Paper Summary Mip-NeRF

Abstract

Neural Fields: Coordinate-based neural networks that parameterized physical properties of scenes or objects across space and time.

Part I: Focusing on neural fields techniques by identifying common components of the neural field methods, including different generalization, representation, forward map, architecture, and manipulation methods.

Part II: Focusing on applications of neural fields to different problems.

1. Introduction

Fields are widely used to continuously parameterize an underlying physical quantity of an object or scene over space and time. For instance, fields can be used to *visualize physical phenomena* [Sab88], *compute image gradients* [SK97], *compute collisions* [OF03], or represent shapes via *constructive solid geometry (CSG)* [Eval15].

Coordinate-based neural networks that represent a field, we refer to these methods as **neural fields**.

1.1 Background

Definition 1: A *field* is a varying physical quantity of spatial and temporal coordinates.

We can represent a field as a continuous function $f : U \to R^n$ that is defined for any coordinate x in space and time $U \in \{R^m, t\}$ and that produces quantity q.

Universal Approximation Theorem [KA03].

Neural Fields: A *neural network* can represent any field through its parameters Θ. Thus:

Definitions 2: A *neural field* is a field that is parameterized fully or in part by a neural network.

Within visual computing, **neural fields** have been called *implicit neural representations*, *neural implicits*, or *coordinate-based neural networks*.

Part I. Neural Field Techniques

A typical neural fields algorithm in visual computing proceeds as follows: Across space-time, we **sample coordinates** and feed them into a **neural networks** to produce physical field values. Field values are samples from the desired **reconstruction domain** of our problem. Then, we apply a **forward map** to relate the reconstruction to the sensor domain, where supervision is available. Finally, we calculate the reconstruction error or **loss** that guides the NN optimization process by comparing the reconstructed signal to the sensor measurement.

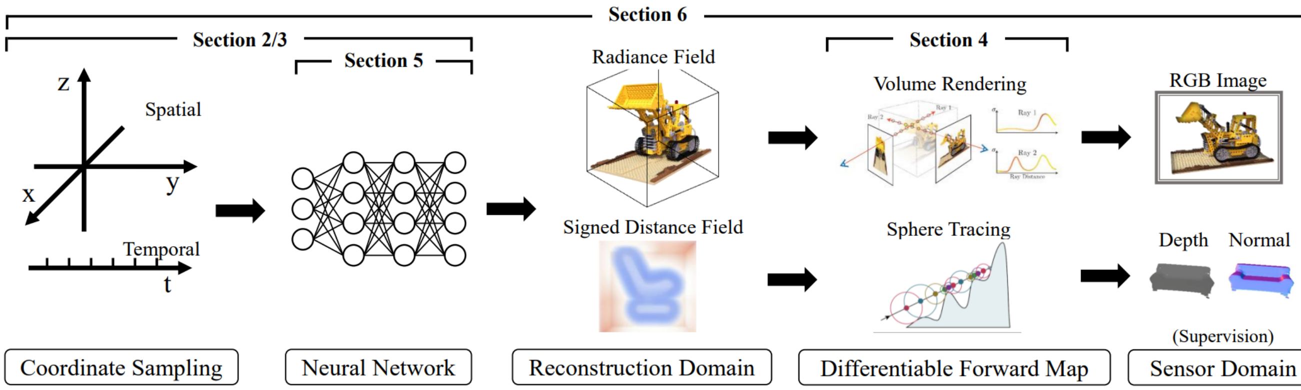

Figure 3: ***A typical feed-forward neural field algorithm.*** *Spatiotemporal coordinates are fed into a neural network which predicts values in the reconstruct a domain. Then, this domain is mapped to the sensor domain where sensor measurements are available as supervision.*

Class and Section	**Problems Addressed**
Generalization (Section 2)	Inverse problems, ill-posed problems, edit ability, symmetries.
Hybrid Representations (Section 3)	Computation & memory efficiency, representation capacity, edit ability.
Forward Maps (Section 4)	Inverse problems.
Network Architecture (Section 5)	Spectral bias, integration & derivatives.
Manipulating Neural Fields (Section 6)	Edit ability, constraints, regularization.

Table 2: *The five classes of techniques in the neural field toolbox each addresses problems that arise in learning, inference, and control.*

2. Generalization

A neural network expresses a prior via the function space of its architecture and parameters Θ, and generalization is influenced by the **inductive bias** of this function space (Section 5). Suppose we wish to **vary** our **prior** over the estimation of the plausible 3D shape depending on characteristics of the point cloud, or more broadly over characteristics of a reconstruction domain, we can **condition** our neural field.

2.1 Conditional Neural Fields

A conditioning latent variable **z**. Observation-specific info can be encoded in the conditioning latent variable **z**, while shared info can be encoded in the neural field parameters.

Disentanglement: useful information separation.

There are three components in a conditional neural field:

1. An encoder or inference function ε that outputs the conditioning latent variable (code) z given an observation $O : \varepsilon(O) = z$

2. $*$ A mapping function Ψ b/t $\mathbf{z}$ and neural field parameters $\Theta : \Psi(\mathbf{z}) = \Theta$

3. The neural field itself Φ.

The encoder ε finds the most probable $\mathbf{z}$ given the observations O : $arg\ max_z\ P(\mathbf{z}|O)$. The decoder maximizes the inverse conditional probability to find the most probable O given $\mathbf{z}$: $arg\ max_O\ P(O|\mathbf{z})$.

2.1.1 Encoding the conditioning Variable $\mathbf{z}$

Auto-decoders:

The latent codes $\mathbf{z}$ are free variables to be directly optimized. Every training observation is initialized with its own latent code $\mathbf{z}_i$. To optimize a particular latent code $\mathbf{z}_i$, we map it to neural field parameters Θ via the mapping function Ψ, $\Psi(\mathbf{z}) = \Theta$. Compute a reconstruction loss, and back-propagate that loss to $\mathbf{z}_i$ to take a gradient descent step. At training time, the per-observation latent codes $\mathbf{z}_i$, the neural field Φ, and the conditioning function Ψ are jointly optimized.

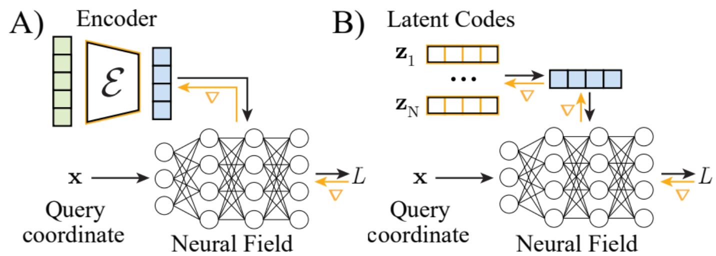

Figure 4: ***Inference: Encoding vs. Auto-decoding.*** *(A) In amortized or encoder-based inference, encoder $\mathcal{E}$ maps observations $\mathcal{O}$ to latent variable $\mathbf{z}$. Parameters of $\mathcal{E}$ are jointly optimized with the conditional neural field. (B) Auto-decoding lacks an encoder, and each neural field is represented by a* separately-optimized *latent code $\mathbf{z}_i$, which are jointly optimized with the conditional neural field. Gradient flow displayed in orange.*

How to optimize the latent code $\hat{\mathbf{z}}$ during testing still need to be clear.