foundations

MDPI

Review

Data-Driven Field Representations and Measuring Processes

Wanrong Hong [†], **Sili Zhu** [†] **and Jun Li** [*]

School of Computer Science, Australian Artificial Intelligence Institute (AAII), University of Technology Sydney, Sydney 2007, Australia; wanrong.hong-1@student.uts.edu.au (W.H.); sili.zhu@student.uts.edu.au (S.Z.)
[*] Correspondence: jun.li@uts.edu.au
[†] These authors contributed equally to this work.

Abstract: Natural mathematical objects for representing spatially distributed physical attributes are 3D field functions, which are prevalent in applied sciences and engineering, including areas such as fluid dynamics and computational geometry. The representations of these objects are task-oriented, which are achieved using various techniques that are suitable for specific areas. A recent breakthrough involves using flexible parameterized representations, particularly through neural networks, to model a range of field functions. This technique aims to uncover fields for computational vision tasks, such as representing light-scattering fields. Its effectiveness has led to rapid advancements, enabling the modeling of time dependence in various applications. This survey provides an informative taxonomy of the recent literature in the field of learnable field representation, as well as a comprehensive summary in the application field of visual computing. Open problems in field representation and learning are also discussed, which help shed light on future research.

Keywords: field; data-driven; neural networks; measuring process

check for updates

Citation: Hong, W.; Zhu, S.; Li, J. Data-Driven Field Representations and Measuring Processes. *Foundations* **2024**, *4*, 61–79. https://doi.org/10.3390/foundations4010006

Academic Editor: Michal Hnatič

Received: 8 December 2023
Revised: 31 December 2023
Accepted: 16 January 2024
Published: 30 January 2024

1. Introduction

Fields are useful notions that are ubiquitous in all scientific areas. In generic terms, a field can be treated as a mapping from spatial locations to a physical variable of interest. Given a reference frame, a location is specified by a set of real-valued coordinates, and the most common form of a field function is $f : \mathbb{R}^3 \mapsto \mathcal{Y}$. $\mathcal{Y}$ represents the domain of the quantities of interest.

The concept of fields serves as a fundamental cornerstone in various scientific disciplines, offering a versatile framework to understand and describe diverse phenomena. Across scientific areas, field representations take on distinct forms tailored to the unique demands of each discipline. In the related physics and mathematics subjects, fields are often construed as formal mathematical entities, inviting analytical scrutiny to uncover their intricate characteristics. In the domain of quantum field theory (QFT) [1–3], fields cease to be mere abstractions and emerge as dynamic entities permeating space and time. These fields—whether scalar, vector, or tensor—are not fixed values; instead, they are dynamic and fluctuating entities that embody the fundamental nature of quantum interactions. Transitioning from the microscopic to the macroscopic, the gravitational field introduces a different facet of field representation. In the study of gravitation [4], fields manifest as gravitational fields, bending the geometry of space–time itself. In contrast to the analytical focus of QFT, gravitational fields lead toward a geometrical interpretation, weaving a narrative where masses dictate the curvature, and objects move along paths carved by this curvature. In applied mathematics, the study of partial differential equations (PDEs) [5] further diversifies our perspective on field representations. PDEs are ubiquitous in describing a myriad of physical phenomena, from heat diffusion to wave propagation.

While the theoretical underpinnings of fields find elegance in formal mathematics, the practical application of field concepts in engineering demands a shift in perspective. When grappling with real-world engineering problems, computational aspects take center stage.

In the area of representing and performing computations on fields, a prevalent approach involves discretization, or quantization, of space, offering a practical bridge between theory and application [6,7]. Depending on the application fields, various quantization schemes have been found effective. For example, Eulerian quantization [8] finds practical application in finite element analysis, where structures are divided into smaller, discrete elements, and computations are performed at each node or element. Contrasting Eulerian quantization, Lagrangian quantization [9] follows the motion of individual particles or elements through space and time. In simulating the behavior of fluids, Lagrangian methods track the movement of fluid particles, allowing for a detailed examination of fluid flow, turbulence, and interactions. Similarly, in simulating the dynamics of solids, Lagrangian quantization enables the study of deformations, collisions, and structural responses under varying conditions.

Recently, the development of data-driven approaches has revolutionized various scientific and engineering domains. The emergence of data-driven techniques, powered by advancements in machine learning and computational capabilities, offers a promising ability to handle complex, high-dimensional data. Specifically, a useful technique involves employing flexible parameterized representations to model the family of appropriate field functions, particularly through neural networks. Originating from computational geometry and vision, this technique aims to recover fields for subsequent computational vision tasks, such as representing light-scattering fields. Despite the rapid development of data-driven field modeling techniques, there is currently no comprehensive review summarizing the advancements in this field. Therefore, this review aims to provide an overview of state-of-the-art learnable computational models for 3D field functions. Specifically, we focus on models that utilize neural networks as function approximators, showcasing the capabilities and flexibility of neural networks in handling various fields.

The structure of this review is as follows: In Section 2, we present the technical definition of field functions and neural networks. In Section 3, we review neural-network-based and parametric-grid-based approximations for practical field functions. In Section 4, we discuss the developments of the measurement process in the data-driven frameworks by reviewing existing works contributing to the observation process. This section aims to reveal the challenges in this domain and suggests emerging research problems. Finally, we conclude the review.

2. Background

A field function describes a spatially (and temporally) distributed quantity

$$f : \mathbf{x} \mapsto \mathbf{u} \tag{1}$$

where $\mathbf{x} \in \mathcal{X}$ represents the domain in which the interested physical process is defined, usually an extent of 3D space with an optional time span. $\mathbf{u}$ represents the range of the target physical properties of interest.

We consider a parameterized function space:

$$\mathcal{H} : \{h(\mathbf{x}, \mathbf{o}; \theta) | \theta \in \Theta\} \tag{2}$$

from which a candidate function h^* is selected to approximate the underlying field f in Equation (1). In Equation (2), h represents a candidate approximator for f. $\mathbf{x}$ specifies a space–time point. The symbol θ represents learnable parameters. There is an extra input variable to the learning field model, $\mathbf{o}$, which specifies extra conditions to make observations of the underlying field (The observation condition in Equation (2) should be considered without loss of generality. In practice, useful modeling of a physical system can be formulated by specifying the observation process. For example, a quantum mechanical system with a state vector $|x\rangle$ can often be formulated with an observation operator $H|x\rangle$).

The field model is determined in a data-driven manner. In the design stage, the parametric model family is constructed, typically by selecting a specific neural network structure. Once the network structure is fixed, specific parameter values define the particular field function. In the data-driven modeling approach, these parameters are set during an optimization process (learning). The optimization objective is formulated to ensure that the field model aligns with a set of observed variables. The modeling process involves specifying how the observable quantities used to constrain the model were generated.

Neural Networks as Function Approximator

One of the most successful families of function approximators is the neural network models [10]. It is not surprising that multiple-layer perceptrons (MLPs) have become a prevalent framework for generic function approximation to represent a target field function. An MLP is composed of interconnected nodes, referred to as neurons, organized in layers. Each neuron is linked to every neuron in the preceding and succeeding layers. The output of a neuron is adjusted by weights and biases to approximate complex functions, allowing the capture of intricate patterns and relationships in the data.

Generally, the computation of an MLP involves multiple primary units, where, for a unit $a \in \mathcal{A}$,

$$f^a(\boldsymbol{x}) = \varphi\Big(\sum_{i=1}^{k} w_i^a x_i + b^a\Big) \tag{3}$$

where f denotes a specific layer with input vector $\boldsymbol{x}$. The parameters $\{w_i^a\}$ and b^a determine the behavior of the particular unit. $\varphi(\cdot)$ is the activation function applied to the summation of weighted inputs and bias. This design draws inspiration from biological neurons, where synaptic weights modulate excitatory and inhibitory stimuli, akin to the weights $\{w\}$ and bias b in Equation (3). The status of a neuron is determined by its response to the stimuli, akin to the activation function φ. The unit in Equation (3) is commonly referred to as a *neuron*. MLP represents a specific class of computational models containing multiple units $a \in \mathcal{A}$. In an MLP, the units $\mathcal{A}$ are organized in an acyclic graph, and the computations are hence sorted in multiple stages. Units belonging to the same computational step, which can be evaluated simultaneously, are often referred to as forming a *layer* in the network. To be specific, the inputs to a certain layer are given by:

$$x_i^l = f^{l-1,i}(\dots) \tag{4}$$

where the superscripts on the right-hand side specifies a neural in $\mathcal{A}$: the ith neuron output in the $(l-1)$th layer serves as the ith input to the neurons in the lth layer. The inputs to the lth layer expand recursively as in Equation (4), and the inputs to the first layer, $\boldsymbol{x}^0$, are the raw inputs of the MLP model.

MLP can serve as a generic computational model as a function approximator. Given a function $g : \mathbb{R}^n \mapsto \mathbb{R}$ that is Borel-measurable, Hornik et al. [10] established that for any $\epsilon > 0$ and for any compact subset $K \subset \mathbb{R}^n$, there exists a two-layer neural network f such that the supremum norm of the difference between g and f over K is less than ϵ. The neural network can be represented as:

$$f^{(2)}(x) = \sum_{i=1}^{N} w_i^{(2)} f^{(1)}(x) + b^{(2)}$$

Theoretical investigations of neural network expressivity have been conducted since then.

3. Computational Model of Learnable Field

This section concerns the use of the universal approximation capacity of neural network models to represent physical fields. Learnable models have been gaining attention in the areas of fluid simulation [11], electromagnetics [12], climate science [13], acoustics field [14], medical imaging [15–19], etc.

The brief overview in the preceding section demonstrated that the neural network family contains sufficient expressivity and can be used to approximate most practical field functions. However, most theoretical investigations have been either nonconstructive or have considered the properties of the model asymptotically. In practical field modeling, specific designs and techniques need to be developed to ensure effective modeling. The inputs to a field function specify a location in the spatial domain where the field of interest is evaluated. Although the field can also evolve over time and the function differentiates one time from another [20–22], in this section, we focus on formulating the computations and specifically consider a static field for simplicity. We explore recent methodologies and advancements in analytics for neural-network-based field learning by sampling works in a burgeoning research topic, such as the field of light scattering [23]. However, the principles of computational design for field models extend beyond the exemplary cases and are generally applicable to other tasks involving field function approximation using neural networks as learning models.

3.1. Neural Network-Based Approximator

In [23], the authors present a field f that maps a location $p \in \mathbb{R}^3$ to interested values such as density σ and color c (Equation (5)). Furthermore, the variable p continues to represent the location or position in the remainder of this section.

$$f : (p, d) \mapsto (\sigma, c) \tag{5}$$

where d represents the observing direction, which is selected based on the specific interest and is not limited to the particular observing direction.

The researchers employed a neural network to approximate two field functions using shared parameters. Firstly, they sample locations along the ray from observing direction. Then, each location's coordinates are fed into the neural network, which output density values and geometry feature vectors. In this stage, the neural network approximates the scene's geometry as the geometry function denoted as f_σ, where $f_\sigma \in \mathcal{H}$. In the second stage, the inputs of the neural network are the observing direction and geometry feature vectors. The neural network approximates the appearance function denoted as g_c, where $g_c \in \mathcal{H}$, and outputs visual attributes such as color at that location. Then, the overall field approximation can be written as a composition of a geometry field and an appearance field:

$$f = g_c(f_\sigma(p), d) \tag{6}$$

In the implementation, the two field functions are approximated by the same neural network, see Figure 1a. This approximator type has also been utilized in other studies, such as [24–26]. However, all the parameters of the approximator are shared, which means each field cannot be updated and adjusted individually. Additionally, combining two fields results in an increased number of layers in the network, leading to longer computation times during training and evaluation.

Garbin et al. [27] implemented two field approximations by two separate neural networks and achieved a faster and more flexible approximation as shown in Equation (5). The geometry and appearance fields have their own parameters, which are not shared with each other, as depicted in Figure 1b. When a change happens, the two approximations can be updated more flexibly, such as storing one approximation and adjusting the other.

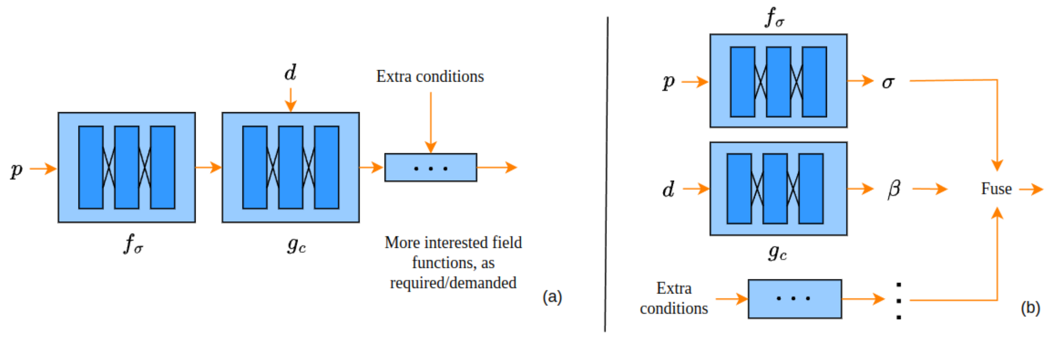

Figure 1. Different computational architectures to implement the function family in Equation (2). (**a**) Different field functions with different physical properties can be learned by a shared neural network, such as the combination of f_σ interested in density and g_c interested in appearance in Equation (6). However, the approximators are not limited to geometry and appearance. One of the "extra conditions" can be time, and the field function can approximate time-related changes [20]. It should be mentioned that the order of different fields in this figure is just an example. In practice, the order must be carefully considered per the requirements or demands. (**b**) Each field can be approximated by individual neural networks, which can be updated and adjusted separately. However, not all field functions can be simply fused. Both physical relations and computational skills need to be considered in real implementations.

3.2. Eulerian Representation via Parameters on Spacial Grids

The above field approximated using neural networks is a global approximation where all small regions within the field are interconnected. When a change is observed at a specific location (see Section 4), the neural-network-based approximation needs to update all parameters of the field function accordingly. However, in most cases, a change only affects a small region surrounding it. An alternative computational model associates function approximations with particular regions in the domain. The most straightforward specification of a location is via Cartesian coordinates. In the case of a 3D field, the 'regions' can be arranged by a grid with Cartesian coordinates [16,28–32].

Liu et al. [28] developed a grid-based approximator to express local properties in specific regions (Figure 2). The grid was utilized to spatially organize the field, where each grid cell represents a small region within the field. They defined a grid as $G = \{V_1, ..., V_N\}$ containing N small cells V. $\mathcal{G}_\theta^i \in \mathcal{H}$ represents the local field function within the ith cell. We use the same example of geometry representation as mentioned above. The geometry field approximator can be written as follows: cite number

$$\mathcal{G}_\theta^i : p \mapsto \sigma, p \in V_i \tag{7}$$

It can be seen that $\mathcal{G}_\theta^i$ has the same function as the neural network mentioned above, which maps a location to the corresponding values of interest. However, it has a different structure from neural networks. Unlike neural networks, the parameters are shared between layers, and the learnable parameters are associated with the location of the cell. For example, if the cell is a small cube, the parameters can be assigned to the eight vertices, and the local properties are bound within the volume of the cube. When p falls in a small region V_i, its feature $\mathcal{V}_i(p)$, where $\mathcal{V} \in \mathcal{H}$, is calculated based on the features stored for that region. This implies that only local parameters are utilized to represent and observe p.

The grid approximation can also be combined with other approximating functions, such as g_c:

$$\mathcal{G} = \{\mathcal{G}_\theta^0, \mathcal{G}_\theta^1, \ldots, \mathcal{G}_\theta^i, \ldots\}$$
$$f = g_c(\mathcal{G}, d)$$

The authors demonstrated the effectiveness of utilizing a grid-based strategy for approximating a field.

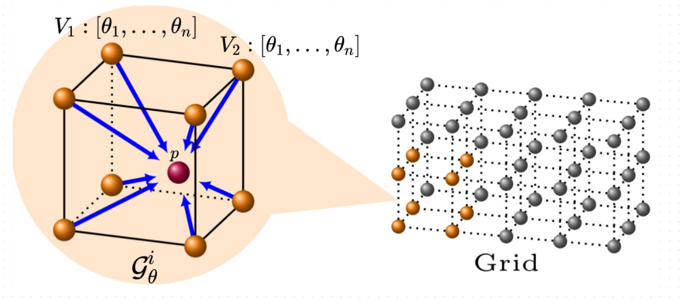
$V_1 : [\theta_1, \ldots, \theta_n]$
$V_2 : [\theta_1, \ldots, \theta_n]$
p
$\mathcal{G}^i_\theta$
Grid

Figure 2. Grid-based approximator of a field. If the smallest cell in a grid is a cube. The learnable parameters can be assigned to the eight vertices, and the local properties are represented within the cube. The feature of any position p in the cube is computed using the selected interpolation strategy. The grid arrangement provides spatial indexing, offering efficient spatial querying, rapidly retrieving objects or points within a specific region of interest.

Yu et al. [33] also employed similar grid-based approximators $\mathcal{G}$. However, they did not utilize a neural network to compose the other fields. They leveraged local priors such as spherical harmonic (SH), which assumes light scatters on a small area of the target [34]. They demonstrated that grid-based approximations can work without neural networks, and each region can output different interests directly.

As grid-based approaches associate the parameters with spatial location, it is possible to use a more efficient structure to arrange parameters with fluctuation in a region. One common choice is using an octree.

Octree is a hierarchical data structure used to efficiently represent spatial information. When applying an octree for a grid-based field, the empty or full property of a tree node (Figure 3) is associated with the parameters of the approximator. Octree also represents the property of the target function. When the target function changes very quickly in a region, more samples are needed for approximation, which leads to a higher density of parameter distribution in that region and vice versa [28,35]. Octree can be used to prune or skip uninterested regions in the field and recursively partition the interested ones into smaller cells. By distributing parameters according to the function's fluctuation in each region, the octree can effectively capture the complex region of the field, allowing for a more precise approximation of the target function.

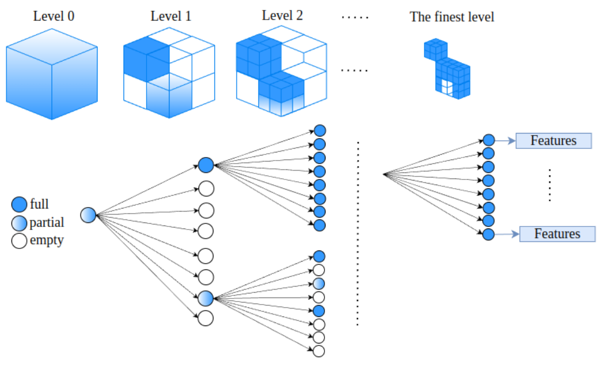

Figure 3. Octree representation of a field. Consider the grid-based field in Equation (7). The root level represents the bounding box for the entire field, while the finest leaf nodes correspond to the smallest units determined by settings, such as the vertices of a cube area. A white node indicates that all its child nodes are empty, allowing it to be skipped during the computing process. Conversely, blue nodes represent important spaces where all the vertices possess valuable features and should not be skipped. Partial nodes indicate that only certain nodes are empty and cannot be skipped. The full and partial nodes can be subdivided into smaller cells until a termination condition is met. Regions with more significant details or rapid changes are subdivided with more nodes, while less important regions are represented with fewer nodes. As all the nodes have corresponding learnable parameters, this adaptive nature enables a higher density of parameter distribution in important regions.

Different from previous researchers focusing on grid-based approaches, Chen et al. [36] factorized the field function through compact tensors for scene modeling. Taking a geometry field as an example, the original field function maps a location to a density interest and is denoted as G. The tensor representation T aims to approximate G through low-rank factorization. The rank $r = [1, 2, \cdots, R]$ is the number of tensors or matrices required to recreate the original elements of G (Figure 4).

The three mode tensors can be indexed using (i, j, k), and v_i^X is the ith value of the vector v^X. For CP decomposition, a tensor element T_{ijk} is computed by summing the R outer products of vectors (v_r^X, v_r^Y, v_r^Z) as follows:

$$T_{ijk} = \sum_{r=1}^{R} v_{r,i}^X \otimes v_{r,j}^Y \otimes v_{r,k}^Z$$

where $\otimes$ is the outer product.

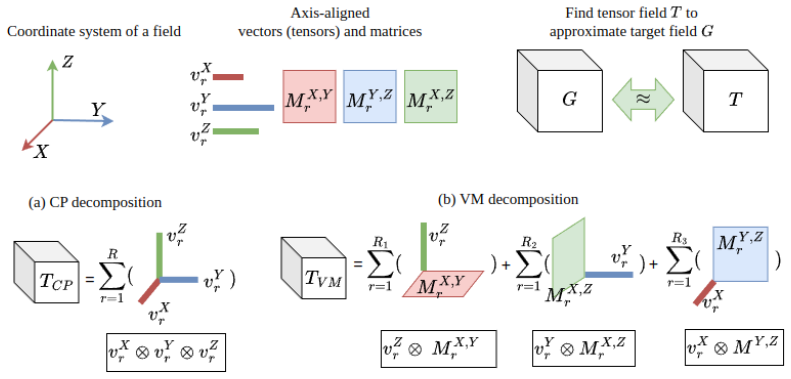
Coordinate system of a field
Z
Y
X
Axis-aligned vectors (tensors) and matrices
v_r^X
v_r^Y
v_r^Z
$M_r^{X,Y}$
$M_r^{Y,Z}$
$M_r^{X,Z}$
Find tensor field T to approximate target field G
G
≈
T
(a) CP decomposition
$T_{CP} = \sum_{r=1}^{R} (v_r^Z \; v_r^Y \; v_r^X)$
$v_r^X \otimes v_r^Y \otimes v_r^Z$
(b) VM decomposition
$T_{VM} = \sum_{r=1}^{R_1} (v_r^Z \; M_r^{X,Y}) + \sum_{r=1}^{R_2} (M_r^{X,Z} \; v_r^Y) + \sum_{r=1}^{R_3} (M_r^{Y,Z} \; v_r^X)$
$v_r^Z \otimes M_r^{X,Y}$
$v_r^Y \otimes M_r^{X,Z}$
$v_r^X \otimes M^{Y,Z}$

Figure 4. Tensor factorization: G represents a field function. A tensor field T is the approximation of G. According to the X, Y, and Z axes of a field, three modes vectors (tensors) v^X, v^Y, v^Z or matrices $M^{X,Y}, M^{Y,Z}, M^{X,Z}$ are used for the decomposition. (**a**) The CP decomposition represents canonical polyadic decomposition [37], which is vector-only decomposition, and (**b**) VM decomposition refers to vector–matrix decomposition [36]. Tensor decomposition can be expressed as the sum of R outer products of vectors or vector–matrix. A smaller value of R indicates a more compact field representation, but accuracy is sacrificed.

For VM decomposition, an element is computed by vectors and matrix:

$$T_{ijk} = \sum_{r=1}^{R} v_{r,i}^X \otimes M_{r,jk}^{Y,Z} + v_{r,j}^Y \otimes M_{r,ik}^{X,Z} + v_{r,k}^Z \otimes M_{r,ij}^{X,Y}, v_r^X \in \mathbb{R}^I, v_r^Y \in \mathbb{R}^J, v_r^Z \in \mathbb{R}^K$$

Also, a tensor function can be combined with other interesting functions like the appearance function discussed above [38–40].

In summary, neural-network-based and parametric-grid-based approximations were discussed for representing fields. The brief overview demonstrated the capabilities and flexibility of using neural networks to handle various fields. The learning parameters can belong to a single field or be shared by multiple-field approximations, which approximate global information well but may not efficiently handle local changes. On the other hand, a grid-based approximator can effectively capture local changes but may have limitations in capturing global information. To address these limitations, researchers have explored the combination or fusion of neural networks and grid-based approximators [28,30]. This approach allows for leveraging the strengths of each method and potentially achieving a more comprehensive approximation.

4. Computational Models of the Measurement Process

Learning models are constructed using data-driven methods. In addition to the spatial and temporal distribution of the interested physical properties, it is also necessary to incorporate the observable variables in the models, i.e., we need to describe the process of how data are measured from the physical fields, such as images from a scattering field of light. The computational observation model allows the information of the physical fields to be extracted from the sensory data and hence learn the underlying model, for example, learning a magnetic field from sensory data [41]. In this section, we discuss several challenges and design choices in this aspect and illustrate the ideas using examples of learning light-scattering fields from sensory observations. The relationship between 2D observations and field functions is shown in Figure 5.

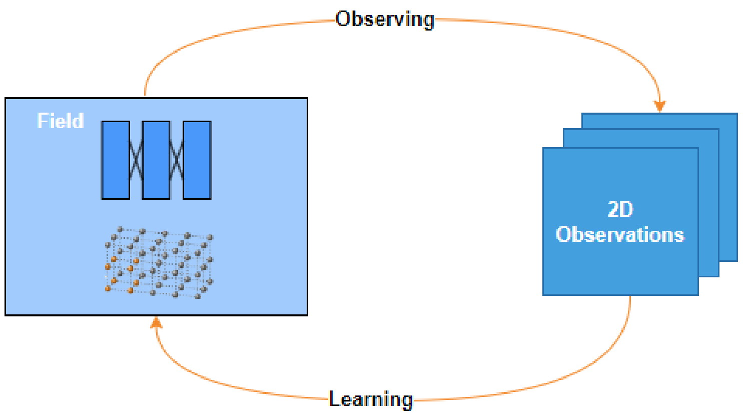

Figure 5. The correlation between fields and 2D observations. The diagram provides a visual representation of how data-driven field models interact with and manifest in observable 2D data.

Formally, recall the field approximator function family discussed in previous sections. An example instantiating Equation (2) was discussed in Section 3,

$$F : (\mathbb{R}^3, S^2) \to (\mathbb{R}^3 \times \mathbb{R})$$

where F specifies the light scattering at a location in 3D space $\mathbb{R}^3$; S^2 represents the observation condition of the light field, i.e., the view perspective from the two-sphere (indicating all possible viewing directions). The discussion so far focused on expressing F in an adequate function family.

However, despite the conceptual clarity of the model, the field cannot be directly learned from measurements. In other words, a function approximation model $\mathcal{H}^a : \{h(\mathbb{R}^3, S^2; \theta)\} | \theta \in \Theta$ alone cannot be used to represent practical fields. This is because learning (data-driven) models aim to find an optimal approximation to F from observations. Typically, this is achieved by minimizing the difference between the observed quantities and what the field function has specified. In the task above, actual measured sensory data are not directly related to F; instead, the data are obtained from a process of forming observations by reducing the 3D field to variables distributed on a 2D plane. The result of the observation process is 2D images.

To learn from data, constructing a computational model that mimics the process of deriving observations from the field is essential. The computational observation model facilitates inferring field information from the data. To faithfully, efficiently, and effectively formulate this process for learning, attention must be paid to several key aspects.

Specifically, we consider three aspects that underscore the significance of the observation process: the integral in 3D space, the discretization of the integral, and the diverse computational processes involved in executing the integral.

In measuring the field, quantifiable values depend on the field's extent rather than a specific point, making direct empirical verification of field equations impractical. Thus, constructing a computational model to depict the observation process becomes necessary. Given the observational nature across a region, a logical approach is to integrate the field over that region, assuming the observation aligns with the expected value of the field function across the field.

The assumption is only valid when the interested physical quantities are reasonably homogeneous. In practical scenarios where this condition is not met, adjustments to the computational model become imperative. For instance, in the observation process within the neural radiance field, the mathematical expression involves the volumetric rendering equation, as shown in Equation (8) [42].

$$C(r) := \int_{t_{near}}^{t_{far}} T(t)\sigma(r(t))c(r(t), d)dt \tag{8}$$

Here, the integral encapsulates the integration over the entire depth range from t_{near} to t_{far}, where t refers to the distance between two points along a line. The variable d corresponds to the observing direction. σ is the quantity value illustrating the propensity of the volume to scatter photons—essentially, quantifying "how many" photons can be scattered. For example, a volume on the surface can scatter more photons and contribute more to the final observations. c denotes the color at the specific position concerning observing direction. $T(t)$ represents the probability that photons do not contribute to the volume within the range t_{near} to t. This introduces a probabilistic element, accounting for the likelihood that photons traverse the specified distance without interaction, formally defined as:

$$T(t) := e^{-\int_{t_{near}}^{t} \sigma(r(s))ds}$$

The computational model can be altered based on changes in the observation process. For instance, if the observation emphasizes the geometric structure of scene objects, necessary modifications to the computational model are warranted [43–50]. This might entail a transition to a model that first reconstructs the surface before undertaking the observation process [51–59].

The second aspect involves the discretization process within the computational framework. A conventional method used to estimate the integral in Equation (8) is a process known as quadrature [24,25,27,29,30,33,36,60–64]. This entails discretizing the continuous line into a set of discrete samples, obtained via a spatial sampler. These discrete samples function as a representative approximation of continuous volumetric data.

Specifically, in the example of NeRF, a coarse-to-fine strategy is adopted [65]. In the coarse stage, for each line integral, a stratified sampling approach is implemented to uniformly partition the depth range $[t_{near}, t_{far}]$ into N bins. The mathematical definition of the sampling process within each bin is as follows:

$$t_i \sim \mathcal{U}[t_{\text{near}} + \frac{(i-1)}{N}(t_{\text{far}} - t_{\text{near}}), t_{\text{near}} + \frac{i}{N}(t_{\text{far}} - t_{\text{near}})]$$

where t_i represents the sampled points within each bin, ensuring a stratified distribution along the line integral. This strategic sampling approach aims to capture the varied characteristics of the scene at different depths, providing a more comprehensive representation for the subsequent integral computation.

The estimation of the integral in Equation (8) can then be expressed in a discrete form as follows [66]:

$$\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - e^{(-\sigma_i\delta_i)})c_i \tag{9}$$

In this formulation, $\hat{C}$ represents the estimated observing color, which is computed by summing over all the sampled points within the stratified bins. Each term in the summation accounts for the contribution of a specific sampled point i, considering its transmission probability T_i, the volumetric scattering term $(1 - e^{(-\sigma_i \delta_i)})$, and the color value c_i at that point. $\delta_i = t_{i+1} - t_i$ refers to the distance between adjacent samples.

Specifically, the transmission probability T_i is calculated using the discrete transmission function, which encapsulates the cumulative effect of volumetric absorption along the line, influencing the contribution of each sampled point to the final observation:

$$T_i = e^{(-\sum_{j=1}^{i-1} \sigma_j \delta_j)}$$

In the fine stage, points are sampled based on the volume that is predicted in the coarse stage, which samples more points in the regions the model expects to contain visible content. To achieve this, Equation (9) is rewritten based on the result from the coarse stage as:

$$\hat{C}_c(r) = \sum_{i=1}^{N_c} w_i c_i$$

$$w_i = T_i(1 - e^{(-\sigma_i \delta_i)})$$

The points are then sampled based on the normalized weights: $\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$.

The sampling strategy needs to be adjusted based on different observation processes. For instance, when the integral range encompasses an unbounded scene rather than a bounded one [67], a stratified sampling strategy may not be appropriate. In cases where the observation prioritizes the geometric structure of scene objects, the sampling strategy may have less significance in the model, given that the surface can be approximated.

4.1. Integration Adjustments

In Zhang et al. [68], the observation model tackles the challenge regarding the scene range. In the previous discussion, Equation (8) integrates attributes from t_{near} to t_{far}. When the dynamics range of the depth in the scene is small, a finite number of samples is enough for estimating the integral. However, when the scene is an outdoor scene or a 360-degree unbounded scene, the distance range could be large (from nearby objects to distant elements like clouds and buildings). Specifically, for expansive scenes, where distances vary significantly, the integration range becomes extensive.

The challenge arises when the observation is determined by the field evaluation over a large extent. For example, when forming images in a NeRF model, if the device is focusing outward, the distance range is large. The observation model, rooted in Equation (8), faces complexity, as highlighted in [68], due to the need for sufficient resolution in both foreground and background areas within the integral. However, employing a sampling strategy in a Euclidean space, as demonstrated [23], proves challenging to meet this requirement.

Alternatively, when all cameras are forward-facing toward the scene, NeRF utilizes normalized device coordinates (NDCs) [69] for 3D space reparameterization. While effective for forward-facing scenes, this reparameterization strategy limits possible viewpoints and is unsuitable for unbounded scenes.

To overcome this challenge, NeRF++ [68] employs the inverted sphere parameterization. This approach maps coordinates in Euclidean space to a new space, as illustrated in Figure 6, offering a solution to the limitations imposed by the traditional Euclidean and NDC approaches.

The fundamental concept behind this parameterization strategy involves a division of 3D space into two distinct regions: an inner unit sphere encapsulating the foreground and an outer space accommodating the background. The re-parameterization process is selectively applied solely in outer space. To model these two regions effectively, the framework employs two MLPs separately, each dedicated to capturing the characteristics of the inner sphere and outer space. In practical terms, to determine the color of a pixel, Equation (4) is applied independently for the two MLPs, with the results then merged in a final composition step.

Specifically, as shown in Figure 6, the inverted sphere parameterization in outer space reparameterizes a 3D point (x, y, z), where $r = \sqrt{x^2 + y^2 + z^2} > 1$, into a quadruple $(x', y', z', 1/r)$. The resulting unit vector (x', y', z') satisfies $x'^2 + y'^2 + z'^2 = 1$, indicating its orientation along the same direction as the original point (x, y, z) and representing a direction on the sphere.

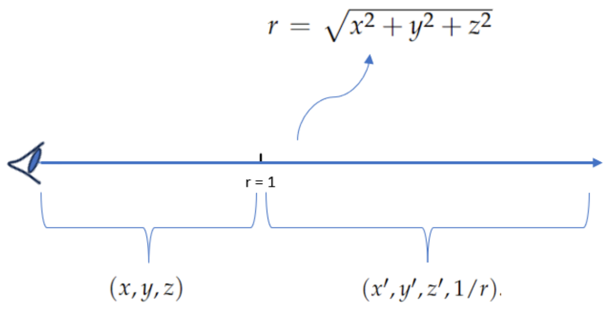

Figure 6. Inverted sphere parameterization, illustrating the scene parameterization, showcasing a mapping of coordinates from Euclidean space to a new spatial representation. Specifically, the coordinates in $r < 1$ are not changed, while the coordinates in $r > 1$ are mapped to $(x', y', z', 1/r)$.

In contrast to Euclidean space, where distances can extend infinitely, the parameterized quadruple imposes bounds: $x', y', z' \in [-1, 1]$, and $1/r \in [0, 1]$. This finite numerical range encapsulates all values within manageable limits. Furthermore, the parameterization acknowledges that objects at greater distances should exhibit lower resolution. By incorporating $1/r$ into the scheme, it naturally accommodates reduced resolution for farther objects. This not only preserves the scene's geometric characteristics but also establishes a foundation for subsequent modeling and computational processes.

With the inverted sphere parameterization, the integral Equation (8) can be updated as follows:

$$C(r) = \int_{t_0}^{t'} \sigma(\mathbf{o} + t\mathbf{d}) \cdot c(\mathbf{o} + t\mathbf{d}, \mathbf{d}) \cdot e^{-\int_{s=0}^{t} \sigma(\mathbf{o}+s\mathbf{d})ds} dt$$
$$+ e^{-\int_{s=0}^{t'} \sigma(\mathbf{o}+s\mathbf{d})ds} \cdot \int_{t'}^{\infty} \sigma(\mathbf{o} + t\mathbf{d}) \cdot c(\mathbf{o} + t\mathbf{d}, \mathbf{d}) \cdot e^{-\int_{s=t'}^{t} \sigma(\mathbf{o}+s\mathbf{d})ds} dt$$

where $\mathbf{o} + t\mathbf{d}$ refers to the line to be integrated; $\mathbf{o}$ is an observation point; $\mathbf{d}$ is the viewing direction.

Mip-NeRF [60] introduces a fundamental transformation to the integral process, the starting point in the computational observation process. The main change provided with Mip-NeRF is integration along 3D cones originating from each pixel instead of a line, as shown in Figure 7. The integration along these cones captures a broader set of spatial relationships, providing a distinctive characteristic. This adjustment in the observation process results in images with reduced aliasing artifacts. By integrating along cones, Mip-NeRF extends its reach to a richer set of spatial relationships within the scene. This departure from line integration allows for a more comprehensive observation of the field representation.

Traditionally, the field was integral along each line, as shown in Equation (8), capturing information discretely along a one-dimensional path. Mip-NeRF, however, redefines this approach by integrating along a 3D cone originating from every pixel. With the alteration from line to cone, the subsequent point sampling process undergoes a corresponding shift. Instead of uniformly sampling points along a line, Mip-NeRF adapts its sampling strategy to efficiently capture information within these 3D cones. This adjustment is reflected in the use of multivariate Gaussian distributions to approximate intervals within the conical frustums. Additionally, instead of mapping a point in 3D space to a positional encoding feature, Mip-NeRF maps each conical frustum to an integrated positional encoding (IPE).

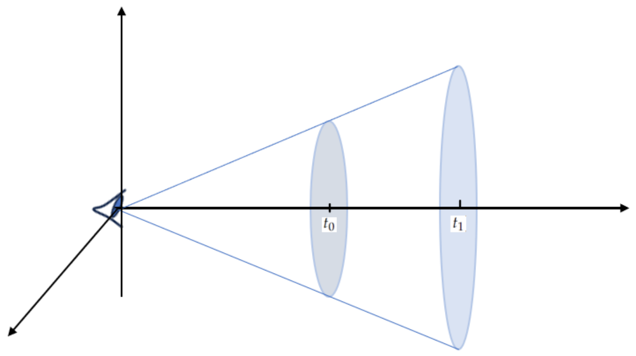

Figure 7. Integration process along a 3D cone. Compared to integrating along a line, which only utilizes the information in the line, integrating along a cone, which utilizes the information within a 3D conical frustum, captures more information.

The discretization process of Mip-NeRF involves dividing the cast cone into a series of conical frustums, essentially cones cut perpendicular to their axis. However, the utilization of a 3D conical frustum as input is not a straightforward task. To address this challenge, Mip-NeRF employs a multivariate Gaussian to adeptly approximate the characteristics of the conical frustum. A Gaussian is employed to fully characterize the conical frustum with three essential values: the mean distance along the line μ_t, the variance along the line σ_t^2, and the variance perpendicular to the line σ_r^2. A midpoint $t_\mu = (t_0 + t_1)/2$ and a half-width $t_\delta = (t_1 - t_0)/2$ are utilized to parameterize the quantities. This process is as follows:

$$\mu_t = t_\mu + \frac{2t_\mu t_\delta^2}{3t_\mu^2 + t_\delta^2}$$

$$\sigma_t^2 = \frac{t_\delta^2}{3} - \frac{4t_\delta^4(12t_\mu^2 - t_\delta^2)}{15(3t_\mu^2 + t_\delta^2)^2}$$

$$\sigma_r^2 = r^2(\frac{t_\mu^2}{4} + \frac{5t_\delta^2}{12} - \frac{4t_\delta^4}{15(3t_\mu^2 + t_\delta^2)})$$

where t_0 and t_1 correspond to the start and end depth of a conical frustum, as shown in Figure 7, and r refers to the width of the pixel in world coordinates scaled by $2/\sqrt{12}$.

Crucially, as shown in Equations (10) and (11), the coordinate frame of the conical frustum, now represented by the Gaussian, transforms world coordinates. After the coordination transformation, IPE is applied to obtain the encoding of each conical frustum. Since this is not our focus, we do not provide further details here.

$$\mu = \mathbf{o} + \mu_t \mathbf{d} \tag{10}$$

$$\mathbf{\Sigma} = \sigma_t^2(\mathbf{d}\mathbf{d}^T) + \sigma_r^2(\mathbf{I} - \frac{\mathbf{d}\mathbf{d}^T}{\|\mathbf{d}\|_2^2}) \tag{11}$$

In the earlier discussion, the field measuring process was designed around the distribution of physical properties. Conversely, when the observation emphasizes the geometric structure of scene objects, necessary adjustments should be implemented in the computational model. When observing a field with a specified surface, leveraging geometric information can improve the accuracy of the observation.

An example in the radiance field is the NeuS model and its follow-ups [43,70–78]. NeuS extracts the surface by approximating a signed distance function (SDF) in the radiance field. With the extracted surface, the observation process is different from that of the original NeRF.

Before delving into the measuring process in NeuS, it is crucial to comprehend how a surface is represented in the radiance field. The surface S of the objects in a scene is represented by a zero-level set of its SDF. As previously mentioned, σ in Equation (8) denotes the quantity value illustrating the volume's propensity to scatter photons. For surface reconstruction, NeuS introduces a weight function $w(t)$ that signifies the contribution of a point's color to the final color of a pixel by combining σ and T. The measuring process can then be expressed as Equation (12). NeuS asserts that the weight function should satisfy two conditions for surface approximation: unbiased and occlusion-aware.

$$C(r) := \int_0^{+\infty} w(t)c(r(t), d)dt \tag{12}$$

The "unbiased" criterion necessitates a weight function that ensures the intersection of the integrating line with the zero-level set of the SDF contributes the most to the pixel color. The "occlusion-aware" property of the weight function requires that when a ray passes multiple surfaces sequentially, the integration process correctly utilizes the color of the surface nearest to the device to compute the output color. NeuS formulates the weight function as Equation (13) to make the function unbiased in the first-order approximation of the SDF.

$$w(t) = \frac{\phi_s(f(\mathbf{p}(t)))}{\int_0^{+\infty} \phi_s(f(\mathbf{p}(u)))du} \tag{13}$$

where $\mathbf{p}(t)$ refers to a point in an integral line.

To render the weight function occlusion-aware, NeuS introduces an opaque density function $\rho(t)$ to replace $\sigma(t)$ in the standard observation process. The weight function can then be expressed as follows:

$$w(t) = T(t)\rho(t) \tag{14}$$

$$T(t) = e^{(-\int_0^t \rho(u)du)} \tag{15}$$

By combining Equations (13)–(15), $\rho(t)$ can be obtained:

$$\rho(t) = \frac{-\frac{d\phi_s}{dt}(f(\mathbf{p}(t)))}{\Phi_s(f(\mathbf{p}(t)))}$$

4.2. Discretization of the Integration

As mentioned previously, NeRF++ proposes a reparameterization strategy achieved by extending the concepts in NDC to handle unbounded scenes. However, the efficacy of this strategy relies on points sampled from a one-dimensional line. With Mip-NeRF, this line evolves into a 3D cone, necessitating an extension from NeRF++'s original parameterization. The challenge arises because the reparameterization strategy designed by NeRF++ is not straightforward for the extended line-to-cone transition in Mip-NeRF. In response to this challenge, Mip-NeRF360 [67] introduces a parameterization strategy and a sampling scheme. Departing from the conventional approach of NeRF++, this strategy harmonizes with the 3D cone structure, offering an adaptive and optimized strategy for sampling distances, extending the integral range compared with that of Mip-NeRF.

Technically, Mip-NeRF360 contributes to the observation model in two main aspects. Firstly, it introduces a parameterization method for the 3D scene, mapping 3D coordinates to an alternate domain. Secondly, it provides an approach for sampling t in the discretization process.

To achieve Gaussian reparameterization, Mip-NeRF360 leverages the classical extended Kalman filter. The coordinate transformation is outlined in Equation (16). Applying this operator to a 3D space maps the coordinates of an unbounded scene to a ball with a radius of two. Similar to NeRF++, points within a radius of one remain unaffected by this transformation.

$$contract(x) = \begin{cases} x & \|x\| \leq 1 \\ (2 - \frac{1}{\|x\|})(\frac{x}{\|x\|}) & \|x\| > 1 \end{cases} \quad (16)$$

Additionally, the sampling scheme is designed by introducing a parameterization in the 3D space. In a bounded scene, NeRF employs uniform point sampling from the near to far planes during the coarse stage. When employingn NDC parameterization, the uniformly spaced series of samples in the sampling process becomes uniformly spaced in inverse depth, fitting well with forward-facing scenes. However, this approach proves unsuitable for an unbounded scene in all directions. Therefore, Mip-NeRF360 suggests a sampling strategy, linearly sampling distances t in disparity, as proposed by Mildenhall et al. [79]. Barron et al. achieved this by mapping the Euclidean distance t to another distance s, establishing the relationship as follows:

$$s \triangleq \frac{g(t) - g(t_n)}{g(t_f) - g(t_n)}$$

$$t \triangleq g^{-1}(s \cdot g(t_f) + (1 - s) \cdot g(t_n))$$

where $g(\cdot)$ represents an invertible scalar function. To attain samples that are linearly distributed in disparity within the t space, Mip-NeRF360 chooses $g(x) = 1/x$. This choice ensures that the samples become uniformly distributed in s space.

In practical implementations, Mip-NeRF360 modifies the coarse-to-fine sampling strategy. Unlike NeRF, which calculates weights based on coarse-stage results and subsequently utilizes them repeatedly in the fine-stage sampling procedure, Mip-NeRF360 adopts a distinct approach. It recommends incorporating an additional MLP exclusively for predicting weights, excluding color prediction. The predicted weights are then used to sample s intervals.

Significantly, the supplementary MLP in Mip-NeRF360 is smaller than the NeRF MLP, specifically designed to constrain the weights generated by the NeRF MLP. As a result, the resampling process in Mip-NeRF360 involves reduced computational complexity compared to NeRF, owing to the considerably smaller size of the additional MLP compared to the entirety of the NeRF MLP.

4.3. Computational Process of Summing Over Sampled Points

Numerical integration is commonly used for computing integrals in practice, including Riemann sums, quadratures, or Monte Carlo methods [80]. Based on these, authors [61]introduced a framework called AutoInt, which utilizes neural networks to approximate integrals in the evaluation stage. As shown in Equation (8), the integral involves two nested integrations: line integration and transmission, Equation (4), which is not trivial to approximate. Therefore, AutoInt divides each line into piecewise sections. Following this division, the observing integral can be refined as follows:

$$C(r) := \sum_{i=1}^{N} \bar{\sigma}_i \bar{c}_i \bar{T}_i \delta_i$$

$$T(t) := e^{-\sum_{j=1}^{i-1} \bar{\sigma}_j \delta_j}$$

where

$$\bar{\sigma}_i = \delta_i^{-1} \int_{t_{i-1}}^{t_i} \sigma(t) dt$$

$$\bar{c}_j = \delta_i^{-1} \int_{t_{i-1}}^{t_i} c(t) dt$$

where $\delta_i = t_i - t_{i-1}$ is the length of each section. After dividing the line into sections, AutoInt is applied to evaluate each integral.

5. Conclusions

In conclusion, this review focused on data-driven field models, exploring diverse implicit fields grounded in various computational structures. Different field approximators were explored, including neural-network-based and parametric-grid-based approximations for practical field functions, demonstrating the capabilities and flexibility of using neural networks to handle various fields. Specifically, we summarized two approaches based on how the field models are updated:

- Global model with shared parameters: The most common choice in this way is neural networks, which update the entire field. It tends to smooth out local changes or details in the data and may face challenges in capturing and representing small variations. Also, its time consumption has been shown to be heavy in the studies mentioned above.
- Local model with parametric grid: Regardless of the smallest unit of a grid, such as an indexed cube or an element in tensor, in the grid-based method, the field space is arranged, offering efficient spatial querying, allowing for the rapid retrieval of objects or data points within a specific region of interest. When a change is observed, only its surrounding area needs to be updated. Furthermore, this approach allows for skipping or pruning unimportant areas and the adaptive concentration of parameters in areas containing more significant and informative data. However, this approach may not be suitable for certain physical phenomena, such as light phenomena, which often require additional techniques like spherical harmonics for accurate representation [33]. Additionally, the resolution limitations of the grid can restrict its ability to capture fine-grained details, and the memory requirements can grow rapidly as the complexity of the problem increases.

Our examination also extended beyond the theoretical frameworks, delving into the intricate details of measuring models designed for the field functions. Furthermore, the critical challenges and design choices of observation processes were discussed, including the integration adjustments, discretization of the integration, and computational strategies. Following our classification framework, we identified key research problems, with each aspect representing a distinct avenue for further exploration:

- In the area of integration adjustments, our synthesis of the literature revealed the diverse reparameterization strategies employed to accommodate varying scene scopes. This encompasses scenarios ranging from a limited depth of the scene to an infinite depth and extends to the intricate challenges posed by a 360-degree unbounded scene.
- When it comes to the discretization of integration, our exploration encompasses a discussion on how existing methods strategically sample from space. This sampling is intricately tied to the specific design choices made in integration adjustments, forming a crucial aspect of the computational framework.
- In describing the computational processes, we delved into the widely adopted method and the nuanced numerical integration approach embraced by AutoInt.

In essence, our review not only identifies existing research problems but also serves as a guide for future investigations, offering a roadmap for researchers to navigate and contribute to the evolving field of spatial attribute representation.

Author Contributions: Conceptualization, W.H., S.Z. and J.L.; methodology, W.H., S.Z. and J.L.; formal analysis, W.H., S.Z. and J.L.; writing—original draft preparation, W.H., S.Z. and J.L.; writing—review and editing, W.H. and S.Z.; visualization, W.H. and S.Z.; supervision, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China Scholarship Council grant number 202108200020.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Peskin, M.E.; Schroeder, D.V. *An Introduction To Quantum Field Theory*; CRC Press: Boca Raton, FL, USA, 1995.
2. Anastopoulos, C.; Plakitsi, M.E. Relativistic Time-of-Arrival Measurements: Predictions, Post-Selection and Causality Problems. *Foundations* **2023**, *3*, 724–737. [CrossRef]
3. Santos, E. Stochastic Interpretation of Quantum Mechanics Assuming That Vacuum Fields Are Real. *Foundations* **2022**, *2*, 409–442. [CrossRef]
4. Misner, C.; Thorne, K.; Wheeler, J.; Kaiser, D. *Gravitation*; Princeton University Press: Princeton, NJ, USA, 2017.
5. Olver, P.J. *Introduction to Partial Differential Equations*; Springer: Cham, Switzerland, 2014; Volume 1.
6. Mikki, S. On the Topological Structure of Nonlocal Continuum Field Theories. *Foundations* **2022**, *2*, 20–84. [CrossRef]
7. Saari, P.; Besieris, I.M. Conditions for Scalar and Electromagnetic Wave Pulses to Be "Strange" or Not. *Foundations* **2022**, *2*, 199–208. [CrossRef]
8. Liu, G.; Quek, S. *Finite Element Method: A Practical Course*; Elsevier Science: Oxford, UK, 2003.
9. Koschier, D.; Bender, J.; Solenthaler, B.; Teschner, M. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. In Proceedings of the 40th Annual Conference of the European Association for Computer Graphics, Eurographics 2019-Tutorials, Genoa, Italy, 6–10 May 2019.
10. Hornik, K.; Stinchcombe, M.; White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
11. Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; Battaglia, P. Learning to Simulate Complex Physics with Graph Networks. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 8459–8468.
12. Li, L.; Wang, L.G.; Teixeira, F.L.; Liu, C.; Nehorai, A.; Cui, T.J. DeepNIS: Deep Neural Network for Nonlinear Electromagnetic Inverse Scattering. *IEEE Trans. Antennas Propag.* **2018**, *67*, 1819–1825. [CrossRef]
13. Nowack, P.; Braesicke, P.; Haigh, J.; Abraham, N.L.; Pyle, J.; Voulgarakis, A. Using Machine Learning to Build Temperature-based Ozone Parameterizations for Climate Sensitivity Simulations. *Environ. Res. Lett.* **2018**, *13*, 104016. [CrossRef]
14. Zhong, C.; Jia, Y.; Jeong, D.C.; Guo, Y.; Liu, S. Acousnet: A Deep Learning Based Approach to Dynamic 3D Holographic Acoustic Field Generation from Phased Transducer Array. *IEEE Robot. Autom. Lett.* **2021**, *7*, 666–673. [CrossRef]
15. Zang, G.; Idoughi, R.; Li, R.; Wonka, P.; Heidrich, W. IntraTomo: Self-supervised Learning-based Tomography via Sinogram Synthesis and Prediction. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 1940–1950.
16. Wu, Q.; Feng, R.; Wei, H.; Yu, J.; Zhang, Y. Self-supervised Coordinate Projection Network for Sparse-view Computed Tomography. *IEEE Trans. Comput. Imaging* **2023**, *9*, 517–529. [CrossRef]
17. Li, H.; Chen, H.; Jing, W.; Li, Y.; Zheng, R. 3D Ultrasound Spine Imaging with Application of Neural Radiance Field Method. In Proceedings of the 2021 IEEE International Ultrasonics Symposium (IUS), Xi'an, China, 11–16 September 2021; pp. 1–4.
18. Shen, L.; Pauly, J.M.; Xing, L. NeRP: Implicit Neural Representation Learning with Prior Embedding for Sparsely Sampled Image Reconstruction. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 770–782. [CrossRef]
19. Reed, A.W.; Blanford, T.E.; Brown, D.C.; Jayasuriya, S. Implicit Neural Representations for Deconvolving SAS Images. In Proceedings of the IEEE OCEANS 2021: San Diego–Porto, San Diego, CA, USA, 20–23 September 2021; pp. 1–7.
20. Pumarola, A.; Corona, E.; Pons-Moll, G.; Moreno-Noguer, F. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 10318–10327.
21. Huang, X.; Zhang, Q.; Ying, F.; Li, H.; Wang, X.; Wang, Q. HDR-NeRF: High Dynamic Range Neural Radiance Fields. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 18377–18387.
22. Liu, J.W.; Cao, Y.P.; Mao, W.; Zhang, W.; Zhang, D.J.; Keppo, J.; Shan, Y.; Qie, X.; Shou, M.Z. DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 36762–36775.
23. Mildenhall, B.; Srinivasan, P.; Tancik, M.; Barron, J.; Ramamoorthi, R.; Ng, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Cham, Switzerland, 2020; Volume 12346, pp. 405–421.
24. Sitzmann, V.; Martel, J.N.P.; Bergman, A.W.; Lindell, D.B.; Wetzstein, G. Implicit Neural Representations with Periodic Activation Functions. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7462–7473.
25. Tancik, M.; Srinivasan, P.P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.T.; Ng, R. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7537–7547.
26. Sun, Y.; Liu, J.; Xie, M.; Wohlberg, B.; Kamilov, U.S. CoIL: Coordinate-Based Internal Learning for Tomographic Imaging. *IEEE Trans. Comput. Imaging* **2021**, *7*, 1400–1412. [CrossRef]
27. Garbin, S.J.; Kowalski, M.; Johnson, M.; Shotton, J.; Valentin, J.P.C. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 14326–14335.
28. Liu, L.; Gu, J.; Lin, K.Z.; Chua, T.S.; Theobalt, C. Neural Sparse Voxel Fields. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 15651–15663.

29. Sun, C.; Sun, M.; Chen, H.T. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5449–5459.
30. Müller, T.; Evans, A.; Schied, C.; Keller, A. Instant Neural Graphics Primitives with A Multiresolution Hash Encoding. *ACM Trans. Graph.* **2022**, *41*, 102. [CrossRef]
31. Zha, R.; Zhang, Y.; Li, H. NAF: Neural Attenuation Fields for Sparse-View CBCT Reconstruction. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Singapore, 18–22 September 2022; Springer: Cham, Switzerland, 2022; pp. 442–452.
32. Xu, J.; Moyer, D.; Gagoski, B.; Iglesias, J.E.; Grant, P.E.; Golland, P.; Adalsteinsson, E. NeSVoR: Implicit Neural Representation for Slice-to-Volume Reconstruction in MRI. *IEEE Trans. Med. Imaging* **2023**, *42*, 1707–1719. [CrossRef]
33. Yu, A.; Fridovich-Keil, S.; Tancik, M.; Chen, Q.; Recht, B.; Kanazawa, A. Plenoxels: Radiance Fields without Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5491–5500.
34. Green, R. Spherical Harmonic Lighting: The Gritty Details. In Proceedings of the Archives of the Game Developers Conference, San Jose, CA, USA, 4–8 March 2003; Volume 56, p. 4.
35. Rückert, D.; Wang, Y.; Li, R.; Idoughi, R.; Heidrich, W. NeAT: Neural Adaptive Tomography. *ACM Trans. Graph.* **2022**, *41*, 55. [CrossRef]
36. Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; Su, H. TensoRF: Tensorial Radiance Fields. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Cham, Switzerland, 2022; pp. 333–350.
37. Hitchcock, F.L. The Expression of a Tensor or a Polyadic as a Sum of Products. *J. Math. Phys.* **1927**, *6*, 164–189. [CrossRef]
38. Zhang, K.; Kolkin, N.I.; Bi, S.; Luan, F.; Xu, Z.; Shechtman, E.; Snavely, N. ARF: Artistic Radiance Fields. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Cham, Switzerland, 2022; pp. 717–733.
39. Shao, R.; Zheng, Z.; Tu, H.; Liu, B.; Zhang, H.; Liu, Y. Tensor4D: Efficient Neural 4D Decomposition for High-Fidelity Dynamic Reconstruction and Rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16632–16642.
40. Jin, H.; Liu, I.; Xu, P.; Zhang, X.; Han, S.; Bi, S.; Zhou, X.; Xu, Z.; Su, H. TensoIR: Tensorial Inverse Rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 165–174.
41. Khan, A.; Ghorbanian, V.; Lowther, D. Deep Learning for Magnetic Field Estimation. *IEEE Trans. Magn.* **2019**, *55*, 7202304. [CrossRef]
42. Kajiya, J.T.; Herzen, B.V. Ray Tracing Volume Densities. In Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, Minneapolis, MN, USA, 23–27 July 1984.
43. Wang, P.; Liu, L.; Liu, Y.; Theobalt, C.; Komura, T.; Wang, W. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 27171–27183.
44. Wizadwongsa, S.; Phongthawee, P.; Yenphraphai, J.; Suwajanakorn, S. Nex: Real-time view synthesis with neural basis expansion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8534–8543.
45. Yariv, L.; Gu, J.; Kasten, Y.; Lipman, Y. Volume Rendering of Neural Implicit Surfaces. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4805–4815.
46. Oechsle, M.; Peng, S.; Geiger, A. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 5569–5579.
47. Goel, S.; Gkioxari, G.; Malik, J. Differentiable Stereopsis: Meshes from multiple views using differentiable rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 8625–8634.
48. Worchel, M.; Diaz, R.; Hu, W.; Schreer, O.; Feldmann, I.; Eisert, P. Multi-View Mesh Reconstruction with Neural Deferred Shading. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 6177–6187.
49. Munkberg, J.; Hasselgren, J.; Shen, T.; Gao, J.; Chen, W.; Evans, A.; Müller, T.; Fidler, S. Extracting Triangular 3D Models, Materials, and Lighting From Images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 8270–8280.
50. Chen, Z.; Funkhouser, T.A.; Hedman, P.; Tagliasacchi, A. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16569–16578.
51. Xiang, F.; Xu, Z.; Havsan, M.; Hold-Geoffroy, Y.; Sunkavalli, K.; Su, H. NeuTex: Neural Texture Mapping for Volumetric Neural Rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 7115–7124.

52. Lorensen, W.E.; Cline, H.E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM Siggraph Comput. Graph.* **1987**, *21*, 163–169. [CrossRef]
53. Chen, Z.; Zhang, H. Neural Marching Cubes. *ACM Trans. Graph.* **2021**, *40*, 251. [CrossRef]
54. Remelli, E.; Lukoianov, A.; Richter, S.R.; Guillard, B.; Bagautdinov, T.M.; Baqué, P.; Fua, P. MeshSDF: Differentiable Iso-Surface Extraction. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22468–22478.
55. Gao, J.; Chen, W.; Xiang, T.; Tsang, C.F.; Jacobson, A.; McGuire, M.; Fidler, S. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9936–9947.
56. Shen, T.; Gao, J.; Yin, K.; Liu, M.Y.; Fidler, S. Deep Marching Tetrahedra: A Hybrid Representation for High-Resolution 3D Shape Synthesis. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 6087–6101.
57. Yang, B.; Bao, C.; Zeng, J.; Bao, H.; Zhang, Y.; Cui, Z.; Zhang, G. NeuMesh: Learning Disentangled Neural Mesh-based Implicit Field for Geometry and Texture Editing. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Cham, Switzerland, 2022; pp. 597–614.
58. Tang, J.; Zhou, H.; Chen, X.; Hu, T.; Ding, E.; Wang, J.; Zeng, G. Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement. *arXiv* **2023**, arXiv:2303.02091.
59. Wei, X.; Xiang, F.; Bi, S.; Chen, A.; Sunkavalli, K.; Xu, Z.; Su, H. NeuManifold: Neural Watertight Manifold Reconstruction with Efficient and High-Quality Rendering Support. *arXiv* **2023**, arXiv:2305.17134.
60. Barron, J.T.; Mildenhall, B.; Tancik, M.; Hedman, P.; Martin-Brualla, R.; Srinivasan, P.P. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 5835–5844.
61. Lindell, D.B.; Martel, J.N.P.; Wetzstein, G. AutoInt: Automatic Integration for Fast Neural Volume Rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 14551–14560.
62. Reiser, C.; Peng, S.; Liao, Y.; Geiger, A. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 14315–14325.
63. Hedman, P.; Srinivasan, P.P.; Mildenhall, B.; Barron, J.T.; Debevec, P.E. Baking Neural Radiance Fields for Real-Time View Synthesis. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 5855–5864.
64. Hu, T.; Liu, S.; Chen, Y.; Shen, T.; Jia, J. EfficientNeRF: Efficient Neural Radiance Fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12902–12911.
65. Levoy, M. Efficient Ray Tracing of Volume Data. *ACM Trans. Graph.* **1990**, *9*, 245–261. [CrossRef]
66. Max, N.L. Optical Models for Direct Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* **1995**, *1*, 99–108. [CrossRef]
67. Barron, J.T.; Mildenhall, B.; Verbin, D.; Srinivasan, P.P.; Hedman, P. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5460–5469.
68. Zhang, K.; Riegler, G.; Snavely, N.; Koltun, V. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv* **2020**, arXiv:2010.07492.
69. McReynolds, T.; Blythe, D. *Advanced Graphics Programming Using OpenGL*; Elsevier: Amsterdam, The Netherlands, 2005.
70. Wang, Y.; Han, Q.; Habermann, M.; Daniilidis, K.; Theobalt, C.; Liu, L. NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 3295–3306.
71. Li, Z.; Muller, T.; Evans, A.; Taylor, R.H.; Unberath, M.; Liu, M.Y.; Lin, C.H. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 8456–8465.
72. Zhang, J.; Yao, Y.; Li, S.; Fang, T.; McKinnon, D.N.R.; Tsin, Y.; Quan, L. Critical Regularizations for Neural Surface Reconstruction in the Wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 6260–6269.
73. Yu, Z.; Peng, S.; Niemeyer, M.; Sattler, T.; Geiger, A. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 25018–25032.
74. Fu, Q.; Xu, Q.; Ong, Y.; Tao, W. Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-View Reconstruction. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 3403–3416.
75. Grabocka, J.; Schmidt-Thieme, L. NeuralWarp: Time-Series Similarity with Warping Networks. *arXiv* **2018**, arXiv:1812.08306.
76. Rosu, R.A.; Behnke, S. PermutoSDF: Fast Multi-View Reconstruction with Implicit Surfaces Using Permutohedral Lattices. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada,17–24 June 2023; pp. 8466–8475.
77. Wang, Y.; Skorokhodov, I.; Wonka, P. Improved surface reconstruction using high-frequency details. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1966–1978.

78. Zhang, Y.; Hu, Z.; Wu, H.; Zhao, M.; Li, L.; Zou, Z.; Fan, C. Towards Unbiased Volume Rendering of Neural Implicit Surfaces with Geometry Priors. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 4359–4368.
79. Mildenhall, B. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Trans. Graph.* **2019**, *38*, 29. [CrossRef]
80. Davis, P.J.; Rabinowitz, P. *Methods of Numerical Integration*; Courier Corporation: North Chelmsford, MA, USA, 2007.