Neural Fields in Visual Computing and Beyond

- Neural Fields in Visual Computing and Beyond
- **[v4]** Tue, 5 Apr 2022 18:19:13 UTC
- Coordinate-based neural networks that represent a field is referred as *neural fields*
- An inverse problem in science is **the process of calculating from a set of observations the causal factors that produced them**
- ill-posed problem

1. Intro

> field in physics: A *field* is a quantity defined for all spatial and/or temporal coordinates (spacetime coordinates)
>
> baidu: 场是一个以时空为变量的物理量

- We can represent a field as a function mapping a coordinate **x** to a quantity, which is typically a scalar or vector.
- Why we use neural fields?
  - the underlying field generation process may not have a known analytic form. Thus, functions may be described by parameters Θ that are hand crafted, optimized, or learned. A field producing quantity q at location x maybe described as $q = \Phi(x; \Theta)$
- sampled functions are often indexed by discrete values - *Nyquist* sampling rate
  - camera pixels
  - voxels
  - discretized level sets
- Why we use MLP?
  - universal approximation theorem: A multi-layer perceptron (MLP) neural network can approximate any function through their learned parameters

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

- since MLP is *a universal approximator*, we can use MLP to represent a field
- Neural fields are both continuous and adaptive by construction
- Neural fields help to resolve complexity problem by using their parameters only where field detail is present

Part I. Neural Field Techniques

> A typical algorithm:
>
> - Across space-time, we **sample** coordinates and **feed** them into a neural network to produce *field quantities*. The field quantities are samples from the desired

> *reconstruction domain* of our problem
> - We apply a *forward map* to relate the reconstruction to the *sensor domain* (e.g. RGB image), where supervision is available
> - We calculate the reconstruction *error* or *loss* that guides the neural network optimization process by comparing the reconstructed signal to the sensor measurement

5 techniques:

- prior learning and conditioning
  - aid reconstruction from incomplete sensor signals
  - inverse problems, ill-posed problems, edit ability, symmetries
- hybrid representations
  - improve memory, computation, and neural network efficiency
  - computation & memory efficiency, representation capacity, edit ability
- forward maps
  - supervise reconstruction
  - inverse problems
- network architecture
  - overcome neural network spectral biases (blurriness) and efficiently compute derivatives and integrals
  - spectral bias, integration and derivatives
- manipulate neural field
  - add constraints and regularizations, and to achieve editable representations
  - edit ability, constraints, regularization

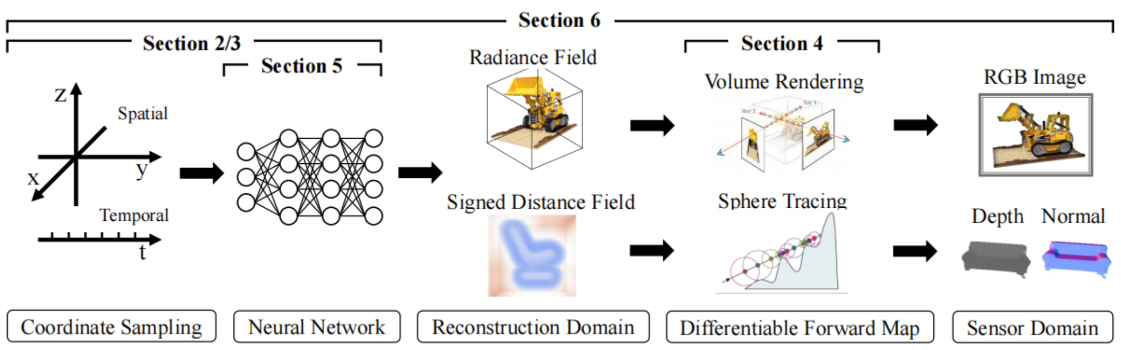


Figure 3: **A typical feed-forward neural field algorithm.** Spatiotemporal coordinates are fed into a neural network which predicts values in the reconstruct a domain. Then, this domain is mapped to the sensor domain where sensor measurements are available as supervision. Figures adapted from [MST*20, LZP*20].

2 Prior Learning and Conditioning

- Why we talk about prior?
  - we need a function/model to implement a task/reconstruction, we need to learn a prior to achieve this
  - prior should reflect your *current beliefs* (either from previous data or from purely subjective sources) about the parameters before you have observed the data
- Why we talk about conditioning?
  - we want the prior to adapt to varying conditions
  - For neural fields, this is accomplished by conditioning the neural field on a set of latent variables **z** that encode the properties of a specific field

2.1. Conditional Neural Fields

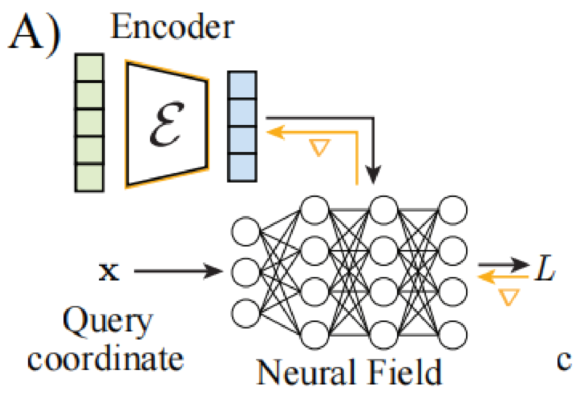
> A conditional neural field lets us **vary** the field by varying a set of latent variables **z**

- latent variables could be:
  - samples from an arbitrary distribution
  - semantic variables describing shape, type,size, color
  - come from an encoding of other data types such as audio data
- **z** is typically a low-dimensional vector, and is often referred to as a *latent code* or *feature code*
- instance-specific information can then be encoded in the conditioning latent variable **z**
- shared information can be encoded in the neural field parameters

2.1.1. Encoding the Conditioning Variable z

Feed-forward Encoders/Amortized Inference/Auto Encoders

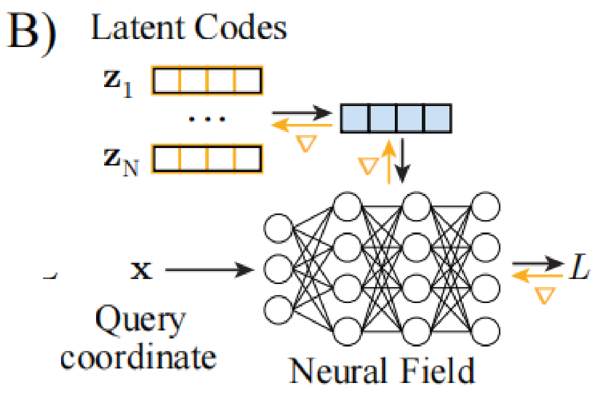
- the conditioning latent code $\mathbf{z} = \mathcal{E}(\mathcal{O})$ is generated via an encoder $\mathcal{E}$, typically a neural network
  - parameters in $\mathcal{E}$ can encode priors that can be pre-trained on data or auxiliary tasks
  - encoder maps observation $\mathcal{O}$ to latent variable
  - parameters of $\mathcal{E}$ are jointly optimized with the conditional neural field
- the decoder is the neural field that is conditioned by the latent code
- fast since both encoding and decoding require only a single forward pass



Auto-decoders

- an auto-decoder directly accepts a latent vector as an input
- allows modeling multiple SDFs with a single neural network
- encoding happens through stochastic optimization
  - the latent code $\mathbf{z} = argmin_z \mathcal{L}(\mathbf{z}, \Theta)$ is obtained by minimizing some loss function $\mathcal{L}$
  - optimization acts as encoder $\mathcal{E}$ to map observations $\mathcal{O}$ to latent variables **z**
- each model/data point is represented by a *separately-optimized* latent code $\mathbf{z}_i$, which will be optimized together during training
- the decoder is explicitly defined
  - during inference/reconstruction, decoder weights are fixed and an optimal latent variable $\hat{z}$ is estimated/selected $(argmin)$

    - because of estimation/optimization, auto-decoding is significantly slower than those requiring only one pass
- auto-decoding does not introduce additional parameters and does not make assumptions about observations $\mathcal{O}$
- an auto-decoder can ingest tuples of pixel coordinates and colors independently of their spatial arrangement



Hybrid Approaches

- initialize **z** with a forward pass of a parametric encoder $\mathcal{E}$
- optimize **z** iteratively via auto-decoding

2.1.2. Global and Local Conditioning

Global Conditioning

- the neural network is *coordinate-independent*
- a single latent code **z** determines the entire neural field Φ across coordinate values
- desirable in representation learning, when we want **z** to be *compact*—to encode as much information as possible
- certain signals are not amenable to the learning of a single, continuous space of latent variables **z**
    - to model a room with many furniture, it is hard to use a single global latent code to represent the whole distribution and geometry of each object

Local Conditioning

- the neural network is *coordinate-dependent*
- the latent code is generated by $\mathbf{z} = g(\mathbf{x})$
    - g is a discrete data structure
- As each latent code **z** only has to encode information about its local neighborhood, it consequently does not need to store information about the configuration anymore
    - split a room into small 3D cubes, and in each cube store a latent code that describes the geometry in that cube
- an encoder only has to encode local properties into latent variables **z**

- This can leverage encoder properties such as translation equivariance, granting better out-of-distribution generalization

Hybrid Approaches

- for human face images, to attempt to disentangle a property that is shared across instances (like hair color) from another that is region specific (like skin wrinkle)

2.1.3. Mapping z to Neural Field Parameters Θ

> Conditional neural fields are expressed as predicting *a subset of the parameters* Θ of a neural field Φ_Θ via a function Ψ

Conditioning by Concatenation/Concatenation-based Condition

- directly concatenate inputs $\mathbf{x}$ with $\mathbf{z}$
  - a neural field $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ becomes $\Phi : \mathbb{R}^{2+n} \to \mathbb{R}^3$ after concatenation
- conditioning via concatenation is equivalent to defining an affine function $\Psi(\mathbf{z}) = \mathbf{b}$ that maps latent codes $\mathbf{z}$ to the vector of biases $\mathbf{b}$ of the first layer of Φ
  - in the case of conditioning via concatenation, the subset of parameters that is predicted by Ψ is only the biases of the first layer of the neural network

    > Why is a bias? Refer to the supplementary material of https://arxiv.org/abs/2006.09662v1

  - Ψ is parameterized as a simple affine mapping
  - conditioning via concatenation is a special case of a hypernetwork

Hypernetworks

- parameterize the function Ψ as a neural network that takes the latent code $\mathbf{z}$ as input and outputs neural field parameters Θ via a forward pass
- a general form of conditioning because every other form of conditioning may be obtained from a hypernetwork:
  - by outputting only subsets of parameters Θ
  - by factorizing parameters of Φ via low-rank approximations
  - via additional scales and biases
  - by varying Ψ architectures such as using only a single linear layer

FiLM and Other Conditioning

- predicting feature-wise transformations
- steps:
  - use a network Ψ to predict a per-layer (and potentially per-neuron) scale γ and bias β vector from latent code: $\Psi(\mathbf{z}) = \{\gamma, \beta\}$
  - input $\mathbf{x}_i$ to layer Φ_i is transformed as $\Phi_i = \gamma_i(\mathbf{z}) \bigodot \mathbf{x}_i + \beta_i(\mathbf{z})$

2.2. Gradient-based Meta-learning

- all neural fields in target distribution are viewed as *specializations* of an underlying *meta-network* with parameters θ

> What is meta-learning: https://blog.csdn.net/qq_23225317/article/details/104016826
>
> The goal of few-shot meta-learning is to train a model that can quickly adapt to a new task using only a few data points and training iteration

- individual instances are obtained from fitting this meta-network to a set of observations $\mathcal{O}$, minimizing a reconstruction loss $\mathcal{L}$ in a small number of gradient descent steps with step size λ: $\Theta^{j+1} = \Theta^j - \lambda \nabla \sum_{\mathcal{O}} \mathcal{L}(\Phi(\mathcal{O}; \Theta_i^j)), \quad \Theta_i^0 = \theta.$
    - At training time, a batch of tasks is drawn,with each task split into 'context' and 'target' observation
    - The model adapts itself using the 'context' observations (train set)
    - The model makes predictions on the 'target' observation (validation set)
    - The model parameters are optimized to minimize the losses on all target observations in the training set, over all tasks
- In a conditional neural field, the prior is expressed via the parameters of Ψ that enforce that the parameters of Φ lie in a low dimensional space as defined by latent variable $\mathbf{z}$
- In gradient-based meta-learning, the prior is expressed by constraining the optimization to not move the neural field parameters Θ too far away from the parameters of the meta-network θ

> Q: not too far away reflects the adaption ability?
>
> Optimizing not only the embeddings but also the net work weights regularized to be close to the original weights can better resolve ambiguities in unobserved regions

- gradient based meta-learning enables fast inference, as only a few gradient descent steps are required to obtain Θ

3. Hybrid Representations

Benefits of discrete data structures:

- They typically reduce computation
- They also allow for more efficient use of network capacity, since large MLP networks have diminishing returns problem in representation capacity
- When representing geometry, discrete structures allow for empty space skipping, and so accelerate rendering
- Discrete structures are also suitable for simulation

2 general approaches to spatial decomposition:

- network tiling
- embedding

Network Tiling

- A collection of separate (usually small) *neural fields* Φ are tiled across the input coordinate space, covering disjoint regions
- The network architecture is shared, but their parameters are distinct for each disjoint region
- $\mathbf{q} = \Phi(\mathbf{x}, \Theta) = \Phi(\mathbf{x}, g(\mathbf{x}))$

Embedding

- store *latent variable* $\mathbf{z}$ in the data structure
- *local* embedding $\mathbf{z} = g(\mathbf{x})$
- $\mathbf{q} = \Phi(\mathbf{x}, \Theta) = \Phi(\mathbf{x}, \Psi(\mathbf{x})) = \Phi(\mathbf{x}, \Psi(g(\mathbf{x}))$
- tiling is a special case of embedding where $\Psi()$ is the identity function

3.1. Defining *g* and its Common Forms

g maps coordinates to quantities using a sum of Dirac delta functions:

$$g(\mathbf{x}) = \alpha_0\delta(\mathbf{x}_0 - \mathbf{x}) + \alpha_1\delta(\mathbf{x}_1 - \mathbf{x}) + \ldots + \alpha_n\delta(\mathbf{x}_n - \mathbf{x}) \qquad (1)$$

- coefficients α_i are the quantities stored at coordinates $\mathbf{x}_i$
- for network tiling, α_i is an entire set of network parameters
- for embedding, α_i is latent variable
- The discrete data structure *g* may use an *interpolation scheme* to define *g* outside the coordinates $\mathbf{x}_i$ of the Dirac deltas, such as nearest neighbor, linear, or cubic interpolation

3.1.1. Regular Grids

- pros
  - simple to index
  - simple to apply standard signal processing techniques
- cons
  - poor memory scaling in high dimensions
  - Nyquist-Shannon theorem requires dense sampling for high-frequency signals
  - solutions:
    - be adaptive or sparse to focus the capacity around higher frequency regions
    - be implemented with data structures like hierarchical trees and textures
- *Grid tiling* discretizes the coordinate domain with a grid and define each local region with smaller neural networks
  - pros
    - help learn larger scale signals
    - make inference faster
    - be suitable for parallel computing
  - cons
    - may increase overfitting given sparse training data
    - possible tile boundary artifacts
      - can be reduced by network parameter interpolation
- *Grids of embeddings*

    - pros
        - similarly model larger-scale signals
        - enable the use of small neural networks
        - benefit from interpolation
        - can be generated from other neural fields

3.1.2. Irregular Grids

- pros
    - avoid the Nyquist-Shannon sampling limit
    - can be morphed to adaptively increase capacity in complex data regions
    - may declare connectivity between coordinates explicitly or implicitly
    - may be organized into hierarchies
- cons
    - irregular sampling pattern

Point Clouds

- a collection of sparse discrete coordinates
- each location can hold an embedding or a network (tiling)
- point clouds can volumetrically define regions through *Voronoi cells/Voronoi diagrams* via nearest neighbor interpolation
- for continuous interpolation, we can use Voronoi cells with natural neighbor interpolation or soft Voronoi interpolation

> 关于维诺的这几个插值都没怎么找到资料……也没怎么看懂那个论文……

Object-centric Representations

- a collection of points but where each has an orientation and a bounding box or volume
- neural field parameters are stored at each point or at each vertex of the bounding volume
- store embeddings or a networks (tiling)

Mesh

- for triangle meshes, embeddings can be stored on vertices and interpolated with barycentric interpolation
- for complex polygons, mean-value coordinates and harmonic coordinates are options

    > 调和坐标有点太复杂了……

4. Forward Maps

- *reconstruction domain*: how we represent the world
- *sensor domain*: how we observe the world
- Solving an *inverse problem* recovers the reconstruction from observations obtained from sensors, i.e., finding the parameters Θ of a neural field Φ given observations from the sensor Ω

> We represent the (unknown) reconstruction as a neural field $\Phi : \mathcal{X} \to \mathcal{Y}$ that maps world coordinates $\mathbf{x}_{recon} \in \mathcal{X}$ to quantities $\mathbf{y}_{recon} \in \mathcal{Y}$. A sensor observation is often also a field $\Omega : \mathcal{S} \to \mathcal{T}$ that maps sensor coordinates $\mathbf{x}_{sens} \in \mathcal{S}$ to measurements $\mathbf{t}_{sens} \in \mathcal{T}$.

> *Forward maps* relate/map reconstruction domains to sensor domains. A forward map is an operator $F : (\mathcal{X} \to \mathcal{Y}) \to (\mathcal{S} \to \mathcal{T})$ ($F : \Phi \to \Omega$), which is a mapping of functions.
>
> - *parameter differentiable* if for $\mathbf{y} = F(\Phi(\mathbf{x}))$ we can calculate $\frac{\partial \mathbf{y}}{\partial \theta}$
> - *input differentiable* if for $\mathbf{y} = F(\Phi(\mathbf{x}))$ we can calculate $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

The optimization problem to solve:

$$\arg\min_{\Theta} \int_{(\mathbf{x}_{recon}, \mathbf{x}_{sens}) \in (\mathcal{X}, \mathcal{S})} \| F(\Phi(\mathbf{x}_{recon})) - \Omega(\mathbf{x}_{sens}) \| \tag{2}$$

4.1. Rendering

> We define a *renderer* as a forward map which **converts** some neural field representation of 3D shape and appearance to an image.
>
> Renderers often utilize a *raytracer* which takes as input a ray origin and a ray direction, and returns some information about the neural field.

4.1.1. Ray-surface Intersection

For those representing a shape's surface

ray-surface intersection amounts to a root finding algorithm

- ray marching
  - numerical methods may fail to converge when the surface is too thin
  - alleviation: stochastically varying the step length and using supersampling
  - interval arithmetic can be used to guarantee convergence at the cost of iteration count
- sphere tracing
  - can efficiently find intersections with guaranteed convergence if the surface is Lipschitz-bounded
- segment tracing
  - can further speed up convergence at the cost of additional segment arithmetic computation at each step

  > Refer to the visual interpretation at about 6:50

- Above methods are all differentiable, but naively back propagating through the iterative algorithm is computationally expensive.
- If a hybrid representation is used, we can exploit a bounding volume hierarchy to speed up raytracing.
- In some cases, the data structure can also be rasterized onto the image to reduce the number of rays

4.1.2. Surface Shading and Lighting

> Once the ray-surface intersection point has been retrieved, we can calculate the *radiance contribution* from the *point towards the camera*. This is done with a *bidirectional scattering distribution function* (BSDF) which can be differentiable or be parameterized as a neural field.

fatser: approximations of incident lighting

- cubemaps
- spherical basis functions
- neural networks

4.1.3. Volume Rendering

> Instead of *Kajiya's* rendering equation, volume rendering uses the *volume rendering integral* based on the equations of radiative transfer, for which the integral can be **numerically approximated** using quadrature (in practice, stochastic ray marching).

- under the assumptions of exponential transmittance, integration can be performed with a simple cumulative sum of samples across a ray
- non-exponential formulations exist and some are not physically accurate but work as an approximation
- one important factor: number of samples
    - higher sample count: more accurate model but more computational cost and memory

4.1.4. Hybrid Volume and Surface Rendering

- Volume rendering is typically under-constrained due to the **stochastic samples taken on intervals**, resulting in noise near surfaces.
- Surface rendering **only provides gradients at the object surface**, which do not smoothly propagate across the spatial domain.

4.2. Physics-informed Neural Networks

> *Physics informed neural networks* (PINNs) use *learning bias* which supervises boundary and initial values (from an incomplete simulation or observations) using a loss, and the rest of space by sampling or regularizing with equations of physics, typically partial differential equations (PDEs)

- the PINN paradigm is often seen with signed distance functions
- the boundary values of SDF are the point cloud $\mathcal{X}$ which correspond to the 0-level set

4.3. Identity Mapping Function

In some applications, the sensor domain may be **the same** as the reconstruction domain. In these cases, the forward model is the identity mapping function.

5. Network Architecture

5.1. Overcoming Spectral Bias

> Real-world signals are complex, making it challenging for neural networks to achieve *high fidelity*
>
> Neural networks are biased to fit functions with *low spatial frequency*

Positional Encoding

- > The coordinate input of the neural network may be transformed by a *position encoding* $\gamma : \mathbb{R}^n \to \mathbb{R}^m$, which is a set of scalar functions $\gamma_i : \mathbb{R}^n \to \mathbb{R}$ that maps a coordinate vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ to a vector of embedded coordinates:
  >
  > $$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \gamma_2(\mathbf{x}), \ldots \gamma_m(\mathbf{x})] \tag{3}$$
- $\sin()$ and $\cos()$ are widely used
- positional encodings can thus also be seen as controlling the interpolation properties of a neural field
- the choice of the frequency of γ_i influences the learning ability and reconstruction of the network

> 个人理解：其实positional encoding也就是换了个数学表达来表示一个坐标，这个数学表达往往具有多级导数等性质，在进行反向传播、求偏微分、插值等操作时可能会更具可操作性

Activation Functions

- An alternative approach to enable the fitting of high-frequency functions is to replace standard, monotonic nonlinearities with periodic nonlinearities
- It has been pointed out that positional encoding with Fourier features is equivalent to periodic nonlinearities with one hidden layer as the first neural network layer

5.2. Integration and Derivatives

Derivatives

- A key benefit of neural fields is that they are highly flexible general function approximators whose derivatives $\nabla_x \Phi(\mathbf{x})$ are easily obtained via automatic differentiation
- Specifically, the derivatives of the network must be nontrivial (*i.e.,* nonzero) to the degree of the PDE
  - to parameterize solutions of the wave equation — a second order PDE — the second-order derivatives of the neural field must be nontrivial
  - This places restrictions on the activation function used in the network

Integrals

- possible to sample the neural field and approximate the integral using numerical quadrature
- automatic integration is **not** well-explored

    - seminal work: directly parameterize the *antiderivative* of the field as a neural field

6. Manipulating Neural Fields

> Neural fields have limited tools for *editing* and *manipulation*, which significantly limits their use cases

Neural fields can be manipulated by:

- transform spatial or temporal coordinate inputs
- directly manipulating the latent features or the learned network's weights

6.1. Input Coordinate Remapping

6.1.1. Spatial Transformation via Explicit Geometry

- For object modeling problems, structural priors are often available in the form of *bounding boxes* and *coarse explicit geometry*
- For articulated objects, the *kinematic chain* offers explicit control over object geometry through its *joint angles*
    - Effective for a small number of joints, but may introduce spurious correlations in case of long kinematic chains such as human body
    - To avoid the above issue, we can represent target shapes as the *composition of local neural fields*
        - each local field is independently modeled, often producing artifacts around joints with unobserved joint transformations
    - To avoid the above issue, an alternative is to warp an observation space into a canonical space, where the reconstruction is defined, using the articulation of a template model

6.1.2. Spatial Transformation via Neural Fields

> Modeling general dynamic scenes requires flexible representations that can handle arbitrary transformations.
>
> Learning neural fields for spatial transformations is a highly under-constrained problem without 3D supervision. This motivates the use of *regularization loss terms* based on physical intuitions.

Smoothness

- The first derivative of warp fields with respect to spatiotemporal coordinates should be *smooth*, assuming no sudden movements
- This is necessary to constrain unobserved regions

Sparsity

- 3D scenes generally contain large empty space.
- *Enforcing sparsity* of the predicted motion fields avoids suboptimal local minima

Cycle Consistency

- If a representation provides both forward and backward warping, we can employ *cycle-consistency loss*

Auxiliary Image-space Loss

- Image-space information such as optical flows and depth maps can also be used in auxiliary loss functions

6.1.3. Temporal Re-mapping

Conditioning the neural field on temporal coordinates allows time editing such as speed-up/slow-down $\Phi(a \cdot t)$, offset $\Phi(t+b)$, and reversal $\Phi(-t)$

6.2. Editing via Network Parameters

Latent Code Interpolation/Swapping

- For neural fields conditioned on latent codes, interpolation or sampling in the latent space can change properties of the representation

Latent Code/Network Parameters Fine-Tuning

- After pretraining, we can fine-tune parameters to fit new, edited observations at test time
- The neural field is coupled to the explicit supervision via differentiable forward maps

Editing via Hypernetworks

- Hypernetworks can learn to map a new statistical distribution to a pre-trained neural field, by replacing its parameters.

Part II. Applications of Neural Fields

Problems in visual computing:

- 3D shape and appearance reconstruction
- novel view synthesis
- human modeling
- medical imaging
- ...

> 这一部分开始主要是概述各种论文的成果。

7. 3D Scene Reconstruction

> *Reconstruction* is the solution to an inverse problem that maps available observations to a representation.

For 3D, available observations are **discrete** (due to sensors), often **sparse** (few images), **incomplete** (partial point clouds), residing in a **lower dimension** (2D images), or **lack vital topological information** (point clouds).

7.1. Reconstruction of 3D Shape and Appearance

7.1.1 Reconstructing 3D Scenes with 3D Supervision

- Example of 3D supervision: point clouds
- Work on neural fields for geometry reconstruction often focuses on learned priors for reconstruction
- listing a lot of works...

7.1.2 Differentiable Rendering

> Differentiable Rendering allow reconstruction of 3D neural fields representing shape and/or appearance *given only 2D images*, instead of 3D supervision

- NeRF combined *volume rendering* with a *single ReLU MLP*, *parameterizing a monolithic neural field*, and added *positional encodings*. By fitting a single neural field to *a large number of images of a single 3D scene*, this achieved photo-realistic novel view synthesis from only 2D images of arbitrary scenes for the first time
    - Nerf++ improves representation of unbounded 3D scenes via an *inverted-sphere background parameterization*
    - DoNeRF proposes to jointly train a NeRF and a ray depth estimator for fewer samples and faster rendering at test time.
    - Mip-NeRF proposes to control the frequency of the positional encoding for multi-scale resolution control.
    - NeRF−−, BARF and iNeRF propose to back-propagate into camera parameters to enable camera pose estimation given a reasonable initialization.
    - PixelNeRF and GRF perform prior-based reconstruction by extracting features from images with a fully convolutional CNN, and, when querying a 3D point, projecting it on the image plane to use image features for local conditioning-via concatenation
- While volume rendering has better convergence properties than surface rendering and enables photorealistic novel view synthesis, the quality of the reconstructed geometry is worse, due to the lack of an implicit, watertight surface representation
    - IDR leverages an *SDF parameterization* of geometry, a sphere-tracing based surface renderer, and positional encodings to enable high-quality geometry reconstruction
    - Neural Lumigraph Rendering distills an IDR model into a Lumigraph after rendering to enable fast novel view synthesis at test time
    - UNISURF, NeuS and VolSDF propose to relate the occupancy function of a volume to its volume density, thereby combining volume rendering and surface rendering, leading to improved rendering results and better geometry reconstruction
- Ray marching requires many samples along a ray to faithfully render complex 3D scenes. Even for relatively simple scenes, *rendering requires hundreds or even thousands of evaluations of the neural scene representation per ray*
    - Light Field Networks parameterize the *light field* of a scene, which maps every ray to a radiance value, but requiring a multi-view consistency prior

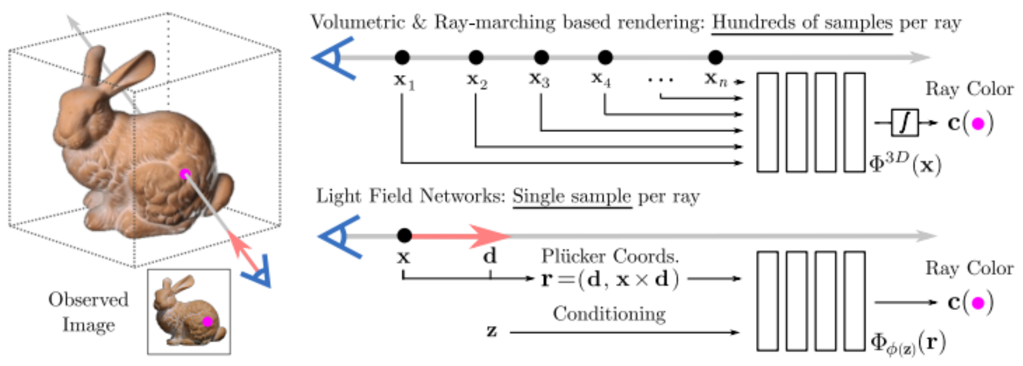
- 
Volumetric & Ray-marching based rendering: Hundreds of samples per ray
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\cdots$ $\mathbf{x}_n$
Ray Color
$\mathbf{c}(\bullet)$
$\Phi^{3D}(\mathbf{x})$
Light Field Networks: Single sample per ray
$\mathbf{x}$ $\mathbf{d}$
Plücker Coords.
$\mathbf{r} = (\mathbf{d}, \mathbf{x} \times \mathbf{d})$
Conditioning
$\mathbf{z}$
Ray Color
$\mathbf{c}(\bullet)$
$\Phi_{\phi(\mathbf{z})}(\mathbf{r})$
Observed Image

Figure 11: Instead of encoding a 3D scene with a 3D neural field that requires hundreds of samples along a ray to render, appearance & geometry may be encoded as a neural light field directly mapping an oriented ray to a color, enabling rendering with a single sample per ray, but requiring a multi-view consistency prior. Figure adapted from [SRF*21].

- To prevent overfitting to the input views and allow view synthesis, we can learn multi-view consistency via global conditioning, hypernetworks, and inference via auto-decoding, or through ray embedding spaces
  - NeuLF parameterizes forward-facing scenes via light fields, overfitting on single scenes, and addresses multi-view consistency via enforcing similarity of randomly sampled views with the context views in the Fourier domain
  - NeX parameterizes a set of multi-plane images as a 2D neural field, where the network inputs are pixel locations. Rendering is computationally efficient without ray marching

> 以上的论文并不是综述里提及的全部论文......
>
> 个人总结一下问题：
>
> - 速度慢
> - 只适用于静态场景
> - 几何体重建得不够好
>   - 已有的方案是用SDF或者occupancy function
> - ray marching速度太慢了
>   - 参数化light field，减少生成novel views时采样的次数（事实上这正是后来的NeRV做的事情）

7.2. Reconstruction of Scene Material and Lighting

> The goal of *material reconstruction* is to **estimate** the material properties of a surface or participating media from sparse measurements such as images.
>
> For opaque surfaces, this may be the parameters of a *bidirection scattering distribution function* (BSDF).
>
> For participating media, this may be phase functions.

- The forward modeling of light transport involves surface rendering and volume rendering for participating media, which both equations having recursive integrals with no closed form solution for forward modeling

    - Approaches to inversely solve these equations differ in *the degree of approximation* they make
- Some methods reconstruct *appearance* as approximate incident radiance, and other methods attempt to separate appearance into *materials* via explicit scattering distribution functions and *lighting*, enabling applications such as relighting
- Many papers use a neural field to *parameterize the parameter space of existing material models*
    - Neural Reflectance Fields can reconstruct both the SVBRDF and geometry by assuming a known point light source, and use a neural field which parameterizes density, normals, and parameters of a microfacet BRDF with a volume rendering forward map with one bounce direct illumination
    - NeRV handles more varied lighting setups with an environment map and a one bounce indirect illumination forward map, along with an additional neural field which parameterizes visibility, but assume the environment map is known a priori

> 以上的论文并不是综述里提及的全部论文......

7.3. Dynamic Reconstruction

> The challenges of modeling dynamic scenes are that the input data is even *sparser* in spacetime and often *no 4D ground-truth data are available*.
>
> With *careful design choices*, *regularization loss terms*, and the *strong inductive bias of neural fields*, several works have proposed solutions to this inverse problem.

Embedding

- Modeling temporally changing objects or scenes requires additional embedding that encodes frame information / processing temporal coordinate
- While embedding based on temporal coordinates automatically incorporates temporal coherency as inductive bias, per-frame latent codes can enable the captures of more scene details

Warp Representation

- To model dynamic scene from limited in put data, we can split the problem into modeling a scene in the canonical space and warping it into each time frame

8. Digital Humans

Face Modeling

> The explicit representations (data-driven parametric morphable model) lack realism and impose topological limitations making it difficult to model hair, teeth, etc. The expressiveness of fields, such as *SDF* and *radiance fields*, has made them an excellent candidate to address these limitations.

- NeRF can be adopted for photo-realistic view synthesis of human heads
- Another line of works enable the semantic control of radiance fields by conditioning on head pose and facial expression parameters obtained from a 3D morphable face model
- Another line of works enable the semantic control of radiance fields by conditioning on head pose and facial expression parameters obtained from a 3D morphable face model

Body and Hand Modeling

> Neural fields have demonstrated efficacy in 3D reconstruction of *clothed humans* from image inputs or point clouds.

- Due to substantial variations in shape and appearance of clothed human bodies, a global latent embedding does not lead to plausible reconstruction.
- Human bodies are dynamic as they are both articulated and deformable. Several works show that providing the structure of human bodies significantly improves the learning of radiance fields
- Template-mesh registration is another important task in body modeling
- Several recent works model a parametric model of human bodies, clothed human hands, or clothing as *neural implicit surfaces*
- One unique property of human body is the articulation with non-rigid deformation

9. Generative Modeling

> Assuming a dataset of samples drawn from a distribution $\mathbf{y} \sim \mathcal{D}$, generative modeling defines a latent distribution $\mathcal{Z}$, such that every sample $\mathbf{y}$ can be identified with a corresponding latent $\mathbf{z} \sim \mathcal{Z}$. The mapping from $\mathcal{Z}$ to samples from $\mathcal{D}$ is performed via a learned generator $\mathbf{g}$, parameterized as a deep neural network, $\mathbf{g}(\mathbf{z}) = \mathbf{y}$.

Neural fields provide greater flexibility for generative modeling because their outputs can be sampled at arbitrary resolutions, and because input samples can be shifted by simply applying transforms to the input coordinates

Generative Modeling of Images

- In generative modeling of images, samples $\mathbf{y}$ are images, and we assume that there exists a distribution $\mathcal{D}$ of “natural” images.
- The neural field Φ maps 2D pixel coordinates to RGB colors, $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$

Generative Modeling of 3D Shape and Appearance

- Instead of directly modeling the distribution of images, neural fields can parameterize distributions over 3D shape and appearance given only an image dataset.
- The neural field $\Phi : \mathbb{R}^3 \to \mathbb{R}^n$ then maps a *3D* coordinate to *a quantity that encodes shape and appearance*. This is combined with a neural forward rendering model that renders an image given camera parameters.

Multi-modal Generative Modeling

- An advantage of neural fields is that they are, in principle, agnostic to the signal they parameterize

> Q: 没看懂为什么这是个优点

10. 2D Image Processing

> A compelling feature of 2D neural fields is the ability to represent *continuous images*.

> The first networks were not fit via gradient descent, instead relying on architecture search in a genetic algorithm framework, and could thus not represent images with fine detail.

- Unlike grid-based convolutional architectures, continuous images can be sampled *at any resolution*, which can be used for a variety of below image processing tasks

Image-to-image Translation

- These techniques take an input image and map it to another image that preserves some representation of the content.
- Common tasks include
  - image enhancement
  - image super resolution
  - denoising
  - inpainting
  - semantic mapping
  - generative modeling
- Since this task requires learning a prior from data, an encoder-decoder architecture is often used, where the encoder is a *convolutional neural network*, and the decoder is a *locally conditioned 2D neural field*

Image Reconstruction

11. Robotics

> Robotics requires complex perception systems that allow agents to efficiently **infer**, **reason** about, and **manipulate** representations of *real-world scenes*.

11.1. Localization via Camera Parameter Estimation

> Cameras observe the 3D world via 2D images.
>
> The projection of a world location onto the image plane is obtained through the extrinsic and intrinsic matrices, where the extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ defines the 6DoF transformation between the world coordinate frame and the camera coordinate frame, while the intrinsics matrix $\mathbf{K}$ describes the projection of a 3D point onto the 2D image plane.

- Since neural rendering is end-to-end differentiable, camera parameters can be jointly estimated with the neural field making them useful for Simultaneous Localization and Mapping (SLAM) and Absolute Pose Regression (APR)

  > Q: 这个没看懂……什么叫端到端可微？原因和结果又是什么关系？

- Representing the extrinsic matrix, particularly rotation, however, is a long-standing challenge. Since the 3-by-3 rotation matrix lies on $SO(3)$, *continuity is not guaranteed*
  - alternative parameterizations are used
- Joint reconstruction and registration is a long standing chicken-and-egg problem: camera parameters are needed to reconstruct the scene, and a reconstruction is needed to estimate camera parameters
  - One simplification is to assume known reconstruction and solve only the registration problem

        - problem: limited use case of registering new, un-posed images to an already-reconstructed scene
        - problem: the optimization is known to be non-convex
        - solution: a coarse initialization of camera parameters alleviates this
    - More challenging problem of estimating unknown camera parameters, while jointly reconstructing the scene

11.2. Planning

> Planning in robotics is the problem of identifying a sequence of valid robot configuration states to achieve an objective:
>
> - path planning for navigation
> - trajectory planning for grasping or manipulation
> - planning for interactive perception

11.3. Control

> Controllers are responsible for realizing plans, while ensuring that physical constraints and mechanical integrity are preserved

- Control can be achieved either by relying on a planner or directly from observations

12. Lossy Compression

> The goal of lossy data compression is to **approximate** a signal as best as possible with as few bits as possible. These opposing forces naturally form a **tradeoff** which can be characterized as a Pareto frontier: the rate-distortion curve

- In practice, signals are often stored as discrete sequences of data which are transformed into *an alternate basis* such as the discrete cosine transform which help to decorrelate the signal (making downstream tasks like quantization and entropy coding more effective)
- Recent work has explored the potential of neural fields as an alternate signal storage format which directly represents the continuous signal with a parameteric, continuous function. Compression may be achieved in one of two ways
    - by leveraging the inductive bias of the network architecture itself, and *simply overfitting a neural field to a signal*
        - store the parameters and architecture
    - *prior-based* compression schemes achieve compression via *learning a space of low-dimensional latent code vectors* $\mathbf{z}$ that may be decoded into neural field parameters, where the storage cost of the decoder is amortized over many latent codes
- There are many problems in the study of taking neural fields as a fata format. While neural fields for compression remains in its infancy at the time of writing, it is nonetheless a valuable perspective to consider signals as functions and neural networks as a data format.

13. Beyond Visual Computing

Visual computing problems are a subset of all inverse problems which can be parameterized by neural fields. These problems often share the same challenges involving incomplete observations and the need for a flexible parameterization.

13.1. Alternative Signal Modalities

> Neural fields can also model alternative signal modalities such as non-line-of-sight imaging, non-visible x-rays for computed tomography, magnetic resonance imaging (MRI), pressure waves for audio, chemiluminescence, time-of-flight imaging, as well as volumetric light displays.

- **Medical Imaging**
  - CT and MRI
- **Audio**

13.2. Physics-informed Problems

> Physics-informed problems have solutions that are restricted to a set of partial differential equations (PDEs) based on laws of physics.

- solutions are often continuous in spatio-temporal coordinates
- Neural fields are therefore a natural parameterization of the solution space, given that neural networks are continuous, differentiable, and universal function approximators. These neural fields are also referred to as physics-informed neural networks (PINNs)
- Parameterizing the solution problems constrained by nonlinear PDEs via neural networks reformulates these problems as *optimization*, rather than *simulation*, which is more data-efficient

Discussion & Conclusion

Several factors that have resulted in the progress of neural fields:

1. The idea of parameterizing a continuous *field* using an MLP without the need to use more complex neural network architectures has simplified the training of fields and reduced the entry barrier. Neural fields provide an approach to signal processing that is faithful to the original *continuous* signal.
2. Techniques such as positional encoding and sinusoidal activations have significantly improved the quality of neural fields leading to large leaps in applications focused on quality. Applications in novel view synthesis and 3D reconstruction have been particularly important in popularizing neural fields because of the visually appealing nature of these applications.
3. Researchers have realized that *differentiable volume* and *voxel rendering* commonly used in novel view synthesis methods can be useful in solving others tasks like 3D reconstruction and even semantic segmentation.

Future direction for techniques:

- poor generalization

    - integration of stronger priors
- to build a common framework
- lack of shared datasets and benchmarks

Future direction for applications:

- semantic tasks
- to explore the fusion of multiple modalities
- to make use of weakly-supervised or self-supervised learning

Societal Impact:

- negative
    - illegal impersonating
    - implications for privacy
    - decrease the cost of surveillance
    - computational resources wasting
- positive
    - helps artists and content creators
    - relaxes hardware constraints for 3D content creation
    - may help build robotic automation for computer vision

Other Resources

1. NeRF及其发展
2. 2D基本图形的Sign Distance Function (SDF)详解（上）
3. 2D基本图形的Sign Distance Function (SDF)详解（下）
4. 计算机图形学九：隐式曲面(代数形式,CSG, 距离函数,分型几何)与显式曲面
5. IQ的网站
6. 立体角(Solid Angle)详解
7. 渲染方程Rendering equation
8. 常用3D表示
9. Alpha Composition
10. Differentiable Volumetric Rendering