_\mathrm{s}} \to \mathbb{R}$, $f_b : \mathbb{R}^{D_\mathrm{s}} \to \mathbb{R}^{D_\mathrm{z}}$ are 2-layer MLPs with SiLU activations [36] in-between. They can be summarized as:

$$f_a \coloneqq \{\mathrm{Linear}(D_\mathrm{s}, D_\mathrm{z}) \to \mathrm{SiLU} \to \mathrm{Linear}(D_\mathrm{z}, 1)\}, \tag{14}$$

$$f_b \coloneqq \{\mathrm{Linear}(D_\mathrm{s}, D_\mathrm{z}) \to \mathrm{SiLU} \to \mathrm{Linear}(D_\mathrm{z}, D_\mathrm{z})\}. \tag{15}$$

In all experiments, we use $D_\mathrm{o} = 512$, $D_\mathrm{z} = 512$, and $D_\mathrm{s} = D_\mathrm{o} + 2d = 516$ in $d = 2$ dimensions, or $D_\mathrm{s} = 518$ in $d = 3$ dimensions.

Neural field conditioning We condition the neural field using FiLM [33]. Following the implementation details of FiLM, in practice, we use the following equation for a FiLM layer:

$$\mathbf{h}' \coloneqq \text{FiLM}(\mathbf{h}, \mathbf{z}) = (1 + \alpha(\mathbf{z})) \odot \mathbf{h} + \beta(\mathbf{z}), \tag{16}$$

where $\mathbf{h}$ is the encoded input in the first FiLM layer, or the conditioned input in subsequent FiLM layers, and $\alpha : \mathbb{R}^{D_\text{z}} \to \mathbb{R}^{D_\text{h}}, \beta : \mathbb{R}^{D_\text{z}} \to \mathbb{R}^{D_\text{h}}$ are MLPs. This equation deviates slightly from Section 2, since it predicts the residual of a multiplicative modulation. This approach can be beneficial during the early stages of training, since it initially defaults to an identity transformation for zero-initialized weights, while the alternative can "zero out" the network outputs. For both α and β we use 2-layer MLPs with SiLU activations in-between.

$$\alpha \coloneqq \{\text{Linear}(D_\text{z}, D_\text{h}) \to \text{SiLU} \to \text{Linear}(D_\text{h}, D_\text{h})\} \tag{17}$$

$$\beta \coloneqq \{\text{Linear}(D_\text{z}, D_\text{h}) \to \text{SiLU} \to \text{Linear}(D_\text{h}, D_\text{h})\}. \tag{18}$$

Unless specified otherwise, in all experiments, we use $D_\text{h} = 512$.

Full neural field The full neural field is a 3-layer MLP with SiLU [36] activations in-between, and FiLM layers after the first two linear layers, and outputs a latent force field. The neural field can be summarized as:

$$\mathbf{f}(\mathbf{v} \mid \mathbf{z}) = \{\text{Linear} \to \text{FiLM} \to \text{SiLU} \to \text{Linear} \to \text{FiLM} \to \text{SiLU} \to \text{Linear}\}. \tag{19}$$

A.1.2 Aether as a variational autoencoder

Here we present our graph network architecture that closely follows Graber and Schwing [14], Kofinas et al. [24]. The model is formulated as a variational autoencoder [21, 37] with latent edge types that infers a latent graph structure. The encoder is tasked with predicting interactions between object pairs, while the decoder uses the sampled graph structure to make predictions. As mentioned in Section 3.3, our full architecture also integrates G-LoCS, *i.e.* the augmented node states include the predicted forces exerted at the target node, as well as the state of the auxiliary origin-node, expressed in the local frame of the target node.

Encoder Equations (20) to (22) describe the message passing steps of our graph network. In these equations, we process each timestep independently.Then, in Equations (23) and (24) we compute the evolution of edge embeddings over time with LSTMs [16], and in Equations (25) and (26) we estimate the posterior and the learned prior over our edges.

$$\mathbf{h}_{j,i}^{(1),t} = f_e^{(1)}\left(\left[\mathbf{v}_{j|i}^t, \boxed{\mathbf{f}_{j|i}^t}, \mathbf{v}_{i|i}^t, \boxed{\mathbf{f}_{i|i}^t}, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) \tag{20}$$

$$\mathbf{h}_i^{(1),t} = f_v^{(1)}\left(g_v^{(1)}\left(\left[\mathbf{v}_{i|i}^t, \boxed{\mathbf{f}_{i|i}^t}, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j \in \mathcal{N}(i)} \mathbf{h}_{j,i}^{(1),t}\right) \tag{21}$$

$$\mathbf{h}_{j,i}^{(2),t} = f_e^{(2)}\left(\left[\mathbf{h}_i^{(1),t}, \mathbf{h}_{j,i}^{(1),t}, \mathbf{h}_j^{(1),t}\right]\right) \tag{22}$$

$$\mathbf{h}_{(j,i),\text{prior}}^t = \text{LSTM}_\text{prior}\left(\mathbf{h}_{j,i}^{(2),t}, \mathbf{h}_{(j,i),\text{prior}}^{t-1}\right) \tag{23}$$

$$\mathbf{h}_{(j,i),\text{enc}}^t = \text{LSTM}_\text{enc}\left(\mathbf{h}_{j,i}^{(2),t}, \mathbf{h}_{(j,i),\text{enc}}^{t+1}\right) \tag{24}$$

$$p_\phi\left(\boldsymbol{z}^t | \mathbf{x}^{1:t}, \boldsymbol{z}^{1:t-1}\right) = \text{softmax}\left(f_\text{prior}\left(\mathbf{h}_{(j,i),\text{prior}}^t\right)\right) \tag{25}$$

$$q_\phi\left(\boldsymbol{z}_{j,i}^t | \mathbf{x}\right) = \text{softmax}\left(f_\text{enc}\left(\left[\mathbf{h}_{(j,i),\text{prior}}^t, \mathbf{h}_{(j,i),\text{enc}}^t\right]\right)\right) \tag{26}$$

The functions $f_e^{(1)}, f_v^{(1)}, f_e^{(2)}, g_v^{(1)}, f_\text{prior}, f_\text{enc}$ denote MLPs.

Decoder The decoder samples $\boldsymbol{z}_{(j,i)}^t$ using Gumbel-Softmax [27, 19]. The following equations formalize a message passing scheme performed for the current timestep, and another one performed for the hidden node states. Both are used to update the hidden node states, and to make predictions for the next timestep. As mentioned in Section 2, we make predictions in the local coordinate frame of each node. Thus, we perform an inverse transformation for each node to transform the predictions back to the global coordinate frame.

$$\mathbf{m}_{j,i}^t = \sum_k z_{(j,i),k}^t f^k \left(\left[\mathbf{v}_{j|i}^t, \boxed{\mathbf{f}_{j|i}^t}, \mathbf{v}_{i|i}^t, \boxed{\mathbf{f}_{i|i}^t}, \mathbf{v}_{\mathcal{O}|i}^t \right] \right) \tag{27}$$

$$\mathbf{m}_i^t = f_v^{(3)} \left(g_v^{(3)} \left(\left[\mathbf{v}_{i|i}^t, \boxed{\mathbf{f}_{i|i}^t}, \mathbf{v}_{\mathcal{O}|i}^t \right] \right) + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{j,i}^t \right) \tag{28}$$

$$\mathbf{h}_{j,i}^t = \sum_k z_{(j,i),k}^t g^k \left(\left[\mathbf{h}_j^t, \mathbf{h}_i^t \right] \right) \tag{29}$$

$$\mathbf{n}_i^t = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_{(j,i)}^t \tag{30}$$

$$\mathbf{h}_i^{t+1} = \mathrm{GRU}\left(\left[\mathbf{n}_i^t, \mathbf{m_i}^t \right], \mathbf{h}_i^t \right) \tag{31}$$

$$\boldsymbol{\mu}_i^{t+1} = \mathbf{x}_i^t + \mathbf{R}_i^t \cdot f_v^{(4)} \left(\mathbf{h}_i^{t+1} \right) \tag{32}$$

$$p(\mathbf{x}_i^{t+1} | \mathbf{x}^{1:t}, \boldsymbol{z}^{1:t}) = \mathcal{N}\left(\boldsymbol{\mu}_i^{t+1}, \sigma^2 \mathbf{I} \right) \tag{33}$$

Our GRU [1] is identical to the one used in [23].

Training Our full VAE model is trained by minimizing the negative Evidence Lower Bound (ELBO), which comprises the reconstruction loss of the predicted trajectories (positions and velocities) and the KL divergence.

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\boldsymbol{z})] - \mathrm{KL}[q_\phi(\boldsymbol{z}|\mathbf{x}) || p_\phi(\boldsymbol{z}|\mathbf{x})] \tag{34}$$

Following Graber and Schwing [14], the reconstruction loss and the KL divergence take the following form:

$$\mathbb{E}_{q_\phi(\boldsymbol{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\boldsymbol{z})] = -\sum_i \sum_t \frac{||\mathbf{x}_i^t - \boldsymbol{\mu}_i^t||}{2\sigma^2} + \frac{1}{2} \log(2\pi\sigma^2), \tag{35}$$

$$\mathrm{KL}[q_\phi(\boldsymbol{z}|\mathbf{x}) || p_\phi(\boldsymbol{z}|\mathbf{x})] = \sum_{t=1}^{T} \left(\mathbb{H}(q_\phi(\boldsymbol{z}_{ji}^t|\mathbf{x})) - \sum_{\boldsymbol{z}_{ji}^t} q_\phi(\boldsymbol{z}_{ji}^t|\mathbf{x}) \log p_\phi(\boldsymbol{z}_{ji}^t|\mathbf{x}^{1:t}, \boldsymbol{z}^{1:t-1}) \right), \tag{36}$$

where $\mathbb{H}$ denotes the entropy operator. In all experiments, we set the variance $\sigma^2 = 10^{-5}$.

We train Aether using Adam [21]. Unless stated otherwise, in all experiments, we use a learning rate of $5\mathrm{e}{-4}$.

A.1.3 Aether architecture in Lorentz force field setting

The Lorentz force field setting, proposed by [10] uses only a single timestep as input and the task is to predict the positions for a single timestep in the future. Thus, we have to modify our architecture for this setting. This way, we also ensure a fairer comparison with other methods. We do not use an NRI [22] or dNRI [14] backbone in this setting.

$$\mathbf{h}_{j,i}^{(1)} = f_e^{(1)} \left(\left[\mathbf{v}_{j|i}, \boxed{\mathbf{f}_{j|i}}, \mathbf{v}_{i|i}, \boxed{\mathbf{f}_{i|i}}, q_i q_j, \|\mathbf{r}_{j,i}\|_2 \right] \right) \tag{37}$$

$$\mathbf{h}_i^{(1)} = f_v^{(1)} \left(g_v \left(\left[\mathbf{v}_{i|i}, \boxed{\mathbf{f}_{i|i}} \right] \right) + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_{j,i}^{(1)} \right) \tag{38}$$

$$\mathbf{h}_{j,i}^{(l)} = f_e^{(l)} \left(\left[\mathbf{h}_i^{(l-1)}, \mathbf{h}_{j,i}^{(l-1)}, \mathbf{h}_j^{(l-1)} \right] \right) \tag{39}$$

$$\mathbf{h}_i^{(l)} = f_v^{(l)} \left(\mathbf{h}_i^{(l-1)} + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_{j,i}^{(l)} \right) \tag{40}$$

$$\hat{\mathbf{p}}_i = \mathbf{p}_i + \mathbf{R}_i \cdot f_o\left(\mathbf{h}_i^L \right) \tag{41}$$

with $l \in \{2, \dots, L\}$. We use 4 layers, *i.e.* $L = 4$. The functions $f_e^{(l)}$ denote 2-layer MLPs with SiLU [36] activations after each layer. The functions $f_v^{(l)}$ denote 2-layer MLPs with SiLU activations in-between, and doubling the dimensionality in-between. Finally, g_v denotes a linear layer, while f_o denotes a 3-layer MLP with SiLU activations in-between. Following [10], we use a hidden dimension of 64. For this experiment, we use a learning rate of $1e-3$.

$$f_v^{(l)} = \{\text{Linear} \rightarrow \text{SiLU} \rightarrow \text{Linear}\} \tag{42}$$

$$f_e^{(l)} = \{\text{Linear} \rightarrow \text{SiLU} \rightarrow \text{Linear} \rightarrow \text{SiLU}\} \tag{43}$$

$$f_o = \{\text{Linear} \rightarrow \text{SiLU} \rightarrow \text{Linear} \rightarrow \text{SiLU} \rightarrow \text{Linear}\} \tag{44}$$

For the neural field, we use the input positions and velocities as input, as well as the particle charges, *i.e.* $\mathbf{f} = f(\mathbf{p}, \mathbf{u}, q)$. We do not use any input encoding for positions or velocities in this setting, but we use an embedding for the charges that maps them to 16 dimensions. We concatenate the charge embeddings with positions and velocities and feed them as input to the neural field. The neural field is a 3-layer MLP with SiLU activations in-between. We use a hidden dimension of 32 in the neural field.

$$f = \{\text{Linear} \rightarrow \text{SiLU} \rightarrow \text{Linear} \rightarrow \text{SiLU} \rightarrow \text{Linear}\} \tag{45}$$

A.2 G-LoCS

Encoder

$$\mathbf{h}_{j,i}^{(1),t} = f_e^{(1)}\left(\left[\mathbf{v}_{j|i}^t, \mathbf{v}_{i|i}^t, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) \tag{46}$$

$$\mathbf{h}_{i}^{(1),t} = f_v^{(1)}\left(g_v^{(1)}\left(\left[\mathbf{v}_{i|i}^t, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{h}_{j,i}^{(1),t}\right) \tag{47}$$

$$\mathbf{h}_{j,i}^{(2)} = f_e^{(2)}\left(\left[\mathbf{h}_i^{(1),t}, \mathbf{h}_{j,i}^{(1),t}, \mathbf{h}_j^{(1),t}\right]\right) \tag{48}$$

$$\mathbf{h}_{(j,i),\text{prior}}^t = \text{LSTM}_{\text{prior}}\left(\mathbf{h}_{j,i}^{(2),t}, \mathbf{h}_{(j,i),\text{prior}}^{t-1}\right) \tag{49}$$

$$\mathbf{h}_{(j,i),\text{enc}}^t = \text{LSTM}_{\text{enc}}\left(\mathbf{h}_{j,i}^{(2),t}, \mathbf{h}_{(j,i),\text{enc}}^{t+1}\right) \tag{50}$$

$$p_\phi\left(\boldsymbol{z}^t | \mathbf{x}^{1:t}, \boldsymbol{z}^{1:t-1}\right) = \text{softmax}\left(f_{\text{prior}}\left(\mathbf{h}_{(j,i),\text{prior}}^t\right)\right) \tag{51}$$

$$q_\phi\left(\boldsymbol{z}_{j,i}^t | \mathbf{x}\right) = \text{softmax}\left(f_{\text{enc}}\left(\left[\mathbf{h}_{(j,i),\text{prior}}^t, \mathbf{h}_{(j,i),\text{enc}}^t\right]\right)\right) \tag{52}$$

Decoder

$$\mathbf{m}_{j,i}^t = \sum_k z_{(j,i),k}^t f^k\left(\left[\mathbf{v}_{j|i}^t, \mathbf{v}_{i|i}^t, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) \tag{53}$$

$$\mathbf{m}_i^t = f_v^{(3)}\left(g_v^{(3)}\left(\left[\mathbf{v}_{i|i}^t, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{m}_{j,i}^t\right) \tag{54}$$

$$\mathbf{h}_{j,i}^t = \sum_k z_{(j,i),k}^t g^k\left(\left[\mathbf{h}_j^t, \mathbf{h}_i^t\right]\right) \tag{55}$$

$$\mathbf{n}_i^t = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{h}_{(j,i)}^t \tag{56}$$

$$\mathbf{h}_i^{t+1} = \text{GRU}\left(\left[\mathbf{n}_i^t, \mathbf{m_i}^t\right], \mathbf{h}_i^t\right) \tag{57}$$

$$\boldsymbol{\mu}_i^{t+1} = \mathbf{x}_i^t + \mathbf{R}_i^t \cdot f_v^{(4)}\left(\mathbf{h}_i^{t+1}\right) \tag{58}$$

$$p(\mathbf{x}_i^{t+1} | \mathbf{x}^{1:t}, \boldsymbol{z}^{1:t}) = \mathcal{N}\left(\boldsymbol{\mu}_i^{t+1}, \sigma^2\mathbf{I}\right) \tag{59}$$

A.3 Source oracle

The *source oracle* modifies LoCS [24] to use virtual nodes. Since graph networks are permutation invariant, we cannot just include the sources as nodes of the graph and perform message passing, as the network would not be able to distinguish particles from sources. Thus, we treat the sources separately in the message passing so that the network can identify them. More specifically, we introduce a new message function that computes $\mathrm{field\ source} \rightarrow \mathrm{particle}$ messages. Furthermore, we introduce a separate aggregation function in the update step that only aggregates the messages from field sources. We denote the set of field sources as $\mathcal{S}$. The state of a field source $s \in \mathcal{S}$ is denoted as $\mathbf{v}_s$, while the same state expressed in the local coordinate frame of node i is denoted as $\mathbf{v}_{s|i}$. The source oracle graph network is defined as follows:

$$\mathbf{h}^t_{j,i} = f_e\left(\left[\mathbf{v}^t_{j|i}, \mathbf{v}^t_{i|i}\right]\right), \tag{60}$$

$$\mathbf{h}^t_{s,i} = f_s\left(\left[\mathbf{v}^t_{s|i}, \mathbf{v}^t_{i|i}\right]\right), \tag{61}$$

$$\mathbf{\Delta x}^{t+1}_{i|i} = f_v\left(g_v\left(\mathbf{v}^t_{i|i}\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{h}^t_{j,i} + \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\mathbf{h}^t_{s,i}\right), \tag{62}$$

where f_s is an MLP. We predict future states for all the “observable” particles, but not for the “source” particles.

A.4 Parallel aether architecture

$$\mathbf{h}^{(1)}_{j,i} = f^{(1)}_e\left(\left[\mathbf{v}_{j|i}, \mathbf{v}_{i|i}, q_i q_j, \|\mathbf{r}_{j,i}\|_2\right]\right) \tag{63}$$

$$\mathbf{h}^{(1)}_{i} = f^{(1)}_v\left(g_v\left(\mathbf{v}_{i|i}\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{h}^{(1)}_{j,i}\right) \tag{64}$$

$$\mathbf{h}^{(l)}_{j,i} = f^{(l)}_e\left(\left[\mathbf{h}^{(l-1)}_{i}, \mathbf{h}^{(l-1)}_{j,i}, \mathbf{h}^{(l-1)}_{j}\right]\right) \tag{65}$$

$$\mathbf{h}^{(l)}_{i} = f^{(l)}_v\left(\mathbf{h}^{(l-1)}_{i} + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{h}^{(l)}_{j,i}\right) \tag{66}$$

$$\hat{\mathbf{p}}_i = \mathbf{p}_i + \mathbf{R}_i \cdot f_o\left(\mathbf{h}^L_i\right) + \boxed{\mathbf{f}_i} \tag{67}$$

A.5 Aether with EGNN backbone

Our method is agnostic to the choice of equivariant graph network; we expect that it would be beneficial for a number of strictly equivariant networks. To test this hypothesis, we combine EGNN [41] with our method; starting from the velocity formulation of EGNN, we modify the message and velocity equations to incorporate the predicted forces for each node, as follows:

$$\mathbf{m}_{j,i} = \phi_e\left(\mathbf{h}^l_i, \mathbf{h}^l_j, \left\|\mathbf{p}^l_j - \mathbf{p}^l_i\right\|^2_2, a_{j,i}, \boxed{\mathbf{f}_i}, \boxed{\mathbf{f}_j}\right), \tag{68}$$

$$\mathbf{u}^{l+1}_i = \phi_v\left(\mathbf{h}^l_i, \boxed{\mathbf{f}_i}\right)\mathbf{u}^l_i + C\sum_{j\neq i}\left(\mathbf{p}^l_j - \mathbf{p}^l_i\right)\cdot\phi_x(\mathbf{m}_{j,i}), \tag{69}$$

$$\mathbf{p}^{l+1}_i = \mathbf{p}^l_i + \mathbf{u}^{l+1}_i, \tag{70}$$

$$\mathbf{m}_i = \sum_{j\in\mathcal{N}(i)}\mathbf{m}_{ji}, \tag{71}$$

$$\mathbf{h}^{l+1}_i = \phi_h\left(\mathbf{h}^l_i, \mathbf{m}_i\right). \tag{72}$$

The remaining EGNN components remain unaltered.

A.6 Computing resources

All experiments were performed on single GPUs. We used 2 different GPU models, namely the Nvidia RTX 2080 Ti, and Nvidia GTX 1080 Ti. Our source code was written in PyTorch [32], version 1.4.0, and CUDA 10.0.

B Dataset details

B.1 Electrostatic field

Kipf et al. [23] introduced a dataset of interacting charged particles. Charged particles interact via electrostatic Coulomb forces. We assume a set of N particles, and each particle has a position $\mathbf{p}_i^t \in \mathbb{R}^D$ and a charge $q_i \in \mathbb{R}$. The force $\mathbf{f}_{j,i}^t$ exerted from particle j to particle i is computed as follows:

$$\mathbf{p}_{j,i}^t = \mathbf{p}_j^t - \mathbf{p}_i^t, \tag{73}$$

$$\hat{\mathbf{p}}_{j,i}^t = \frac{\mathbf{p}_{j,i}^t}{\left\|\mathbf{p}_{j,i}^t\right\|}, \tag{74}$$

$$\mathbf{f}_{j,i}^t = C \cdot q_i q_j \frac{\hat{\mathbf{p}}_{j,i}^t}{\left\|\mathbf{p}_{j,i}^t\right\|^2}. \tag{75}$$

Since forces only depend on positions at the current timestep, in the following equations, we omit the time indices to reduce clutter. The total force exerted at particle i is:

$$\mathbf{f}_i = \sum_{j=1, j\neq i}^{N} \mathbf{f}_{j,i} = C \cdot q_i \sum_{j=1, j\neq i}^{N} q_j \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2}. \tag{76}$$

The electric field is a vector field, whose value at the test position $\mathbf{p}_i$ assumes a positive test charge $q_i = 1$, and is defined as:

$$\mathbf{E}_{j,i} = \frac{\mathbf{f}_{j,i}}{q_i} = C \cdot q_j \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2}, \tag{77}$$

$$\mathbf{E}_i = \sum_{j=1, j\neq i}^{N} \mathbf{E}_{j,i} = C \cdot \sum_{j=1, j\neq i}^{N} q_j \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2}. \tag{78}$$

Our first experiment aims to study the effect of static fields, *i.e.* a single field across all train, validation, and test simulations. We extend the charged particles dataset by adding a number of immovable sources. Overall, these sources act like regular particles, exerting forces on the observable particles, except we ignore any forces exerted to them, and fix their positions and velocities to zero. We use $N = 5$ "observable" particles and $M = 20$ "source" particles. In all experiments, we assume unit charges, $q_i = \pm 1$, and $C = 1$. The probabilities of positive or negative charges are equal. Then, the forces and the electric field can be simplified as:

$$\mathbf{f}_{j,i} = \operatorname{sign}(q_i q_j) \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2}, \tag{79}$$

$$\mathbf{E}_i = \sum_{j=1, j\neq i}^{N} \operatorname{sign}(q_j) \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2}. \tag{80}$$

The net force exerted at a particle $i \in \{1, \ldots, N\}$ is computed as:

$$\mathbf{f}_i = \sum_{j=1, j\neq i}^{N+M} \mathbf{f}_{j,i} = \underbrace{\sum_{j=1, j\neq i}^{N} \mathbf{f}_{j,i}}_{\text{particles}} +. \underbrace{\sum_{j=N+1}^{N+M} \mathbf{f}_{j,i}}_{\text{field}} \tag{81}$$

Following Satorras et al. [41], Fuchs et al. [12], Kofinas et al. [24], we remove virtual borders that cause elastic collisions. We generate a dataset of 50,000 simulations for training, 10,000 for validation and 10,000 for testing. The datasets contains *only* the positions and velocities for the "observable" particles, while the field sources are only used for visualization. Following Kipf et al. [23], each simulation lasts for 49 timesteps. During inference, we use the first 29 steps as input and predict the remaining 20 steps.

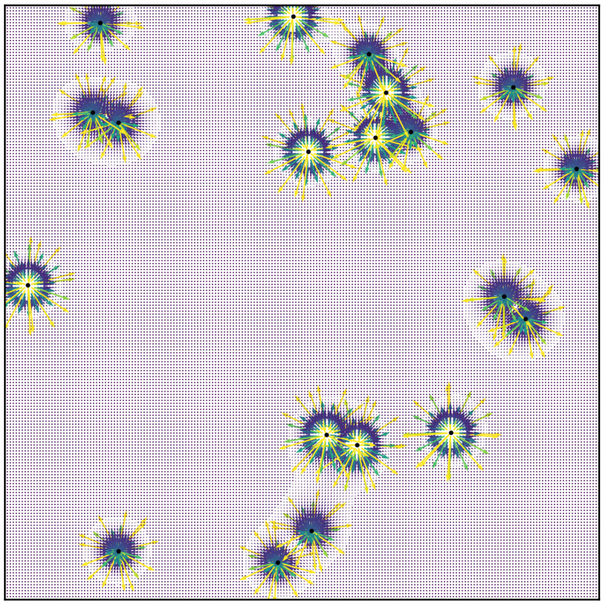


Figure 7: Visualization of the static field in the electrostatic field setting

B.2 Traffic scenes - inD

InD [4] is a real-world traffic scenes dataset that comprises trajectories of pedestrians, vehicles, and cyclists. It contains 33 recordings, recorded at 4 different locations in Aachen, Germany. We hypothesize that discovering a latent traffic force field will be beneficial for trajectory forecasting in traffic scenes. For simplicity, we focus on static field discovery in traffic scenes. We create a subset that contains scenes from a single location. Namely, we choose "Frankenburg, Aachen", since it is the location with most interactions in the dataset. The subset corresponds to 12 recordings; we use 8 for training, 2 for validation, and 2 for testing. We follow a similar experimental setting with Graber and Schwing [14], Kofinas et al. [24]. We divide each scene into 18-step sequences. We use the first 6 time steps as input and predict the next 12 time steps.

B.3 Gravitational n-body dataset

In this experiment, we study the influence of dynamic fields, *i.e.* fields that are different across simulations. Similar to the electrostatic field setting, we extend the gravitational n-body dataset by Brandstetter et al. [5] by adding gravitational sources. The equation that describes the forces is similar to Equation (75). Namely, we have

$$\mathbf{f}_{j,i}^{t} = C \cdot m_i m_j \frac{\hat{\mathbf{p}}_{j,i}^{t}}{\left\|\mathbf{p}_{j,i}^{t}\right\|^2}, \tag{82}$$

where m_i, m_j are the particle masses. We create a dataset of 50,000 simulations for training, 10,000 for validation and 10,000 for testing. We use $N = 5$ particles and $M = 1$ source. We set the masses of particles to $m_p = 1$, while the source has a mass of $m_s = 10$. Similarly to the electrostatic field experiment, the datasets contains *only* the positions and velocities for the "observable" particles, while the field source is only used for visualization. We generate trajectories of 49 timesteps. We use the first 44 timesteps as input and predict the remaining 5 steps. All other dataset details are identical to Brandstetter et al. [5].

B.3.1 2D gravitational n-body dataset

We also experiment with a smaller variant of the dynamic gravitational fields, using a 2D setting. We create a dataset of 5,000 simulations for training, 1,000 for validation and 1,000 for testing. All other dataset details are the same with the full 3D dataset. We report results in Figure 21. We showcase predicted trajectories in Appendix D.3, and the learned fields in Appendix D.3.1.

C Extra experiments

C.1 Alternative equivariant network backbones

To further test the applicability of our method, we combine it with different equivariant graph network backbones. First, we combine our method with GMN [18]. Similar to Equations (4) and (5), we concatenate the predicted forces for each node with the message $\mathbf{Z}_{ji}$ in $[\mathbf{Z}_{ji}, \mathbf{f}_i, \mathbf{f}_j]$. The formulation above is further motivated by SGNN [15], which extends GMN by including gravity as an external force term, as well as object-aware information (see appendix A.3 in [15] for a comparison). In our case, we replace the gravity term with the predicted forces per node. Thus, instead of $[\mathbf{Z}_{ji}, \mathbf{g}]$, we have $[\mathbf{Z}_{ji}, \mathbf{f}_i, \mathbf{f}_j]$.

We train and evaluate GMN on the Lorentz force field setting. We then add the force terms using the formulation above. We show the results in the table below. Indeed, using Aether greatly enhances the performance of GMN, which further enhances our hypothesis.

Table 4: Ablation study on the choice of equivariant GNN backbone. Position prediction MSE on Lorentz force field.

Method	MSE (↓)
GMN [18]	0.0365
GMN+Aether (ours)	**0.0261**

Next, we integrate our method in EqMotion [55], a recent equivariant method with state-of-the-art performance on trajectory forecasting. We incorporate Aether in EqMotion by treating the predicted forces as geometric features similar to velocities. Namely, after computing the forces for each object at each timestep, we compute the magnitudes of the force vectors and the force angle sequence, *i.e.* the angles between forces in consecutive timesteps. We concatenate these quantities to the existing features for the feature initialization step.

We train and evaluate EqMotion and Aether with EqMotion on inD [4] following our experimental setup. We report the results in Figure 8. We see that Aether is beneficial even for a state-of-the-art trajectory forecasting method, which further strengthens our claims.

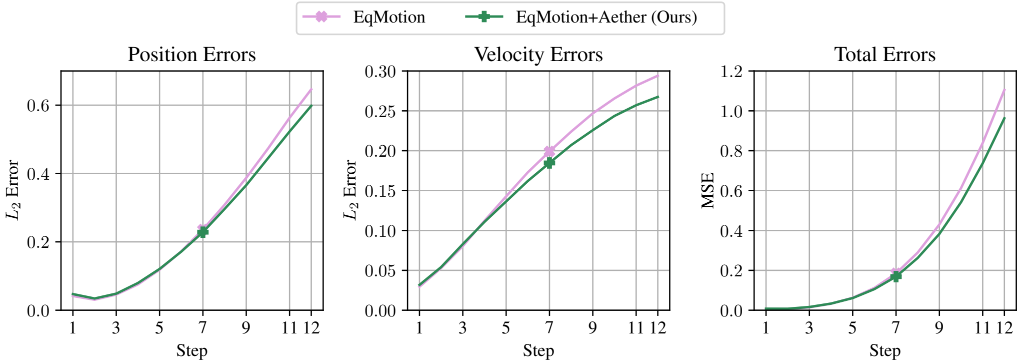


Figure 8: Ablation study on the choice of equivariant GNN backbone. Results on inD.

C.2 Non-equivariant network with neural field

Our ability to capture global components with a neural field stems from our overall architecture, which promotes disentanglement. Using a graph network that respects the underlying symmetries and has an inductive bias towards using local interactions, *i.e.* any equivariant graph network, allows the neural field to "solve for" the global components, by "subtracting" the local interactions from the observable net effects.

The choice of an equivariant network is crucial here; a non-equivariant graph network like NRI [23] or dNRI [14] would merely gather "redundant" information from the neural field. We demonstrate this mathematically in the following equations for the static field, in which we first compute the force $\mathbf{f}$ at a target position $\mathbf{p}$, and then compute the node embedding using equations from NRI/dNRI, including the forces.

$$\mathbf{f} = f(\mathbf{p}) = \mathrm{MLP}_1(\mathbf{p}) \tag{83}$$

$$\mathbf{h} = g(\mathbf{p}, \mathbf{u}, \mathbf{f}) = \mathrm{MLP}_2([\mathbf{p}, \mathbf{u}, \mathbf{f}]) \tag{84}$$

We can see that the node embeddings depend on positions twice, one explicit and one through another MLP, in an architecture similar to a concatenated residual connection. In this case, we do not expect the neural field to isolate global forces, or to be helpful for future forecasting. We test this hypothesis with an ablation experiment on the electrostatic field setting, by combing our neural field with dNRI, instead of an equivariant network. In Table 5, we report the MSE at the final prediction timestep, i.e. MSE@20. We can see that adding a neural field to a non-equivariant network does not enhance performance, and in fact, it results in performance degradation, which enhances our hypothesis.

Table 5: Ablation study on the suitability of non-equivariant networks with Aether. Combining dNRI –a non-equivariant graph network– with a neural field does not enhance performance. Results on the electrostatic field setting.

Method	MSE@20 (↓)
dNRI [14]	1.20
dNRI+Aether	1.37
Aether	**0.69**

C.3 Choice of conditioning mechanism

FiLM [33] is used in the dynamic field setting to condition neural fields. To examine its influence on performance, we perform an ablation study on the 3D gravitational setting, where we replace FiLM layers with conditioning by concatenation, a very simple and successful conditioning mechanism. We term this model *Concat Aether*, and report the results in Table 6. We see that conditioning by concatenation underperforms, scoring almost on par with LoCS. This is perhaps expected, since concatenation is a rather weak form of conditioning, and our task is very challenging. On the other hand, FiLM is a powerful mechanism and is able to condition effectively.

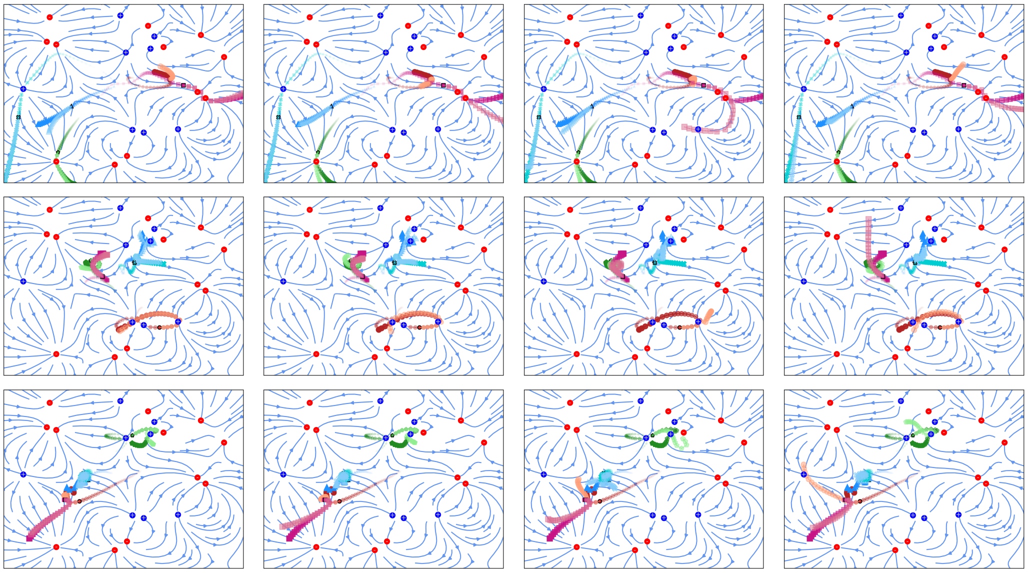
Table 6: Ablation study on the choice of conditioning mechanism. Results on the 3D gravitational field setting.

Method	MSE@5 (↓)
LoCS [24]	0.1308
Aether	**0.0660**
Concat Aether	0.1474

D Qualitative results

D.1 Electrostatic field

Figure 9 shows qualitative results on the electrostatic field setting.



(a) Aether (b) G-LoCS (c) LoCS (d) dNRI

Figure 9: Predictions on the electrostatic field setting. Lighter colors indicate predictions, and darker colors indicate the groundtruth. Predictions start where markers have black edges. Markers get bigger and more opaque as trajectories evolve in time. The background streamplots indicate the groundtruth field, and are not given as input to the networks. Similarly, the blue $\oplus$ markers and the red $\ominus$ markers, are merely shown for illustrative purposes, indicating the charges of the field sources, and are not given as input to the networks. *Best viewed in color.*

D.1.1 Discovered electrostatic field

In Figure 10 we visualize the discovered electrostatic field compared to the groundtruth one.

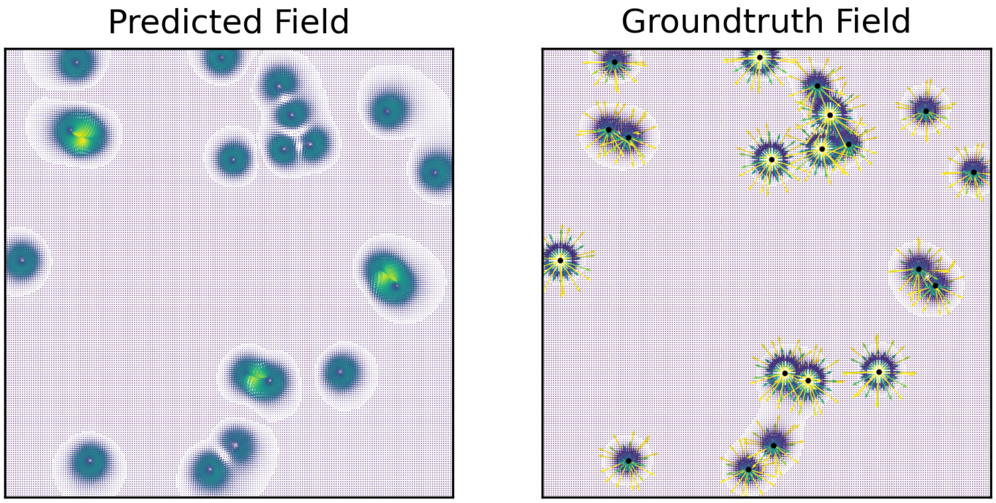
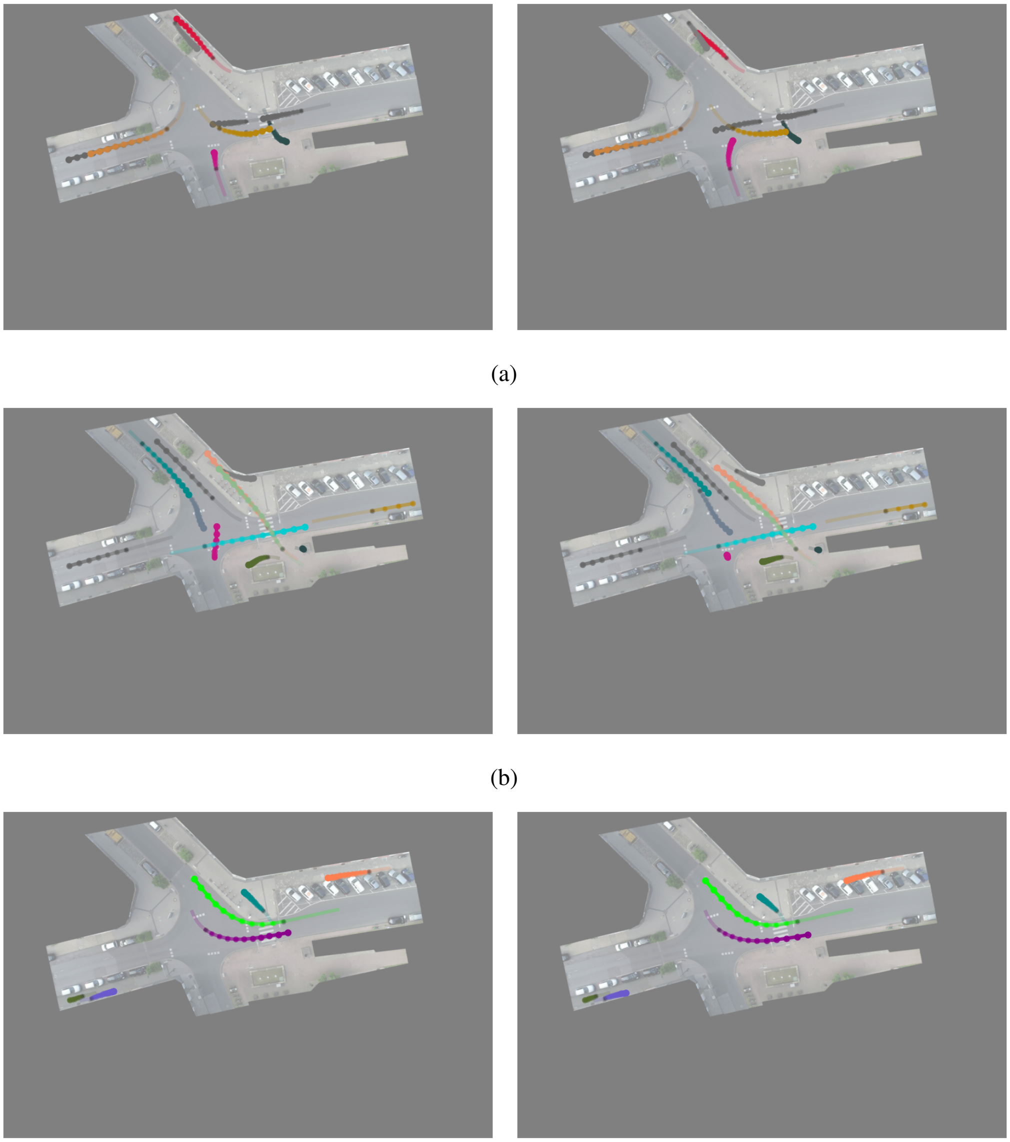


Figure 10: Learned Field (left) in electrostatic field setting compared to groundtruth (right).

D.2 InD

Figure 11 shows qualitative results of our method on inD [4]. Figures 12, 13 and 14 show qualitative results for G-LoCS, LoCS, and dNRI, respectively.

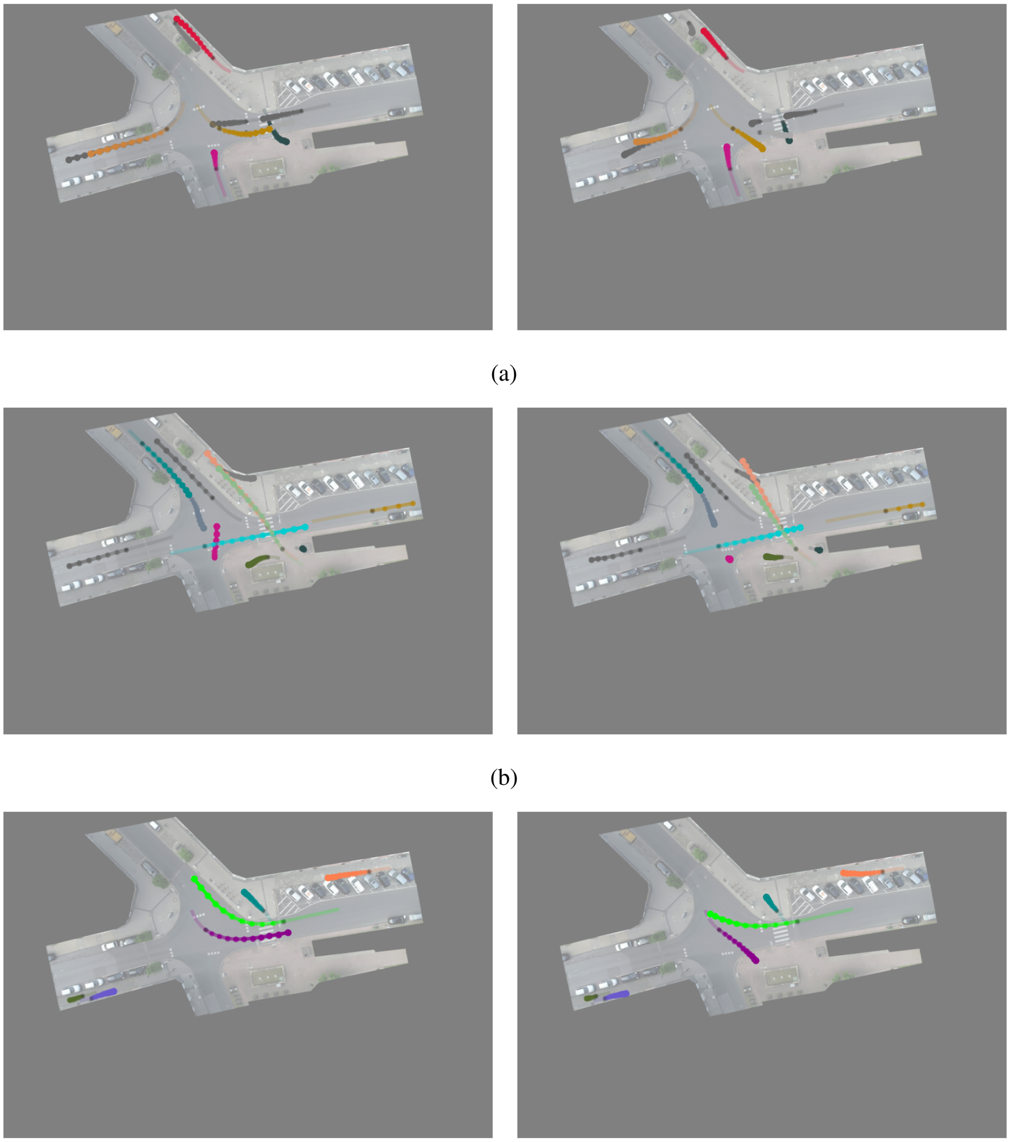


(a)

(b)

(c)

Figure 11: Aether predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*

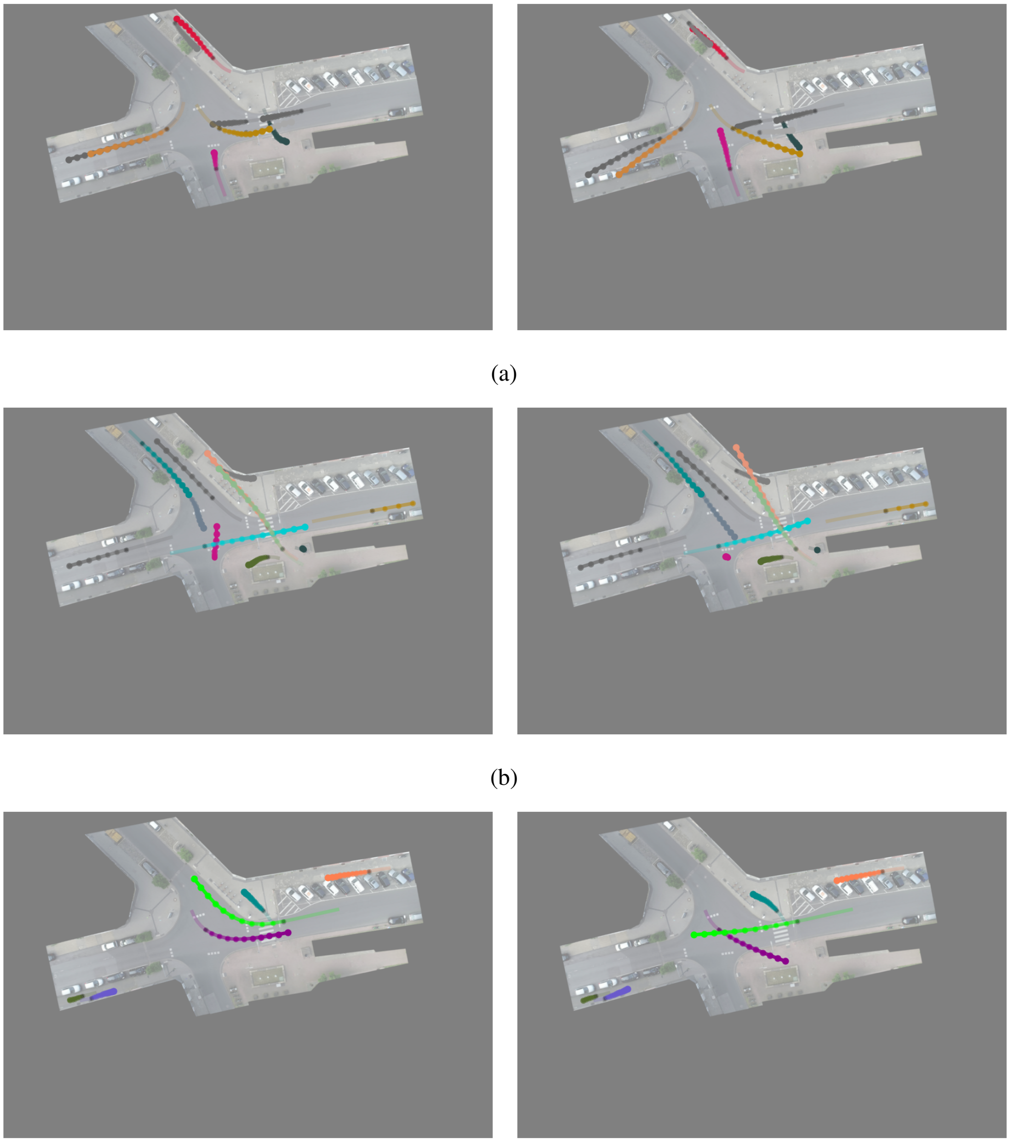


(a)

(b)

(c)

Figure 12: G-LoCS predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*

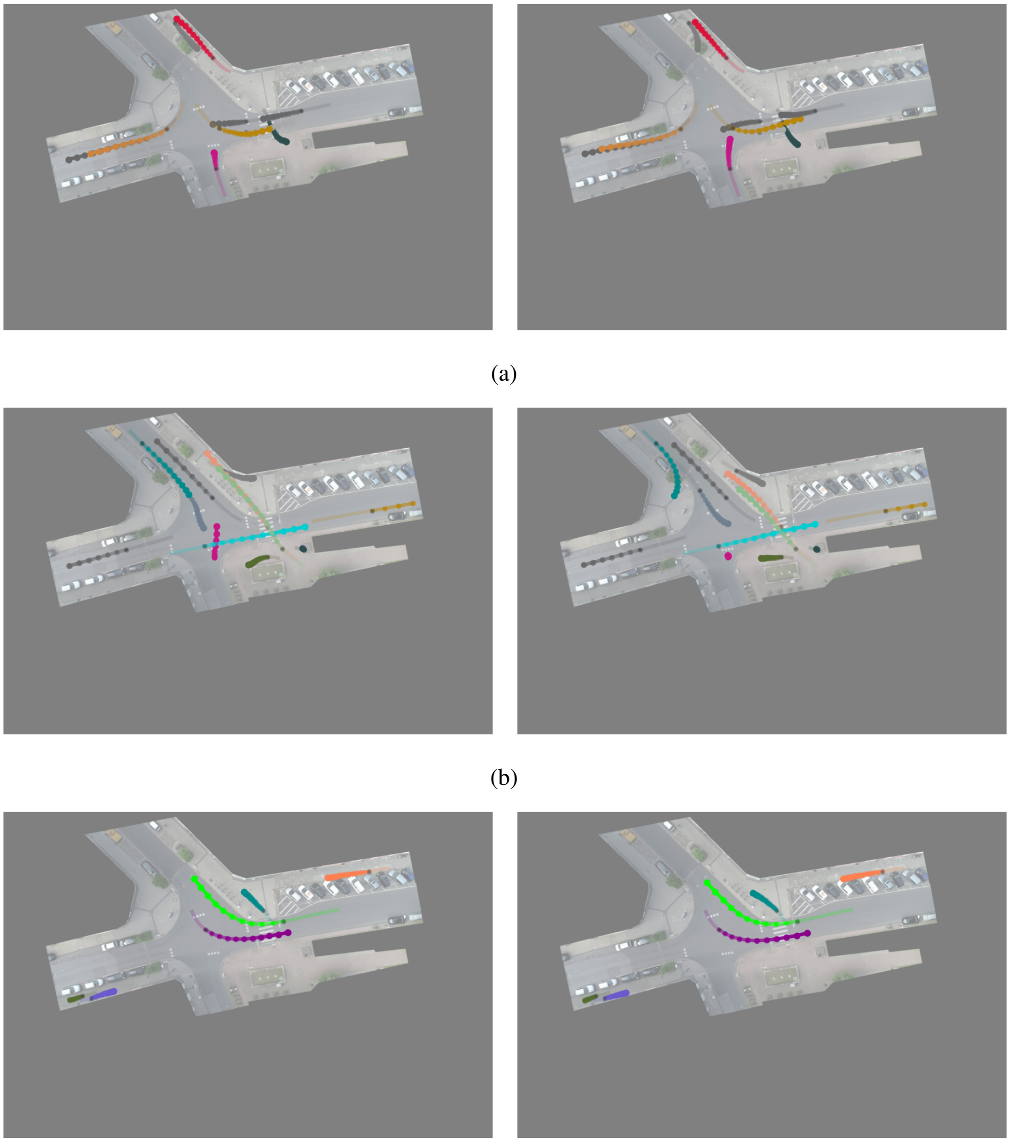


(a)

(b)

(c)

Figure 13: LoCS predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*



(a)

(b)

(c)

Figure 14: dNRI predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*

D.2.1 Discovered traffic force field

In Figure 15 we visualize the discovered traffic force field on inD. In the supplementary material, we provide video visualizations of the learned field, with input orientations evolving over time.

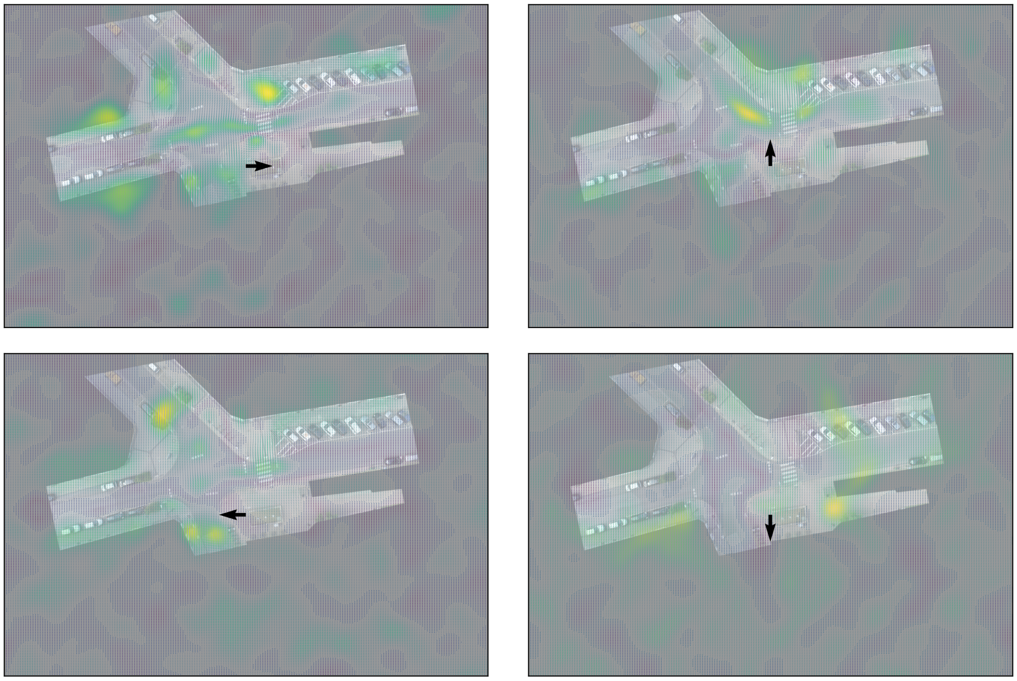
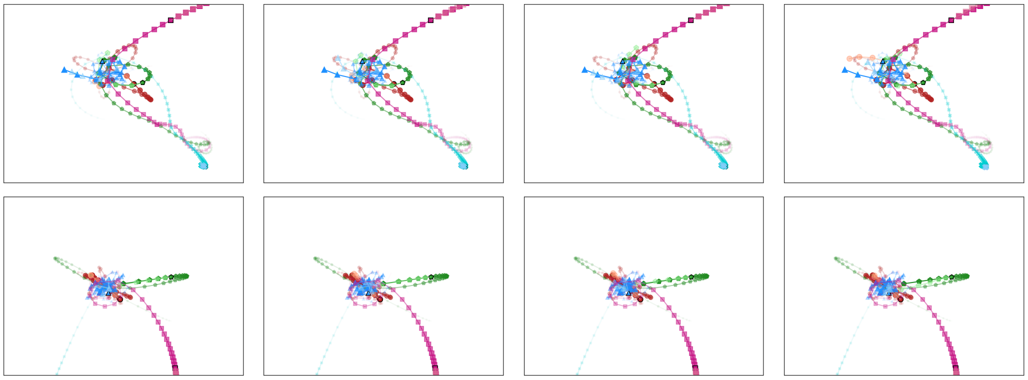


Figure 15: Discovered field on inD [4]. For simplicity, we only visualize the field for discrete input orientations in $\mathrm{C}_4 = \left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}$. *Best viewed in color.*

D.3 2D gravity

Figure 16 shows qualitative results on the 2D gravitational n-body problem.



(a) Aether (b) G-LoCS (c) LoCS (d) dNRI

Figure 16: Predictions on gravity. Lighter colors indicate predictions, and darker colors indicate the groundtruth. Predictions start where markers have black edges. Markers get bigger as trajectories evolve. *Best viewed in color.*

D.3.1 Discovered 2D gravitational fields

Figure 17 shows examples of discovered fields compared to the groundtruth ones.

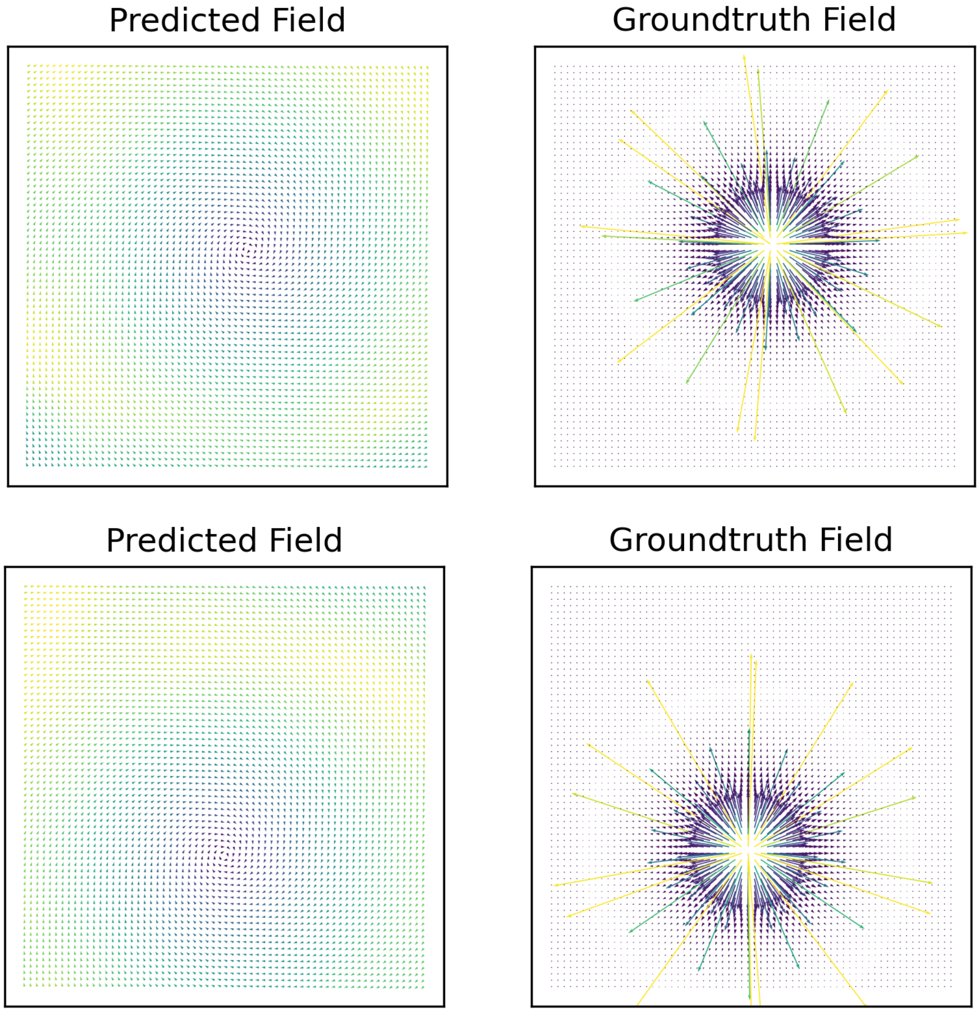


Figure 17: Learned *dynamic* fields (left) in 2D gravitational field setting vs groundtruth (right).

E Quantitative results

In all settings, we report the *total errors*, *i.e.* the mean squared errors of positions and velocities over time, $E(t) = \frac{1}{ND}\sum_{n=1}^{N}\|\mathbf{x}_n^t - \hat{\mathbf{x}}_n^t\|_2^2$. Following Kofinas et al. [24], we also separately report the L_2 norm *position errors*, $E_p(t) = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{p}_n^t - \hat{\mathbf{p}}_n^t\|_2$, and *velocity errors*, $E_u(t) = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{u}_n^t - \hat{\mathbf{u}}_n^t\|_2$.

E.1 Electrostatic field

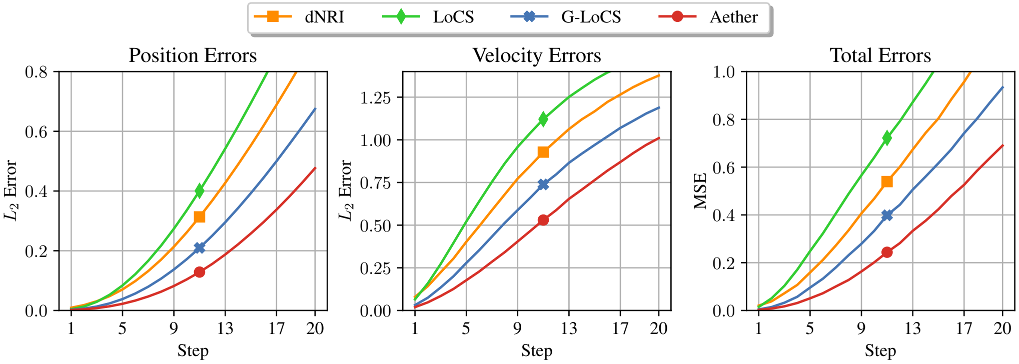


Figure 18: Results in the electrostatic field setting.

E.2 InD

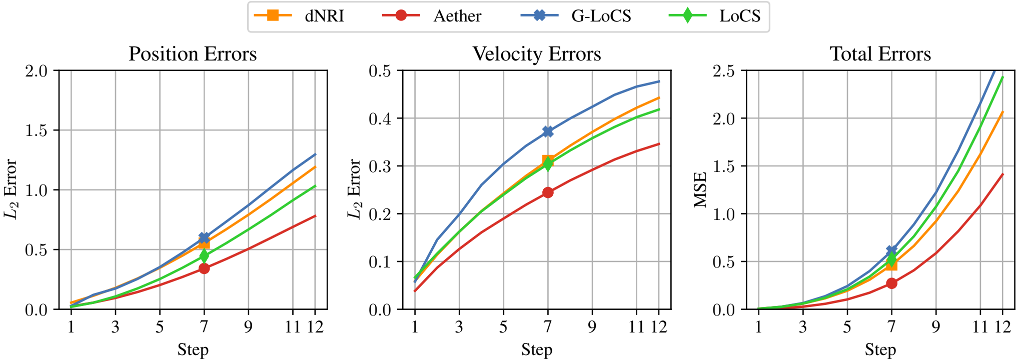


Figure 19: Results in inD.

E.3 Gravity

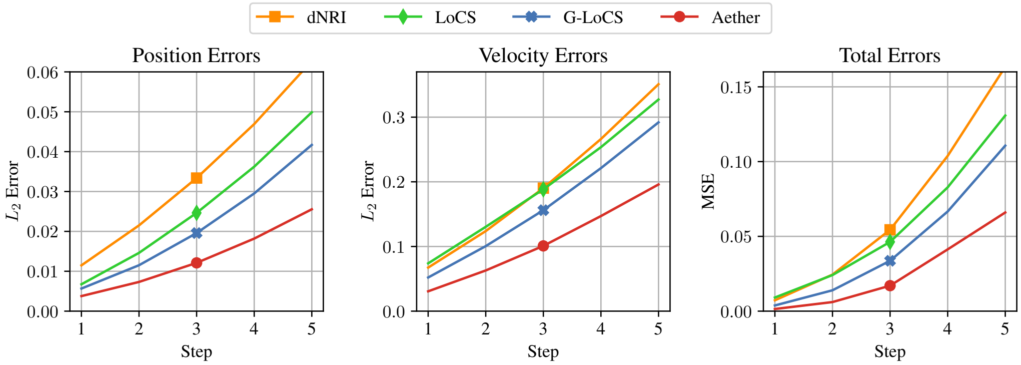


Figure 20: Results in the dynamic gravitational field setting.

E.4 2D gravity

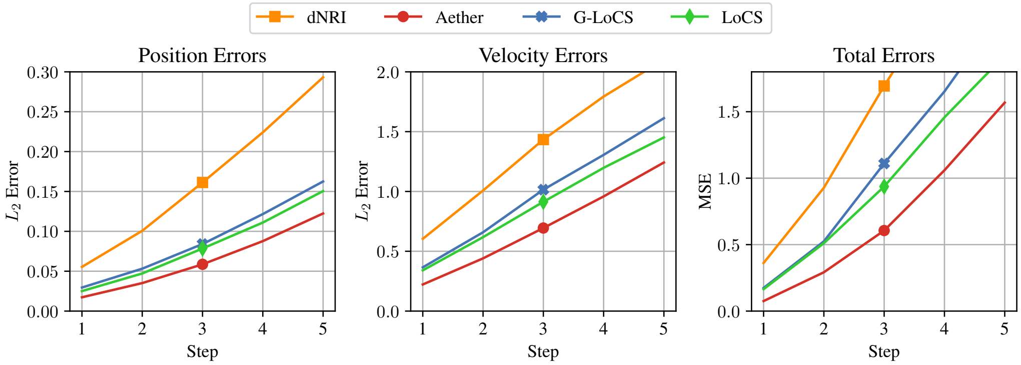


Figure 21: Results in the dynamic 2D gravitational field setting.

E.5 Significance of the discovered field

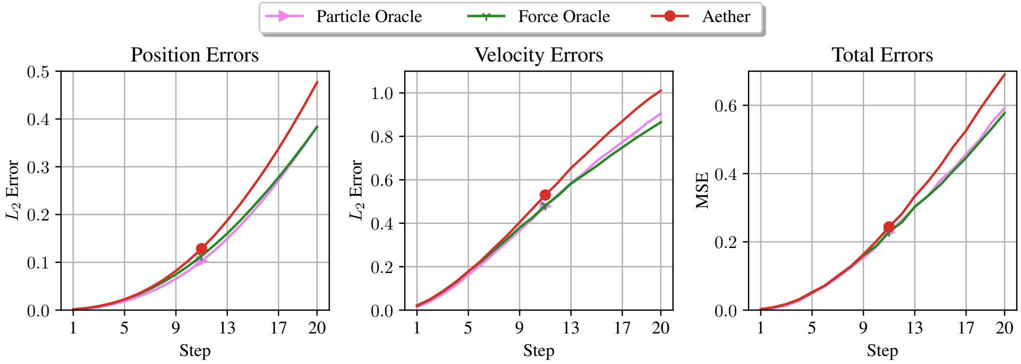


Figure 22: Ablation study on the significance of the discovered field. Results in the electrostatic field setting.