DualNeRF – Image Based 3D Reconstruction via Implicit Neural Representation and Neural Rendering without 3D Supervision

Tianhao (Walter) Wu[1]

MEng Computer Science

Lourdes De Agapito Vicente

Submission date: 1st May 2021

[1]**Disclaimer:** This report is submitted as part requirement for the MEng at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged

Abstract

We propose DualNeRF, a deep learning framework that reconstructs the 3D geometry and appearance from a single RGB image input, without requiring any ground truth 3D labels as training data. Traditional deep learning approaches for reconstruction task usually rely on explicit representations of the 3D world, such as point cloud, mesh and voxel grid, which contain undesired properties such as discretisation. Our proposed method instead predicts an implicit neural representation, which is a continuous and differentiable function stored in the weights of a neural network. With this representation, our model is able to render differentiable novel views of the target object. Compared to existing approaches, our method further removes the dependency on the point feature input and enforces the neural network to fully condition on the input image to predict a reliable implicit representation. We conduct extensive experiments on both synthetic ShapeNet dataset and real-life Stanford Car Dataset to demonstrate the performance of our model in comparison with the state-of-the-art method and also carry out a throughout ablation study to evaluate each component of our method.

Contents

Chapter 1

Introduction

1.1 Image-based 3D Reconstruction

Accurately capturing and reconstructing the 3D geometry of an object from 2D images is a long-standing problem in computer vision. It is fundamental to many applications such as 3D modelling, virtual reality (VR), 3D scene understanding, robotics and medical imaging. Without additional information such as the materials, lighting and camera parameters, the reconstruction problem becomes ill-posed due to lack of constraints in the images alone. While traditional multi-view stereo (MVS) [1] approaches apply general assumptions to constrain the problem and achieve reliable performance when a dense set of calibrated views is available, they cannot be applied to situations where the input is limited to be a single image. To extract the correct 3D shape from a single 2D image, the method must incorporate a good understanding of the 3D space and a strong prior knowledge of the reconstruction target to compensate for the insufficient information.

1.2 Recent Advances in Deep Learning Based Methods

Recent advances in deep learning based methods have yielded rapid progress in 3D reconstruction. Image inputs are typically processed with convolutional neural networks (CNN) to directly predict a 3D model in explicit representations such as voxels [2, 3, 4], point clouds [5, 6] and mesh [7, 8, 9]. However, those representations come with issues including undesirable discretisation and difficulty in computing gradient due to rasterisation. As an alternative, reconstruction via implicit neural representation has been proposed. This class of method models the 3D geometry as an implicit function encoded in the weights of a neural network. Due to their properties of continuous and naturally differentiable, loss on either the 3D geometry or the rendering can be easily used to supervise the neural network to learn the 3D shape, appearance or lighting of the target object.

However, models that directly optimise the 3D geometry loss apparently require ground truth 3D supervision, which is expensive to obtain for a realistic dataset. Therefore, synthetic dataset such as ShapeNet [10] is widely used as training data instead, potentially causing the models to be unreliable in practical scenarios. On the other hand, many models trained without ground truth 3D supervision either require test-time optimisation [11, 12] or are designed to work with a single scene or object only, resulting in extremely poor performance when generalising to unseen objects [13, 14].

1.3 Project Achievement

Through this project, we propose DualNeRF, a deep learning framework that takes as input a single RGB image with corresponding camera parameters and produces a continuous volumetric radiance field, which is a continuous and differentiable implicit representation of 3D geometry and appearance. The deep learning networks are trained end-to-end using images of different objects with camera intrinsics and extrinsics, without the need to access any ground truth 3D geometry. The trained model is able to synthesise novel views of unseen objects under the same category given a posed RGB image input. When an explicit representation is needed, algorithms such as marching cubes [15] can be applied to easily convert the continuous volumetric radiance field to an explicit mesh representation. In particular, our approach consists of the following components:

- An CNN encoder that extracts a 3D feature volume from an input RGB image. Each pixel in the input image corresponds to a 1D local feature vector, which is produced by up-sampling and concatenating the outputs of several convolutional layers in the encoder.
- A multilayer perceptron (MLP) decoder with skip connections to decode the local feature vector concatenated with a depth and view direction into the RGB value (radiance) and density, which fully describe the geometry and appearance of the 3D space in an implicit way.
- An additional MLP decoder that decodes the full image feature concatenated with 3D coordinate and view direction into a different set of RGB and density predictions. Predictions from two decoders are averaged to produce a final result.
- A differentiable volumetric renderer that renders samples along each ray into a pixel value. By sampling sufficient points in the 3D space, the image from a specified viewpoint and view direction can be rendered.

Our key contributions are:

- We propose an autoencoder framework that learns rich class-specific priors and can extract an implicit representation of geometry and appearance from a single image input. The framework achieves remarkable generalisation ability and can reconstruct any unseen object under the same category at test time.
- We build on the existing continuous volumetric radiance field approach NeRF [13] and the subsequent work pixelNeRF [16] and further breaks the dependency between implicit function output and the query input by replacing the 3D coordinate in the query with a single depth value, which represents how deep the queried point is into the scene. This enforces the implicit function to depend less on the coordinate input but more on the concatenated image feature, therefore achieving a higher generalisation ability.
- We propose to use an additional global decoder that takes the full image feature into account. The use of global feature may lead to coarse and blurry prediction, but it guides the framework to produce a more globally consistent reconstruction.
- We carry out extensive experiments and evaluations to demonstrate the performance and our methods, in comparison with existing baseline approaches.

In the rest of this paper, we first briefly discuss the recent development in the field of 3D reconstruction and novel view synthesis to introduce the advantages of implicit neural representation over traditional explicit representation. We then present the reader the necessary technical background to fully comprehend the challenges and our solution, including the classic pinhole camera model, volume rendering and the existing NeRF approach, which our method builds and improves on. We explain in details how our method works and demonstrate its performance both quantitatively and qualitatively on the synthetic ShapeNet [10] dataset and the real-life Stanford Car dataset [17], with comparison to the state-of-the-art baseline methods.

Chapter 2

Literature Review

2.1 Classic Multi-view Stereo Reconstruction

Classic 3D reconstruction approaches usually consist of a Structure-from-Motion to Multi-view Stereo (SfM-MVS) pipeline, where an SfM algorithm first extracts and matches visual features from a set of unstructured images, then estimates the camera parameters for each image relying on the feature correspondence. An MVS algorithm then takes the images and camera parameters as input and reconstructs a dense 3D model of the underlying geometry.

Due to the rapid growth of computational complexity with the number of input views, MVS algorithms that directly produce a global 3D geometry are unable to deal with large scale reconstruction. Furukawa et al. [18] overcome this difficulty and achieve city-scale 3D reconstruction by splitting the unstructured image inputs into overlapping clusters, where existing MVS algorithms can be run in parallel on each cluster. The reconstructions from each cluster are then merged together to form a global geometry.

Despite the success of MVS approaches, relying on general hand-crafted prior knowledge limits their performance under challenging scenarios. For example, MVS can only reconstruct the 3D geometry covered by the input views, and therefore cannot produce a full reconstruction from sparse views. This limitation of MVS leads to the development of data-driven approaches, where either general or specific priors can be extracted automatically from the training data.

2.2 Deep Learning Based Reconstruction via Traditional Representation

The development in deep learning has led to promising results in image-based reconstruction. Several methods have been proposed to learn a direct mapping from one or multiple RGB images to an explicit 3D representation, including voxels [2, 3], point clouds [5, 6] and mesh representation [7, 8]. Due to the requirement of ground truth 3D shape as supervision, the above methods usually rely on the synthetic dataset for training and may not work well with real-life scenes. The use of explicit 3D representation also means that the quality of reconstruction is limited by resolution. Using a high resolution can produce sharper and more detailed reconstruction, but would also increase the amount of information to be learnt by the neural network, therefore requiring more data, larger network and more storage. Because the network directly produces a discrete 3D representation, the resolution of reconstruction is fixed and the network cannot easily scale to a

different resolution at inference time.

2.3 Learning Implicit Neural Representation with 3D Supervision

Implicit neural representation has recently emerged as a novel direction for 3D shape and appearance reconstruction. It is a continuous and differentiable function modelled by neural networks, which implicitly represent the underlying scene by associating each 3D spatial location with its geometric feature such as colour, density, reflectance or distance to the closest surface. By sampling sufficient points in the continuous 3D space and querying the neural networks with those 3D locations, an explicit representation with customisable resolution and precision can be obtained.

Compared with traditional explicit 3D representations, implicit representation only needs to store a set of weights for neural network and thus is more memory efficient. In addition, because it is a continuous function, it naturally avoids rasterisation and does not suffer resolution constraint, nor does it require approximation to become differentiable [19, 20].

2.3.1 Signed Distance Function

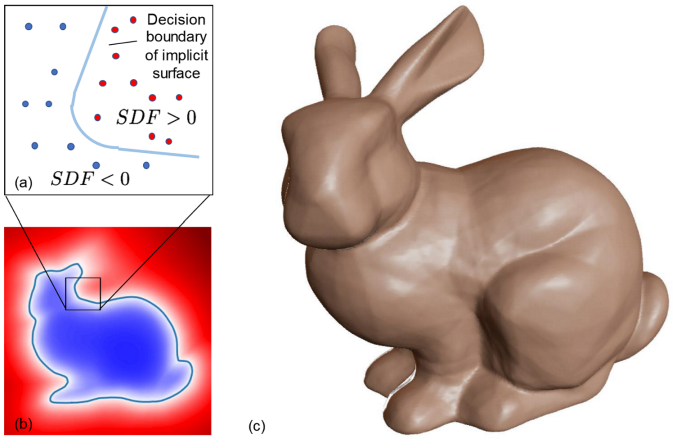

Figure 2.1: **Illustration of an SDF implicitly representing the Stanford Bunny from DeepSDF. [19].** (a) signs of values returned from SDF near a surface. The zero level set indicates the surface. (b) a 2D slice of SDF values in the 3D space. Colour represents the sign and brightness represents the magnitude. (c) a rendering of the surface represented by this SDF.

Signed distance function (SDF) is an implicit function that represents an arbitrary surface in a 3D space. It is a continuous and differentiable function that takes as input 3D coordinates and outputs a real value, whose sign indicates whether the point is inside or outside the surface and the magnitude indicates the distance to the closest surface. The 3D surface is therefore implicitly defined by the zero level set of the function. Given an SDF, 3D shape of the underlying object can be explicitly found via algorithms such as marching cubes [19].

The use of SDF in the 3D reconstruction task has achieved reliable results. Park et al. [19] use an MLP with skip connection to model an SDF. In order to reconstruct unseen objects, the network uses an auto-decoder architecture where a latent code is optimised at inference time for each target object. The latent code is concatenated with the 3D coordinate and passed into an MLP

decoder to obtain a signed distance. This use of auto-decoder reduces computational complexity and improves the generalisation ability. DISN proposed by Xu et al. [20] further improves the model's ability to capture fine details by incorporating a CNN encoder to extract a feature vector from the input RGB image. In addition to a conventional feature vector that is extracted from the full image, the encoder also produces a local feature pyramid, which is passed to a separate local decoder network to produce more detailed predictions. Our method of combining local and global features is heavily inspired by it, but we incorporate a different implicit representation that can be naturally applied with neural rendering to allow training without ground truth 3D supervision. Both of the above approaches can reconstruct the 3D surface based on an input image and can generalise to unseen objects, but they cannot model complex scenes containing multiple objects. Runz et al. [21] develop a framework that detects, segments and groups individual objects in the images of a complex scene. Multiple sparse views of the same object are embedded into a latent space and are fused to form one feature vector, which is then decoded into both a dense SDF representation and a sparse point cloud representation to allow fast evaluation and training in a coarse-to-fine approach. Nevertheless, above methods all require 3D supervision during training and do not reconstruct object appearance.

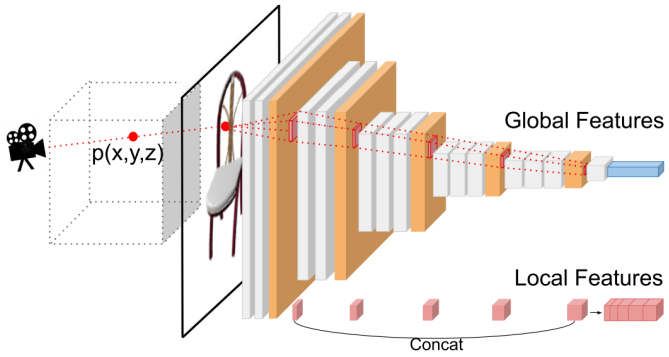

Figure 2.2: **Illustration of DISN local feature extraction [19].** In addition to the global feature computed from the full image, a local feature specific to the 3D point being queried is also extracted. This is done by first projecting the 3D point back to the image plane to find out its coordinate on the image, then extracting the intermediate features correspond to this pixel coordinate from the outputs of several layers and concatenating them together.

2.3.2 Structured Implicit Functions

Genova et al. [22] propose structured implicit functions (SIF) as a general 3D shape representation with a great scalability to complex or detailed objects. It is defined as a set of parameters describing multiple independent anisotropic 3D Gaussian models, each accounts for the geometry of a local space. The full 3D space is therefore described by the sum of all Gaussian functions as a level contour, where the object surface is represented by a certain level set defined by the model. While this type of shape representation has already been introduced in traditional computer graphics, Genova et al. incorporate a multi-view autoencoder network that infers the Gaussian parameters directly from a set of posed depth views. This pipeline generates a strongly consistent shape template across all shape categories, where a Gaussian model tends to describe the local area with the same semantic meaning (e.g., chair legs). Therefore, this representation naturally includes rich shape prior information and can be easily used to find correspondence between shapes.

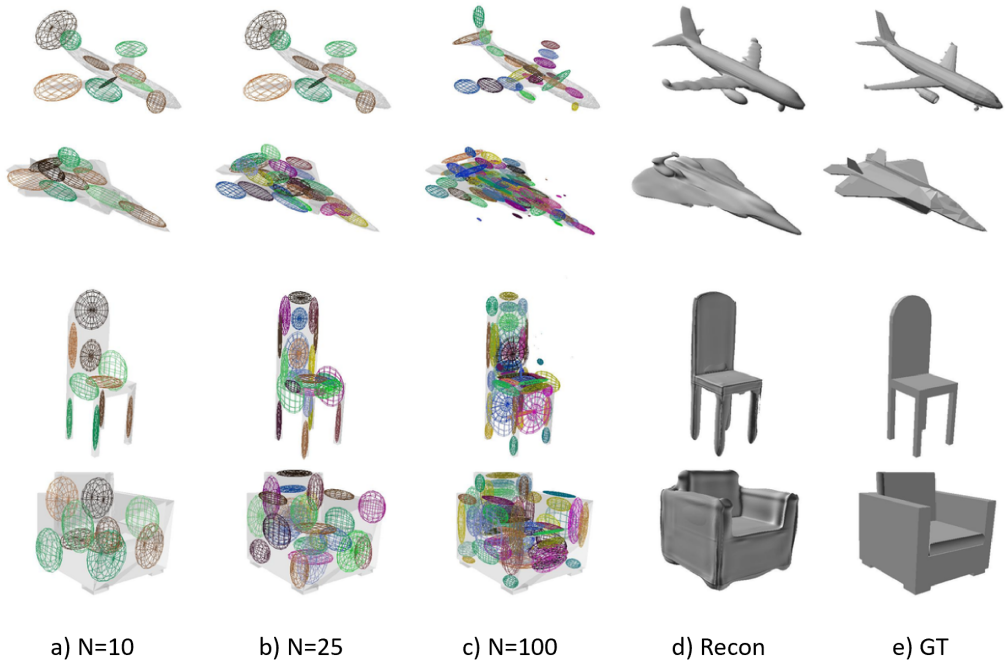

Figure 2.3: **An illustration of representing shapes as SIF modified from [22].** (a, b, c) SIF representation of the objects, where N indicates the number of Gaussian models used in SIF. (d) a rendering of surface reconstructed from SIF in (c). (e) the ground truth object.

Genova et al. [23] build on SIF and further propose local deep implicit functions, which are the combination of SIF and a local point feature decoder. Given one or more posed depth image, the SIF network is first used to learn the 3D Gaussian parameters as a coarse model. Points are then sampled around each local area described by individual Gaussian functions and passed through a local autoencoder network to provide a more detailed shape prediction in a coarse-to-fine manner. By limiting the prediction target to be a small local space of the object, the local autoencoder requires much less capacity and can be trained very efficiently.

2.4 Learning with Image Supervision via Neural Rendering

Neural rendering is a method that renders 2D images in a deep learning approach, where the properties of rendering, such as viewpoint, illumination, geometry and appearance, can be specified explicitly or implicitly [24]. The use of neural rendering allows the computation of gradient on the rendered image, which is usually non-differentiable due to the rasterisation step in the traditional rendering process. With a computable gradient, the photometric loss on the rendering can be back-propagated to the network to supervise the learning, therefore the network can be trained without any 3D ground truth labels. While this approach can be applied to learn a traditional 3D representation via neural network, there also exists several implicit representation that is designed to work with neural rendering and yield a great performance in terms of geometry and apperance reconstruction.

2.4.1 Learning Traditional Representation with Image Supervision

Recent advance in neural rendering techniques has led to many deep learning frameworks that can be trained without ground truth 3D supervision. To enforce the neural network to learn a correct 3D representation instead of simply memorising the 2D image, additional techniques are usually required. Dupont et al. [4] achieve the 3D reconstruction of unseen objects in the same category with only image and camera pose supervision. Relying on the fact that transformation should be equivariant in both original and reconstructed 3D space, the network is trained using image pairs of the same object and the relative rotations angle of the camera. The reconstructed voxel grid from the first image is trained to match the second image after rotating by their relative angle and vice versa. Guandao et al. [25] relax the supervision requirements further by developing a unified framework that can be trained using a mixture of images with and without class label and camera pose. In addition to applying a view-independent constraint similar to [4], they also incorporate Generative Adversarial Networks (GANs) to encourage realistic-looking reconstructions. Wu et al. [26] focus on reconstruction of face and other deformable objects from in-the-wild images without additional supervision. They assume that faces and common objects are highly symmetrical and incorporate additional constraints by flipping the input image and reconstructing on both original and flipped images. To deal with asymmetric aspects such as hair, they also predict two confidence maps for original and flipped images to learn what parts of object are potentially asymmetric. Without access to any 3D ground truth supervision, the above methods can still accurately reconstruct the geometry of unseen objects. However, the use of explicit representation limits their abilities to model fine details and unique characteristics of each object.

2.4.2 Learning SDF with Image Supervision

While the SDF solely describes geometry features and cannot be learnt directly with RGB image supervision, it is possible to combine the SDF with a neural rendering model and learn them together from images. Yariv et al. [14] propose an SDF-based architecture to learn the geometry, appearance and lighting of an object from a set of masked and posed images. They define the pixel values from a rendering as a function of object SDF, appearance and camera parameters. By approximating the rendering process, this function is made differentiable and thus can be trained end-to-end in a deep learning pipeline. However, it is an optimisation-based method that only learns the reconstruction of a single object at a time.

Lin et al. [12] propose a method that learns in extremely general condition, where only a single RGB image for each object is assumed to exist in the training dataset. Similar to [14], they use separate SDF and colouring functions to render the view from a specified viewpoint and compare it with the ground truth view to obtain a photometric loss as supervision. They additionally make use of information from a 2D silhouette to provide a lower bound on the signed distance for 3D points outside of the surface. Despite the limited amount of supervision, this model is able to reconstruct the geometry of unseen objects under the same category.

2.4.3 Scene Representation Networks

Scene Representation Networks (SRNs) [11] mainly consist of a neural network that represents arbitrary scenes by mapping each 3D coordinate into a feature vector, which implicitly contains all geometric properties needed to render the local area. This feature is then decoded through a pixel generator network into the RGB value for a pixel. SRNs also include a ray marcher with learnable

step size to allow fast tracking of the intersection point between a concrete surface and the camera ray in the 3D space, hence the 3D coordinate that needs to be queried to get the correct pixel value can be efficiently computed.

The SRNs representation naturally comes with a differentiable pixel renderer and can therefore be supervised with images and camera poses only. The framework incorporates a Hypernetwork, which is an MLP network that decodes an object feature vector obtained using an auto-decoder architecture from [19] into the weights of an MLP scene representation network for each target object. This network can then be queried with 3D coordinates to return geometry features which are then decoded into RGB values by the pixel renderer. By optimising the object feature vector, learnable ray marcher and all decoder networks jointly, the framework is trained to achieve 3D reconstruction of unseen objects under the same category. However, the use of object feature vector requires test-time optimisation for each novel object being reconstructed, hence reducing the algorithm efficiency at inference time.

2.4.4 Continuous volumetric radiance field

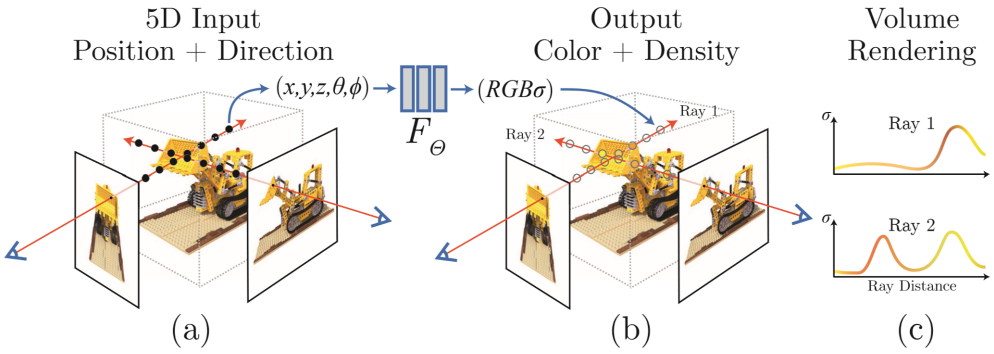

Figure 2.4: **A continuous volumetric radiance field representing a LEGO model and the volume renderings from different viewpoints.** (a, b) the continuous volumetric radiance field that takes a position and a direction, and returns the RGB and density. Views of the object from any viewpoint can be rendered by sampling this function. (c) the volume rendering approach, where points along a ray are integrated into a pixel value. Points with a higher density are more likely to stop the ray, hence their radiance would weight more in the rendering. Illustration from NeRF [13]

A continuous volumetric radiance field [13] is a 5D function that maps each 3D spatial location and 2D radiance direction into an RGB colour (radiance) and a density value, which implies the occupancy of a particle on this point. This representation can naturally model both the 3D geometry and appearance of an arbitrary complex scene, and is able to model non-Lambertian effects, meaning the brightness of a surface may vary with the view direction. Through querying points in the 3D space and collecting their density values, a depth map of the object can be rendered from any viewpoint, and the field can be easily converted to an explicit mesh representation via marching cubes algorithm. A 2D view of the field from specified viewpoint and view direction can also be rendered via traditional volume rendering [27], which avoids rasterisation and has a gradient that can be easily computed.

NeRF [13] uses an MLP architecture similar to [19] to model the continuous volumetric radiance field. The network is supervised with the photometric loss between the volume rendered view and

the ground truth view. By optimising over a set of images of the same object, NeRF is able to accurately learn the underlying 3D geometry and appearance of a very complex object or scene. The subsequent work NSVF [28] combines the implicit continuous volumetric radiance field with explicit voxel grid representation and further improves rendering efficiency. However, both methods are designed to learn the reconstruction of a single scene or object at a time, and therefore lose the ability to generalise to unseen targets. To reconstruct a different object, a new network must be trained from scratch and a dense set of views of the target object has to be collected.

Subsequent work improve upon the implementation of NeRF and achieve reconstruction on unseen objects. GRF [29] extracts spatiality-aware local features for each pixel of the input image by concatenating the RGB channels with the spatial location of the input camera, then passing them into a CNN encoder to obtain per-pixel local image features. For a 3D point being queried, GRF retrieves local features from multiple input images of the same object and applies *attention aggregation* to combine all useful information into a single feature vector, which is then concatenated with the view direction feature and decoded into the radiance and density values. pixelNeRF [16] incorporate the rotational equivariant similar to [4] by requiring the network to reconstruct a target view based on an input view from a different viewpoint. It extracts a local image feature similar to [20] and concatenates it with spatial point features transformed into an input view camera coordinate system. This enforces the network to make predictions by conditioning on the input image feature, and also prevents the deep learning model from overfitting to the canonical world coordinate system. Compared to pixelNeRF, our method further reduces the dependency on the coordinate system by providing only a depth value as spatial information, and we additionally incorporate a decoder that conditions on the full image feature to provide predictions based on global clues.

2.5 Novel View Synthesis

2.5.1 Traditional Novel View Synthesis

Novel view synthesis involves generating new views of a target scene or object from unseen viewpoints given a set of input views. A class of approaches achieve this by compositing input views based on their poses, without directly reconstructing the underlying geometry. Levoy et al. [30] interpret input images as 2D slices of a 4D light field and generate novel views by interpolating input views in this free space. This 4D light field is simplified from the 5D plenoptic function [31] by assuming the radiance of a point remains constant along a direction, and is possible because only blocked points along a ray would give different radiance, but they are visually occluded and therefore not needed for rendering views.

After the emergence of sophisticated MVS methods, some approaches incorporate off-the-shelf MVS algorithms to obtain an explicit global geometry, then apply blending to a novel view based on the input views. Hedman et al. [32] develop an image-based-rendering (IBR) pipeline that achieves real-time free viewpoint rendering of complex indoor scenes from a set of RGB and RGB-D images. To resolve the issues with misalignment in object edges and global geometry estimation, the pipeline works by first reconstructing a coarse global mesh from RGB-D images, then refining it using high resolution RGB images to create per-view mesh. The mesh visible from the queried viewpoint is then blended based on the input views by optimising over an IBR cost, which involves a relative angle term that accounts for view-dependent effects in the rendering, and a distance term to encourage the use of input views that are close to the rendering surface, so that sharper

and more accurate views can be rendered.

2.5.2 Deep Learning Approaches

More recently, deep learning techniques are incorporated into the traditional novel view synthesis pipeline. Blending is a complex step in the pipeline because it needs to deal with difficult view-dependent effects and compensate for potential errors in the previous geometry reconstruction step. Traditional approaches usually estimate the optimal blending weights using a set of hand-crafted heuristics. Hedman et al. [33] improve upon this by training a CNN to predict the blending weights for pixels from different input views in a single forward pass. To blend a novel view, the pixels from input views are ranked by their IBR cost and organised into several *mosaics*, where each pixel in the first mosaic has the lowest IBR cost to the corresponding pixel in the novel view. A CNN takes a raw view of the unblended mesh view concatenated with the top four mosaics and extracts the blending weights for each pixel.

Chapter 3

Technical Background

3.1 Perspective Projection Model

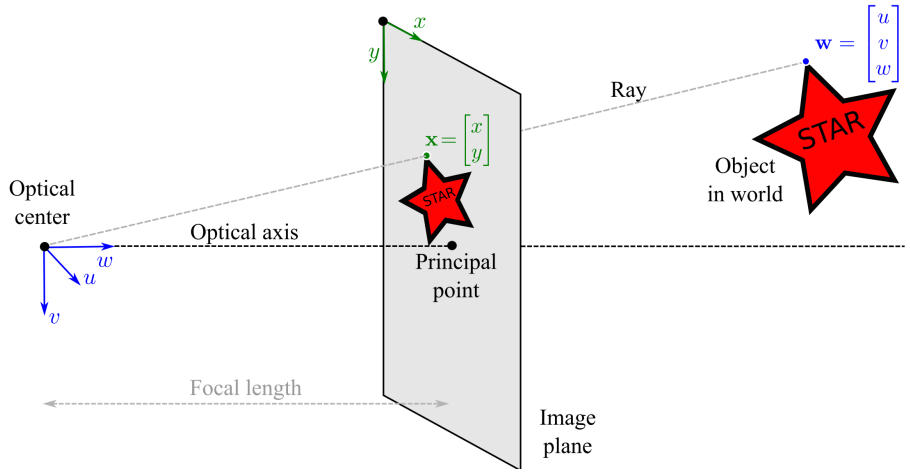

Figure 3.1: **The pinhole camera model.** In theory, the model uses a virtual image plane located between the camera and object, thus the formed image is not inverted. Illustration from [34].

The pinhole camera is a widely used camera model in computer graphics and vision. The model is fully parameterised by a 3×4 matrix $\mathbf{A}$ up to a scale factor. The matrix $\mathbf{A}$ is usually decomposed into the product of a 3×3 matrix $\mathbf{K}$ representing the camera intrinsics, and a 3×4 matrix $\mathbf{P}$ representing the extrinsics or pose.

$$\begin{aligned} \mathbf{A} &= \mathbf{KP} \\ \mathbf{K} &= \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{P} &= \left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{array}\right) = [\mathbf{R}|\mathbf{T}] \end{aligned} \tag{3.1}$$

K contains the intrinsic parameters of the camera: f_x, f_y represent the focal lengths in vertical and horizontal directions. c_x, c_y represent the coordinate of the principal point, which is defined as the projection of the optical centre (camera origin) on the image plane. s defines the image skewness, which is needed when the image plane is not perpendicular to the optical axis.

P contains the extrinsic parameters that describe the location and orientation of the camera. It can be further decomposed into a 3×3 matrix **R** describing the rotation and a 1×3 vector describing the translation.

Due to a large number of unknowns in the pinhole camera model, simplifications are often made to reduce its complexity, including assumptions that focal length is the same in both directions ($f_x = f_y = f$), no skewness is present in the image ($s = 0$) and the principal point is at the centre of any un-cropped image ($c_x = c_y = 0$).

3.2 Volume Rendering

For a view point with position $\mathbf{o} \in \mathbb{R}^3$ and view direction $\mathbf{d} \in \mathbb{R}^3$, every pixel with coordinate i, j on the image has a associated ray parameterised as $\mathbf{r}_{i,j}(t) = \mathbf{o} + t\mathbf{d}_{i,j}$, where t represents the Euclidean distance between the 3D point to the view point, $\mathbf{d}_{i,j} \in \mathbb{R}^3$ represents the unit length ray direction and can be obtained by first calculating the relative direction in the camera coordinate system, then transforming from camera coordinate system into world coordinate system by pre-multiplying the camera-to-world rotation matrix $\mathbf{R}_{c2w}$:

$$\begin{aligned} \mathbf{d}'_{i,j} &= [\frac{i - W/2}{f}, \frac{j - H/2}{f}, 1]^T \\ \mathbf{d}_{i,j} &= \mathbf{R}_{c2w}\mathbf{d}'_{i,j} \end{aligned} \tag{3.2}$$

where W, H are the width and height of the image.

Classic volume rendering [27] can be used to render a pixel value by integrating over the radiance of all points on the ray, weighted by their densities and distances to the camera.

$$\begin{aligned} \hat{\mathbf{C}}(\mathbf{r}) &= \int_{t_{near}}^{t_{far}} T(t)\sigma(t)\mathbf{c}(t)dt \\ T(t) &= exp\left(-\int_{t_{near}}^{t_{far}} \sigma(s)ds\right) \end{aligned} \tag{3.3}$$

where t_{near} and t_{far} are the near and far bounds, $\sigma(t)$ and $\mathbf{c}(t)$ are the density and radiance of the point respectively.

In practice, the integrals are approximated using numerical quadrature with points sampled along the ray:

$$\begin{aligned} \tilde{\mathbf{C}}(\mathbf{r}) &= \sum_{i=1}^{N} \tilde{T}(t_i)\alpha(t_i)\mathbf{c}(t_i) \\ \tilde{T}(t_i) &= exp\left(-\sum_{j=1}^{i-1} \sigma(t_j)\delta_{\mathbf{t}}(j)\right) \\ \alpha(t_i) &= 1 - exp\Big(-\sigma(t_i)\delta_{\mathbf{t}}(i)\Big) \end{aligned} \tag{3.4}$$

where $\delta_{\mathbf{t}}(i)$ represents the distance between sample i and $i+1$ and is evaluated to infinity for the last sample.

$$\delta_{\mathbf{t}}(i) = \begin{cases} t_{i+1} - t_i, & \text{if } i < N \\ \infty, & \text{if } i = N \end{cases} \tag{3.5}$$

3.3 NeRF

This section briefly reviews the work of NeRF [13], which lays the foundation for our work. NeRF introduces a simple MLP architecture to learn the 3D geometry, appearance and lighting of a scene from a set of images with the corresponding camera poses.

3.3.1 NeRF Model

The NeRF model is an MLP network that represents a 3D scene by encoding a continuous volumetric radiance field of colour and density. Given the point features, which are the 3D coordinate $\mathbf{x} \in \mathbb{R}^3$ and the view direction $\mathbf{d} \in \mathbb{R}^3$, NeRF returns the density σ and the RGB colour $\mathbf{c}$ of the point.

$$NeRF(\gamma_{L_{\mathbf{x}}}(\mathbf{x}), \gamma_{L_{\mathbf{d}}}(\mathbf{d})) = (\sigma, \mathbf{c}) \tag{3.6}$$

where $\gamma_L(x)$ is a positional encoding function that enforces the network to learn the high frequency functions to better fit the data. It simply maps the low dimensional input x into corresponding L dimensional Fourier feature:

$$\gamma_L(x) = (sin(2^0\pi x), cos(2^0\pi x), ..., sin(2^{L-1}\pi x), cos(2^{L-1}\pi x)) \tag{3.7}$$

Through volume rendering, points are sampled along the camera rays and integrated into a pixel value, which is then compared with the ground truth pixel value to obtain a training loss.

$$\mathcal{L}(\mathbf{r}) = \left\| \tilde{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2 \tag{3.8}$$

3.3.2 Hierarchical Sampling

A hierarchical sampling method is used to improve the efficiency and accuracy of rendering. This method involves training two separate networks: one coarse network that is queried with N_c stratified samples with a fixed interval, one fine network that is queried with N_f samples determined by the result of the coarse network and thus tend to be more informative.

Points for the coarse network are sampled with a stratified method. The distance range $[t_{near}, t_{far}]$ is first partitioned into N_c equal-length intervals, one sample is then drawn randomly from each interval. The sampled N_c points are used to query the coarse network and the rendering of the ray can thus be re-written as a weighted sum:

$$\tilde{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w(t_i)\mathbf{c}(t_i) \tag{3.9}$$

where the weights $w(t_i) = \tilde{T}(t_i)\alpha(t_i)$ are then normalised to form a Probability Density Function (PDF) of the distance t:

$$P(t) = \frac{\sum_{i=2}^{N_c-1} w(t_i)\mathbb{I}[t \in [t_i, t_{i+1}]]}{\sum_{i=2}^{N_c-1} w(t_i)} \tag{3.10}$$

By sampling under this PDF, points that are more important to the final rendering will be selected more often. N_f fine points are hence sampled from this distribution. Together with previously sampled coarse points, $N_c + N_f$ points in total are queried using the fine network to obtain their density and radiance values and finally compute the volume rendered pixel value $\tilde{\mathbf{C}}(\mathbf{r})$

3.3.3 Limitations

NeRF achieves state-of-the-art performance in image-to-3D reconstruction task with its ability to capture extremely fine details of any complex scene and to synthesise photo-realistic novel views. However, it comes with severe limitations as it is an optimisation-based approach that learns and memories the geometry of a single scene. Therefore, the model does not extract any prior knowledge about the reconstruction target and it completely loses the ability to generalise to unseen objects. To work with a different object or scene, the model is required to be re-trained completely. This significantly increases the time and computation requirement of NeRF and also limits its practicability, due to the difficulty in collecting a dense set of views of the target object.

Chapter 4

Methodology

The existing NeRF approach directly optimises an MLP with respect to a dense set of images to learn a continuous volumetric radiance field that agrees with the images. The method is therefore not aware of different objects and cannot generalise to unseen objects at inference time. To overcome this issue, we propose to incorporate a CNN encoder to take RGB image input and extract image features, which are then conditioned by an MLP decoder to produce a neural radiance field model similar to the original NeRF model.

Our experiments revealed that extracting and conditioning on a full image feature severely constraints the method's ability to generalise to uncommon objects. As shown later in section 5.2.3, network conditioned on global feature tends to produce blurry results and perform a retrieval-based reconstruction, where the network memorises the shape and appearance of objects in the training dataset and simply retrieves the most similar one during inference. This behaviour has also been noted by [16]. Inspired by the work of DeepSDF [19] and pixelNeRF [16], we instead up-sample and concatenate outputs from several convolutional layers in the encoder to form a 3D feature volume and condition our decoder on the corresponding local feature. We also perform a coordinate system transform similar to pixelNeRF on the query coordinate and direction to allow the decoder to work in input camera coordinate space. We further simplify the query by replacing the coordinate with a single depth value, removing the dependency on the training data.

Unlike pixelNeRF which works only with the local image feature, We include an additional decoder that takes the global feature, as it may contain helpful global information that can not be well represented in the local features. The final RGB and density values passed to the volume rendering step are the proper averages of the outputs of two decoders, taking their predicted densities into account.

The following sections first present our DualNeRF with a local feature and depth value query only, then show the global decoder component and how the results of two decoders are combined together with respect to the property of volume rendering.

4.1 Local Feature with Depth Query

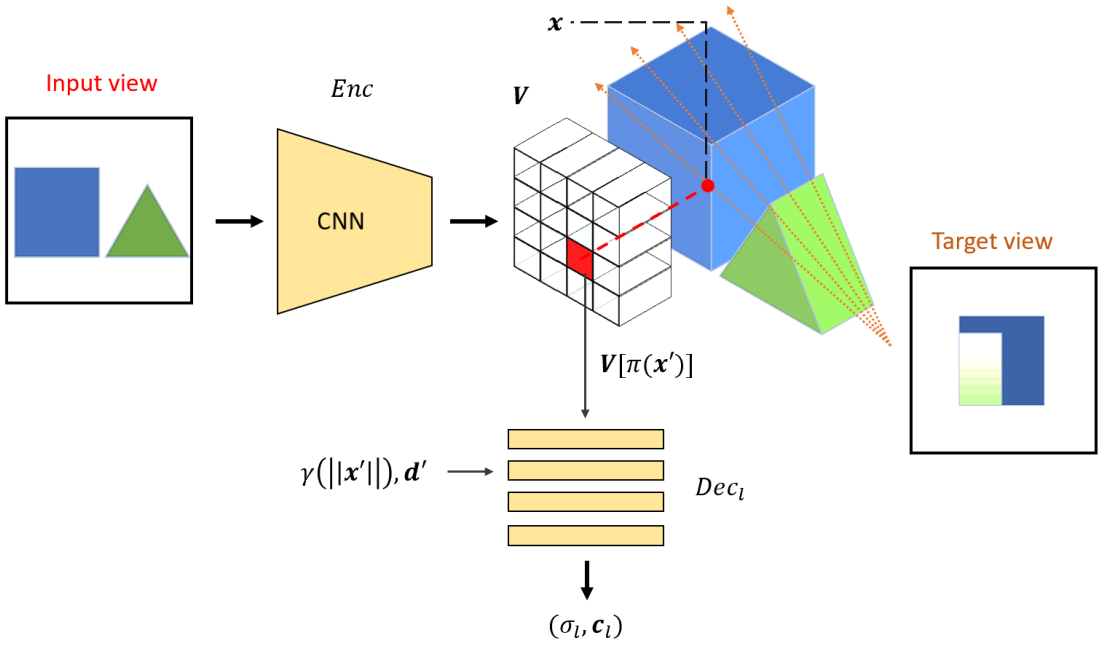

Figure 4.1: **DualNeRF architecture with only local decoder.** Given an input image $\mathbf{I}$, a CNN encoder is first used to extract a feature volume $\mathbf{V}$. A query point with world coordinate $\mathbf{x}$ and view direction $\mathbf{d}$ is sampled on the camera rays from a different target viewpoint. A MLP decoder Dec_l takes the corresponding local feature concatenated with Euclidean distance between camera origin and the point $\|\mathbf{x}'\|$ and the ray direction transformed into input view coordinate $\mathbf{d}'$, and returns the radiance $\mathbf{c}_l$ and density σ_l.

This section describes the main architecture of the proposed DualNeRF method, including the multi-view training scheme, convolutional encoder component and the local decoder with depth query.

Multi-view Training

To ensure the networks are learning the actual underlying geometry instead of memorising the image inputs, two images of the same object from different viewpoints are used for each training step. Two images, an input view $\mathbf{I}$ and a target view $\mathbf{I}_t$, are sampled at random during each training iteration. The input image $\mathbf{I}$ which has camera pose $\mathbf{P} = [\mathbf{R}, \mathbf{T}]$ is passed to the encoder Enc, then 3D points are sampled on camera rays from a different target viewpoint with pose $\mathbf{P}_t = [\mathbf{R}_t, \mathbf{T}_t]$. The sampled points are used to query the decoder conditioned on the image feature to recover the target view, which is then compared with ground truth target view $\mathbf{I}_t$ to obtain a mean squared error as the training loss.

To correctly account for the difference in the source and target viewpoints, we incorporate the coordinate system transform technique from pixelNeRF [16]. Instead of directly working with point features that consist of a 3D coordinates $\mathbf{x}$ and a view direction $\mathbf{d}$ in the world coordinate system, they are first transformed into input view coordinate system using $\mathbf{P}$:

$$\begin{bmatrix} \mathbf{x}' \\ 1 \end{bmatrix} = \mathbf{P}^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$
$$\mathbf{d}' = \mathbf{R}^{-1}\mathbf{d} \tag{4.1}$$

The use of $\mathbf{x}'$ allows the model to link the point being queried with the extracted image feature and therefore enables it to learn how to appropriately lift 2D images to 3D reconstructions. Besides, $\mathbf{d}'$ measures how closely related the two views are. If the input viewpoint is very close to the target viewpoint, the model should be able to rely more on the extracted local feature. Otherwise, the model is required to exploit its prior knowledge to predict sensible results [16].

Feature Volume Encoder

Inspired by DISN [20] and pixelNeRF [16], we use a convolutional encoder Enc that takes an input RGB image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ and extracts a feature volume $Enc(\mathbf{I}) = \mathbf{V} \in \mathbb{R}^{H \times W \times d}$, which consists of outputs from multiple convolutional layers in the encoder. Those outputs are up-scaled via bilinear interpolation to match the height and width of $\mathbf{I}$ and concatenated together to form a three-dimensional feature volume with depth d.

Local Feature and Depth Query

When querying for the radiance and density of a 3D point, the positional embedded point features are concatenated with the corresponding local feature. The local feature is identified by re-projecting the query point $\mathbf{x}'$ onto the image plane of the input image and finding the corresponding pixel coordinate $\pi(\mathbf{x}')$. To do this, the inverse of equation (3.2) is carried out:

$$\begin{aligned} \mathbf{x}' &= [x', y', z']^T \\ \pi(\mathbf{x}') &= [\frac{x' \times focal + W/2}{-z'}, \frac{-y' \times focal + H/2}{-z'}] \end{aligned} \tag{4.2}$$

The coordinate $\pi(\mathbf{x}')$ can therefore be used to locate and extract a 1D local image feature from the 3D feature volume $\mathbf{V}$. As the input view is different from the target view and the near/far boundaries of the sampling are usually rough estimations, it is possible that a sampled point locates outside of the perspective area of the input view. When this happens, we simply replace the local feature vector with a zero vector.

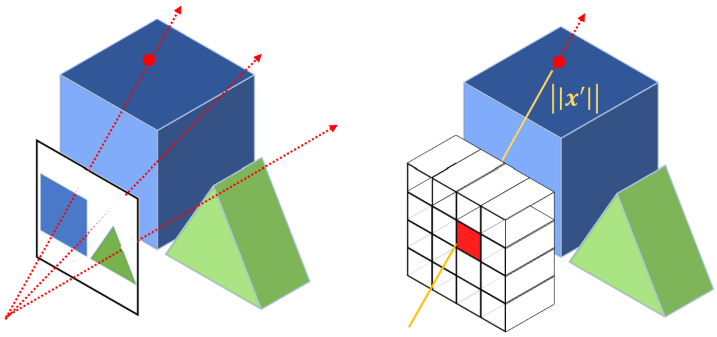

Figure 4.2: **Querying a local image feature with depth information.** The depth $\|\mathbf{x}'\|$ specifies how deep into the local feature the target point is. If the point is very close to the visible surface, the decoder can directly rely on the image feature. Otherwise, if the point is very deep and far away from the input view camera, the decoder should rely more on its prior knowledge instead.

However, unlike pixelNeRF which concatenates the local feature vector with positional embedding of the full transformed coordinate $\gamma_{L_\mathbf{x}}(\mathbf{x}')$, we instead argue that using only the embedding of a single depth value $\gamma_{L_\mathbf{x}}(\|\mathbf{x}'\|)$ is sufficient. This value represents the Euclidean distance between the input view camera and the 3D point, or can be viewed as how deep the queried point is into the 3D space as seen from the input view camera. As shown in figure 4.2, because the corresponding local feature is found by re-projecting point $\mathbf{x}'$ onto the image plane, the same feature is thus used to query all points along the camera ray, where depth is the only degree of freedom that distinguishes between the points.

Because we pass in only a depth value that contains very little dependency to the target prediction, the decoder is therefore forced to rely more on the concatenated image feature, leading to a better ability to generalise to unseen objects based on an image input.

The final decoding with the MLP decoder is thus represented as:

$$(\sigma_l, \mathbf{c}_l) = \begin{cases} Dec_l\Big(\gamma_{L_\mathbf{x}}(\|\mathbf{x}'\|), \gamma_{L_\mathbf{d}}(\mathbf{d}'), \mathbf{V}[\pi(\mathbf{x}')]\Big), & \text{if } \pi(\mathbf{x}') \in [0, width) \times [0, height) \\ Dec_l\Big(\gamma_{L_\mathbf{x}}(\|\mathbf{x}'\|), \gamma_{L_\mathbf{d}}(\mathbf{d}'), \mathbf{0}\Big), & \text{otherwise} \end{cases} \tag{4.3}$$

4.2 Global Feature Decoder

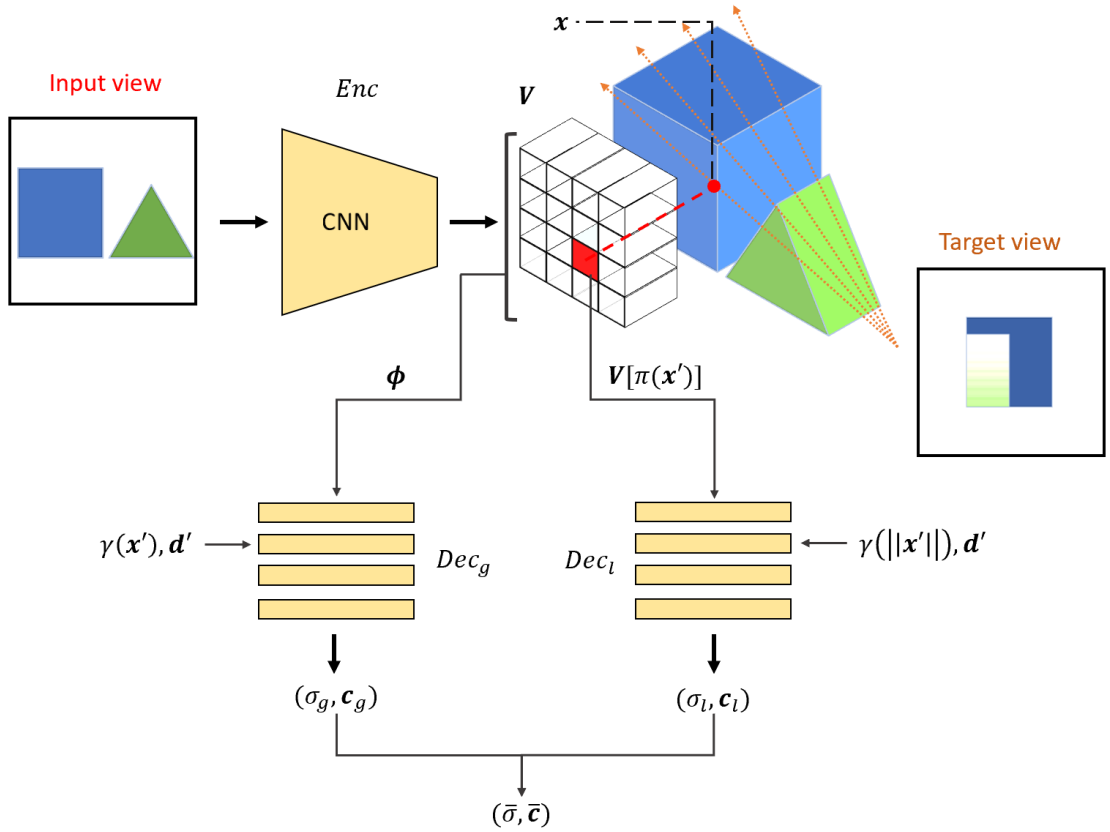

Figure 4.3: **DualNeRF with the both local and global decoders.** In additional to Dec_l that decodes the local image feature, a global decoder Dec_g is incorporated to take global output of Enc concatenated with $\mathbf{x}'$ and $\mathbf{d}'$. The final output is a proper average of the results of two decoders.

An additional global decoder network Dec_g with the same architecture as Dec_l is added to produce predictions based on the full input image. Instead of re-projecting points onto the image plane and taking the local feature vector, Dec_g takes the traditional encoder output $\phi \in \mathbb{R}^d$, which is a feature vector obtained as the output of the last layer in Enc after additional pooling and dense layers. With the global feature, the queried 3D point no longer has depth as the only variable, hence the embedding of full transformed coordinate $\gamma_{L_\mathbf{x}}(\mathbf{x}')$ is used as query input. The decoding with global decoder is represented as:

$$Dec_g\Big(\gamma_{L_\mathbf{x}}(\mathbf{x}'), \gamma_{L_\mathbf{d}}(\mathbf{d}'), \phi\Big) = (\sigma_g, \mathbf{c}_g) \tag{4.4}$$

The predictions from both decoders are combined with respect to the property of volume rendering. As described in section 3.2, a higher density means the camera ray is more likely to be stopped by this point, its radiance would therefore contribute more to the final rendered pixel value. We thus define the final radiance as the weighted average of predicted radiance values, where the weights are defined by their densities, while the final density is a simple average of the two.

$$\begin{aligned} \bar{\mathbf{c}} &= \frac{\sigma_l \mathbf{c}_l + \sigma_g \mathbf{c}_g}{\sigma_l + \sigma_g} \\ \bar{\sigma} &= \frac{\sigma_l + \sigma_g}{2} \end{aligned} \tag{4.5}$$

The combination of global and local decoders allows the model to capture useful global information, including overall lighting condition (brightness) and common shapes of objects. Therefore, the reconstruction appears more natural even on challenging inputs.

Chapter 5

Experiments

We demonstrate the effectiveness of our method by carrying out extensive experiments and evaluations on both synthetic and real-life dataset. We compare quantitatively and qualitatively with the baseline approaches on ShapeNet [10] synthetic car dataset. We also evaluate our method on the same ShapeNet car dataset rendered by SRN [11] to provide more comparable results. We then show the reconstructions of real-life cars from Stanford Car Dataset [17] to evaluate the model's ability to transfer to real domain.

5.1 Experiment Details

5.1.1 Metrics

Following the evaluation protocols of NeRF [13] and pixelNeRF [16], we evaluate all methods with standard image quality metrics PSNR and SSIM [35] as well as a deep learning based metric LPIPS [36].

The peak signal-to-noise (PSNR) is a widely used quality measurement metric that defines the power ratio between maximum possible signal and noise. Given the original image $\mathbf{I}$ and its reconstruction $\mathbf{I}_{pred}$, PSNR is defined as follows:

$$PSNR(\mathbf{I}, \mathbf{I}_{pred}) = 10log_{10}\frac{max(\mathbf{I})^2}{MSE(\mathbf{I}, \mathbf{I}_{pred})} \tag{5.1}$$

where $MSE(\mathbf{I}, \mathbf{I}_{pred})$ is the mean squared error between the two images. A higher value of PSNR represents a stronger signal, and therefore the recovery is better.

The structural similarity index measure (SSIM) is a similarity metric that measures perception-based similarity in structural information. Unlike PSNR, it does not measure the absolute differences between target and reconstructed images, but instead considers various local areas of the images and thus more closely resembles the human perception. A higher value of SSIM means a higher similarity between two images [35].

The Learned Perceptual Image Patch Similarity (LPIPS) is a recently proposed deep learning based metric that also measures perceptual similarity. It is obtained by calibrating the outputs of pre-trained networks, which learned to extract high level perceptual information that are useful for classification and recognition tasks. Therefore, this metric is able to represent human perception to a great extend. A lower value of LPIPS means a smaller difference between two images [36]. We

use the VGG network sub-variant of LPIPS to keep consistent with the prior works. The version of LPIPS used is 0.1.

5.1.2 Implementation Details

The network architecture used in our method is modified based on pixelNeRF implementation, with a different pre-trained encoder and a smaller decoder.

For encoder, we use the first three convolutional blocks of a pre-trained ResNet-50 classifier [37] to extract a feature volume. The outputs from four layers in those blocks are up-sampled with bilinear interpolation to match the height and width of the input image, and then concatenated together to form a $H \times W \times 896$ tensor. Each 896 feature vector is then passed through an MLP top layer to form a compressed $H \times W \times 512$ feature volume. When a global decoder is used, the encoder additionally applies an average pooling layer and a dense top layer on the outputs of the last layer, *conv3_block4_out*, to produce a 512 feature vector.

The exact layers used and their original output shapes are shown in table 5.1. The name of each layer comes from the pre-trained ResNet-50 model from TensorFlow [38].

Layer Index	Layer Name	Output Shape
4	*conv1_relu*	$H/2 \times W/2 \times 64$
6	*pool1_pool*	$H/4 \times W/4 \times 64$
38	*conv2_block3_out*	$H/4 \times W/4 \times 256$
80	*conv3_block4_out*	$H/8 \times W/8 \times 512$

Table 5.1: **ResNet-50 layers used to output the feature volume.** The layer indices are counted from 0.

For local decoder, we set positional embedding hyperparameters to $L_{\mathbf{x}} = 20$ and $L_{\mathbf{d}} = 0$ to produce a point feature similar to the original pixelNeRF implementation. I.e., no positional embedding is performed on the view direction. The point feature $[\gamma_{20}(\|\mathbf{x}'\|)\ \mathbf{d}']$ therefore has length 41. For global decoder, we set $L_{\mathbf{x}} = 6$ and $L_{\mathbf{d}} = 0$ and the point feature $[\gamma_6(\mathbf{x}')\ \mathbf{d}']$ has length 42.

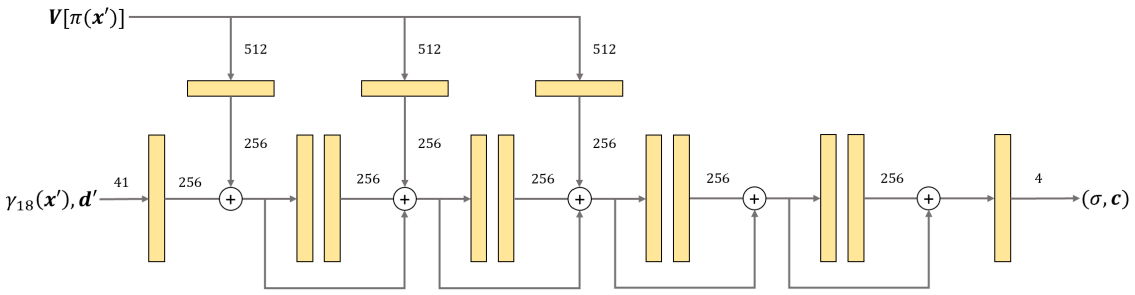

Figure 5.1: **Local decoder architecture.** Each yellow block represents an independent dense layer. The global decoder has the same architecture with different inputs.

The architecture of the MLP decoder is shown in figure 5.1. Due to memory constraint, we use a simplified version of the ResNet-like MLP decoder proposed by pixelNeRF. The network has four residual blocks, each containing two dense layers of width 256. The extracted local feature vector $\mathbf{V}[\pi(\mathbf{x}')]$ (or the global feature vector ϕ for Dec_g) is passed through three independent dense layers

to be compressed into a size of 256, and is introduced as an extra skip component to the first two residual blocks.

Similar to NeRF, we train separate coarse and fine networks for each of the local and global decoders. The sampling numbers for both coarse and fine networks are set to 16 ($N_c = N_f = 16$).

For each training batch, 4 distinct objects are sampled at random and 2 images are drawn as input view and target view for each object. Given each input view, the model predicts 128 randomly selected pixel values in the target view and the mean square error between predictions and the ground truth for all pixels and objects is used as training loss. Adam is used to optimise the network with learning rate 5e$-$5 for 200000 iterations. The first 100 iterations are trained with pre-cropped images. I.e., pixels are selected from a $H/2 \times W/2$ sub-image located at the centre of original image. The purpose of pre-cropped training is to prevent bad initial training batch that consists of too many pixels of empty space, as this would kill neurons in the networks and cause them to always predict 0 density and cannot be trained further. However, it is observed that using too many pre-cropped training causes difficulty in convergence of global decoder, therefore a number of 100 is selected.

For both of the following experiments, the training of our method takes roughly 3 days on a GeForce GTX 1080 Ti GPU.

5.2 ShapeNet Benchmarks

5.2.1 Data

We first present the experiments carried out on a subset of the ShapeNet "car" category. We randomly sample 740 car objects and reserve 240 objects as the test dataset. The rest 500 objects are randomly split into 490 training objects and 10 validation objects. This uneven split scheme is mainly because of the complexity in training the networks, thus hyperparameter tuning becomes difficult to carry out and is not included as the focus of our experiments. At test time, we fix the input as one informative view for each test object and reconstruct the views from all 30 viewpoints.

For each object, blender [39] is used to render 30 views with resolution 64×64 from random viewpoints. The azimuth and elevation are randomly sampled between [0, 360] and [15, 60] respectively. The distance between camera and object centre is fixed as 1, and the t_{near}, t_{far} boundaries for volume render are therefore set to 0, 1.5 for all models trained on this dataset. Two blender "Sun" light sources are placed at opposite sides of the object to provide sufficient lighting. The rendering is done with "blender renderer" engine type.

There are several reasons why we choose not to use the ShapeNet car dataset provided by SRN [11]: 1) SRN training dataset contains 50 views rendered with an elevation between [-90, 90] for every object in the car category. While this provides a high coverage on all possible viewpoints, there exists less informative views that would cause training to be inefficient. For example, views that capture the bottom of the object. Therefore, We decide to simplify the problem and limit the elevation to a smaller range. 2) SRN dataset renders the car objects from a relatively far distance with a higher resolution. When the images are down-sized to 64×64, the objects appear sightly blurry and contain less details. Therefore, we first render our own ShapeNet car dataset and carry out training and evaluation on it.

5.2.2 Baselines

We compare with baselines include NeRF [13] and pixelNeRF↓, a simplified implementation of pixelNeRF baseline following the original paper [16], both quantitatively and qualitatively to demonstrate the performance of our method under a constrained experiment setting, namely with less training data, sampling rate and network capacity.

The NeRF baseline is implemented based on the original setting for the synthetic LEGO dataset published in the code repository of [13]. The positional embedding hyperparameters are set to $L_{\mathbf{x}} = 10$ and $L_{\mathbf{d}} = 6$, the sampling numbers for coarse and fine networks are both set to 64 ($N_c = N_f = 64$). Each training batch contains 1024 pixels randomly sampled from a training image. Adam is used to optimise the network with learning rate 5e−4. The network is trained with 100000 batch iterations.

The encoder and decoder networks in pixelNeRF↓ have the same architecture as in our method, with the only difference being the point feature fed into the decoder. This modification on network architecture is due to the difference in software framework and hardware constraints. Compared to the original implementation proposed in pixelNeRF paper, the pre-trained ResNet-34 [37] encoder is replaced with a ResNet-50 network in pixelNeRF↓, and the decoder MLP is simplified to have only half the width (256 instead of 512 proposed in the original paper) and one less residual block (4 instead of 5). The positional embedding hyperparameters are set according to the original paper, where $L_{\mathbf{x}} = 6$ and $L_{\mathbf{d}} = 0$. The sampling numbers for both coarse and fine networks are reduced to 16 ($N_c = N_f = 16$). The training is the same as our method, where 4 distinct objects are sampled for each training batch, 2 images from each object are selected as input/target view and 128 pixel values are predicted for each object. Adam is used to optimise the network with learning rate 5e−5 for 200000 iterations. The first 30000 iterations are trained with pre-cropped images to speed up convergence.

5.2.3 Results

Quantitative Evaluation

We first present the quantitative metric evaluations on the reserved ShapeNet car test dataset. As shown in table 5.2, our method demonstrates a competing performance to pixelNeRF↓ baseline.

	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	14.903	0.590	0.366
pixelNeRF↓	17.652	**0.671**	**0.305**
Ours	**17.679**	0.666	0.321
Ours (local only)	17.624	0.664	0.316
Ours (global only)	17.234	0.655	0.319

Table 5.2: **Evaluation on ShapeNet car dataset.** We report PSNR, SSIM (higher is better) and LPIPS (lower is better). Ours (local only) is the model with only a local decoder, while ours (global only) only contains a global decoder that conditions on full image feature.

Ablation Study In table 5.2, we show the performance of global and local decoders respectively. Although using only a global decoder that conditions on full image feature leads to retrieval-based reconstruction and causes worse performance, combining it with a local decoder actually improves the PSNR and SSIM.

Qualitative Evaluation

We then present the reconstructions of several unseen objects in the test dataset. As shown in figure 5.2 and 5.3, both our method and pixelNeRF↓ are able to generalise to unseen car objects and can extract correct geometry and appearance information from only a single RGB image input. However, our method generates a more natural-looking shape, as can be seen most obviously from the rendering of front-looking cars in figure 5.2: the front of the car generated by pixelNeRF↓ contains unnatural curves and incorrect shadows, while the ones generated by our method match better with the actual geometry. Moreover, our method captures global information such as lighting conditions more accurately, as can be seen from the reconstructions of the third object.

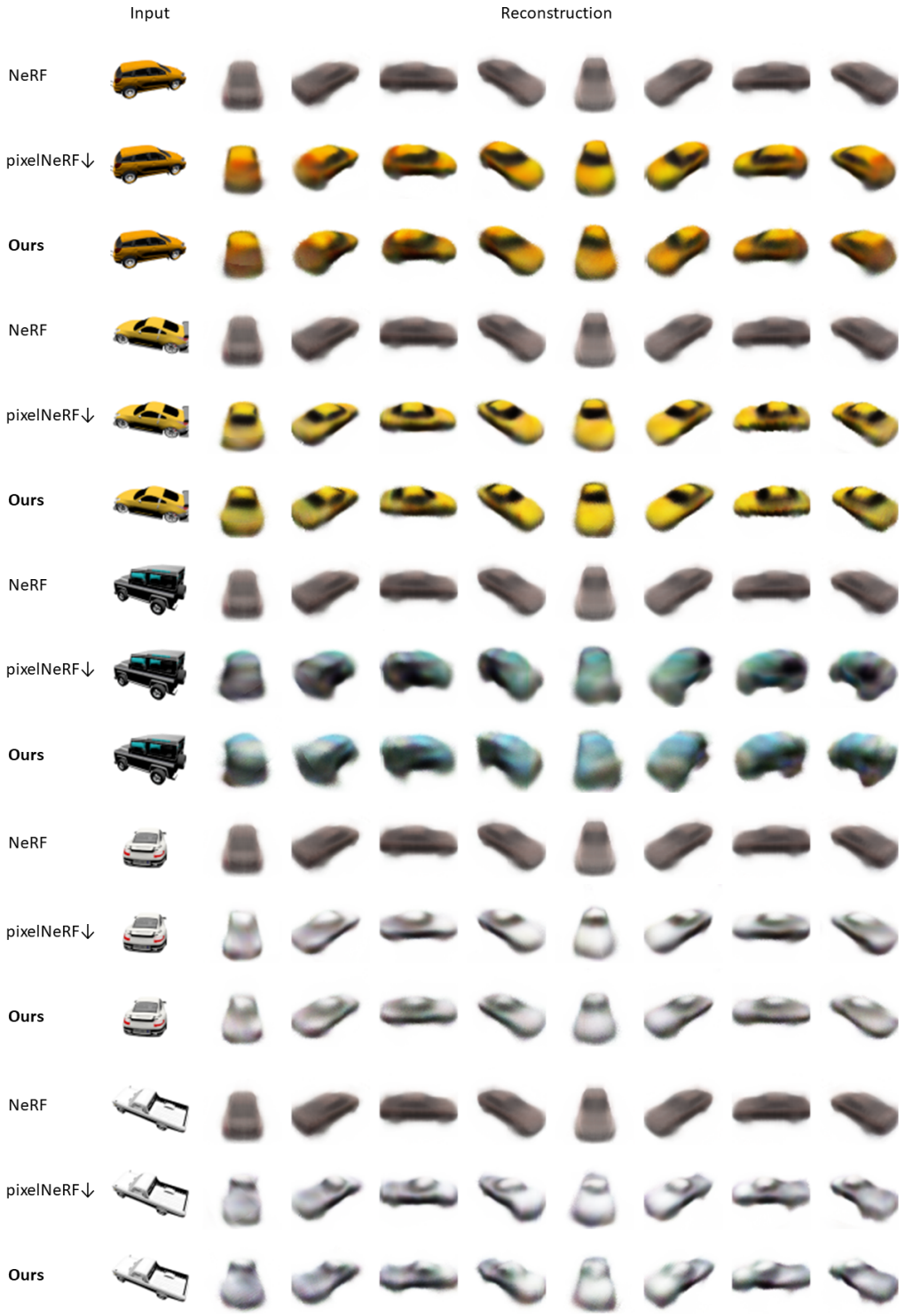

Figure 5.2: **Reconstruction of unseen ShapeNet cars.** We provide a single RGB image and its camera pose as input to the models and reconstruct the views from novel view points. The NeRF baseline is not designed to take image input at inference time, hence it only takes the camera pose input.

Figure 5.3: **Reconstruction of unseen ShapeNet cars.** Similarly, we provide an input image and render the object from a specified target viewpoint. The ground truth views are shown in the right-most column.

Ablation Study In figure 5.3 we present the reconstructions of our method with only local or global decoder. While the local decoder only model shows a similar performance to the pixelNeRF baseline, the global decoder only model tends to perform retrieval-based reconstruction and synthesise blurry or incorrect views on some objects. The full method combining both decoders mitigates those issues and produces the best reconstructions overall.

5.3 SRN ShapeNet Car Dataset

To provide comparable evaluations to the prior works, we re-train our method on the ShapeNet car dataset rendered by SRN [11] and evaluate both quantitatively and qualitatively on the corresponding test dataset.

5.3.1 Data

The SRN training dataset contains 2458 distinct car objects, each with 50 renderings of size 128×128 from random viewpoints in the full sphere around the object centre. The SRN test dataset contains 704 unseen objects, each with 251 renderings of same size. The viewpoints of these renders are arranged in an Archimedean spiral with elevation between $[0, 90]$. For the near/far boundaries, we use the same setting in pixelNeRF and set them to 0.8 and 1.8 respectively.

Due to hardware limitations, we only use 500 cars sampled from the SRN training set for training, while the testing is done on the full test dataset to remain comparable with prior works.

5.3.2 Baseline

Instead of using modified implementation pixelNeRF↓, we use the original pixelNeRF implementation and pre-trained model weights provided in the code repository [16]. The weights were originally trained with multiple input views and hence would perform slightly worse when reconstructing from a single view, but the difference is very small.

5.3.3 Results

Quantitative Evaluation

	PSNR ↑	SSIM ↑	LPIPS ↓
pixelNeRF	**27.58**	**0.942**	**0.095**
Ours	20.03	0.790	0.250

Table 5.3: **Evaluation on SRN ShapeNet car dataset.** The evaluations for pixelNeRF are obtained from the original paper [16].

Qualitative Evaluation

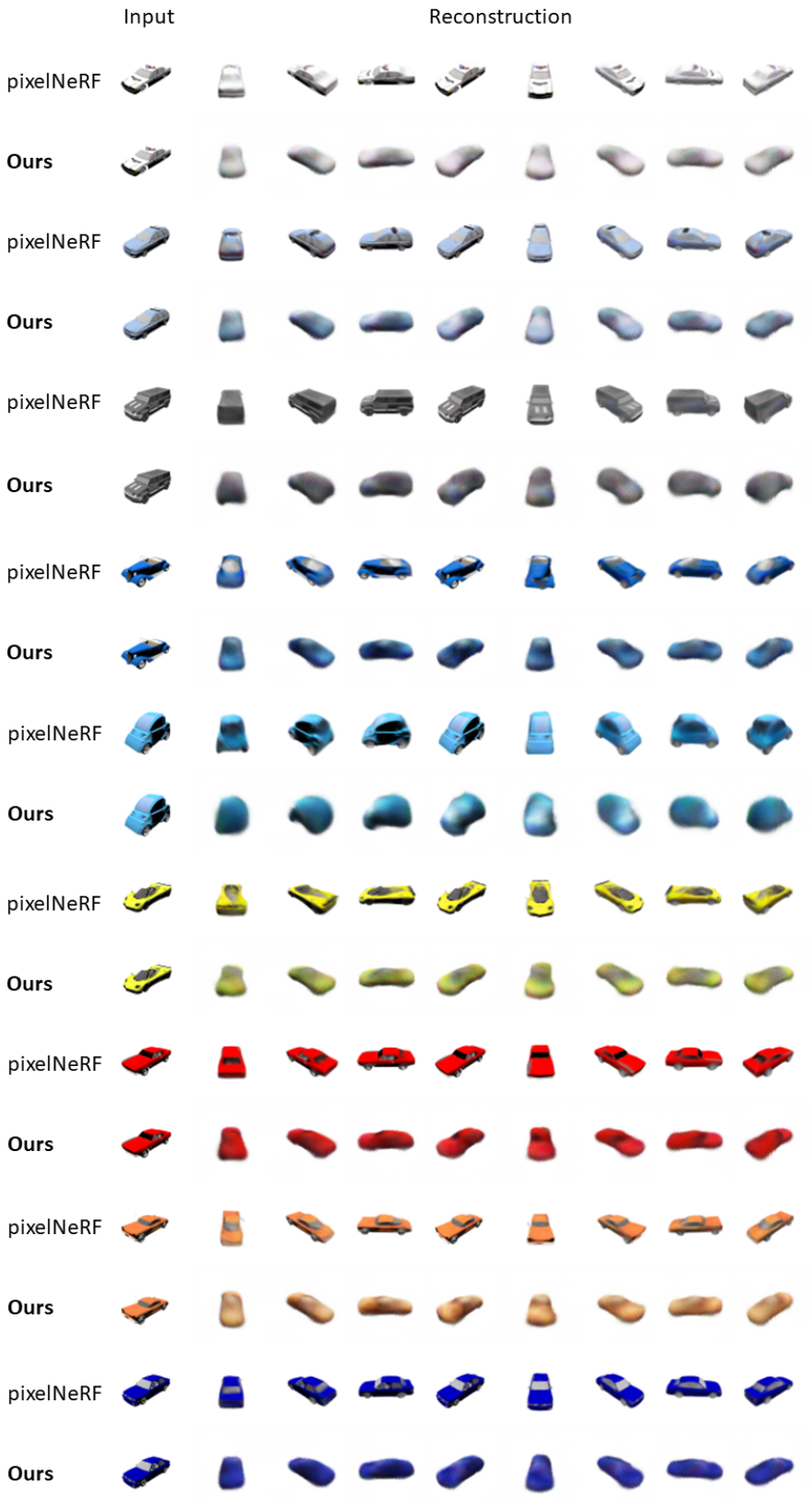

Figure 5.4: **Reconstruction of cars in SRN test dataset.** The provided pixelNeRF model works with images of size 128×128, while our method works with images of size 64×64 due to hardware limitations.

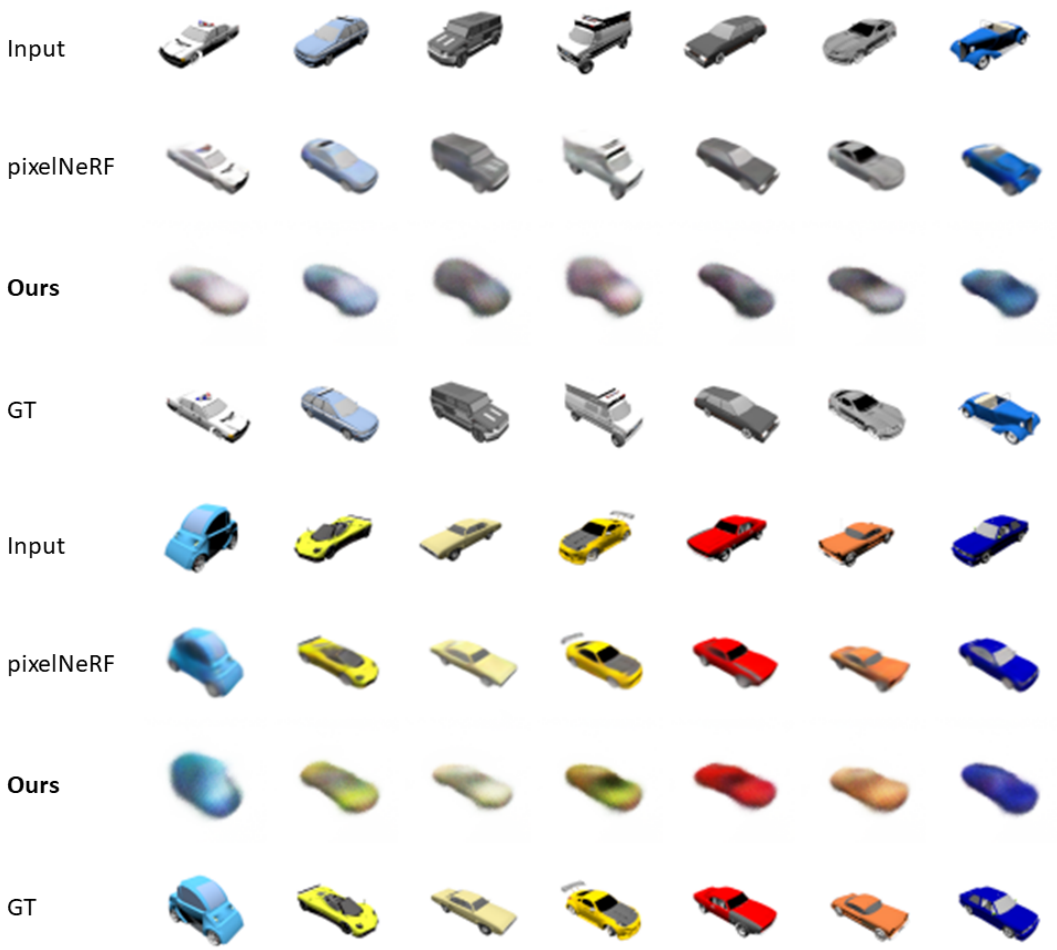

Figure 5.5: **Reconstruction of cars in SRN test dataset.**

Due to the limitations in the training dataset, model capacity and continuous volumetric radiance field sampling rate, our model renders more blurry reconstruction compared to pixelNeRF. However, our model is still able to capture the coarse geometry and appearance of a large number of unseen cars with varying appearance. With larger network capacity and training dataset, we expect our model to achieve a similar level of performance to pixelNeRF.

5.4 Stanford Car Dataset

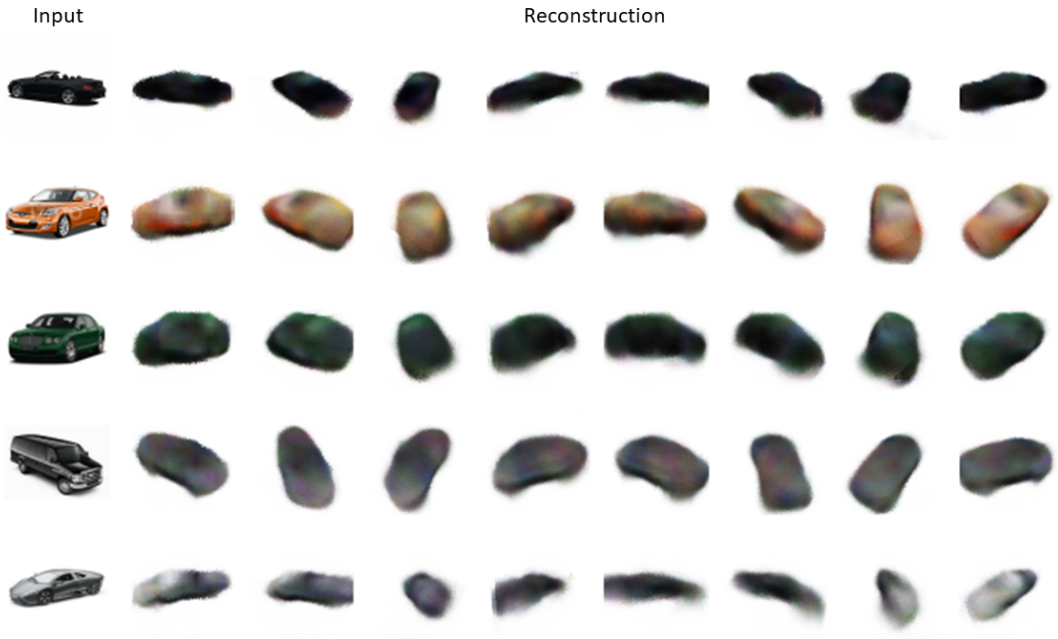

Figure 5.6: **Reconstruction of real-life cars.** We apply our model trained in section 5.2 to the real car images from Stanford Car Dataset [17]. The images are cropped and re-sized to 64×64, and we manually specify a rough estimation of the camera intrinsics and extrinsics.

We present the qualitative results of our model trained in section 5.2 with real-life car images from the Stanford Car Dataset [17]. Several car images without background are selected and passed to the model and novel views are synthesised from viewpoints rotating about the vertical axis. Due to the lack of ground truth camera parameters in the dataset, we manually estimate the intrinsics and extrinsics for each input image.

Results show that our model is able to capture the coarse shape and appearance of real-life cars. However, the reduction in performance compared to synthetic car reconstruction is significant. This is mainly because of: 1) The lack of accurate camera parameters. 2) Insufficient amount of training data. Our model is trained with only 490 different synthetic car objects due to hardware and time constraints, hence the model is limited in its generalisation ability. 3) The domain gap between synthetic and real-life car objects. For example, the real-life car images contain complex lighting and reflection effects, which are absent in synthetic car images.

Chapter 6

Conclusions

6.1 Achievements

We have presented DualNeRF, a deep learning framework that utilises implicit neural representation to achieve the reconstruction of 3D geometry and appearance of unseen objects from a single RGB image. The framework builds on the existing continuous volumetric radiance field method, NeRF, and extends it to allow the extraction of categorical prior information by training with only images and camera parameters, without any ground truth 3D shape labels. Our method attempts to prevent the issue of overfitting to the canonical coordinate system by limiting the network query information to be a depth value only, and also combines global and local image feature decoders in an error-correcting manner. Extensive evaluations demonstrate that our method is able to produce reliable reconstruction on synthetic cars. Although the performance on real-life cars is less satisfactory, we expect it to be mitigated by providing sufficient training data.

6.2 Future Work

The current implementation of continuous volumetric radiance field in this framework does not strictly obey the underlying Physical principle, that is the density of a 3D point should be independent of view direction. As shown in figure 6.1, NeRF [13] enforces this property by separating the query inputs and passing in only the coordinate at the beginning of the network. After the network predicts the density as an intermediate output, the view direction is then passed in to obtain the radiance prediction. However, in our approach, to account for the difference in the input and target viewpoints, the transformed view direction is passed to the decoder network at the beginning to predict both density and radiance values. The density is therefore not completely independent of the view direction input. Overcoming this issue may improve the method's ability to work with real objects and scenes with complex lighting conditions.

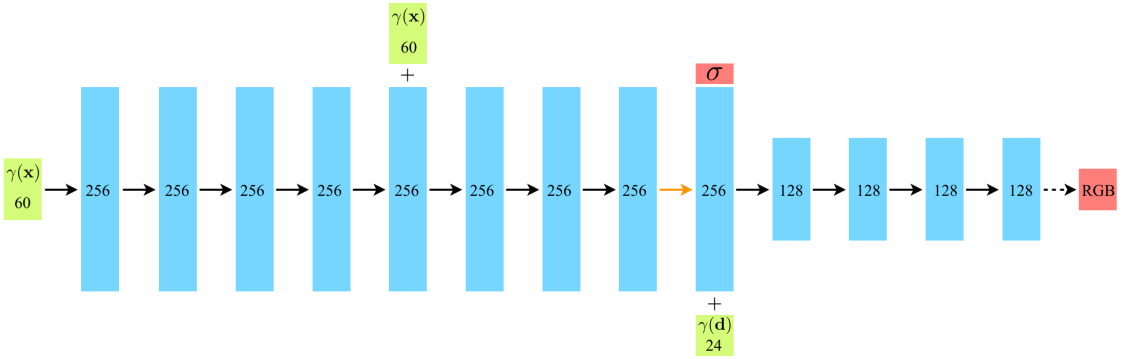

Figure 6.1: **Architecture of NeRF from the original paper [13].** The NeRF network takes only the coordinate input $\gamma(\mathbf{x})$ at the beginning to predict a density value σ. The view direction $\gamma(\mathbf{d})$ is passed to the network after density has been predicted, so that density is independent of view direction.

Another potential task is to improve model inference efficiency, which is one of the main drawbacks of all implicit representations. Due to their implicit nature, an accurate rendering usually requires a large number of queries at different spatial locations. Hence, converting to high quality explicit geometry or rendering is computationally expensive and is far from being done in real-time. The existing solution NSVF [28] combines the implicit continuous volumetric radiance field with explicit voxel representation to allow more effective queries, but extending it to image-based reconstruction of unseen objects is still challenging.

One exciting future direction is to extend the model even further to predict motion, i.e., a 4D video sequence. While modelling time-varying scenes via a neural network has been achieved on 2D images [40], extending it to account for the underlying 3D geometry would be much more difficult, but not impossible with sophisticated implicit neural representations.

Bibliography

[1] *Y. Furukawa, C. Hernández, Multi-view stereo: A tutorial, Foundations and Trends® in Computer Graphics and Vision 9 (1-2) (2015) 1–148.* doi:10.1561/0600000052. *URL* http://dx.doi.org/10.1561/0600000052

[2] *C. B. Choy, D. Xu, J. Gwak, K. Chen, S. Savarese, 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction (2016).* arXiv:1604.00449.

[3] *Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes (2015).* arXiv:1406.5670.

[4] *E. Dupont, M. A. Bautista, A. Colburn, A. Sankar, C. Guestrin, J. Susskind, Q. Shan, Equivariant neural rendering (2020).* arXiv:2006.07630.

[5] *H. Fan, H. Su, L. Guibas, A point set generation network for 3d object reconstruction from a single image (2016).* arXiv:1612.00603.

[6] *P. Achlioptas, O. Diamanti, I. Mitliagkas, L. Guibas, Learning representations and generative models for 3d point clouds (2018).* arXiv:1707.02392.

[7] *N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, Y.-G. Jiang, Pixel2mesh: Generating 3d mesh models from single rgb images (2018).* arXiv:1804.01654.

[8] *T. Bagautdinov, C. Wu, J. Saragih, P. Fua, Y. Sheikh, Modeling facial geometry using compositional vaes, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 3877–3886.* doi:10.1109/CVPR.2018.00408.

[9] *H. Kato, Y. Ushiku, T. Harada, Neural 3d mesh renderer (2017).* arXiv:1711.07566.

[10] *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, F. Yu, ShapeNet: An Information-Rich 3D Model Repository, Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015).*

[11] *V. Sitzmann, M. Zollhöfer, G. Wetzstein, Scene representation networks: Continuous 3d-structure-aware neural scene representations (2020).* arXiv:1906.01618.

[12] *C.-H. Lin, C. Wang, S. Lucey, Sdf-srn: Learning signed distance 3d object reconstruction from static images (2020).* arXiv:2010.10505.

[13] *B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis (2020).* arXiv:2003.08934.

[14] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, R. Basri, Y. Lipman, Multiview neural surface reconstruction with implicit lighting and material (2020). arXiv: 2003. 09852.

[15] W. Lorensen, H. Cline, Marching cubes: A high resolution 3d surface construction algorithm, ACM SIGGRAPH Computer Graphics 21 (1987) 163–. doi: 10. 1145/ 37401. 37422.

[16] A. Yu, V. Ye, M. Tancik, A. Kanazawa, pixelnerf: Neural radiance fields from one or few images (2020). arXiv: 2012. 02190.

[17] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3d object representations for fine-grained categorization, in: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013.

[18] Y. Furukawa, B. Curless, S. M. Seitz, R. Szeliski, Towards internet-scale multi-view stereo, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 1434–1441. doi: 10. 1109/ CVPR. 2010. 5539802.

[19] J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 165–174.

[20] Q. Xu, W. Wang, D. Ceylan, R. Mech, U. Neumann, Disn: Deep implicit surface network for high-quality single-view 3d reconstruction, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2019, pp. 492–502.

[21] M. Runz, K. Li, M. Tang, L. Ma, C. Kong, T. Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, R. Newcombe, Frodo: From detections to 3d objects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[22] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, T. Funkhouser, Learning shape templates with structured implicit functions (2019). arXiv: 1904. 06447.

[23] K. Genova, F. Cole, A. Sud, A. Sarna, T. Funkhouser, Local deep implicit functions for 3d shape (2020). arXiv: 1912. 06126.

[24] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, M. Zollhöfer, State of the art on neural rendering (2020). arXiv: 2004. 03805.

[25] G. Yang, Y. Cui, S. Belongie, B. Hariharan, Learning single-view 3d reconstruction with limited pose supervision, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision – ECCV 2018, Springer International Publishing, Cham, 2018, pp. 90–105.

[26] S. Wu, C. Rupprecht, A. Vedaldi, Unsupervised learning of probably symmetric deformable 3d objects from images in the wild, in: CVPR, 2020.

[27] J. Kajiya, B. von herzen, Ray tracing volume densities, ACM SIGGRAPH Computer Graphics 18 (1984) 165–174. doi: 10. 1145/ 964965. 808594.

[28] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, C. Theobalt, Neural sparse voxel fields (2021). arXiv: 2007. 11571.

[29] A. Trevithick, B. Yang, Grf: Learning a general radiance field for 3d scene representation and rendering (2020). arXiv:2010.04595.

[30] M. Levoy, P. Hanrahan, Light field rendering, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, Association for Computing Machinery, New York, NY, USA, 1996, p. 31–42. doi:10.1145/237170.237199.
URL https://doi.org/10.1145/237170.237199

[31] E. H. Adelson, J. R. Bergen, The plenoptic function and the elements of early vision, in: Computational Models of Visual Processing, MIT Press, 1991, pp. 3–20.

[32] P. Hedman, T. Ritschel, G. Drettakis, G. Brostow, Scalable Inside-Out Image-Based Rendering 35 (6) (2016) 231:1–231:11.

[33] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, G. Brostow, Deep blending for free-viewpoint image-based rendering 37 (6) (2018) 257:1–257:15.

[34] S. J. D. Prince, Computer Vision: Models, Learning, and Inference, 1st Edition, Cambridge University Press, USA, 2012.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity, Trans. Img. Proc. 13 (4) (2004) 600–612. doi:10.1109/TIP.2003.819861.
URL https://doi.org/10.1109/TIP.2003.819861

[36] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: CVPR, 2018.

[37] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition (2015). arXiv:1512.03385.

[38] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL https://www.tensorflow.org/

[39] B. O. Community, Blender - a 3D modelling and rendering package, Blender Foundation, Stichting Blender Foundation, Amsterdam (2020).
URL http://www.blender.org

[40] M. Bemana, K. Myszkowski, H.-P. Seidel, T. Ritschel, X-fields: Implicit neural view-, light- and time-image interpolation (2020). arXiv:2010.00450.

Appendix A

Project Plan

Final Year Project Plan

Degree of Program: MEng

Supervisor: Lourdes De Agapito Vicente

Project Working Title: Unsupervised Generalisable 3D Reconstruction from Images with Neural Rendering

1. Purpose and Background

Accurately capturing the shape and texture of an object and reconstructing it as a 3D model based on images is a challenging task in computer vision. Deep learning based methods have achieved promising results in this area, and the most recent solution involve neural rendering, which is a data-driven approach that renders a 2D image based on potentially incomplete scene representation [1]. Relying on a differentiable rendering equation or a continuous 3D shape representation, loss on the rendering can be calculated and help neural network to learn the 3D shape, appearance or lighting of the target object.

However, the lack of realistic data with ground truth 3D information severely limits the performance and reliability of neural rendering, this is particularly evident for supervised and semi-supervised deep learning methods. As a consequence, synthetic dataset like ShapeNet [2] is widely used in the training of neural rendering models that require ground truth 3D data. Nonetheless, these models tend to be unreliable when dealing with real data [3, 4, 5]. On the other hand, for the models that do not require ground truth 3D data, their training procedure demands multiple views of a single object, which generally results in the loss of ability to inference on unseen objects or scenes [6, 7]. To address those issues, I propose a framework that uses neural rendering technique along with probabilistic neural network to learn the 3D shape and texture of unknown objects using very few images of the target object.

2. Aims and Objectives

This project focuses mainly on 3D reconstruction and neural network, and I have the following aims and objectives:

Aims:

1. To learn the fundamental knowledge and techniques behind 3D reconstruction of objects via neural network, especially 3D reconstruction using images with and without ground truth 3D shape as supervision.

2. To learn the efficient and correct ways of carrying out research. To attempt to solve one of the main challenges in this field, namely the issue with unsupervised model not being able to generalize over different objects, and potentially publish the work.

Objectives:

1. Reviews recent papers related to 3D reconstruction, neural rendering and neural network. Identify the state-of-the-art approaches, set up baseline models, explore their limitations and potential future directions.
2. Collect suitable dataset for the experiment, (likely to be synthetic dataset such as ShapeNet), implement tools to render and reprocess them into a set of images of different objects across categories, along with the corresponding camera pose.
3. Develop an initial convolutional neural network model that takes an image and a 3D point coordinate as input and outputs the radiance and density of that point. The photometric loss can be calculated by rendering the neural representation using volume rendering techniques [6] and comparing with the input image.
4. Apply different priors to improve the reconstruction accuracy, for example, symmetry assumption [8] and transformation equivariant [3].
5. Try to implement different techniques including positional encoding [6, 9], local feature [5], probabilistic neural network and confidence map [8]. Compare the performance with respect to dataset collected and identify the useful ones.
6. Explore the possibility of applying fine-tuning to further improve the generalisation ability and reliability. For example, initial model can be trained using a mixture of views of different objects across categories, then fine-tuned with category-specific data to improve the reconstruction accuracy on that type of objects.

3. Deliverables

The deliverables of this project are:

1. A comprehensive literature review on 3D reconstruction and neural rendering, which identifies the state-of-the-art methods and their current limitations
2. Programs or functions for generating views from synthetic dataset and providing other useful metadata, such as camera pose, field of view, near/far boundaries etc.
3. A new deep learning model that relies on images without ground truth 3D shape supervision, yet achieves inference on unseen objects of same or potentially different categories.
4. A set of experiment results comparing my method and existing methods.

4. Work Plan

The plan and sub-tasks of this project are:

1. By the end of October 2020: carry out literature review, collect dataset and setup baseline models.

2. By the end of November 2020: fix dataset for experiment, implement a most basic version of proposed architecture, run experiments on it as well as baseline models to obtain baseline results.
3. By the end of January 2021: explore different techniques that can improve model performance, including positional encoding, probabilistic neural network, class-specific fine-tuning and GANs.
4. By the end of March 2021: tidy up experiment results and produce documents, code and report.
5. After March 2021: produce conference paper and submit.

References

[1] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, M. Zollhöfer, State of the art on neural rendering (2020). arXiv:2004.03805.

[2] 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, F. Yu, ShapeNet: An Information-Rich 3D Model Repository, Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015).

[3] E. Dupont, M. A. Bautista, A. Colburn, A. Sankar, C. Guestrin, J. Susskind, Q. Shan, Equivariant neural rendering (2020). arXiv:2006.07630.

[4] J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 165–174.

[5] Q. Xu, W. Wang, D. Ceylan, R. Mech, U. Neumann, Disn: Deep implicit surface network for high-quality single-view 3d reconstruction, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2019, pp. 492–502.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis (2020). arXiv:2003.08934.

[7] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, R. Basri, Y. Lipman, Multiview neural surface reconstruction with implicit lighting and material (2020). arXiv:2003.09852.

[8] S. Wu, C. Rupprecht, A. Vedaldi, Unsupervised learning of probably symmetric deformable 3d objects from images in the wild, in: CVPR, 2020.

[9] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains (2020). arXiv:2006.10739.

Appendix B

Code Listing

For a complete reference of the model architecture and training code, the reader is strongly advised to visit the GitHub repository `https://github.com/ChikamaYan/nerf-to-equivariant`*, which contains all the source code and documentation to get started.*

Appendix C

Video

Please refer to the attached additional files for a video demo of the reconstruction on unseen ShapeNet cars.

All videos are generated using the full local and global decoder model trained on SRN ShapeNet car dataset. The model takes in a single input image and reconstructs the views from 40 novel viewpoints orbiting around the object to produce the video.