Summary of Implicit Neural Representations

Jing Li

Abstract

Conventional representations of 3D shapes such as point clouds, meshes, and voxel are limited by their discrete nature and associated drawbacks. However, there has been a growing interest among researchers in the utilization of implicit neural representations(INRs) as an alternative approach. This summary aims to provide a comprehensive overview of the general definition of INRs and highlights various influential neural network architectures that have been applied in the field of computer graphics.

1 3D Shape Representation

1.1 Traditional Representations

Traditional 3D shape representations mainly include point clouds, meshes, and voxel. Point cloud representation utilizes a collection of points in 3D space to depict the shape of an object (Figure 1(a)). Although it is lightweight, it lacks topological connectivity due to its discrete nature. Mesh representation, on the other hand, involves approximating the surface of an object using segmented triangular patches (Figure 1(b)). Two commonly employed methods are utilized for this purpose. The first method involves predefining a template grid[1] and mapping it to the object shape that needs to be represented. The second method employs parameterization algorithms[2, 3] to transform the 2D plane shape into a 3D object. However, a major drawback of these transformations is that the resulting mesh representation remains topologically invariant. In other words, for objects with different topological structures, specific template grids or 2D plane shapes with corresponding topological structures must be designed. Voxel representation, on the other hand, employs regular rectangular cuboid blocks of the same size to represent objects (Figure 1(c)). This can be seen as an extension of 2D image pixel blocks. However, voxel representation has a significant disadvantage in that the memory occupied increases exponentially with the voxel resolution. Consequently, it is only suitable for low-resolution shape representation. Table 1 provides a summary of the drawbacks associated with these three common shape representations.

(a) point (b) mesh (c) voxel

Figure 1: Traditional 3D shape representations.

Shape Representations	Disadvantages
Point	Lack of topological connectivity
Mesh	Difficulty in representing complex topological structures
voxel	Large memory usage and time consumption

Table 1: The main disadvantages of three traditional shape representations.

1.2 Implicit Representations

Traditional shape representations are discretized rather than being represented continuously. However, the real world exists in a continuous manner, and therefore, implicit representation is employed as a continuous representation. Implicit representation commonly used for closed surfaces include the signed distance field SDF and the occupancy function. The SDF represents the directed distance from a point in space to the surface of the object, with its zero isosurface corresponding to the object surface. On the other hand, the occupancy function defines the decision boundary of the object surface. Throughout the subsequent discussion, it is important to note that the closed surface is denoted as S. The signed distance field, denoted as $f : \mathbb{R}^3 \to \mathbb{R}$, expresses the directed distance from a point in 3D space to the surface S. The sign of f depends on whether the point is located inside or outside S, allowing S to be implicitly represented in the following form:

$$S = \{x : f(x) = 0\}$$

In fact, SDF is still the solution to the Eikonal equation. Assuming that S is derived from the surface point cloud χ and the corresponding normal vector N is determined, then the solution of following equation

$$\begin{cases} \|\nabla f\| = 1 \\ f|_\chi = 0 \\ \nabla f|_\chi = N \end{cases}$$

is the signed distance field of S. Some work of neural implicit representations is to construct loss function on this basis, which will be mentioned later.

In addition to SDF, the occupancy function $g : \mathbb{R}^3 \to \{0, 1\}$ is also a commonly used implicit function, which is defined as

$$g(x) = \begin{cases} 0, & x \textit{ outside } S \\ 1, & x \textit{ inside } S \end{cases}$$

In the context of deep learning, g is is characterized by its continuity, that is, the function g takes on values within the entire continuous interval $[0, 1]$. Consequently, the set S can be determined by the decision boundary τ, which lies within the range of $[0, 1]$.

$$S = \{x : g(x) = \tau\}$$

It is worth noting that different values of τ correspond to varying thicknesses of S. To visualize the surface represented by its implicit form, the commonly employed algorithm is Marching cubes[4].

1.3 Implicit Neural Representations

With the remarkable advancements of deep learning in various domains, there has been a growing interest in utilizing deep learning techniques for the representation of 3D shapes. Notably, several approaches have been developed, including the point cloud based neural network architecture PointNet[5], the grid based neural network architecture Surfnet[6], as well as deep learning methods employed in voxel representation[7, 8], and other deep learning techniques applied to traditional shape representation[9, 10, 11, 12]. These approaches primarily involve discrete representations. However, in recent years, researchers have also explored the application of deep learning to implicit expression, resulting in the development of implicit neural representations (INRs). The second chapter of this paper will provide a specific introduction to INRs.

2 Construction Methods

The initial research on INRs was introduced in a series of three papers in 2019[13, 14, 15], which sparked a significant surge of interest in the field. The fundamental concept behind INRs involves training an implicit expression, denoted as f, through the utilization of neural networks $f_\theta : \mathbb{R}^3 \to \mathbb{R}$. These neural networks can take the form of signed distance fields SDF or occupancy functions, where the input corresponds to the point coordinates and the output represents the SDF value or Occupancy value of the given point. Here, θ represents the parameters of the neural network. Consequently, the development of INRs is closely tied to the broader problem of constructing neural networks, determining appropriate training methods, and defining an appropriate loss function.

2.1 Three Classic INRs

This section briefly introduces the basic methods of the three classic works in 2019.

2.1.1 IM-Net

The pioneering work of IM-Net[13]introduced the utilization of neural networks for the construction of implicit representation. The fundamental concept behind this approach involves the implementation of a "encoder-decoder" neural network architecture to facilitate the training of internal parameters. The objective is to minimize the loss function, which represents the sum of the discrepancies between the output occupancy value and the actual value on the observation data set.
Input: Points, Mesh, Voxel(Images)
Output: Occupancy value
Loss(Mean Squared Error):

$$L(\theta) = \frac{\sum_{p \in S} w_p \left\| f_\theta(p) - F(p) \right\|^2}{\sum_{p \in S} w_p}$$

where f_θ represents the neural networks, θ is the parameters, S is the training set, p is the 3D point in the training set, $F(p)$ is the true Occupancy value of the given point p, w_P is the density at p.
Architecture:

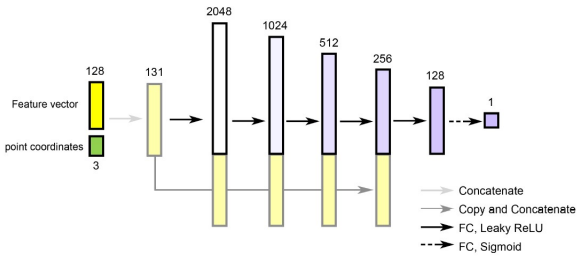

Figure 2: Decoder in IM-Net.

Main ideas: The IM-Net utilizes an encoder-decoder architecture, employing different architectures depending on the input. The author did not create a new encoder, but rather utilized existing encoders. The primary role of the encoder is to learn shape features, such as sharp corners and topology. In this case, the encoder produces 128-dimensional feature vectors. These feature vectors are then concatenated with point coordinates to form a 131-dimensional vector, which serves as the input for the decoder. Figure 2 illustrates the author's design for the decoder, which generates a 1-dimensional Occupancy value as output. The true Occupancy value for each point is present in the training set, allowing for continuous updating of the network parameters through the defined loss function to obtain the final IM-Net.

2.1.2 OccNet

OccNet[14], similar to IM-Net, also employs an Occupancy function and utilizes an encoder-decoder architecture. Nevertheless, there are distinctions between the two approaches in terms of their

decoder design and the formulation of the loss function.
Input: Points, Images, Voxel
Output: Occupancy value
Loss(Cross Entropy):

$$L(\theta) = \sum_{p \in S} -[F(p)log(f_\theta(p)) + (1 - F(p))log(1 - f_\theta(p))]$$

where f_θ represents the neural networks, θ is the parameters, S is the training set, p is the 3D point in the training set, $F(p)$ is the true occupancy value of point p, w_P is the density at p.
Architecture:

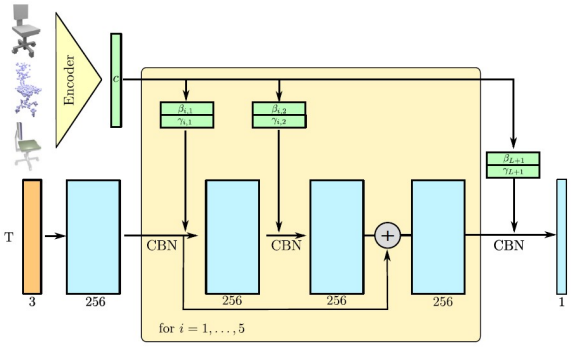

Figure 3: OccNet Architecture.

Multiresolution IsoSurface Extraction:After the completion of the training, an implicit expression was derived, which yielded the Occupancy value as the output. In addition to employing the conventional Marching cubes algorithm for surface extraction, the author introduced a novel approach, namely MISE, for extracting implicit surfaces, as depicted in Figure 4. The algorithm entails the following steps: (1) Calculation of the Occupancy value at each node; (2) Examination of each voxel, and if it contains points with both an Occupancy value of 1 and an Occupancy value of 0, the voxel is marked in red and subdivided into multiple child voxels; (3) Repetition of the aforementioned steps until the termination conditions are satisfied; (4) Extraction of grids using the Marching cubes algorithm; (5) Subdivision of the mesh based on gradient information.

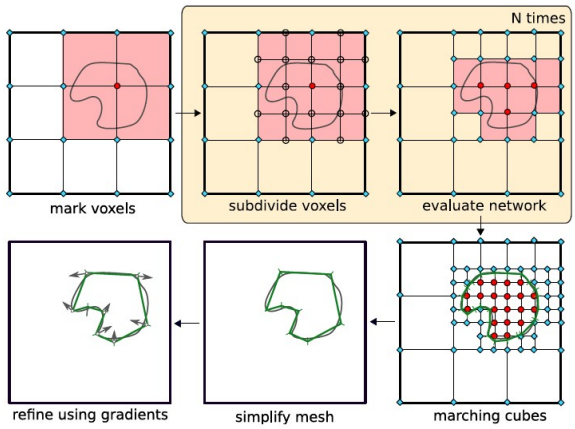

Figure 4: MISE algorithm for extracting implicit surfaces.

2.1.3 DeepSDF

Unlike IM-Net and OccNet, DeepSDF[15] learns the SDF values of surfaces and adopts its own pioneering Auto-decoder neural network architecture.

Input: Points, Mesh, Voxel(Images)
Output: SDF value
Loss:

$$L(\theta) = \sum_{p \in S} \left(g(f_\theta(z, p), F(p)) + \frac{1}{\sigma^2} \|z\|^2 \right)$$

where f_θ is the neural networks, θ is the parameters, S is the training set, p is the 3D point in the training set, $F(p)$ is the true SDF value of point p. The latent code of the input shape is denoted as z, which is a vector of specific length and can be considered as a parameter that requires updating in the neural network. It is important to note that each input shape has its own late code. The function $g(\cdot, \cdot)$ measures the error between the output SDF value $f_\theta(z, p)$ and the true $F(p)$ values:

$$g(f_\theta(z, p), F(p)) = |clamp(f_\theta(z, p), \delta) - clamp(F(p), \delta)|$$
$$clamp(x, \delta) = min\{\delta, max\{-\delta, x\}\}$$

Architecture:

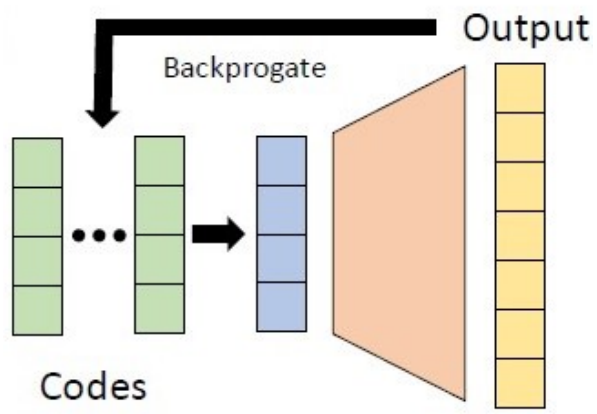

Figure 5: Auto-decoder.

Main ideas: The encoder component of the neural network architecture is eliminated by the author, leaving only the decoder. The decoder takes as input the latent code and point coordinates of the shape. During the training period, both the latent code and network parameters are updated simultaneously. However, in the testing phase, the parameters θ are fixed. To generate a new shape, the latent code is optimized based on the according loss function.

2.2 Unsupervised Learning INRs

This section provides an introduction to the training of INRs without relying on the actual Surface Distance Function SDF value or Occupancy value of the training dataset. Two notable works in this area are SAL[16] and IGR[17]. These works specifically focus on point cloud input and the the loss function. $S \subset \mathbb{R}^3$ represents the point cloud, f_θ denotes the neural network, and θ represents the network parameters. Additionally, it should be emphasized that the point cloud S is sampled on the surface rather than in the entire space.
SAL: The loss function is defined as

$$L(\theta) = \mathbb{E}_x(\tau(f_\theta(p), h_S(x)))$$

where $h_S(x)$ denotes the distance from x to S, and $\tau : \mathbb{R} \times \mathbb{R}_+ \to \mathbb{R}$ is a similarity function which satisfies the following two conditions:
(1) $\tau(-a, b) = \tau(a, b), \forall a \in \mathbb{R}, \forall b \in \mathbb{R}_+$
(2) $\frac{\partial \tau}{\partial a}(a, b) = \rho(a, b)$, ρ is a increasing function and $\rho(0) = 0$

It can be found that the loss function employed by SAL does not require the computation of the actual SDF value or occupancy value of the training data set points. Instead, it directly utilizes the coordinate information of the original input points to train an implicit function, which is a neural network model that operates in an "end-to-end" manner. The metric function h_S is utilized to quantify the distance between a point and the input S. The author highlights that when h_S is chosen as the L_0 norm, the trained outcome corresponds to the occupancy value, whereas selecting h_S as the L_2 norm yields the SDF value as the trained result. The function τ introduces two constraints to capture the similarity between the output value and the true value.

IGR: The construction of IGR loss function comes from Eikonal equation. As mentioned above, SDF is actually the solution of Eikonal equation, so solving SDF according to input is equivalent to solving Eikonal equation. The author defines Loss function as

$$L(\theta) = L_S(\theta) + \lambda \mathbb{E}_x(\|\nabla_x f_\theta(x)\| - 1)^2$$

where

$$L_S(\theta) = \frac{1}{|S|} \sum_{p \in S} \left(|f_\theta(p)| + \tau \left\|\nabla_x f_\theta(p) - n_p\right\|\right)$$

The term $L_S(\theta)$ is designed to ensure that the output of points on the surface is as close as possible to 0 (because their SDF value is 0), and the gradient is as close as possible to the true normal vector n_p is equal, Except for $L_S(\theta)$, the other is to ensure that the modulus of the gradient vector is as long as possible as 1, λ is the trade-off coefficient. The construction of this Loss function is actually derived from Eikonal equation. Unlike SAL, IGR also needs to know the normal vector information of the training data.

2.3 Neural Scene Representations

The initial sections of this study focus on constructing INRs for individual objects or shapes. However, real-world scenes are often more complex, consisting of multiple objects with varying shapes, sizes, and spatial distributions. To address this complexity, previous research has explored methods for constructing INRs in such scenes[18, 19, 20, 21]. For single objects, the later code is typically represented as a simple vector[21]. In large-scale scenes, local neural implicit expressions are commonly used as INRs. These expressions divide an object into multiple overlapping parts, with each part corresponding to a later code. When calculating the SDF value at a specific point, the late codes of the subparts containing the point are separately computed, and the output of the neural network is weighted averaged. This local expression is well-suited for large-scale scenarios.To illustrate the concept of local neural implicit expressions, we will briefly discuss the approach presented in [20]. Figure 6 depicts a 2D image with a $3 * 3$ coordinate grid. The image is divided into four overlapping subregions, each measuring $2 * 2$. During training, each subregion is trained separately, resulting in the acquisition of a late code c_i for each subregion. During testing, for a given point x in the original image, the set of subregions containing point x is denoted as N_x. The final output corresponding to point x is determined by taking the weighted average of the late codes from the subregions in N_x:

$$f(x) = \sum_{i \in N_x} w_i f_\theta(c_i, x_i)$$

where x_i is the local coordinate of x in the region i.

2.4 Texture Reconstruction

The above are all implicit expressions of object shape. In fact, implicit expressions can also be used for texture expression of objects[22, 23, 24], such as color rgb. That is, learning the rgb values at each point:

$$f_\theta : \mathbb{R}^3 \to \mathbb{R}^3$$

The input of f_θ is the coordinates of a point in 3D space, and the output is the rgb value of that point. Similar to the shape representation introduced earlier, a neural network architecture can also be constructed to optimize parameters on the training set, obtaining an implicit expression of the final predicted rgb value.

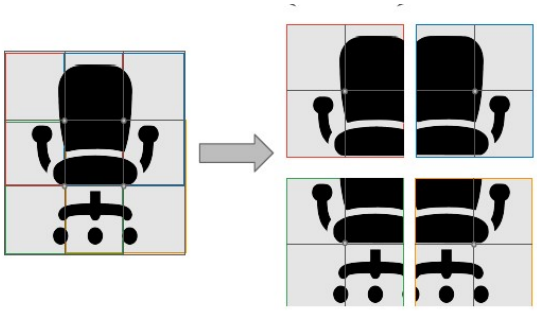

Figure 6: Local Implicit Neural Representations.

For example, the typical job of PIFU[23] is to reconstruct the human body with clothing. In addition to reconstructing the surface of the human body, it is also necessary to reconstruct the color information of each point. Firstly, the human surface is reconstructed, and then the rgb value is reconstructed, as shown in Figure 7. The upper encoder learns the late code of the object shape, while the lower encoder learns the late code of the object color. Finally, the two are combined to obtain the final reconstruction. In addition, LIIF[24] mainly focuses on the reconstruction of 2D color images in the real world, as shown in Figure 8. It implicitly expresses the image using multiple local nerves, but unlike before, the implicit expression here is the rgb value.

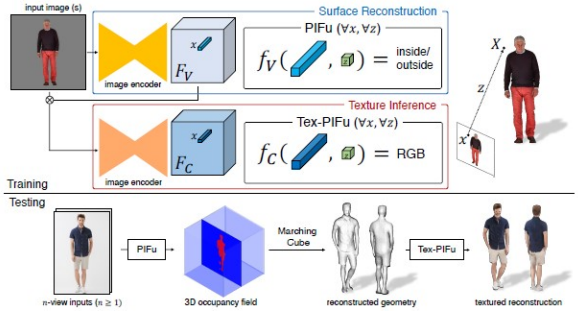

Figure 7: schematic diagram of PIFU.

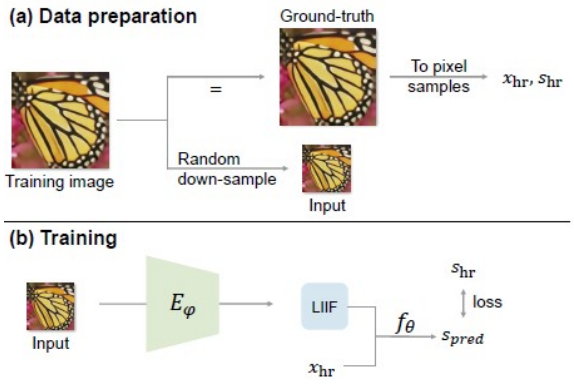

Figure 8: Reconstruction of 2D real images by LIIF.

2.5 Open surface

It has been observed that both SDF and Occupancy representations necessitate the input surface to be a closed surface. However, there have been some studies exploring the implicit representation of

non-closed surfaces[25, 26, 27]. In particular, [25] proposed a Neural Distance Field NDF network that can learn unsigned distance fields without the requirement of the surface being closed. On the other hand, [27] expresses the boundary of the surface explicitly while representing the interior implicitly. By employing geometric measure theory and stochastic gradient optimization methods, the minimum surface problem can be solved, leading to the final implicit expression of the surface with a boundary.

2.6 Learning high-frequency details

In machine learning, Multilayer Perceptron (MLP) models often exhibit a tendency to learn smooth outcomes, which poses challenges in capturing high-frequency details. To address this issue, two commonly employed approaches are available. The first approach involves embedding the inputs into higher dimension space, while the second approach entails selecting alternative activation functions.

2.6.1 Sin periodic activation function

Sitzmann[28] proposed sin periodic activation function, which can better represent details and derivative information in the domain of images and videos. At the same time, this INRs can be used to solve PDE equations, such as Eikonal equation, Poisson equation, etc.

$$f_\theta(x) = W_n(\phi_{n-1} \circ \phi_{n-2} \circ \cdots \phi_0)(x) + b_n$$
$$\phi_i(x) = sin(W_i x + b_i)$$

ϕ_i is the fully connected layer of the i-th layer, w_i is the weight, b_i is the offset. The important factor of sin periodic activation function neural network is the selection of initial parameters value. The author gives a set of initial weight scheme through analysis. The quality of initial weight greatly affects the effect of neural network.

2.6.2 Bspline positional encode

Matthew[29] used Fourier analysis to map low dimensional inputs to high dimensional ones, and then achieved good results through MLP connection (this block is also available in Nerf[30]). Similarly, Wang[31] uses the B-spline position coding method to map the input to high dimensions. Let

$$B^0(x) = \begin{cases} 1, |x| < 0.5 \\ 0, else \end{cases}$$

Then $B^i(x)$ is the i-th convolution of $B^0(x)$, and it can be seen that the support of $B^1(x)$ is $[-1, 1]$, the support of $B^2(x)$ is $[-1.5, 1.5]$, and so on. For a one-dimensional input, divide it equally into K parts to obtain $K+1$ nodes $\{c_i\}_{i=0}^K$, then $B_{\delta,c_i}(x) = B\left(\frac{x-c_i}{\delta}\right)$, where δ is the node spacing, then the B-spline function can be defined as

$$\psi(x) = \sum_{i=0}^{K} W_i B_{\delta,c_i}(x)$$

In the input $x \in \mathbb{R}^d$ of the neural network, denote $D_1, D_2, ..., D_M$ is a set of linear independence unit vector of $\mathbb{R}^d$, and the projection of $x \in \mathbb{R}^d$ is recorded as $x_k = <x, D_K>$, then the B-spline position code is defined as

$$\Psi(x) = (\psi_1(x_1), \psi_2(x_2), \cdots, \psi_M(x_M))$$

3 Applications

INRs refer to a collection of works that employ neural networks to implicitly represent objects or scenes. These techniques have found applications in various domains such as medical imaging, CT scanning, and vascular reconstruction[32, 33, 34]. The underlying approach involves solving partial differential equations using neural implicit representation[35, 36]. Additionally, 4D scene reconstruction, including the generation of human motion sequences, has been achieved[37, 38, 39]. Throughout this period, my primary research focus has been on iterative reconstruction methods, including point

cloud reconstruction, shape completion, shape interpolation, voxel super-resolution (which involves generating high-resolution images from low-resolution inputs), as well as single view and multi-view reconstruction. The majority of the references cited in Sections 1 and 2 also center around iterative reconstruction techniques. The following figures serve as visual representations of the reconstruction outcomes achieved through the use of INRs, as depicted in the literature.

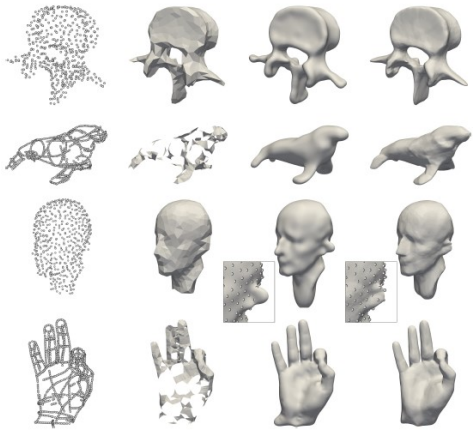

Figure 9: Point Cloud Reconstruction-SAL.

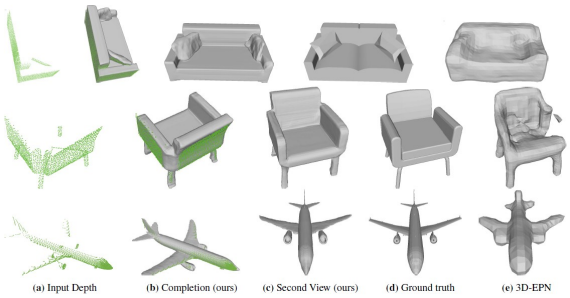

Figure 10: Point Cloud Completion-DeepSDF.

Figure 11: Shape Interpolation-IMNet.

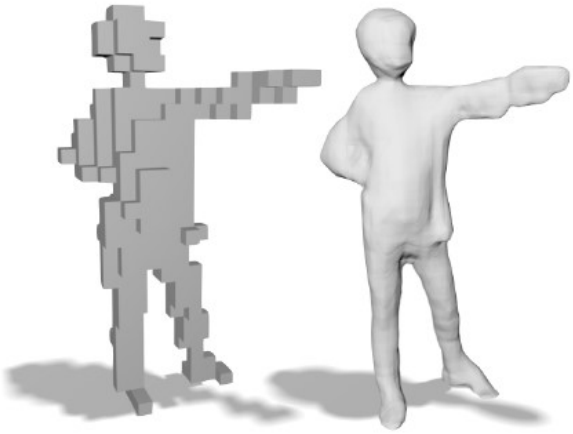

Figure 12: Voxel SuperResolution-IFNet[40].

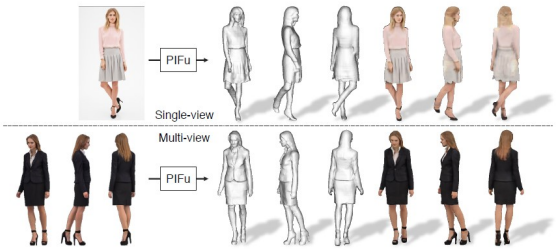

Figure 13: Single view and multi view reconstruction-PIFU.

References

[1] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018.

[2] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[3] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[4] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[6] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6040–6049, 2017.

[7] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 484–499. Springer, 2016.

[8] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 628–644. Springer, 2016.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[11] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 international conference on 3D vision (3DV)*, pages 728–737. IEEE, 2018.

[12] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606, 2018.

[13] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[14] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[15] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[16] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

[17] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[18] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 608–625. Springer, 2020.

[19] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[20] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[21] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020.

[22] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019.

[23] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2304–2314, 2019.

[24] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021.

[25] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020.

[26] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20834–20843, 2023.

[27] David Palmer, Dmitriy Smirnov, Stephanie Wang, Albert Chern, and Justin Solomon. Deepcurrents: Learning implicit representations of shapes with boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18665–18675, 2022.

[28] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[29] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[31] Peng-Shuai Wang, Yang Liu, Yu-Qi Yang, and Xin Tong. Spline positional encoding for learning 3d implicit signed distance fields. *arXiv preprint arXiv:2106.01553*, 2021.

[32] Dieuwertje Alblas, Christoph Brune, Kak Khee Yeung, and Jelmer M Wolterink. Going off-grid: continuous implicit neural representations for 3d vascular modeling. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 79–90. Springer, 2022.

[33] Muhammad Osama Khan and Yi Fang. Implicit neural representations for medical imaging segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 433–443. Springer, 2022.

[34] Qing Wu, Yuwei Li, Yawen Sun, Yan Zhou, Hongjiang Wei, Jingyi Yu, and Yuyao Zhang. An arbitrary scale super-resolution approach for 3-dimensional magnetic resonance image using implicit neural representation. *arXiv e-prints*, pages arXiv–2110, 2021.

[35] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. *arXiv preprint arXiv:2206.02607*, 2022.

[36] Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems*, 34:10368–10381, 2021.

[37] Pablo Cervantes, Yusuke Sekikawa, Ikuro Sato, and Koichi Shinoda. Implicit neural representations for variable length human motion generation. In *European Conference on Computer Vision*, pages 356–372. Springer, 2022.

[38] Weihao Zhuang, Tristan Hascoet, Ryoichi Takashima, and Tetsuya Takiguchi. Optical flow regularization of implicit neural representations for video frame interpolation. *arXiv preprint arXiv:2206.10886*, 2022.

[39] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9054–9063, 2021.

[40] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6970–6981, 2020.