Canonicalization of Neural Fields

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Electronics and Communication Engineering
by Research

by

Rohith Agaram
2021702026
rohith.agaram@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
January 2024

Copyright © Rohith Agaram, 2024
All Rights Reserved

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "*Canonicalization of Neural Fields*" by Rohith Agaram, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. K. Madhava Krishna

To My Family

Acknowledgments

I want to thank my advisor, Professor Madhava Krishna, head of the Robotics Research Center (RRC), IIIT Hyderabad, India, for giving me a spot in his team and introducing the concepts of 3D computer vision. I am also grateful to Professor Srinath Sridhar, the head of the Interactive 3D Vision and Learning Lab (IVL) at Brown University, USA, for shaping the problem statement for this thesis-canonicalization of neural fields, and for generously providing the resources needed for the research. Thank you, Adrien Poulenard, for explaining the math for TFN and all the brainstorming sessions. Thank you, Rahul Sajnani, for helping me get started with the tools to kickstart my research and for providing valuable criticism on my research decisions. Thank you, Shaurya Dewan, for your outstanding collaboration throughout the development cycle of this project and for the countless late-night calls we've had.

Above all, I want to express my sincere thanks to my family for supporting my decision to quit my job and pursue my master's degree. Their encouragement and understanding have meant the world to me, making this academic journey possible and fulfilling in every way.

I want to thank my labmates and friends for being great companions, sharing ideas, and always cheering me on. Their friendship has made my academic journey a lot more fun and meaningful.

I thank the thesis examiners for their valuable feedback to improve this thesis.

Abstract

Among the many 3D representations, Coordinate-based implicit neural networks or neural fields gained much appreciation in recent times for their ability to represent shape and appearance with very high fidelity and accuracy in 3D computer vision. Despite the advances, however, it remained challenging to build generalizable neural fields for the category of the objects without datasets like shapenet that provide "canonicalized" object instances that are consistently aligned for their 3D position and orientation (pose). Aligning the objects in 3D helps in many tasks for better generalization on 3d scene understanding, classification, and segmentation. 3D pose estimation can also be obtained by aligning the objects in the 3D. There are methods that align 3d objects represented as point clouds/meshes. Now that we have a new promising 3d implicit representation, there is a need to develop a method that helps to align the neural-fields so that we can enjoy the same benefits we had in the space of point clouds/meshes. Unlike point clouds/meshes neural-fields are parametrized by deep neural networks which is very hard to interpret. In this thesis, we present Canonical Field Network (CaFi-Net), a self-supervised method to canonicalize the 3D pose of instances from an object category represented as neural fields, specifically neural radiance fields (NeRFs).

Neural-fields, specifically NeRfs describe the 3D scene as a function of density and view-dependent color. Aligning the objects of a category depends on the geometry rather than the color. That's why CaFi-Net uses density alone to align the objects within the category. Canonicalization is tightly coupled with equivariant networks. In this work, we draw inspiration from 3D Equivariant networks and construct a CaFi-Net as an Equivariant network for rotations. This network directly learns from continuous and noisy density fields by employing a Siamese network architecture. Previous work has done this for points, but handling fields, specifically vector fields, require us to consider rotation equivariance in both the position and orientation of the field. So, to incorporate the rotation equivariance in the fields, we chose the gradient of a scalar field density, which is a vector field, as the signal for building the rotation equivariance in the CaFi-Net. We used spherical harmonics as a basic building block for the equivariant convolution kernels for CaFi-Net. To handle the noisy signal, we weighted the features with the density value at the point. We employed density-based clustering for the segregation of the

background and foreground parts, which is utilized in the calculation of the losses. As there is no publicly available dataset, in order to train the CaFi-Net, we created a simulator that renders 54 camera omnidirectional views for 1300 Nerf instances across 13 shapenet object categories.

During inference, our method takes pre-trained neural radiance fields of novel object instances at arbitrary 3D pose and estimates a canonical field with consistent 3D pose across the entire category. As there are no metrics available for canonicalization for neural fields, we used the same metrics used for the point clouds to evaluate the CaFi-Net Performance. Along with the above metrics we have introduced a new metric Ground Truth Equivariance Consistency(GEC) which measures the canonical performance of CaFi-Net to manual labels. Extensive experiments on the above dataset of 1300 NeRF models show that our method matches or exceeds the performance of 3D point cloud-based methods. We conducted ablation studies, which included exploring the choice of the signal, weighing the equivariant features with the density value, assessing the need for the Siamese network, and finally justifying the design choice of the CaFi-Net. In the results section we showed renderings of the Neural-Fields of the object from the canonical pose that are consistent across the category.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Neural fields [58] coordinate-based neural networks that implicitly parameterize signals have recently gained significant attention as representations of 3D shape [28, 23, 6], view-dependent appearance [25, 41], and motion [26].

In particular, neural radiance fields (NeRFs) [25], have been successfully used in problems such as novel view synthesis [3, 4, 65], scene geometry extraction [60, 54], capturing dynamic scenes [29, 30, 51, 33, 20], 3D semantic segmentation [67, 52], and robotics [1, 14, 22]. Despite the progress, it remains challenging to build neural fields that represent an entire category of objects. Previous methods sidestep the problem by overfitting on a single instance [25], or learning [28, 62, 23] on datasets like ShapeNet [5] that contain objects that are manually **canonicalized** – oriented consistently for 3D position and orientation (3D pose) across a category. This strong supervision makes it easier to learn over categories but limits their application to data that contain these labels. Recent work has proposed methods for self-supervised learning of 3D pose canonicalization [42, 46, 39], however, these operate on 3D point clouds, meshes, or voxels but not neural fields. In this paper, we present Canonical Field Network (**CaFi-Net**), a self-supervised method for category-level canonicalization of the 3D position and orientation of objects represented as **neural fields**, specifically neural radiance fields. Canonicalizing neural fields is challenging because, unlike 3D point clouds or meshes, neural fields are **continuous**, noisy, and hard to manipulate since they are parameterized as the weights of a neural network [59]. To address these challenges, we first extend the notion of **equivariance** to continuous vector fields and show how networks for processing 3D point clouds [49] can be extended to operate directly on neural radiance fields. We design CaFi-Net as a Siamese network that contains layers to extract equivariant features directly on vector fields. These field features are used to learn a canonical frame that is consistent across instances in the category. During inference, our method takes as input neural radiance fields of object instances from a category at arbitrary pose and estimates a **canonical field** that is consistent across the cate-

gory. To handle noise in radiance fields from NeRF, our method incorporates density-based feature weighting and foreground-background clustering. Our approach learns canonicalization without any supervision labels on a new dataset of 1300 pretrained NeRF models of 13 common ShapeNet categories in arbitrary 3D pose. We introduce several self-supervision loss functions that encourage the estimation of a consistent canonical pose. In addition, we present extensive quantitative comparisons with baselines and other methods on standardized canonicalization metrics [36] over 13 object categories. In particular, we show that our approach matches or exceeds the performance of 3D point cloud-based methods. This enables the new capability of directly operating on neural fields rather than converting them to point clouds for canonicalization. To sum up, we contribute:

- Canonical Field Network (**CaFi-Net**), the first method for self-supervised canonicalization of the 3D position and orientation (pose) of objects represented as neural radiance fields.
- A Siamese neural network architecture with equivariant feature extraction layers that are designed to directly operate on continuous and noisy radiance fields from NeRF.
- A public dataset of 1300 NeRF models from 13 ShapeNet categories including posed images, and weights for evaluating canonicalization performance.

Chapter 2

Related Work

We focus our review of related work on neural fields, supervised canonicalization, self-supervised canonicalization, and equivariant neural network architectures.

2.1 Neural Fields

Neural fields are emerging as useful representations for solving problems such as novel view synthesis [25, 3, 4, 65, 16], shape encoding and reconstruction [28, 23], dynamic reconstruction [29, 30, 20], appearance modeling [24, 35, 44], and human motion modeling [50, 19, 45, 8]. In contrast to voxel/point cloud-based methods that discretize 3D space, neural fields offer a continuous representation and differentiable framework, enabling more precise modeling of intricate structures. This attribute is particularly advantageous in tasks like shape encoding and reconstruction, where the fine-grained details of objects are crucial.

Generalization of neural fields to object categories remains difficult, but some methods have used pre-canonicalized datasets to circumvent this problem [62]. Please see [58, 48] for more details. In this paper, our focus is on canonicalizing for the 3D pose of neural fields, specifically, neural radiance fields (NeRF) [25].

2.2 Supervised Canonicalization

Canonicalization, the process of aligning 3D shapes to a standardized reference frame, plays a pivotal role in various applications, particularly in tasks like instance-level camera pose estimation, human pose estimation and 3D shape analysis.

Datasets such as ShapeNet [5] and ModelNet40 [57] have 3D shapes that are manually pre-canonicalized. This inductive bias aids category-level generalization in problems such as 3D

reconstruction [28, 23, 47, 12]. We can also use these datasets to formulate canonicalization as a supervised learning problem [53] enabling applications such as 6 degree-of-freedom object pose estimation, multi-view reconstruction [18, 43], and reconstruction of articulating [21, 64] and non-rigid objects [63]. However, our goal is to canonicalize without using manual pose labels.

2.3 Self-Supervised Canonicalization

Traditional supervised approaches rely on labeled datasets, requiring manual annotation of canonical poses for each point cloud, a process that is both labor-intensive and impractical in scenarios where labels are unavailable. The emergence of self-supervised canonicalization methods offers a compelling alternative by using input point clouds without explicit annotations. One common strategy involves predicting the canonical point cloud along with a transformation that maps it to the input point cloud. The loss is then computed between the input and the transformed input, guiding the model to learn canonical representations without relying on labeled data.

Recent research has shown that self/weak supervision is sufficient for learning pose canonicalization on point clouds [37, 36, 46, 42]. None of these previous self-supervised methods can operate directly on neural fields – to the best of our knowledge, ours is the first.

2.4 Equivariant Neural Networks

Equivariant neural networks have emerged as a transformative paradigm in 3D point cloud processing, addressing limitations inherent in traditional 3D point cloud encoders. Conventional methods often struggle to capture and leverage the symmetries and transformations embedded in 3D data, leading to suboptimal representations for subsequent tasks. By explicitly considering the group symmetries of the input data, equivariant architectures ensure that the learned representations are invariant under various transformations, contributing to improved generalization and adaptability.

Pose-equivariant networks are equivariant to input pose [49, 9, 56, 55] by design. Some of these methods use Spherical Harmonic functions [49, 55, 56, 10], or vector neurons [9] to extract equivariant features. Equivariance is closely related to canonicalization since a canonical pose is also equivariant. Thus, previous methods have used pose-equivariance for canonicalization [36, 42, 46, 40]. However, these methods have thus far been limited to 3D point clouds, voxels, or meshes.

Chapter 3

Background

In this chapter, we lay the groundwork for a comprehensive understanding of canonical fields. We start by delving into various methods of representing 3D data and camera modeling. Then pivot to the development of the mathematical tools needed to construct both rotation-equivariant and invariant deep neural networks.

3.1 3D Representations

The world around us is three-dimensional which means that two-dimensional data such as images often prove insufficient to adequately represent the world we inhabit. There is an inherent loss of information of depth in the transition from the 3D world to the 2D world. This has led to extensive research in the field of 3D Computer Vision to develop a compact representation that avoids this loss of information. We cover a few of these representations below with their corresponding pros and cons:

1. **Point Clouds**
   This is a form of unstructured representation consisting of an array of 3D points/coordinates in space. These coordinates specify points in space occupied by some object. The objects or scenes represented in this form are essentially discretized into a set of sampled points. For each point we can optionally store the color and normal information
   Pros:

   - Easiest and most intuitive form of representation.
   - Easy to obtain using sensors with basic processing of sensor data.

   Cons:

- Memory is inefficient as this form requires quite a bit of storage to obtain a decent representation. The more points we sample, the better the representation but the more memory we consume.
- Loss of information such as connectivity and topology due to the discretization of objects.
- Poor representation of texture and appearance information if we don't sample an adequate number of points to faithfully represent the scene.

2. **Meshes**
   Here we represent the underlying occupied regions of space as a collection of vertices, edges, and faces. The vertices are basic building blocks of meshes as they define the corners of faces. The vertices are interconnected to produce a network of faces (where each face is some polygon such as a triangle or quadrilateral) to represent the surface of an object in space. The surface of the object can be textured or colored by applying different attributes to the polygons. This representation is commonly used in graphics applications such as video games.
   Pros:

   - Good representation of both, shape and appearance/texture.
   - With the texture maps we can render photorealistic images.

   Cons:

   - Storing the large and high-quality meshes are memory intensive.
   - Complexity in the representation.
   - Can't be directly obtained from sensors.
   - Direct acquisition from sensors is not feasible. Needed post-processing.
   - Not well-suited representation for modern deep learning frameworks.

3. **Voxels**
   Volumetric pixels (voxels) represent a 3D object or space using a regular grid of volume elements, similar to how pixels represent 2D images. Each voxel represents a small volume of space and contains information about properties such as RGB color, density, etc. of that corresponding volume element in 3D space.

   Pros:

   - Easy to query due to its regular grid structure.

- Easier to process due to its regular grid nature, especially for methods based on deep neural networks.
- The regular grid structure of voxels facilitates parallel processing, enabling efficient computations.

Cons:

- Quality improves with an increase in resolution but at the cost of memory.
- Loss of information due to discretization.
- Lot of redundancy in the representation as it stores information about both occupied and unoccupied regions.
- Poor visual representation of texture and appearance information in 3D.
- Not able to capture the high-frequency details.

4. **SDFs**
   Unlike the above representations, This is an implicit representation where a function outputs the distance of a 3D point from the nearest point on the object's surface. It maps points inside the surface to negative values, those on the surface to 0, and all other points outside the surface to positive values. While this representation on its own carries no information on appearance and texture, it is possible to store this information using another function to map 3D points to their texture or RGB color.
   Pros:

   - Memory-efficient as here we store an implicit representation modeled by some function. It is common practice to use a neural network to represent this function making it memory-efficient.
   - A continuous and differentiable representation since it is an implicit representation.
   - Ray tracing with SDFs is computationally efficient.

   Cons:

   - Creating the accurate SDF for complex scenes can be challenging.
   - SDFs might struggle to accurately represent thin structures or surfaces with high curvature.

As we have seen above, each representation offers something different having its own applications. For the purpose of this thesis, we will primarily focus on a more recent form of representation, namely Neural Radiance Fields (NeRFs).

3.2 Camera Modelling

In this section, we discuss how a 3d point in the world is projected on an image. we can mathematically model the camera which relates the 3D point in the world to the image pixel. Given a 3D point $p \in \mathbb{R}^3$ in the world frame, we initially convert it into homogeneous coordinates $p_w = [X_w, Y_w, Z_w, 1]^T$. Let $\mathbf{R} \in R^{3\times3}$ represent the $world$ to $camera$ rotation and $X_o \in \mathbb{R}^3$ is the camera center in the world frame, we can transform the p_w into the camera frame as follows.

$$p_c = \begin{bmatrix} \mathbf{R} & -\mathbf{R}X_o \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.1}$$

After describing the 3D point in the camera frame, we project it onto the image sensor using the pinhole projection model. Let $p_c = [X, Y, Z, 1]$, where $X, Y, Z \in \mathbb{R}$. By the geometric construction in Fig. 3.1 (right) we derive the following using similar triangles.

$$v = \frac{fY}{Z}, \; u = \frac{fX}{Z} \tag{3.2}$$

Now that we have the projected coordinates on the image sensor, the next step is to transform them into image pixel coordinates by adding the camera sensor centers c_x, c_y to projected points.

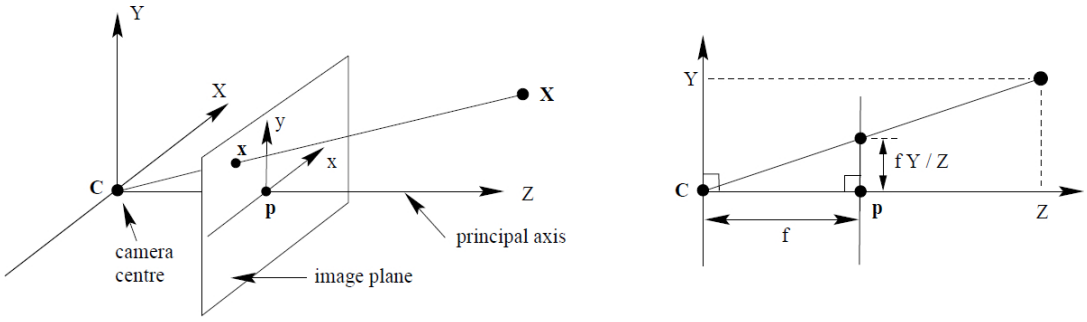

Figure 3.1: **Pinhole camera** projection. Image courtesy [13].

we can combine all of these operations into a single matrix that takes 3D coordinates in the world and transforms them into 2D pixels as follows:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \mathbf{K}[\mathbf{R} \mid -\mathbf{R}X_o] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.3}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ defines the camera intrinsic matrix which has focal length f_x (along x), f_y (along y), the camera centers c_x, c_y and it's shearing parameter s.

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

From this modeling, it's evident that world-to-camera transformation is invertible (rigid transformation), but the pinhole projection is not an invertible transformation. This lack of invertibility results in ambiguous depth along the Z direction. Due to this ambiguity, we will only obtain the ray direction if we multiply the inverse of **K** with image coordinates. The exact 3D point can be found at any distance on that ray.

3.3 Neural Radiance Fields

In this section, we describe Neural Network based 3D scene representation. This method comes under an implicit representation, which means we can query any point in the 3D space to get its attributes. We will primarily be covering the Neural Radiance Fields (NeRFs).

Nerf Models a scene as a 5D vector-valued continuous function which takes the 3D location and 2D viewing direction as inputs and outputs the color and density. 5D vector-valued function is approximated by a series of MLPS which are learned with Ground truth Images and camera poses. Once trained, it can be queried to generate an image of the scene from any camera pose around the scene. Essentially, it is a solution to the Novel View Synthesis (NVS) problem but because of the approach NeRF adopts, it can also be used as an implicit representation of the scene.

For a given camera view, NeRF initially performs ray marching, where it backprojects a ray for each image pixel from that view and marches along the ray through the scene. NeRF samples and collects 3D points along each ray. Next, for each point i along a ray, it queries the neural network to output a density (σ_i) and an RGB value (c_i). It should be noted that the points and direction are not fed directly to the network but are first encoded into a higher-dimensional space using Equation 3.5.

$$\gamma(p) = \sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p) \tag{3.5}$$

This is done because the raw input coordinates result in renderings that perform poorly at representing high-frequency details in color and geometry. The density can be thought of as a value that represents the probability of a particular 3D point in space being occupied. Using

the volumetric rending equation 3.6 we can predict the color along the ray r(t) = o + td within the bounds t_n and t_f. As there is no closed-form solution to 3.6, we will numerically estimate the solution using 3.7, where $T_i = exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$ is the transmittance and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. Transmittance indicates the probability that nothing has blocked the ray till that point in the space along the ray.

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)\, dt, \quad \text{where } T(t) = \exp\left(-\int_{t}^{t_n} \sigma(r(s))\, ds\right) \tag{3.6}$$

$$C(r) = \sum_{i=1}^{n} T_i(1 - exp(-\sigma_i \delta_i))c_i \tag{3.7}$$

Finally, NeRF aggregates this information over all the sampled points along the ray to obtain a final estimate of the RGB color of the corresponding pixel using the numerical approximated rendering equation 3.7.

Nerf uses hierarchical sampling following the uniform sampling along the ray for better reconstruction. This is useful because querying the free space and occluded regions won't contribute to the rendering but still be queried. To overcome this inefficiency Nerf uses two networks namely coarse and Fine networks. A coarse network takes the uniformly sampled points along the ray between the near and far bounds. Once the coarse network is evaluated we can perform important sampling along the ray using outputs of the coarse network.

Once this is done for all the pixels/rays, we have an estimate of the image for that view. Finally, the network is trained to minimize the L_2 loss between the RGB values of the estimated pixels and the ground-truth pixels for the available training views. As can be seen from the above approach, the model is forced to learn an implicit representation that carries information on the geometry and appearance of the scene to be able to perform well at its objective of novel view synthesis.

One thing to note in the design of Nerf is that density is a function of 3D locations whereas color is a function of both 3D location and viewing direction. This choice of design is logically coherent as density should not change with the viewing direction thus making it multi-view consistent.

3.4 Canonicalization

Canonicalization generally refers to converting the different forms of data into standard representation. we are particularly interested in finding a 3D rigid transformation that maps the neural field of an object in an arbitrary pose to a consistently oriented pose across a category of shapes. **Pose canonicalization** is closely related to equivariance.

A function Γ over x is said to be **equivariant** to a group operation $\mathcal{O}$ if its output changes by a fixed mapping M for any $\mathcal{O}$ operating on the input x, *i.e.*, $\Gamma(\mathcal{O}x) = M(\mathcal{O})\,\Gamma(x)$. In this work, we are interested in equivariance to the $SE(3)$ group denoting 3D position and orientation (pose) – in particular, we focus on the rotation group $SO(3)$ since 3D translation equivariance is readily achieved through mean centering [27]. Spherical Harmonics are an example of functions equivariant to rotation that are defined on a unit sphere.

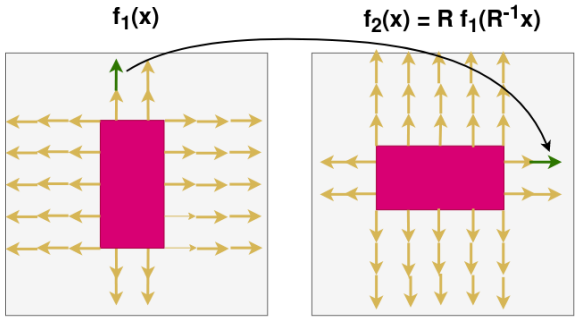

Figure 3.2: **Rotation Equivariance in Vector Fields.** Rotation equivariance in vector fields $\mathcal{F}$ (here, a simulated 2D magnetic field) require two operations: (1) update the position of the green arrow (the vector), and (2) rotating the green arrow at its new position. A function Γ operating on $\mathcal{F}$ is rotationally equivariant over the field if and only if $\Gamma[(\mathcal{R}\cdot\mathcal{F})(\mathbf{x})] = M(\mathcal{R})\,\Gamma[\mathcal{F}(\mathcal{R}^{-1}\,\mathbf{x})]$, where $\mathcal{R}$ is an $SO(2)$ rotation, and M is a rotation-dependent mapping function.

Rotation Equivariance in Vector Fields: Unlike 3D point clouds, rotation equivariance in fields is more involved. Consider a 2D vector field $\mathcal{F}$ defined $\forall\mathbf{x} \in \mathbb{R}^2$ (see Figure 3.2). Rotating this field requires two operations: (1) updating the positions of vectors in the field to new rotated positions, and (2) rotating the directions of the vectors. A function Γ operating on $\mathcal{F}$ is rotationally equivariant over the field if and only if $\Gamma[(\mathcal{R}\cdot\mathcal{F})(\mathbf{x})] = M(\mathcal{R})\,\Gamma[\mathcal{F}(\mathcal{R}^{-1}\,\mathbf{x})]$, where $\mathcal{R}$ is an $SO(2)$ rotation, and M is a rotation-dependent mapping function. In this work, we operate on NeRF's density field (a scalar field) but also extract equivariant features.(vector-valued fields).

3.5 Rotation invariant features in 3D

The spherical harmonics are complex homogeneous polynomials defined over the unit sphere $S^2 \subset \mathbb{R}^3$ there are $2l+1$ spherical harmonics of degree $l \in \mathbb{N}^*$ denoted by Y_m^l for $-l \leqslant m \leqslant l$. The spherical harmonics can be expressed in spherical coordinates (θ, ϕ) on the unit sphere. We can use a function $\varphi : \mathbb{R}^+ \to \mathbb{R}$ as a "radial" component and define the function $\varphi Y_m^l : \mathbb{R}^3 \to \mathbb{C}$. These function can be used as kernels for convolution of $3D$ signals in a neural network. They are particularly interesting because of their behavior under rotation of the input signal. For any rotation $R \in \mathrm{SO}(3)$ and $x \in \mathbb{R}^3$ we have:

$$Y_m^l(Rx) = \sum_{n=-l}^{l} D_{mn}^l(R) Y_n^l(x)$$

where $D^l(R) \in \mathrm{U}_{2l+1,2l+1}(\mathbb{C})$ is a complex unitary matrix called the Wigner D matrix of degree l. We would like to exploit this property of spherical harmonics to learn rotation invariant (and/or equivariant) features in a convolutional neural network. Given a signal $X : \mathbb{R}^3 \to \mathbb{R}$ defined over $\mathbb{R}^3$ we have features given by convolution with the kernels (φY_m^l):

$$X_m^l := X * \varphi Y_m^l : \mathbb{R}^3 \to \mathbb{C}.$$

Now consider the rotated signal $R.X$ defined by:

$$R.X(x) := X(R^{-1}x)$$

The corresponding rotated features behave similarly to spherical harmonics under rotation:

$$\begin{aligned}
(R.X)_m^l(x) &= \int_{\mathbb{R}^3} X(R^{-1}(t-x)) \varphi Y_m^l(t) dt \\
&= \int_{\mathbb{R}^3} X(t-x) \varphi Y_m^l(Rt) dt \\
&= \int_{\mathbb{R}^3} X(t-R^{-1}x) \varphi Y_m^l(Rt) dt \\
&= \sum_{n=-n}^{n} D_{mn}^l \int_{\mathbb{R}^3} X(t-R^{-1}x) \varphi Y_n^l(t) dt \\
&= \sum_{n=-l}^{l} D_{mn}^l R.(X_n^l)(x).
\end{aligned}$$

We would like to find rotation-invariant features. More precisely we are looking for functions $P((X_m^l)_{lm})$ of the features which are rotation invariant in the following sense:

$$P((R.X)_m^l)_{lm}) = R.(P(X_m^l)_{lm}))$$

A simple family of invariants is given by the hermitian products:

$$\sum_{m=-l}^{l} X_m^l \overline{X_m^l},\ l \in [|0, N|]$$

indeed we have:

$$\begin{aligned}\sum_{m=-l}^{l} (R.X)_m^l \overline{(R.X)_m^l} &= \sum_{j=-l}^{l}\sum_{k=-l}^{l}\sum_{m=-l}^{l} (D^l)_{km}^* D_{mj}^l R.(X_j^l) R.(\overline{X_k^l}) \\ &= \sum_{j=-l}^{l}\sum_{k=-l}^{l} \delta_{jk} R.(X_j^l) R.(\overline{X_k^l}) \\ &= R.\left(\sum_{m=-l}^{l} X_m^l \overline{X_m^l}\right).\end{aligned}$$

3.5.1 Spherical harmonics

For any integers $\ell \geqslant 0, m \in [|0, \ell|]$ the associated Legendre polynomial P_ℓ^m is defined by:

$$P_\ell^m(X) = \frac{(-1)^m}{2^\ell \cdot \ell!} \cdot (1 - X^2)^{m/2} \cdot \frac{\partial^{m+\ell}}{\partial X^{m+\ell}} \left[(X^2 - 1)^\ell\right]$$

The definition extends to negative m by:

$$P_\ell^{-m}(X) = (-1)^m \frac{(\ell - m)!}{(\ell + m)!} P_\ell^m(X).$$

The spherical harmonics $(Y_\ell^m)_{l\in\mathbb{N}, m\in[|-l,l|]}$ are defined by:

$$Y_\ell^m(x, y, z) = \sqrt{\frac{(2\ell + 1)}{4\pi} \frac{(\ell - m)!}{(\ell + m)!}}\, P_\ell^m\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \left(\frac{x + iy}{\sqrt{x^2 + y^2}}\right)^m$$

A real basis of spherical harmonics can be defined in terms of their complex analogues by setting

$$
Y_{\ell m} = \begin{cases} \dfrac{i}{\sqrt{2}}\left(Y_\ell^m - (-1)^m\, Y_\ell^{-m}\right) & \text{if } m < 0 \\ Y_\ell^0 & \text{if } m = 0 \\ \dfrac{1}{\sqrt{2}}\left(Y_\ell^{-m} + (-1)^m\, Y_\ell^m\right) & \text{if } m > 0. \end{cases}
$$

$$
= \begin{cases} \dfrac{i}{\sqrt{2}}\left(Y_\ell^{-|m|} - (-1)^m\, Y_\ell^{|m|}\right) & \text{if } m < 0 \\ Y_\ell^0 & \text{if } m = 0 \\ \dfrac{1}{\sqrt{2}}\left(Y_\ell^{-|m|} + (-1)^m\, Y_\ell^{|m|}\right) & \text{if } m > 0. \end{cases}
$$

$$
= \begin{cases} \sqrt{2}\,(-1)^m\ \mathrm{Im}[Y_\ell^{|m|}] & \text{if } m < 0 \\ Y_\ell^0 & \text{if } m = 0 \\ \sqrt{2}\,(-1)^m\ \mathrm{Re}[Y_\ell^m] & \text{if } m > 0. \end{cases}
$$

The corresponding inverse equations are:

$$
Y_\ell^m = \begin{cases} \dfrac{1}{\sqrt{2}}\left(Y_{\ell|m|} - iY_{\ell,-|m|}\right) & \text{if } m < 0 \\ Y_{\ell 0} & \text{if } m = 0 \\ \dfrac{(-1)^m}{\sqrt{2}}\left(Y_{\ell|m|} + iY_{\ell,-|m|}\right) & \text{if } m > 0. \end{cases}
$$

The transition matrix $C^\ell \in \mathcal{M}_{2\ell+1}(\mathbb{C})$ from complex to real spherical degree ℓ spherical harmonics is given by:

$$
\begin{cases} C^\ell_{m,m} = \dfrac{-i}{\sqrt{2}} & \text{if } m < 0 \\ C^\ell_{m,m} = \dfrac{(-1)^m}{\sqrt{2}} & \text{if } m > 0 \\ C^\ell_{m,-m} = \dfrac{1}{\sqrt{2}} & \text{if } m < 0 \\ C^\ell_{m,-m} = \dfrac{i(-1)^m}{\sqrt{2}} & \text{if } m > 0 \\ C^\ell_{0,0} = 1 & \\ C^\ell_{j,k} = 0 & \text{otherwise} \end{cases}
$$

the transition from real to complex harmonics is given by the transpose conjugate matrix $(C^\ell)^*$.

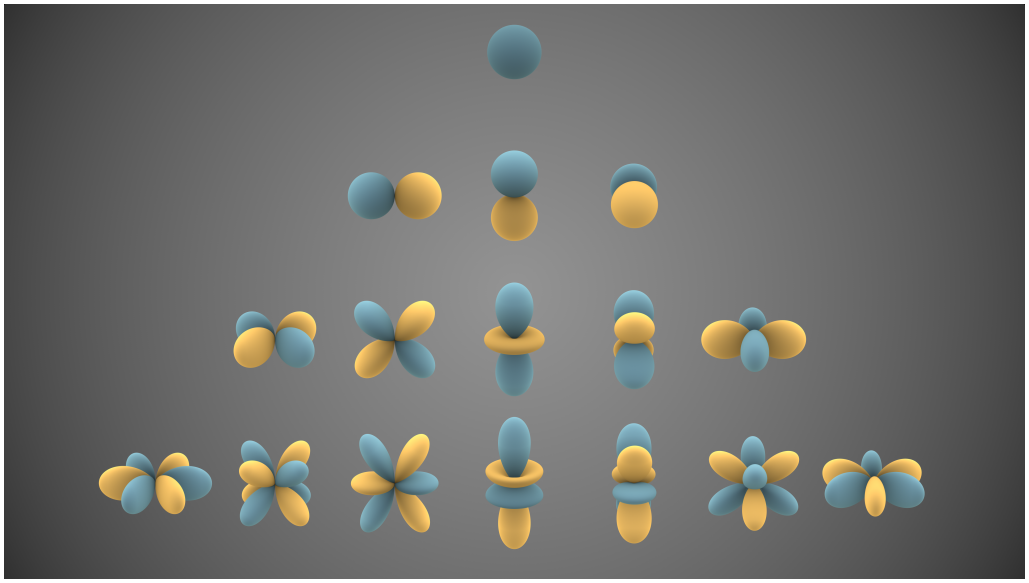

Figure 3.3: Visualization of Real spherical harmonics till type-3. Image courtesy Wikipedia.

Definition 3.5.1 (Wigner d Matrix) *Let $j \in \mathbb{N}$. We define the Wigner d^j matrix for any angle β by:*

$$\begin{aligned} d^j_{m'm}(\beta) &= [(j+m')!(j-m')!(j+m)!(j-m)!]^{1/2} \sum_s \left[\frac{(-1)^{m'-m+s}}{(j+m-s)!s!(m'-m+s)!(j-m'-s)!} \right. \\ &\quad \left. \cdot \left(\cos\tfrac{\beta}{2}\right)^{2j+m-m'-2s} \left(\sin\tfrac{\beta}{2}\right)^{m'-m+2s} \right]. \end{aligned}$$

for $m, m' \in [|-j, j|]$ where the sum over s is over such values that the factorials are nonnegative.

Definition 3.5.2 (Wigner D-matrix) *Let $j \in \mathbb{N}$ the Wigner D^j matrix is defined for any angles α, β, γ by:*

$$D^j(\alpha, \beta, \gamma) := e^{-im'\alpha} d^j_{m'm}(\beta) e^{-im\gamma}$$

for $m, m' \in [|-j, j|]$.

Chapter 4

Method

Our method assumes that we have pre-trained NeRF models of scenes containing single object instances at arbitrary poses from common shape categories. Our scenes contain primarily the object, but also some background – perfect foreground-background segmentation is not available. Our goal is to build a method that estimates a canonicalizing $SE(3)$ transformation that aligns these object NeRFs to a consistent category-level **canonical field**.

Rather than operate on 3D point clouds, meshes, or voxels, our method (see Figure 4.1) operates directly on samples from the continuous neural radiance field. During training, we learn **canonical fields** from a pre-trained NeRF dataset in a fully **self-supervised** manner. We use a Siamese network architecture that extracts equivariant field features for canonicalization (see Figure 4.1). During inference, given a previously unseen NeRF model of a new object instance in an arbitrary pose, our method estimates the transformation that maps it to the canonical field.

4.1 NeRF Sampling

For a pre-trained neural radiance field that maps 3D position (x, y, z) and direction θ, ϕ to color and density $(\mathbf{c}, \sigma)$, the input to our method consists of samples on the density field. We do not use color because it is direction-dependent in NeRF and also varies for different instances.

During both training and inference, we uniformly sample the pre-trained NeRF density field to find the object center and its bounding box. We then sample a uniform grid in the density field of the object's bounding box. We empirically choose uniform grid sampling over fully random sampling since we found no difference. The uniformly sampled density grid is then normalized to obtain density values $\boldsymbol{\sigma}_D$ in the continuous range $[0, 1]$, *i.e.*,

$$\boldsymbol{\sigma}_D := \{1 - exp(-d \cdot \Gamma_r^{\sigma}(x)) \mid x \in \boldsymbol{X}\}, \tag{4.1}$$

where $\boldsymbol{X}$ is the regularly sampled grid within the object bounding box and d is the depth step size used in the coarse NeRF sampling.

Before querying NeRF for occupancy, we do not have the extremities of the object. We query a uniform grid within the unit cube to obtain a 32^3 grid density field to find the object bounds and the center of the object. The density values are normalized to range $[0, 1]$. We then perform an initial K-means clustering with $K = 2$ to separate the foreground and background clusters. The foreground and background are automatically detected by calculating the mean density values within each cluster. The cluster with a high mean is declared as the foreground, while the other one is considered the background. This assumption is generally true in our setting because we have single-object neural radiance fields. The mean of query coordinate locations in the foreground cluster gives us the center of the object $\mathbf{c}$. We then obtain the extremities from the foreground cluster and find its maximal diagonal length ($\boldsymbol{l}$). We re-sample the region within the bounding cube (at center $\boldsymbol{c}$ side length $\boldsymbol{l}$) and use it as input to CaFi-Net. **Translation** and **scale** canonicalization is simply achieved by moving the bounding box's center and scaling it, so our next concern is on rotation canonicalization.

4.2 CaFi-Net

CaFi-Net (Canonical Field Network) is our method for 3D rotation canonicalization using NeRF's density field (red box in Figure 4.1). We now describe its components.

Tensor Field Networks (TFNs): Many methods have been proposed for using equivariant features for rotation canonicalization of 3D data, for instance, spherical CNNs [7, 42], vector neurons [9], or capsule networks [66, 46]. However, these and other methods [37] are limited to 3D point cloud or voxel representations. In this work, we extend Tensor Field Networks [49, 34] pose canonicalization directly on samples from NeRF's density field. TFNs operate by computing Type-ℓ real-valued spherical harmonic functions that are rotation equivariant: $D^\ell(R)Y^\ell(X) = Y^\ell(Rx)$. Where $D^\ell : \text{SO}(3) \rightarrow \text{SO}(2\ell + 1)$ is the Wigner matrix of type ℓ (please see [49, 34] for details). For a 3D point set $P \in \mathbb{R}^{N\times 3}$, a rotation-equivariant TFN convolution layer (EQConv) at point $p \in P$ is defined as: $\text{EQConv}^J(p) = Q^{(n,\ell),J}\left(\sum_{y\in {}^{2R}\mathcal{N}} k_r^n(p-y) \otimes s^\ell[y]\right)$. Where ${}^{2R}\mathcal{N}$ is a set containing neighbors of point p at twice the resolution and $Q^{(n,\ell),J}(\cdot)$ is the Clebsh Gordon decomposition to combine type-n kernel k_r^n and type-ℓ equivariant signal s^ℓ. The $\text{EQConv}(\cdot)$ layer aggregates type-ℓ feature s^ℓ at multiple receptive fields along with learnable weights to perform equivariant convolution. Our

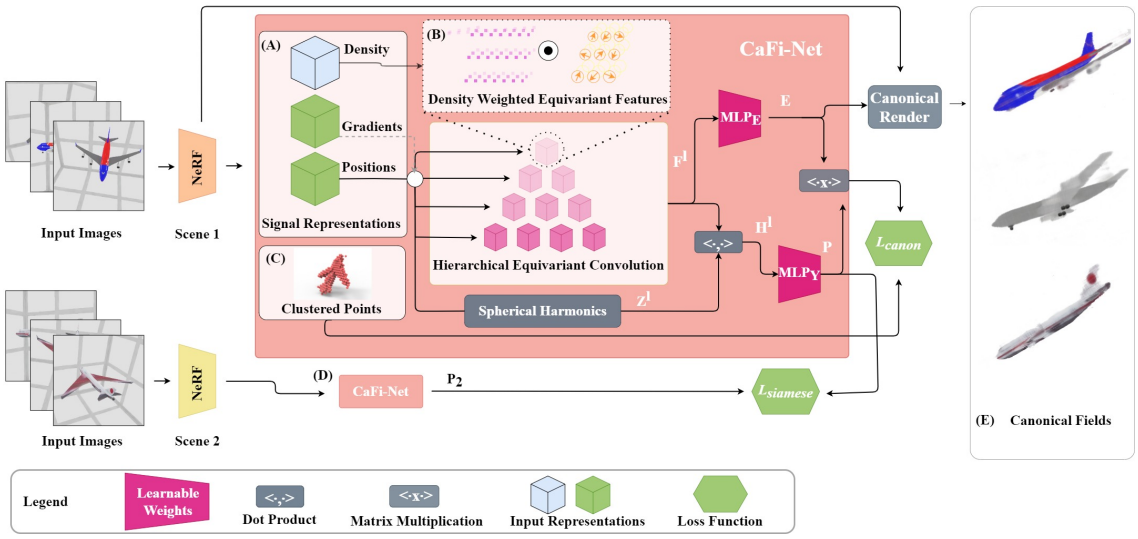

Figure 4.1: **CaFi-Net** samples a density field from NeRF and uses its **density**, **position** and **density gradients** as input signals (A) to canonicalize the field. We predict rotation equivariant features and weigh them by density (B) to guide our learning from the occupied regions of the scene. We then compute an invariant embedding by taking a dot product between equivariant features. This invariant embedding is used to canonicalize the field that enables rendering all the objects in the canonical frame (E). Our method also applies an inter-instance consistency loss (D) that aligns different instances of the same category in the canonical frame. **We do not assume pre-canonicalized fields, and canonicalize in a self-supervised manner**

method uses the EQConv layer as the fundamental building block to canonicalize *continuous* fields.

Signal Representation:

At each of the sampled density field locations, we have several choices of signals to use as the input for CaFi-Net. For instance, we can (1) use the NeRF density value directly, (2) use the xyz location of the sampled field, or (3) use the gradient of the density field. Densities alone do not capture the position and directional components of the field, so we use a combination of densities and gradients of densities at query locations of the grid as our input signal for CaFi-Net. Gradients of density capture the object surface and help in canonicalization (see ablation study in Chapter 5).

Equivariant Convolution: To canonicalize for rotation, we leverage equivariance properties of spherical harmonic functions to obtain rotation-equivariant learnable features on the field [49, 34]. Type-ℓ spherical harmonic functions (Y^ℓ) are degree-ℓ polynomials of points on the

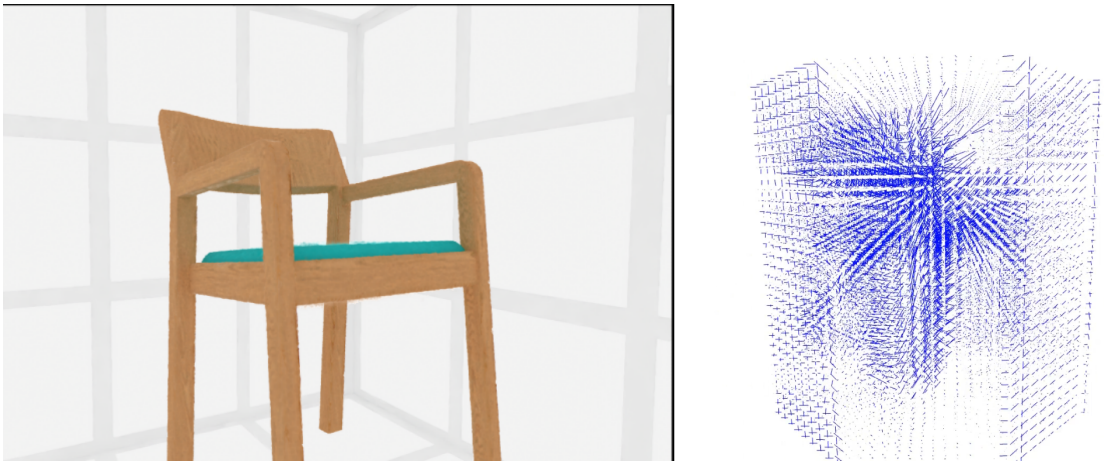

Figure 4.2: Example NeRF renderings of chair on the left and an example 3D visualization of the gradient field of chair on the right.

sphere that are equivariant to $SO(2\ell+1)$. The densities sampled from NeRF reconstructions are often noisy and contain outliers. To promote our method to learn only from occupied regions, we weigh each equivariant feature type by its density. This weighing has no effect on the equivariance of features (see 4.4.2 for proof). We have the option to weigh equivariant features with the local average of the density to smoothen the signal or to weigh directly by the density. We found weighing by density worked better than taking the local average density (see ablation study in Chapter 5).
We learn equivariant features of type ℓ as:

$$ {}^{R}s^{\ell} := f_w({}^{R}x)\left(W^{\ell}(\text{EQConv}^{\ell}(\nabla(\boldsymbol{\sigma}_D({}^{R}x)))) + \delta_{0\ell}b\right), \tag{4.2} $$

$$ f_w({}^{R}x) := \begin{cases} \text{mean}_r(\boldsymbol{\sigma}_D({}^{R}x)), & or \\ \boldsymbol{\sigma}_D({}^{R}x), \end{cases} \tag{4.3} $$

where, W^{ℓ} and b are the weights and biases, ${}^{R}x$ is a 3D point queried at resolution R and $\text{mean}_r(\boldsymbol{\sigma}_D({}^{R}x))$ is the average density at location ${}^{R}x$ from points sampled at radius r. We hierarchically aggregate features at resolutions $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ to obtain global equivariant features. The EQConv^{ℓ} convolution layer is the same as defined in 4.2. We employ non-linearities from [34] that preserve equivariance and capture better equivariant features. We also compute the max-pool of point-wise features of the last layer to obtain global type-ℓ equivariant features F^{ℓ}.

TFN Kernel Implementation: TFN's are equivariant point convolution networks. We are using the modified version of this network for the feature extraction. We will describe the

implementation details of the TFN in this section. TFN network takes the 32x32x32 grid sampled density values as inputs and aggregates the features at multiple scales i.e., 32,16,8 and 4. For input down-sampling, we use 3D avg-pooling with kernel size 2 and stride length 1. We store the resultant down-sampled inputs in a list which will be used in features calculation.

Similar to the 2D convolution, where a kernel operates on a signal to extract features, here also we need a kernel and a signal. At each level, the current resolution serves as the kernel, while its immediately down-sampled resolution serves as the signal. For example, when operating at 32x32x32, 16x16x16 becomes the signal and 32x32x32 becomes the kernel. we refer to kernel points as source points and signal points as target points.

In the convolution process at each target point, 32 points are chosen from the source, determined by their Euclidean distance from the target point. A specified radius is defined, and if any of these points extend beyond the radius from the target point, their contribution to the overall convolution at the target point is set to zero. Once these 32 points are selected, we use the relative distance of these points from the target to evaluate spherical harmonic polynomials to get equivariant features. All these distance measures are taken relative to target *i.e.* $(x_j - y_i)$ where y_i is the target and x_j is the source point. In this work, features up to type-3 are utilized.

After obtaining the kernel features, the Next step would be taking the tensor product between the signal and kernel equivariant features. For the signal, we choose the gradient of the density as type-1 and one's as type-0 signal. The gradients are calculated discretely by taking the difference between the adjacent samples along the x,y, and z directions to get a 3d equivariant signal. All these equivariant signals are implemented using Python dictionaries, where the key corresponds to the feature type and the value is a 4D a vector representing batch, points, feature type, and feature dimensions. Following the tensor- product calculation between the kernel and signal, Clebsch-Gordon coefficients are utilized to decompose the final features into irreducible representations.

Now, we will describe the decomposition process using an example. TFNs are constructed using 3D steerable kernels, which are defined by Equation (4.4).

$$\kappa_{\mathrm{rm}}^{\ell} = \phi_r \left(\|x\|_2 \right) Y_{\ell}^{m} \left(\frac{x}{\|x\|_2} \right) \tag{4.4}$$

In the Equation (4.4), l represents the type of the signal. In the CaFi-Net, we consider up to type-3. Y_{ℓ}^{m} is a spherical harmonic function that takes a 3D vector x as input and produces type l vector of the length $(2l + 1)$.

Suppose we have the target point y_j and source point x_i, then the convolution is defined by Equation (4.4), where $(x_i - y_j)$ is the relative vector and f_i is a signal at x_i. In the CaFi-Net we start with the type-0 (all ones) and type-1 signals (discrete gradient of the density values along

x, y, z directions).

$$y = \sum_i \kappa(x_i - y_j) f_i \tag{4.5}$$

Let's take a numerical example and calculate the dimensions of the convolution. we have,

1. Signal type = 1
2. Kernel type = [0,1,2,3]
3. Target points size = 1024
4. Neighbour hood size = 32

By evaluating the kernel using (4.4) for all the above types, we obtain the following dimensional vector for types 0, 1, 2, and 3 respectively.

- $K^0(x_i - y_j) \rightarrow$ 1024 x 32 x 1
- $K^1(x_i - y_j) \rightarrow$ 1024 x 32 x 3
- $K^2(x_i - y_j) \rightarrow$ 1024 x 32 x 5
- $K^3(x_i - y_j) \rightarrow$ 1024 x 32 x 7

As all these kernels will operate on the type-1 signal to simplify calculations, we can combine the signals of all the kernels into a single extensive vector with dimensions 1024 x 32 x 16. Using the Equation (4.5), we can obtain the features of the dimension (1024 x 16 x 3) given the kernel function vector (1024 x 32 x 16) and the signal vector (1024 x 32 x 3). Now we can unstack the features based on the kernel type into the following dimensions:

- $y[0] \rightarrow$ 1024 x 1 x 3
- $y[1] \rightarrow$ 1024 x 3 x 3
- $y[2] \rightarrow$ 1024 x 5 x 3
- $y[3] \rightarrow$ 1024 x 7 x 3

Consider the features obtained through the convolution of the type-2 kernel with the type-1 signal, represented as $y[2]$ with dimensions 1024x5x3. Reshaping these vectors to 1024 x 15 yields a type-7 (2*7 + 1 = 15) equivariant vector. The type-7 equivariant feature vector is then decomposed into lower types using Clebsch-Gordon coefficients (Q). For a given two equivaraint signals j_1, j_1 the size of the clebsch-Gordon decomposition matrix will be $((2j_1 +$

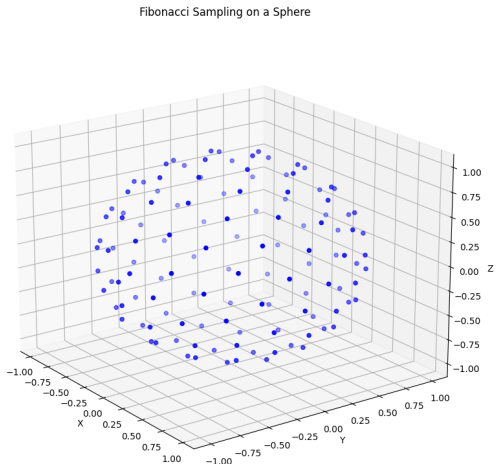

Figure 4.3: Visualization of 128 points distributed on a sphere using the Fibonacci sampling method,

$1) * (2j_2 + 1))$ x $((2j_1 + 1) * (2j_2 + 1))$. After multiplying the features with the Q, the result will be decomposed into multiple type features in the range $(abs(j_1 - j2)$,$(j_1 + j_2 + 1))$. In our case j_1=2 and j_2=1, which makes the clebsch-Gordon decomposition matrix to be of dimension 15 x 15. After multiplying the Q with $y[2]$ we get 1024x15 dimensional vectors which will be split into type 1,2 and 3 feature signals.

Up to this point, there has been no learning involved. We introduced a trainable MLP without a bias term for types greater than 1. For the non-linearities, we use [34] which interprets equivariant features as coefficients of functions in the spherical harmonics basis. For each point, we can compute inverse Spherical Harmonic Transform (SHT), which results in a function on the unit sphere that is equivariant under $\mathrm{SO}(3)$. Inverse SHT functional representation allows us to apply non-linearities in a pointwise fashion. We further use the Forward Spherical harmonic transform to compute the coefficients of the spherical harmonic basis using Equation (4.6). Unfortunately, there is no closed form for the integral; in practice, we have to rely on a discrete approximation of the SHT and its inverse. For this, we sample 128 points on a sphere using Fibonacci sampling as shown in figure 4.3. Please refer to Algorithm 1 for pseudo-code.

$$\int \int_{\text{sphere}} f(\theta, \phi) \cdot Y_{lm}^{*}(\theta, \phi) \, \sin\theta \, d\theta \, d\phi \tag{4.6}$$

$x =$ Density Grid
$y =$ Grid coordinates
$n = 4$
$z[0] = 1$
for $i < n$ **do**
  x_1 = Downsample(x)
  y_1 = Downsample(y)
  k = Gradient(x)
  k_1 = Gradient(x_1)
  $z[1]$ = concat(z[1],k)
  z = Equivariant Convolution (y, y_1, k, z)
  $z[1]$ = concat(z[1],k_1)
  z = apply-linear-weights(z)
  z = compute-iSHT(z)
  z = apply-non-linearties(z)
  z = compute-fSHT(z)
  $x = x_1$
  $y = y_1$
end for
z = compute-iSHT(z)
y = max-pool(z,dim=1)
return y

Algorithm 1: Density TFN Network

Canonicalization: After obtaining the global type-ℓ equivariant feature F^ℓ, we compute its dot product with point-wise spherical harmonics Z^ℓ scaled by their norms (refer Eqn. 4.7) to obtain a pose-invariant embedding H^ℓ [36].

$$Z^\ell(\boldsymbol{X}) = ||\boldsymbol{X}||_2 \, Y^\ell(\boldsymbol{X} / \, ||\boldsymbol{X}||_2) \tag{4.7}$$

As the two vectors F^ℓ and Z^ℓ are both equivariant to input rotation, their dot product is invariant. We use a linear layer on top of the pose-invariant embedding to estimate grid coordinates in the canonical frame ($P \in \mathbb{R}^{H \times W \times D \times 3}$) for each point in $\boldsymbol{X}$ and M equivariant transformations $E \in \mathbb{R}^{M \times 3 \times 3}$ [36] that orient the canonical coordinates to the input coordinates. We then choose the best canonicalizing transform E_b and penalize it for canonicalization. The invariant embedding H, grid coordinates P and equivariant rotation E are given as,

$$H^{\ell}(\boldsymbol{\sigma}_D, \boldsymbol{X}) := \langle F^{\ell}(\boldsymbol{\sigma}_D), Z^{\ell}(\boldsymbol{X}) \rangle, \tag{4.8}$$

$$P := \text{MLP}_Y(H(\boldsymbol{\sigma}_D, \boldsymbol{X})), \tag{4.9}$$

$$E := \text{MLP}_E(F(\boldsymbol{\sigma}_D, \boldsymbol{X})). \tag{4.10}$$

Here, we have dropped the type-ℓ notation for H and F as we concatenate all equivariant feature types.

Siamese Network Architecture: Experimentally, canonicalizing noisy density fields is difficult without guidance on shape similarity within a category. We therefore use a Siamese training strategy [27] where two different object instances are forced to be consistently canonicalized (see the lower branch in Figure 4.1).

Clustering: CaFi-Net predicts a canonical coordinate for every grid point, but unlike point clouds, fields have many unoccupied regions that do not provide any information. We penalize the network on foreground regions only by performing K-means clustering on the densities within the object bounding box with $K = 2$ and choose the cluster $\boldsymbol{C}_f$ with a higher mean. Note that we still operate on continuous fields, but only cluster samples to guide our training.

4.3 Training

CaFi-Net is trained on a large dataset of pre-trained NeRF models over 13 object categories. We use the following loss functions to train our model.

Canonicalization Loss: To self-supervise our learning, we transform the predicted canonical grid coordinates to the input grid using the canonical rotation E_b and compute a point-wise L2 distance between them. This loss forces the predicted invariant grid coordinates P to reconstruct the shape and forces the predicted canonicalizing transform to be equivariant to the input transformation. The final loss is computed over the canonicalizing transform that minimizes this loss:

$$E_b := \min_j \left(\text{mean}_{i \in \boldsymbol{C}_f} ||\boldsymbol{X}_i - E_j P_i||_2^2 \right), \tag{4.11}$$

$$\mathcal{L}_{canon} := \text{mean}_{i \in \boldsymbol{C}_f} ||\boldsymbol{X}_i - E_b P_i||_2^2, \tag{4.12}$$

where C_f is a set of points belonging to the foreground.

Orthonormality Loss: The predicted equivariant transformations E should have orthonormal vectors and we use the following to force orthonormality using:

$$U_j, D_j, V_j := \text{SVD}(E_j), \tag{4.13}$$

$$\mathcal{L}_{ortho} := \frac{1}{M} \sum_j \left|\left|E_j - U_j V_j^T\right|\right|_2, \tag{4.14}$$

where $\text{SVD}(E_j)$ computes the singular value decomposition of the j^{th} equivariant transformation.

Siamese Shape Loss: We use a Siamese loss that penalizes for consistency between different shapes belonging to the same category. Given two un-canonicalized fields in different frames of reference $\boldsymbol{\sigma}_D^1$ and $\boldsymbol{\sigma}_D^2$, we canonicalize both the fields to the canonical frame and compute chamfer distance between their predicted canonical coordinates for points in the foreground cluster C_f, *i.e.*,

$$\mathcal{L}_{siamese} := \text{CD}(P(\boldsymbol{\sigma}_D^1), P(\boldsymbol{\sigma}_D^2)). \tag{4.15}$$

This loss regularizes the training by ensuring that instances of the same category should be aligned in the canonical frame.

Architecture and Hyper-parameters: CaFi-Net predicts a 128 dimensional invariant embedding for each point in space and performs convolution at resolutions $1/2$, $1/4$, and $1/8$. Each equivariant convolution layer aggregates features from 512 neighboring points at twice the resolution. We then use equivariant non-linearities introduced in [34] by applying Batch Normalization [15] and ReLU activation [2] in the inverse spherical harmonic transform domain. We use a three-layer MLP with intermediate Batch Normalization and ReLU layers followed by a final linear layer to predict canonical coordinates. To train our network, we weigh the canonicalization loss ($\mathcal{L}_{canon}$) the highest with weight 2.0 and 1.0 for all the other loss functions. All our models are written in PyTorch [31] and are trained for 300 epochs with a batch size of 2 on an Nvidia 1080-Ti GPU. We use the Adam Optimizer [17] with an initial learning rate of $6e-4$ and L2 weight regularization of $1e-5$ for all our experiments. Please see our experiments in Chapter 5 for more details.

4.4 Equivariance Properties of CaFi-Net

4.4.1 Averaging Equivariant Signals is Equivariant

A tensor field of type ℓ (ℓ-field) is a map $f : \mathbb{R}^3 \to \mathbb{R}^{2\ell+1}$. We have an action of $\text{SO}(3)$ on any ℓ-field f given by $(R.f)(x) := D^\ell(R) f(R^{-1}x)$ for any rotation $R \in \text{SO}(3)$ and $x \in \mathbb{R}^3$

where $D^\ell(R) \in \mathrm{SO}(2\ell+1)$ is the Wigner matrix of type ℓ. We say that transformation F transforming tensor fields of type p into tensor fields of type q if it commutes with the action of $\mathrm{SO}(3)$, i.e. for all type p-field f and rotation $R \in \mathrm{SO}(3)$ we have $F(R.f) = R.F(f)$. We will call such transformations (p,q)-equivariant transformations.

Lemma 1 *For any $r > 0$ the local average operator mean_r defined over p-fields by:*

$$\mathrm{mean}_r(f)(x) := \int_{B(x,r)} f(y)dy$$

where $B(x,r) \subset \mathbb{R}^3$ is the open ball of radius r centered at $x \in \mathbb{R}^3$ is a (p,p)-equivariant transform of fields.

Proof: Let $x \in \mathbb{R}^3$ and $R \in \mathrm{SO}(3)$ we have:

$$\begin{aligned}
\mathrm{mean}_r(R.f)(x) &= \int_{B(x,r)} (R.f)(y)dy \\
&= \int_{B(x,r)} D^p(R) f(R^{-1}y)dy \\
&\underset{u=R^{-1}y}{=} D^p(R) \int_{R^{-1}B(x,r)} f(u)du \\
&= D^p(R) \int_{B(R^{-1}x,r)} f(u)du \\
&= (R.\mathrm{mean}_r(f))(x)
\end{aligned}$$

4.4.2 Locally Averaged Density Weighted Equivariant Vectors are Equivariant

Lemma 2 *Scaling an equivariant field with density is a (1, 1)-equivariant transformation.*

Proof: Let σ be a type-0 density field and f be a type-1 field at the corresponding location, weighing f by the average of σ is:

$$\begin{aligned}
&\text{mean}_r(\sigma)(x) \cdot f(x) := \left(\int_{B(x,r)} \sigma(y)dy\right) f(x) \\
&\text{mean}_r(R \cdot \sigma)(x) \cdot (R \cdot f)(x) \\
&= \left(\int_{B(x,r)} (R \cdot \sigma)(y)dy\right) D^1(R)(f)(R^{-1}x) \\
&= \left(\int_{B(x,r)} (\sigma)(R^{-1}y)dy\right) D^1(R)(f)(R^{-1}x) \\
&= \left(I \cdot \int_{R^{-1}B(x,r)} (\sigma)(u)du\right) D^1(R)(f)(R^{-1}x) \\
&= D^1(R) \cdot \text{mean}_r(\sigma)(x) \cdot (f)(R^{-1}x)
\end{aligned}$$

Thus, $\text{mean}_r(R \cdot \sigma)(x) \cdot f(x) = D^1(R) \cdot \text{mean}_r(\sigma)(x) \cdot f(R^{-1}x)$ proving the result that scaling equivariant features by average density do not break the equivariance property.

4.4.3 Gradient is Type-1 Equivariant

Lemma 3 *The gradient operator is a* $(0, 1)$*-equivariant transformation.*

Proof: Let f be a 0-type field, by the chain rule of differentiation, for any $x, h \in \mathbb{R}^3$ and $R \in \text{SO}(3)$ we have

$$\begin{aligned}
\langle \nabla(R.f), h \rangle &= D_x(R.f)(h) \\
&= D_x f \circ R^{-1}(h) \\
&= D_{R^{-1}x} f \circ D_x R^{-1}(h) \\
&= \langle \nabla_{R^{-1}x} f, R^{-1}h \rangle \\
&= \langle R\nabla_{R^{-1}x} f, h \rangle \\
&= \langle R\nabla_{R^{-1}x} f, h \rangle
\end{aligned}$$

thus $\nabla(R.f) = R.\nabla f$ which concludes the proof.

Chapter 5

Experiments and Results

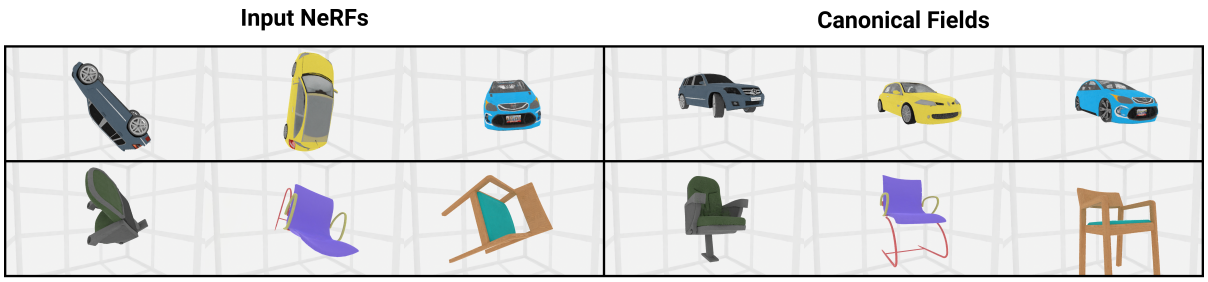

Figure 5.1: Canonical Field Network (**CaFi-Net**), a self-supervised method for 3D position and orientation (pose) canonicalization of objects represented as neural fields. We specifically focus on neural radiance fields (NeRFs) fitted to raw RGB images of arbitrarily posed objects (left), and obtain a **canonical field** (fixed novel view shown on right) where all objects in a category are consistently positioned and oriented.

5.1 Dataset

All our experiments use a new large synthetic NeRF dataset of shapes from 13 categories (that overlap with [12, 46, 36]) from the ShapeNet [5] dataset.

We created a simulator in Blender to render images for Nerf training. The simulator consists of a cube with each side panel hosting 9 cameras directed at the origin. Figure 5.2 shows renderings for one of the instances in the dataset. We avoided the objects having a white texture, given that the background also has a white texture. This decision aims to enhance Nerf training. In each category, we select 100 instances from ShapeNet, randomly rotate them, and place the object at the origin. Finally, we render the images from 54 omnidirectional views. Different from other datasets, we also simulate cluttered backgrounds.

These posed views are used to train NeRF models using the public PyTorch implementation of NeRF [61] for a total of **1300 NeRF models**. To our knowledge, this is one of the largest (compared to [11, 52]) 360° synthetic NeRF datasets. Each NeRF model is trained for 400 epochs with 1024 randomly selected rays per iteration, a coarse sampling resolution of 64 points, and a fine sampling resolution of 128 points along each ray. Rather than train NeRF to maximize PSNR on a novel view, we fix the number of epochs for all models to enable a fair comparison between instances. As a consequence, some NeRF models may have a lower PSNR, but **our goal is not to increase NeRF quality but rather improve canonicalization quality** – hence we fix the number of epochs. For each category, we set aside 20% of the models for testing.

Figure 5.2: Visualization of renderings from all 54 views of a car object. Each row corresponds to the Front, Right, Back, Left, Top, and Bottom views, respectively.

5.2 Metrics

Since there are no known metrics for evaluating canonical fields, we resort to 3 metrics used for 3D point cloud canonicalization [36]. For a fair comparison, we extracted the point clouds from the Pre-trained NeRFs and used them to train the point cloud methods. All these point cloud methods use surface points for training. So, we tried to extract the surface points from the pre-trained Nerf as follows:

We perform K-means clustering with k=2 on the density values obtained from pre-trained NeRF instances, which are queried with points sampled from a uniform grid. For each cluster, we calculate the mean density value and designate the cluster with the higher density as foreground (replacing the densities with 1) and the other as background (replacing the densities with 0). On the modified density grid, we perform the gradient calculation to get the boundary points which will be used for training point cloud methods. Boundary points are those having zero crossing of gradient values. For all the metric calculations, we fixed 180 rotations sampled on a sphere. All these point clouds are mean-centred as follows:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.1}$$

$$x = x - \bar{x} \tag{5.2}$$

(1) **Instance-Level Consistency (IC)**: This metric aims to measure the canonical performance across the instances under different rotations. We use the chamfer distance between two canonical outputs corresponding to differently rotated inputs of the same instance. For better canonicalization, we expect a smaller value of overall IC. The mean of the IC metric is defined as follows:

$$\mathrm{IC} := \frac{1}{|\chi||\mathbf{R}|}\sum_{X_i \in \mathcal{X}}\sum_{R_j \in \mathbf{R}} \mathrm{CD}[(R_j.X_i)^c, X^c] \tag{5.3}$$

(2) **Category-Level Consistency (CC)**: This metric aims to measure the canonical performance across different instances in the same category. We compute the chamfer distance across all the possible pairs of the canonical outputs for all the rotations. For better canonicalization, we expect a smaller value of overall CC. The mean of the CC metric is defined as follows:

$$\mathrm{CC} := \frac{1}{|\chi|N}\sum_{X_i \in X}\sum_{X_j \in N} \mathrm{CD}[X_i^c, X_j^c]. \tag{5.4}$$

(3) **Ground Truth Equivariance Consistency (GEC)**, a variant of the Ground Truth Consistency (GC) metric proposed in [36] to measure canonicalization performance compared to

manual labels. We use the GEC because we observed that the original GC metric is prone to degeneracy when the canonicalizing transform is identity. We fix this issue by taking three point clouds P_i, P_j, and P_k from a canonicalized dataset $\mathcal{P}$ and rotating P_i by R_1, P_j by R_2 where both R_1 and R_2 are random rotations. Let $\mathcal{C}(P)$ predict a canonicalizing rotation for a point cloud P, we then compute the Ground Truth Equivariance consistency as:

$$\text{GEC} := \frac{1}{|\mathcal{P}|^3} \sum_{P_i, P_j, P_k \in \mathcal{P}} \text{CD}(\mathcal{C}(R_1 P_i) R_1 P_k, \mathcal{C}(R_2 P_j) R_2 P_k), \tag{5.5}$$

where CD refers to the Chamfer distance. Here, we apply canonicalizing transforms of shape P_i and P_j to the shape P_k. This modified metric will not be degenerate for identity canonicalizing transforms and is more rigorous in evaluating ground truth consistency.

We first analyze the performance of our method on the three metrics across 13 different categories. In Table 5.1, we highlight our method that operates on a field as **Ours (F)**. Our method performs well across all categories with low variance between different categories. We observe similar trends across metrics for a set of categories (*e.g.*, firearms, lamps) indicating that category shape plays a role in canonicalization performance.

Next, we compare our method against three benchmarks: Principal Component Analysis (PCA) [32], Canonical Capsules (CaCa) [46], and ConDor [36]. All prior methods canonicalize clean point clouds with all points on the surface. To have a fair comparison, we train the above methods on point clouds (P) sampled from NeRF using the method described in the 5.2. Most scenes have points in the range of 120–1200 when sampled uniformly on the 32^3 grid. We resample the scene when points are less than 1024 and train both Canonical Capsules and ConDor on the resulting point clouds. CaFi-Net estimates a canonicalizing transform directly from the density field. For the metric calculation, it is possible that extracted surface point clouds from pre-trained Nerf can be noisy which will affect the evaluation. To address the noise issue, we save the input pose and canonical poses for each instance using different methods. We then utilize these poses to transform the corresponding ShapeNet point cloud for metric calculation. We follow the same protocol as [36] to measure and benchmark canonicalization performance for all the methods.

Analysis: Table 5.1 compares the metrics for PCA (P), CaCa (P), ConDor (P), and CaFi-Net [Ours (F)]. We list point cloud canonicalizers with (P) and field canonicalizer as (F) in the table. On average, our method performs better than point cloud-based methods in the CC and GEC metrics and is on par with the state-of-the-art method (ConDor) in the IC metric. This is despite the fact that our method directly operates on the field. We observe that PCA underperforms in all categories suggesting that the noisy point clouds make it unreliable. Similarly, we observe that CaCa fails to canonicalize in categories where high noise is observed. Given that

	bench	cabinet	car	cellph.	chair	couch	firearm	lamp	monitor	plane	speaker	table	water.	avg.	med.
Instance-Level Consistency (IC) ↓															
PCA (P)	14.65	5.94	6.13	**0.67**	6.13	7.63	15.07	12.84	6.78	7.90	5.40	10.31	10.72	8.47	7.63
CaCa (P) [46]	2.89	2.09	2.01	2.81	1.08	1.90	**0.23**	**2.97**	2.35	1.51	2.21	2.42	**2.60**	2.08	2.21
ConDor (P) [36]	**0.56**	**1.27**	**0.36**	**0.53**	**0.96**	**0.32**	**0.60**	**3.78**	**0.44**	**0.64**	**1.97**	**1.32**	**2.15**	**1.15**	**0.64**
Ours (F)	**2.62**	**1.15**	**0.39**	1.28	**0.81**	**1.43**	0.74	3.87	**0.41**	**0.36**	**0.60**	**1.09**	2.74	**1.34**	**1.09**
Category-Level Consistency (CC) ↓															
PCA (P)	14.55	9.68	7.48	4.26	9.90	13.45	15.13	11.87	13.09	8.15	5.51	10.43	11.07	10.36	10.43
CaCa (P) [46]	2.79	**0.38**	1.05	**0.83**	1.82	**1.59**	1.31	**4.34**	**0.39**	1.83	**0.65**	**1.91**	**2.06**	**1.61**	1.59
ConDor (P) [36]	**1.98**	1.68	**0.47**	**1.04**	**1.28**	**0.88**	**0.96**	4.44	1.58	**1.26**	2.10	2.10	2.56	1.72	**1.58**
Ours (F)	**2.77**	**1.36**	**0.48**	1.26	**0.75**	**1.59**	**0.92**	**3.92**	**0.58**	**0.63**	**0.77**	**1.48**	**2.35**	**1.45**	**1.26**
Ground Truth Equivariance Consistency (GEC)↓															
PCA (P)	14.40	5.86	6.32	1.64	6.32	8.12	15.01	12.49	6.82	8.16	5.32	10.17	11.27	8.61	8.12
CaCa (P) [46]	3.65	2.28	2.40	2.74	1.90	2.22	1.28	4.69	2.48	2.09	**2.22**	3.01	**2.79**	2.59	2.40
ConDor (P) [36]	**2.12**	**1.79**	**0.50**	**1.17**	**1.34**	**0.93**	**1.04**	**4.58**	**1.59**	**1.31**	**2.22**	**2.24**	**2.67**	**1.81**	**1.59**
Ours (F)	**3.26**	**1.51**	**0.54**	**1.43**	**0.94**	**1.90**	**1.01**	**4.37**	**0.63**	**0.66**	**0.81**	**1.63**	3.02	**1.67**	**1.43**

Table 5.1: This table compares the canonicalization performance of our method (F - operating on fields) with other 3D point cloud-based methods (P) on three standard metrics (IC, CC and GEC) on our dataset of 13 categories. We compare with PCA, Canonical Capsules (CaCa) [46] and ConDor [36]. All metrics are multiplied by 100 for ease of reading. The top two performing methods are highlighted in **magenta** (best) and **blue** (second best). We are better than SOTA [36] on the Ground Truth Equivariance Consistency (GEC) and Category-Level Consistency (CC) with lower mean and median canonicalization error. However, we perform on par with ConDor [36] on the Instance-Level Consistency.

ConDor also uses TFNs in a different architecture, we conclude that the choice of TFNs makes canonicalization more robust to noise. Our method has the additional advantage of operating directly on the field. Please see 5.3 for a qualitative comparison of the different methods.

5.3 Ablations

We conduct additional experiments to justify key design choices in our method. For all the ablation studies, we use a smaller subset of our data consisting of 4 categories: bench, cellphone, chair, and airplane.

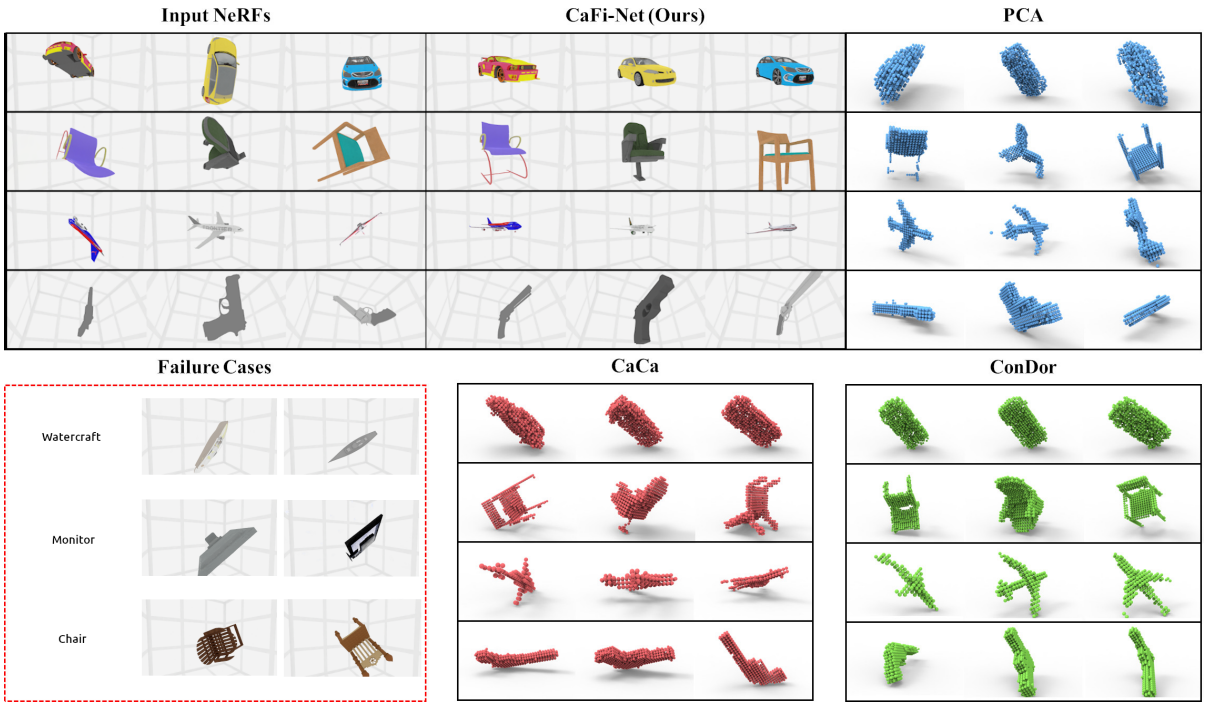

Figure 5.3: This figure visually compares the canonicalization performance of our method with 3D point cloud-based methods. **Input NeRFs** shows one of the RGB views used to learn our NeRF input (notice how orientations of instances are random). **CaFi-Net (Ours)** shows the results of our canonical field rendered from a novel view unseen during NeRF training. **PCA** (blue), **CaCa** [46] (red), and **ConDor** [36]'s (green) results are also presented for the same instances. Our method matches or exceeds ConDor. We also show some failure cases for several instances (red box) which occur either due to thin structures or symmetry.

5.3.1 Choice of Signal Representation

To select the appropriate signal representation to use (see 4), we conducted an ablation study using just the density field xyz locations or its gradients.

The intuition behind this design choice is that gradients of the density capture the surface of the object, which can provide clues for the network. Providing the xyz grid coordinates does not offer specific information about the object, as it remains the same for every instance; then the network must rely solely on weighing the features with density to learn the relevant geometric features for canonicalization. However, even though Nerf densities may not be constant within the foreground and background of objects, gradients can reveal boundaries by generating zero-crossings at these boundaries. This helps the network to learn better geometric

	bench	cellphone	chair	plane	Average
Ground Truth Equivariance Consistency (GEC)↓					
xyz	**2.17**	1.84	1.72	2.07	1.95
Gradient signals	3.26	**1.43**	**0.94**	**0.66**	**1.57**
Instance-Level Consistency (IC)↓					
xyz	**2.04**	1.6	1.4	1.88	1.73
Gradient signals	2.62	**1.28**	**0.81**	**0.36**	**1.26**
Category-Level Consistency (CC)↓					
xyz	**1.99**	1.49	1.52	1.9	1.72
Gradient signal	2.77	**1.26**	**0.75**	**0.63**	**1.35**

Table 5.2: **Choice of Signal Representation** - Canonicalization metrics for using Gradients vs. xyz locations as input signal. Gradients capture the object surface that help in canonicalization.

features which in turn helps in canonicalization. As seen in Table 5.2, the average canonicalization error over 4 categories is lower when using gradient as compared to using density xyz locations on all three metrics.

5.3.2 Weighing Features by Local Average Density

Next, we justify why weighing the equivariant features using the density is helpful. Table 5.3 shows results for weighing features by local average density compared to weighing features by density. The main intuition behind this design choice is to suppress outliers. Upon visualizing Nerf-density fields, we observed the presence of high-density outlier samples in the background. These outliers can influence the overall canonicalization process if we simply weigh the features based on individual density alone. To overcome this, we thought of taking the local average density around the point to weigh features. However, it turns out that

	bench	cellphone	chair	plane	Average.
Ground Truth Equivariance Consistency (GEC)↓					
w/o Local Average Density	**3.26**	**1.43**	**0.94**	0.66	**1.57**
Local Average Density	3.38	1.63	1.3	**0.64**	1.73
Instance-Level Consistency (IC)↓					
w/o Local Average Density	**2.62**	**1.28**	**0.81**	**0.36**	**1.26**
Local Average Density	2.72	1.63	2.64	0.39	1.85
Category-Level Consistency (CC)↓					
w/o Local Average Density	**2.77**	**1.26**	**0.75**	0.63	**1.35**
Local Average Density	2.82	1.54	1.07	**0.51**	1.49

Table 5.3: **Weighing Equivariant Signals by Local Average Density** deteriorates the performance by smoothing out important details of the shape. We show canonicalization with (**w**) and without (**w/o**) weighing by the local averaged density.

weighing features by local average smoothens out details that are necessary to canonicalize the volume. This is especially observed in thin structures. Our average ground truth canonicalization error increases from 1.57 to 1.73 if we weigh features by local averaged densities. In future works, we will delve into better methods for reducing input signal noise that does not smoothen out the geometry details of objects to further improve our canonicalization.

5.3.3 Siamese Architecture

Finally, we justify the need for a two-branch Siamese network architecture. Although our method can canonicalize even without this architecture, it makes it easier to learn over instances in a category as observed in [27]. In this approach, we introduced an additional Chamfer loss between the predicted canonical point clouds from the two branches that are fed with

	bench	cellphone	chair	plane	Average
Ground Truth Equivariance Consistency (GEC)↓					
w/o siamese	3.52	1.61	1.02	1.31	1.86
with siamese	**3.26**	**1.43**	**0.94**	**0.66**	**1.57**
Instance-Level Consistency (IC)↓					
w/o siamese	2.73	1.3	0.82	1.1	1.48
with siamese	**2.62**	**1.28**	**0.81**	**0.36**	**1.26**
Category-Level Consistency (CC)↓					
w/o siamese	2.95	1.44	0.83	1.26	1.62
with siamese	**2.77**	**1.26**	**0.75**	**0.63**	**1.35**

Table 5.4: **Siamese Training** improves performance on all canonicalization metrics on average. We show canonicalization performance *with siamese* and without **(w/o) siamese** training. The average of Ground Truth Equivariance Consistency GEC metric reduces to 1.57 from 1.86

two randomly selected instances with different rotations. Both the branches share the same weights, so we perform two forward passes with different instances, average out the losses, and do one backward pass. Table 5.4 compares our method using a single branch architecture and a Siamese architecture. Clearly, the Siamese architecture helps improve results confirming our hypothesis. Enforcing different instances of the same category to align with each other reduces the Ground Truth Canonicalization error from 1.86 to 1.57. We also see a decrease in all the other metrics including Instance-Level Consistency.

5.4 Qualitative Results

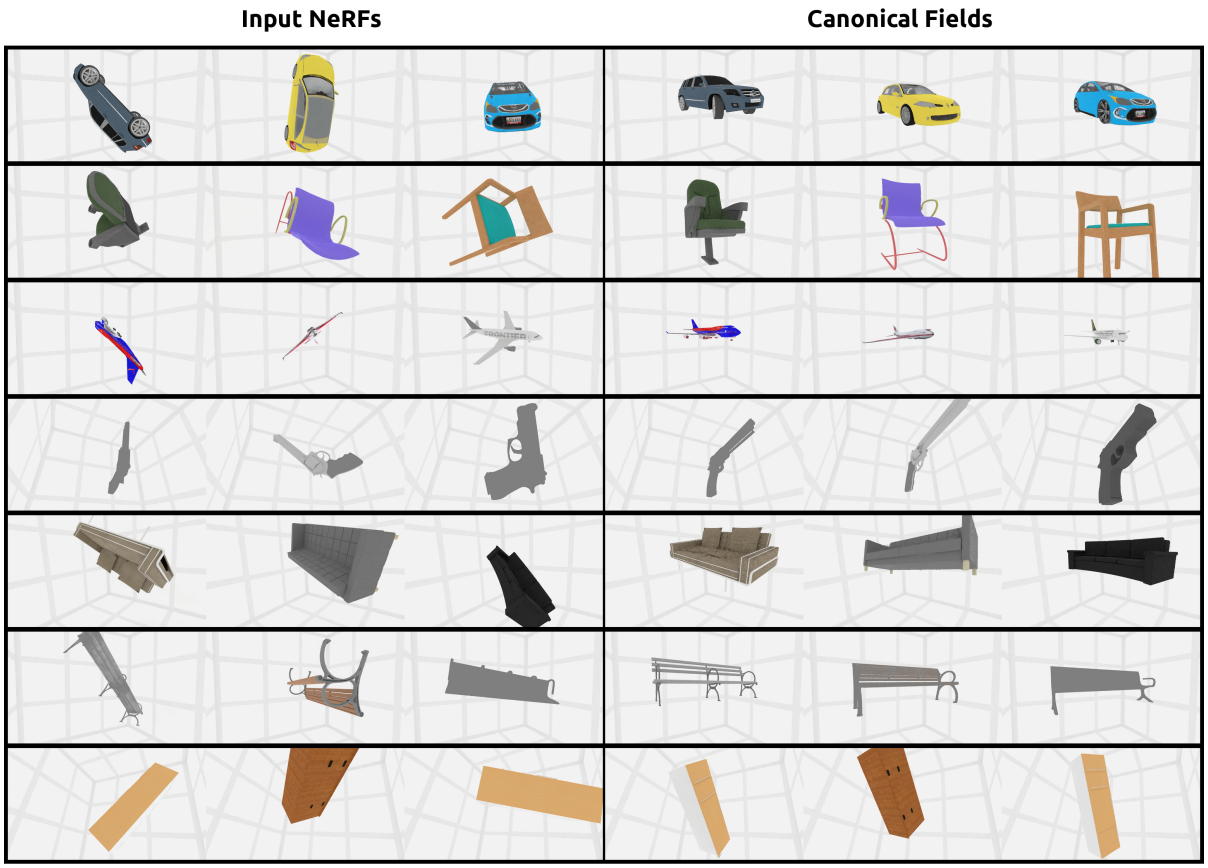

Figure 5.4: **CaFi-Net** qualitative canonicalization results for 7 categories (see following pages for more results).

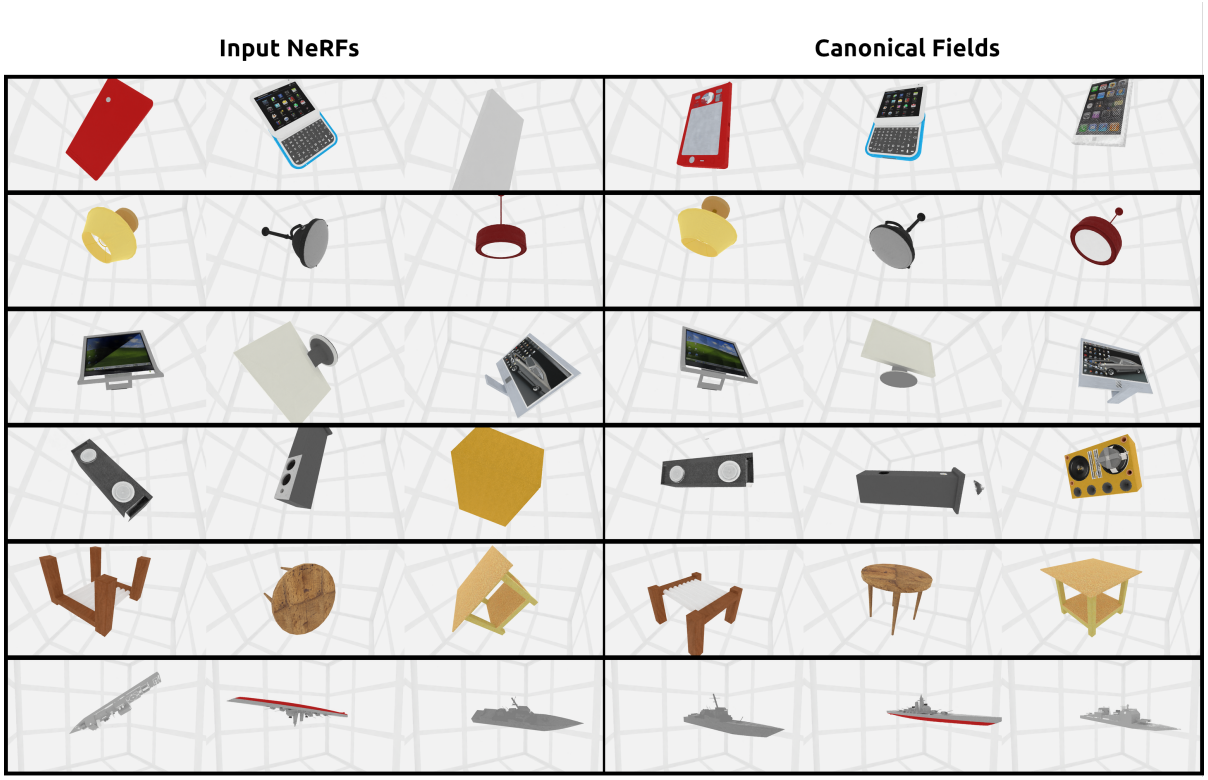

Figure 5.5: **CaFi-Net** qualitative canonicalization results for the remaining 6 categories.

Chapter 6

Conclusion

We presented Canonical Field Network (CaFi-Net), a method for self-supervised category-level canonicalization of the 3D pose of objects represented as neural radiance fields. CaFi-Net consists of a Siamese network architecture with rotation-equivariant convolution layers to extract features for pose canonicalization. Our approach operates directly on NeRF's noisy but continuous density field. We train our method on a large dataset of 1300 NeRF models obtained for 13 common ShapeNet categories. During inference, our method is able to canonicalize arbitrarily oriented NeRFs. Experiments show that our method matches or outperforms 3D point cloud-based methods.

Limitations & Future Work: Our approach has several limitations that future work should investigate. First, we require a 360° NeRF model of objects and cannot handle partial views from front-facing NeRFs. Second, canonicalizing densities with uncertainty is more difficult as the uncertainty increases for very thin/small structures that can be missed. Furthermore, we also inherit the issues that TFNs have with symmetry. In future work, we plan to extend our approach to create a large real dataset of common object categories without requiring manual pose canonicalization and combat the symmetry issue in CaFi-Net as in [38]

Related Publications

1. **Canonical Fields: Self-Supervised Learning of Pose-Canonicalized Neural Fields**
   Rohith Agaram, Shaurya Dewan, Rahul Sajnani, Adrien Poulenard, K.Madhava Krishna, Srinath Sridhar
   The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2023

Other Publications

1. **HyP-NeRF: Learning Improved NeRF Priors using a HyperNetwork**
   Bipasha Sen, Gaurav Singh, Aditya Agarwal, **Rohith Agaram**, K Madhava Krishna, Srinath Sridhar
   Conference on Neural Information Processing Systems(NeurIPS) 2023

Bibliography

[1] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613, 2022.

[2] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[3] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[4] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[6] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[8] E. Corona, T. Hodan, M. Vo, F. Moreno-Noguer, C. Sweeney, R. Newcombe, and L. Ma. Lisa: Learning implicit shape and appearance of hands. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20533–20543, 2022.

[9] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12200–12209, October 2021.

[10] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.

[11] R. Gao, Y.-Y. Chang, S. Mall, L. Fei-Fei, and J. Wu. Objectfolder: A dataset of objects with implicit visual, auditory, and tactile representations. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2021.

[12] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. 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.

[13] H. Group. Pinhole camera model. 2019.

[14] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. Oct 2021.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[16] A. Jain, M. Tancik, and P. Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[18] J. Lei, S. Sridhar, P. Guerrero, M. Sung, N. Mitra, and L. J. Guibas. Pix2surf: Learning parametric 3d surface models of objects from images. In *European Conference on Computer Vision*, pages 121–138. Springer, 2020.

[19] R. Li, J. Tanke, M. Vo, M. Zollhofer, J. Gall, A. Kanazawa, and C. Lassner. Tava: Template-free animatable volumetric actors. *arXiv preprint arXiv:2206.08929*, 2022.

[20] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, R. Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5521–5531, 2022.

[21] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3706–3715, 2020.

[22] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba. 3d neural scene representations for visuomotor control. In *Proceedings of Robotics: Science and Systems*, 2021.

[23] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[24] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022.

[25] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[26] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[27] D. Novotny, N. Ravi, B. Graham, N. Neverova, and A. Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7688–7697, 2019.

[28] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. 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.

[29] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[30] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[33] S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, X. Zhou, and H. Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021.

[34] A. Poulenard and L. J. Guibas. A functional approach to rotation equivariant non-linearities for tensor field networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13174–13183, 2021.

[35] V. Rudnev, M. Elgharib, W. Smith, L. Liu, V. Golyanik, and C. Theobalt. Nerf for outdoor scene relighting. In *European Conference on Computer Vision (ECCV)*, 2022.

[36] R. Sajnani, A. Poulenard, J. Jain, R. Dua, L. J. Guibas, and S. Sridhar. Condor: Self-supervised canonicalization of 3d pose for partial shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16969–16979, 2022.

[37] R. Sajnani, A. Sanchawala, K. M. Jatavallabhula, S. Sridhar, and K. M. Krishna. Draco: Weakly supervised dense reconstruction and canonicalization of objects. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10302–10309. IEEE, 2021.

[38] A. Seo, B. Kim, S. Kwak, and M. Cho. Reflection and rotation symmetry detection via equivariant learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[39] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[40] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. 2022.

[41] V. Sitzmann, M. Zollhoefer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[42] R. Spezialetti, F. Stella, M. Marcon, L. Silva, S. Salti, and L. Di Stefano. Learning to orient surfaces by self-supervised spherical cnns. *arXiv preprint arXiv:2011.03298*, 2020.

[43] S. Sridhar, D. Rempe, J. Valentin, S. Bouaziz, and L. J. Guibas. Multiview aggregation for learning category-specific shape reconstruction. *arXiv preprint arXiv:1907.01085*, 2019.

[44] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.

[45] S.-Y. Su, F. Yu, M. Zollhöfer, and H. Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Advances in Neural Information Processing Systems*, 34:12278–12291, 2021.

[46] W. Sun, A. Tagliasacchi, B. Deng, S. Sabour, S. Yazdani, G. Hinton, and K. M. Yi. Canonical capsules: Unsupervised capsules in canonical pose. *arXiv preprint arXiv:2012.04718*, 2020.

[47] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019.

[48] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, Y. Wang, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Niessner, J. T. Barron, G. Wetzstein, M. Zollhoefer, and V. Golyanik. Advances in neural rendering. Nov 2021.

[49] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[50] G. Tiwari, D. Antic, J. E. Lenssen, N. Sarafianos, T. Tung, and G. Pons-Moll. Pose-ndf: Modeling human pose manifolds with neural distance fields. In *European Conference on Computer Vision (ECCV)*, October 2022.

[51] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular

video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021.

[52] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. S. M. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes, 2021.

[53] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[54] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[55] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NIPS*, pages 10381–10392, 2018.

[56] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant cnns. In *Proc. CVPR*, pages 849–858, 2018.

[57] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. CVPR*, pages 1912–1920, 2015.

[58] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

[59] G. Yang, S. Belongie, B. Hariharan, and V. Koltun. Geometry processing with neural fields. *Advances in Neural Information Processing Systems*, 34:22483–22497, 2021.

[60] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2021.

[61] L. Yen-Chen. Nerf-pytorch. `https://github.com/yenchenlin/nerf-pytorch/`, 2020.

[62] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[63] K. Zakka, A. Zeng, J. Lee, and S. Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020.

[64] G. Zhang, O. Litany, S. Sridhar, and L. Guibas. Strobenet: Category-level multiview reconstruction of articulated objects. *arXiv preprint arXiv:2105.08016*, 2021.

[65] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[66] Y. Zhao, T. Birdal, H. Deng, and F. Tombari. 3d point capsule networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1009–1018, 2019.

[67] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.