Towards Photo-Realistic 3D Reconstruction from Casual Scanning

Jeong Joon Park

A Dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Steve Seitz, Chair

Ira Kemelmacher-Shlizerman

Qi Shan

Richard Szeliski

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Towards Photo-Realistic 3D Reconstruction from Casual Scanning

Jeong Joon Park

Chair of the Supervisory Committee:
Prof. Steve Seitz
Computer Science and Engineering

In this thesis, I address the problem of obtaining photo-realistic 3D models of small-scale indoor scenes from a stream of images captured with a hand-held camera.

Recovering 3D structure of real-world scenes has been an important topic of research in computer vision, due to its wide applicability in virtual tourism, augmented reality, autonomous-driving or robotics. While numerous reconstruction methods have been proposed, they typically present trade-offs between practicality of capture and the realism of the reconstructed model. I introduce novel 3D reconstruction techniques that effectively navigate the trade-off curve, in order to produce photo-realistic models from user-friendly capture setups. Finally, I suggest new directions for learning generalizable scene priors to enable capture from partial inputs.

Creating a photo-realistic digital replica of a physical scene involves careful modeling of geometry, surface materials, and scene lighting, all of which I address in this thesis. At the same time, a reconstruction system should be easy to use for casual users to truly unlock 3D-related applications. This thesis suggests three criteria required for a casual reconstruction system that could greatly reduce the time and resources during scanning: i) the input method should be from a hand-held consumer-grade camera, ii) the system should reconstruct full appearance from a handful of input views of a scene as opposed to a dense view-sampling, and iii) it should automatically complete unobserved parts of a scene. The thesis proposes

novel techniques to tackle each of these criteria.

I first describe a technique to reconstruct the appearance of shiny objects, leveraging the infrared laser system of an RGB-D sensor as a calibrated point light source to recover surface reflectance. This method takes video as an input from a hand-held camera and the scene lighting captured with a 360° camera to generate a realistic replication of a scene, featuring high-resolution texture and specular highlight modeling. The output model allows virtually rendering the captured scene from any viewing direction.

Next, I discuss joint reconstruction of photo-realistic scene appearance and environment lighting of a target scene using a hand-held sensor. I achieve this through a joint optimization of a segmentation neural network, and a material-specific lighting model to reconstruct the input images, and adopt a neural network-enhanced rendering technique that achieve exceptional realism. The combination of physics and machine learning achieves both photo-realism and the ability to extrapolate to new views, reducing the range of required views by the users.

While the first two approaches allow realistic reconstruction from casual scanning, they can only model surfaces that are captured during scanning, i.e., they do not complete missing surfaces. Completing unobserved regions typically calls for machine learning algorithms to extract and apply scene/object priors from a large database. Traditionally, the lack of efficient 3D representations has limited the development of deep learning approaches in 3D. To facilitate machine learning in 3D, I devise my DeepSDF [182] approach that describes 3D surface as a decision boundary of a neural network, which is highly efficient in memory and at the same time can model continuous surfaces. The new representation, along with a newly proposed learning algorithm, allows reconstructing a full, plausible shape from a partial and noisy object scan. I show through experiments that the new representation is highly effective in learning geometric priors from a dataset of objects.

Finally, I extend the DeepSDF representation to model multi-object scenes. Specifically, I

introduce a new method of training a generative model of unaligned objects via an adversarial training in the feature space. I show that reconstructing a multi-object scene from a noisy, partial scan amounts to simply optimizing the randomly initialized latent vectors of the generative model to fit the observed points.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Humans have natural desires to record and share their experiences. Lascaux Cave, located in southwestern France, contains one of the oldest form of wall paintings, where the cave wall is beautifully decorated with sophisticated drawings of wild animals and hunting activities. Although we could only guess why the cave people drew the paintings, it is clear that these were an important part of their culture, as the scientific analysis shows that some of the materials used were unusually difficult to acquire such as the scarce manganese oxide minerals [31].

As the technologies developed, our ancestors were equipped with increasingly sophisticated media to record their experiences via sculptures, paintings, or writings, etc. Notably, the advent of the modern camera in the 19th century made it possible to take a photo-realistic snapshot of the world with just a tap of the finger.

Fast forward a couple centuries, billions of people nowadays have a powerful digital photography machine in their own pockets, i.e., a smartphone, which allows taking photos or videos whenever we feel like sharing a moment. The democratization of the portable camera has radically changed the dynamics of content creation, where billions of content consumers are now only a 'touch' away from becoming a content creator via uploading their photos/videos to online websites.

The driving force behind this evolution of the recording media (from wall paintings to plaster sculptures to selfies on your iPhones) has been the series of technological revolutions that push the boundaries of more realistic but more convenient modes of recording experiences. History, then, prompts a question – what will be the next dominant mode of recording?

I argue that a key aspect of next-generation media will be the ability to capture the world in '3D.' Imagine you want to record your room before moving out. Capturing your room in 3D can enable new user experiences such as virtually walking around in your room, looking closely at an old doodle on your desk, and potentially turning on the lamp you liked. The ability to digitally 're-render' a scene from new viewpoints and interact with the scene is a strong value proposition that is likely to overshadow the traditional static 2D videos.

For the last few decades, very good progress has been made in several areas of 3D reconstruction. Today, for example, you can pick up a phone and take a '3D photograph' that allows rotating a static image or video in 3D [154, 101], or can wave a depth sensor to acquire a highly detailed 3D model of the scene [175, 243].

Despite this encouraging progress, however, existing approaches for 3D reconstruction present trade-offs between visual realism and ease of the capture process. On one hand, methods that provide the highest quality reconstructions often require laboratory capture setups or dense view sampling, i.e., the user needs to capture a scene from a very wide range of view angles, significantly limiting the applicability of these approaches [245, 167, 80, 84]. On the other hand, user-friendly systems that only require casual scanning using hand-held cameras have been more widely adopted in industry (e.g., Apple ARKit, HoloLens). While these methods have simple usage setups, the resulting reconstructions often lack extrapolation ability [167, 134, 101] (i.e., cannot predict views far away from captured views) or the quality of the 3D models is not photo-realistic enough to be used for immersive visual experiences [175, 243, 180, 65].

In this thesis, I address the problem of achieving photo-realistic 3D reconstruction from user-friendly capture processes. I discuss various techniques for obtaining high quality digital reconstruction of a real scene for novel view predictions. Specifically, I focus on three important aspects of 3D reconstruction that strongly affect the realism of the reconstructed scene appearance, which are modeling (i) high-resolution textures, (ii) specular highlights for glossy surfaces, and (iii) scene environment lighting.

At the same time, since I aim for high quality reconstruction with a casual scanning

process, a critical question to ask in this thesis is "what makes a 3D reconstruction system easy to use for casual users?" I identify and address three different barriers to ease of capture, as follows:

1. Casual Input: A user should be able to create 3D content just by moving a hand-held camera around to shoot a video of a target scene, and the camera required by the system should be consumer-grade. These two requirements are in contrast with many of the existing methods that require lab settings such as professional laser scanners, calibrated scene lightings, or robotic arms for camera movements.

2. Generalization Across Viewpoints: A surface viewed in one direction could appear differently from another direction, due for example to specular effects for a shiny surface. It would be extremely challenging if a user needed to capture all possible viewpoints of a surface (e.g., around the hemisphere). Thus, a casual scanning system should only require capturing each surface from several sparsely distributed viewpoints and should be able to automatically predict how the surface would appear from a wide range of uncaptured viewpoints.

3. Completion of Unobserved Parts: It would be unreasonable for casual users to scan every corner of a room. Therefore, a user-friendly system should automatically complete the unscanned parts of the scene.

To achieve the above goal of photo-realistic casual capture, I introduce novel systems, techniques, and frameworks to address each of the aforementioned subproblems throughout this thesis. In Chapter 3, I describe my casual system, *Surface Light Field Fusion* [184], for reconstructing high quality textures and specular highlights from a hand-held RGB-D camera. While commodity RGB-D cameras have been widely adopted for high quality shape reconstruction due to their ability to predict depth measurements, the *appearance* of scanned models is often unconvincing, as view-dependent specular reflections are not modeled [273, 175] or the textures are often washed out [176, 175, 243].

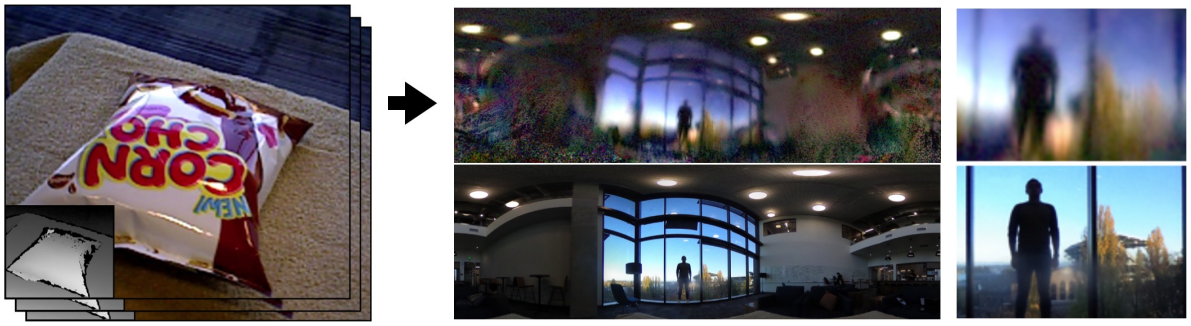
Figure 1.1: RGB-D input video streams to the system [184]. From left to right: sample RGB image, depth map image, infrared channel image. The system recovers surface specularity property from the infrared laser system of the RGB-D camera.

Therefore, I developed a system for interactively scanning highly reflective objects with a commodity RGB-D sensor. In addition to shape, this approach models the surface light field [245], encoding scene appearance from all viewing directions. To achieve this capability, I factor the surface light field into view-dependent and wavelength-dependent components. Then, I exploit the IR laser attached to commodity depth cameras to measure the view-dependent specular surface properties in the IR channel (Figure 1.1) and reconstruct the wavelength-dependent diffuse texture using my novel high-resolution texture tracking and fusion system.

As such, the sysem is able to convincingly reproduce specular highlights, while capturing high quality diffuse textures, with about the same effort needed to recover shape alone. We simply need a handful of IR images that come with the RGB-D scanning and environment lighting captured via a 360° panorama camera for rendering to provide a compelling baseline for photo-realistic casual scanning as shown in Figure 1.2.

An ideal user-friendly system, however, would only require a hand-held camera for 3D reconstruction, so the additional step of capturing scene environment with a panorama camera could be a barrier to a casual experience as described in the above Criteria 1. Therefore, in Chapter 4, I introduce an approach to *automatically* estimate the scene environment just from the specular reflections of shiny surfaces [183]. For instance, from a video of a shiny bag

Figure 1.2: Ground truth images and renderings. First column: Ground truth photographs. Second column: Synthetic rendering of the same pose. Third and fourth column: Rendering of different camera poses.



(a) Input images **(b)** Estimated environment (top), ground truth (bottom) **(c)** Zoom-in

Figure 1.3: From a hand-held RGB-D sequence of an object (a), we reconstruct an image of the surrounding environment (b, top) that closely resembles the real environment (b, bottom), entirely from the specular reflections. Note the reconstruction of fine details (c) such as a human figure and trees with fall colors through the window. We use the recovered environment for novel view rendering.

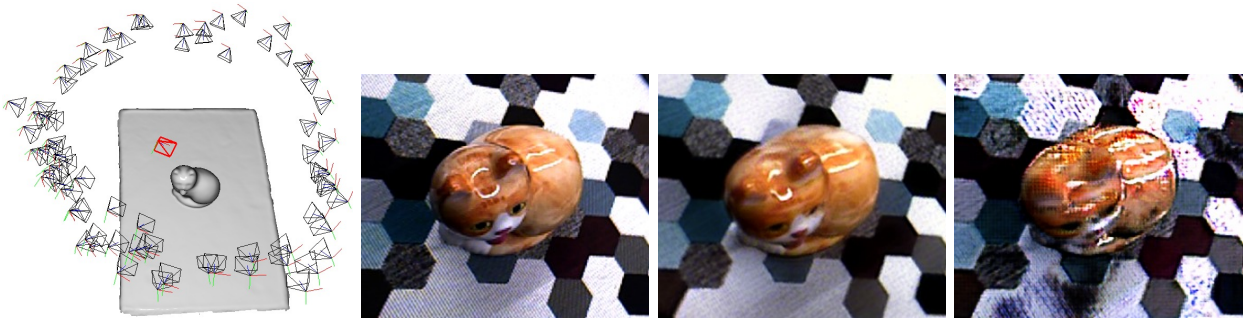
of chips, we can reconstruct a detailed image of the surrounding room including windows, human silhouette, or even trees outside the window (Figure 1.3).

These environment reconstructions, which I call *specular reflectance maps* (*SRMs*), represent the distant environment map convolved with the object's specular surface material. As most scenes are composed of a mixture of materials, each scene has multiple basis SRMs, one for each material. I therefore reconstruct a global set of SRMs, together with a weighted material segmentation of scene surfaces through a joint optimization. Based on the recovered SRMs, together with additional physically motivated components, I build a neural rendering network capable of faithfully approximating the appearance of a complex scene from all viewpoints (Fig. 1.4).

Using the proposed system, a user can obtain a surprisingly detailed image of the environment simply by waving a hand-held camera. The environment image can then be used as an input to the neural rendering network to construct a photo-realistic scene model, replacing the panorama image required in Chapter 3. This rendering scheme can model the

(a) Ground Truth (b) Synthetic Rendering (c) Specular Component

Figure 1.4: Novel view rendering results for various scenes. The recovered SRMs allow rendering the specular component (c) for each view, which is provided as input to the rendering neural network that predicts a photo-realistic synthetic rendering (b) that appears close to the actual photographs (a). Note that some of the ground truth reference images have black "background" pixels inserted near the top and left borders where reconstructed geometry is missing, to provide equal visual comparisons to rendered images.



(a) Camera Trajectory (b) Reference Photo (c) Ours (d) Thies [230]

Figure 1.5: View extrapolation to extreme viewpoints. Novel view synthesis experiment on test views (red frusta) that are very far from any of the input views (black frusta) (a). The view predictions of Thies [230] (d) are notably different from the reference image (b). Our method (c) produces images with highlights appearing at correct locations.

appearance of complex scenes with specular surfaces, interreflections, and convex objects, effects that are most often ignored by the works in the literature. Notably, the combination of physically-based and learning-based neural network modeling leads to excellent generalization across viewpoints, evidenced by its superior performance in view-extrapolation, a task of predicting views that are far from any of the input views (Fig. 1.5). This ability to reconstruct the full scene appearance from sparse input images significantly reduces the user's efforts for scanning, as described in the above Criteria 2.

While the above system achieves scanning with a hand-held camera, it can only reconstruct the part of the scene visible in the input video sequence. To reconstruct all surfaces, an user must scan every corner of a scene, which is infeasible for most cases. Therefore, an ideal casual scanning system should be able to complete unobserved parts of the scene.

Automatically completing partial data, however, is a challenging problem that requires extracting recurring features and patterns from a large dataset. In the 2D domain, deep convolutional neural networks (CNNs) [135], which operate on 2D grids of pixels, have been successfully adopted for image completion [146]. Processing a 3D grid of voxels with a 3D

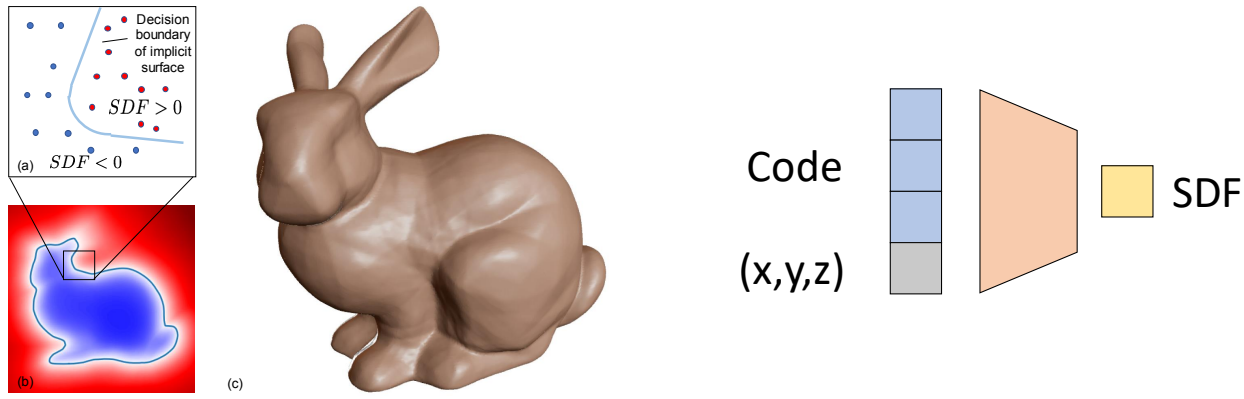
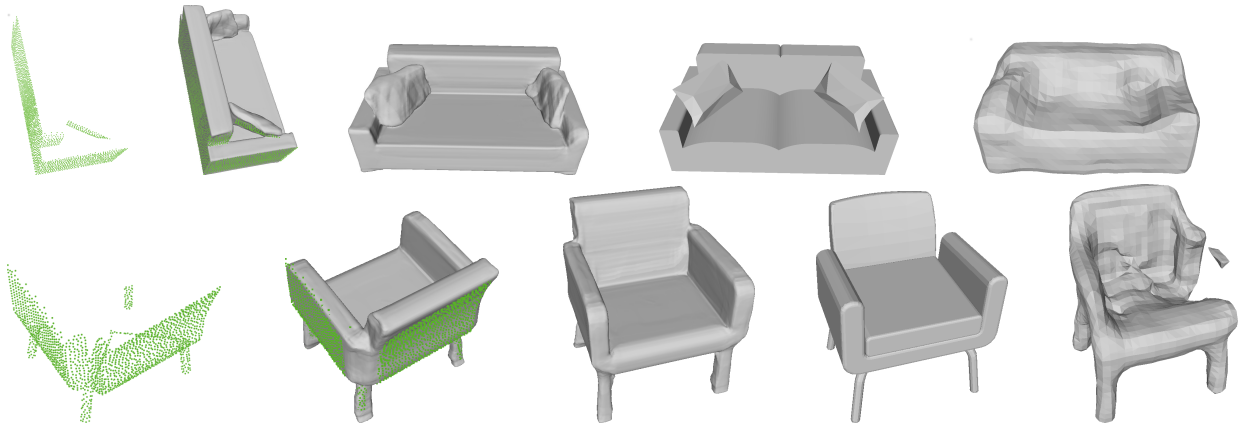


Figure 1.6: Left: DeepSDF [182] representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Right: DeepSDF network architecture takes input as a shape-conditioned latent code concatenated with a query xyz position and outputs the SDF prediction.



(a) Input Depth (b) Completion (c) Second View (d) True Shape (e) 3D-EPN

Figure 1.7: For a given depth image visualized as a green point cloud, we show a comparison of shape completions from our DeepSDF approach against the true shape and 3D-EPN [50].

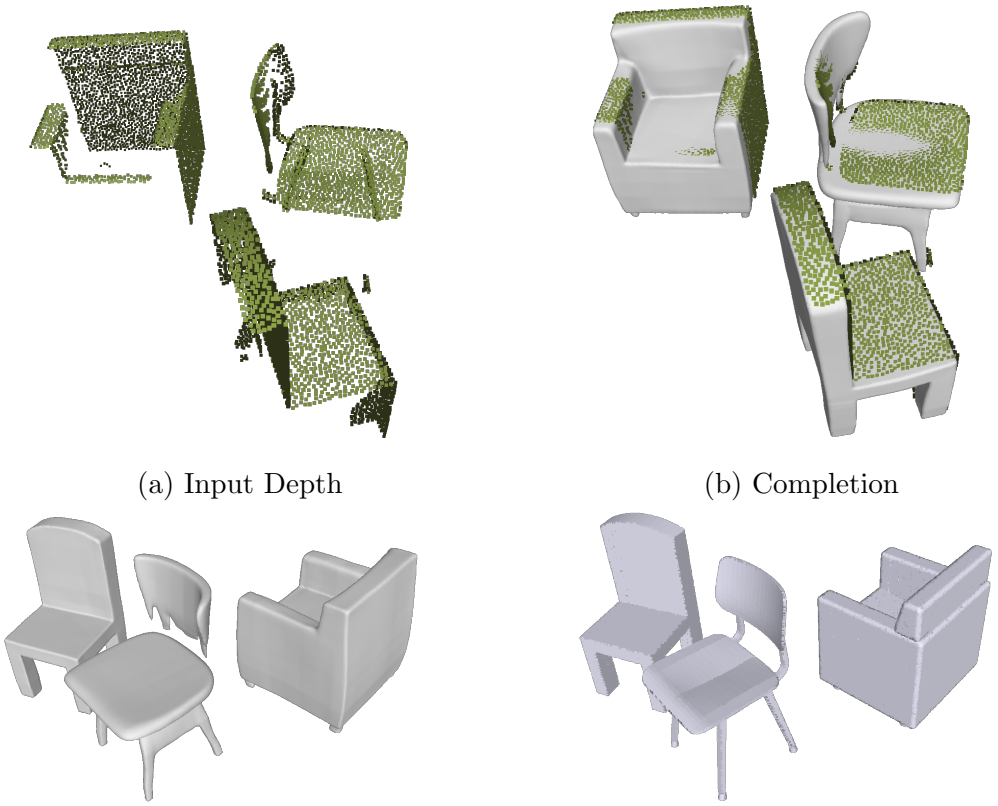
CNN is perhaps the most intuitive extension to the 3D domain, but the 3D CNNs scale poorly in memory and time [249, 50]. Furthermore, more compact 3D representations such as point clouds cannot describe dense surfaces, and triangle meshes come with arbitrary topology that makes them difficult to process with a neural network. The lack of effective representation has limited the fidelity and applicability of deep learning approaches in 3D.

To overcome the above challenges, I designed a new 3D representation suitable for deep learning, which I call DeepSDF, that is more efficient and expressive [182]. This approach uses the concept of a Signed Distance Function (SDF), but unlike common surface reconstruction techniques which discretize this SDF into a regular grid for evaluation and measurement denoising, we instead learn a generative model to produce such a continuous field.

The key idea is to directly regress the continuous signed distance field using a neural network. A signed distance function is a volumetric field where the magnitude at a point represents the distance to the closest surface boundary, and the sign indicates whether the point is inside or outside of the shape. Thus the surface is implicitly represented as zero-level-set, where the SDF value is equal to zero (Fig. 1.6). I regress this signed distance function using a fully connected neural network that takes as input an xyz coordinate and outputs the predicted SDF value of that position.

The proposed continuous representation may be intuitively understood as a 3D classifier that predicts whether a point is inside or outside of the shape, and thus its decision boundary is the surface of the shape itself. The learned DeepSDF models high quality continuous surfaces with complex topologies, and obtain state-of-the art results in shape modeling and completion (Fig. 1.7), while having orders of magnitude smaller network size than voxel-based methods. These results suggest that the proposed coordinate-based volumetric field modeling is a promising new direction for using deep learning in 3D.

Training of DeepSDF model, however, requires a dataset of aligned single objects, limiting its applicability to real-world scenes with objects in arbitrary poses. Therefore, in Chapter 6, I build on top of the continuous representation to improve two important features: modeling transformation of objects and multi-object scenes. Directly extending DeepSDF to



(a) Input Depth (b) Completion

(c) Second View (d) Ground truth

Figure 1.8: Multi-object completion result. For a given depth image of a scene with three objects visualized as a green point cloud (a), I run my shape completion algorithm of Sec. 6.3.2. The completion result (b,c) is of high quality and close to the ground truth shapes (d).

non-aligned shape modeling would be ineffective, as DeepSDF relies on absolute global coordinates. I solve this problem by dividing the target volume into a grid of voxels and apply the DeepSDF technique within the local voxel coordinate system to introduce translational invariance. Then, I train a generative model that produces a voxel grid of local DeepSDF models given a latent vector.

The proposed generative model can reconstruct an object with arbitrary pose by projecting the object shape onto the latent space via gradient descent optimization. In order

to model a scene with more than one object, one can simply initialize some number of latent vectors and jointly optimize them in order to fit the scene surface points. Because the generative model contains priors of a completed object, the projection of partial multi-object observations onto the latent space will automatically result in a scene with completed objects (Fig. 1.8). The proposed technique for modeling multi-object scenes opens up a door for real-world scene completion to build a casual reconstruction system, as described in Critera 3.

Overall, this thesis makes the following two contributions towards casual photo-realistic reconstruction. First, I introduce realistic appearance and environment reconstruction systems from a casual scanning. Second, I propose an effective representation for 3D machine learning that helps reconstructing a plausible scene shape from partial observations. The remainder of the thesis is organized as follows. Chapter 2 reviews the related works in the literature ranging from traditional 3D reconstruction approaches to recent developments of neural implicit representations. In Chapter 3, I describe a casual system for modeling high quality appearance from a hand-held RGB-D camera and its IR laser. Chapter 4 introduces a system capable of jointly recovering high quality environment image and a scene appearance model only from a hand-held camera. In Chapter 5, I introduce the new continuous 3D representation suitable for 3D deep learning, and, in Chapter 6, I extend it to be applied to the multi-object scene completion task, which could ultimately alleviate the users' need to scan every corner of the target scene.

Chapter 2

RELATED WORK

In this report, I describe previous work broadly related to 3D reconstruction. This not only includes physically motivated 3D geometry and appearance reconstruction, but also includes learning-based techniques using various 3D representations. This section also surveys the recent development of neural network-based implicit 3D representations, and deep representation learning techniques that enable automatically discovering the features needed for data processing, generation, or completion.

2.1 3D Reconstruction and Inverse Rendering

The image formation process starts with environment light, which gets reflected, refracted, or scattered on a 3D surface point, directing some of the light to the camera. The incoming light is then recorded by the camera sensor – this process is commonly approximated using the pinhole camera model with lens distortion compensations. The geometric relationship between a 3D point and the camera coordinate frame is generally described with a 4x4 extrinsic matrix that represents the rotation and translation of the camera. For more details about the image formation process, see [224]. For many years, the computer vision community has made attempts to invert the physical image formation process to infer 3D geometry, surface reflectance, or environment lighting that produces the 2D images.

SLAM and SfM Simultaneous Localization and Mapping (SLAM) techniques jointly compute the trajectory of the moving camera along with the map of the 3D scene geometry. SLAM algorithms typically assume a continuous input stream such as a video sequence and thus are useful for many real-time applications including Augmented Reality or Robotics.

Structure from Motion (SfM) methods similarly achieve camera localization and scene mapping, but they often assume offline global processing of input images that are not necessarily from a same video sequence.

SLAM or SfM methods start by processing input images with 2D point descriptors such as SIFT [153] or SURF [14], which are known to be robust to scale, rotation, brightness, and, ultimately, viewpoint changes. These point features are used to find matching regions between images – the corresponding 2D points that are likely to be displaying the same 3D point from different viewpoints. Once the correspondences are established and the outliers filtered out (e.g. RANSAC [70]), one can solve for the relative camera positions between pairs of images and estimate sparse 3D surface points via triangulation [224, 216].

Both SfM and SLAM methods suffer from drift, which is caused by accumulation of relative pose estimation errors across images, especially when the camera trajectory is long and non-overlapping. In order to mitigate drift accumulation, SfM methods conduct offline optimization to close loops [71] or global bundle adjustment [216, 217], which involves jointly optimizing for all camera poses and correspondences to minimize the reprojection error of point features. Early works in the real-time SLAM literature adopts Kalman Filters [98, 54, 215] and Particle Filters [190] to recursively estimate the state vectors (i.e., camera pose) and landmark parameters. PTAM [133] first demonstrated that the high quality tracking (estimating the camera pose of a new frame based on current map) and mapping (updating the current map using global optimization of key frames) can be conducted simultaneously. More recently, ORB-SLAM systems [170, 171] expanded the robustness and versatility of PTAM by incorporating new and existing techniques such as loop detection, pose graph optimization for removing drifts, ORB features [200], and flexible key frame management. For a more comprehensive review of the SLAM field, see [28].

Dense Surface Reconstruction The above feature-based methods generate a sparse map of the environment composed of sparse 3D feature points. However, a more dense reconstruction of scenes is often desired because it allows a better visualization experience

accounting for occlusions and virtual-to-real collision simulation for AR.

Multi-View Stereo (MVS) [67, 79, 74, 207] refers to a class of techniques that recovers dense 3D geometry from a set of images of a scene taken from multiple viewpoints with known, calibrated cameras. MVS methods typically use a plane sweeping strategy to compute a photo-consistency cost volume [209], iteratively refine a surface or volume as in [136], or compute a set of 2D depth maps and merge them [143]. Traditional MVS techniques were developed assuming laboratory settings with Lambertian assumptions, fixed lighting, and small-scale scenes [207]. As online photograph datasets become widely available, newer methods emerged for recovering dense geometry of large scale landmarks from uncalibrated images [80]. Other works considered temporal changes of outdoor scenes or support relighting capabilities [211]. There is a large corpus of work on MVS over the decades; I refer readers to [207, 73] for a more detailed overview of the field.

To enable dense scene reconstruction in real-time, direct SLAM methods [66, 65, 64, 176] track the movements of all pixels in frames, as opposed to tracking feature points, optimizing for the per-pixel depth and camera pose by minimizing the photometric errors (e.g., brightness, gradients). The direct methods not only provide denser scene reconstruction but also enable much more robust camera tracking in featureless environments, and perform well even with extreme motion blur [176]. On the other hand, because there are millions of 3D points in the reconstructed scene, it is highly expensive to perform global optimization of all points and frames. As a result, direct visual SLAM methods often lack loop closure capabilities [64, 66].

Volumetric Fusion Image-based depth maps generated by MVS or active laser sensor systems, e.g., Kinect, can be merged into a single 3D surface through volumetric fusion, which typically involves a voxel grid representation of a signed distance function (SDF) [103, 47]. A signed distance function is a volumetric field where the sign indicates the inside-outside state of a watertight surface, and its magnitude indicates the distance to the closest surface. Fusing the input depth maps using such a non-parametric SDF representation was

pioneered by [47], which showed that a simple weighted-average of projective SDF across diverse viewpoints could be an effective approximation for the true SDF. KinectFusion [175] adopted this SDF representation combined with the Iterated Closest Point (ICP) algorithm [19] to estimate the live camera pose using model-to-frame alignment and achieves real-time dense reconstruction. Several extensions of KinectFusion enable reconstruction of large-scale [243], dynamic [174], or reflective [184] scenes.

Reflectance Estimation Surface reflectance is defined by how a surface reflects the incoming light and is most commonly described via the Bidirectional Reflectance Distribution Function (BRDF), that models outgoing radiance as a function of incoming and outgoing ray directions. Perhaps the simplest, and thus the most common, way of modeling BRDF is applying Lambertian assumption, in which the incoming ray is scattered uniformly in the surface hemisphere [13, 211, 269].

Recovering more general forms of surface BRDFs in a natural, in-the-wild setting is challenging because it requires solving for the complex global illumination process [63]. As a result, most existing reflectance estimation methods limit their systems' operating range to an isolated single object [246, 81], controlled lighting [140, 81], or spatially uniform materials [130, 163].

Environment Lighting Estimation The most straightforward way of recovering environment lighting is using light probes [55] (e.g., a mirror ball) or taking a 360° panorama image [184]. In many cases, however, directly capturing the environment is infeasible, motivating research on lighting estimation using existing cues in the scene. Indirectly estimating lighting from a photo of a general scene is highly under-constrained, so many methods rely on human inputs [124, 272], or apply manually designed intrinsic image priors to their solutions [125, 13, 12, 20, 149].

Layer separation techniques recover the reflected environment from a video of plane reflectors such as a glass picture frame [225, 213, 258, 93, 91, 271]. Recovering reflections

from a general surface is significantly more challenging, even for humans, as the reflected content depends strongly and nonlinearly on surface shape and spatially varying material properties. Therefore, a number of methods seek to recover low-frequency components of the lighting from general curved surfaces [278, 180, 155]. Such methods adopt spherical harmonics lighting representations (following [195]) to simultaneously infer lighting and refine geometry with diffuse surfaces. Other physically-based illumination estimation methods solve for the BRDF and scene lighting for a single convex object, often assuming controlled laboratory settings [246, 61, 254].

More recent learning-based approaches infer lighting from a single image to mitigate the under-constrained nature of the lighting and material separation problem. A number of methods train a neural network to estimate a panoramic representation of the environment from a single image both for indoor and outdoor scenes [75, 218, 139, 104]. While these methods produce plausible environment images, the results are not accurate and lack details (i.e. only get the general direction of lighting)

Image-based Rendering Image-based rendering methods avoid modeling the complex global light transport process, and instead seek to model the scene appearance by reusing the observed radiance values from the input images. Early image-based rendering techniques use proxy geometry to project texture and appearance onto the surface from the images to render novel views [208, 57]. Light-field rendering methods [84, 142] instead mitigate the need for proxy geometry by directly modeling the 5D *light field* of the scene through dense view sampling. While the geometry-free methods require large number of input views, they have an advantage of handling thin-structures, e.g. hair, or view-dependent effects. The surface light field method [245] assumes laser-scanned accurate geometry to achieve high quality renderings with a compact representation.

Recent work in the literature use convolutional neural networks (CNNs) to achieve high quality image-based rendering. These include [102] that used CNNs to learn blending heuristics, [230] that attached learned features to 2D texture maps or point clouds, and [3] to render

realistic novel viewpoints. Other learning-based methods provide impressive results for local light field reconstruction from a small number of input images [167, 166, 219, 43, 119, 275]. These methods, however, do not explicitly model physics, e.g., specular highlights, and thus fail to extrapolate well to novel viewpoints that are far from the input viewpoints.

2.2 Learning-based 3D Reconstruction

Data-driven machine learning approaches have seen remarkable success in 2D computer vision. Convolutional neural networks have been the cornerstone for 2D vision research, enabling downstream tasks such as object detection, segmentation, and realistic image generation [100, 110]. For the past several years, there have been numerous attempts at applying these learning-based techniques to 3D computer vision with various ways of representing 3D geometry and appearance. These methods typically adopt such representations as voxels, point clouds, or meshes in training their deep networks. In this section, I briefly describe work in this area and refer readers to [182] for a more comprehensive review.

2.2.1 Voxel-based

Perhaps the most natural extension from 2D to 3D deep learning has been voxel-based representations, where occupancy or other values are non-parametrically represented with a regular volumetric grid. These methods typically train a generative model that outputs voxel-based 3D shapes given an image or a random code [44, 249]. Other methods train a network that takes partially filled discrete SDFs and output a completed version of the discrete SDF in voxel form [50, 268]. These voxel-based learning approaches, however, suffer from lack of detail and sharpness of the result because the space complexity grows exponentially with the resolution.

2.2.2 Point-Based

PointNet [191, 192] pioneered data-driven learning methods for point-based representation by applying multi-layer perceptrons (MLPs) to the point coordinates and abstract all the

points using a max-pooling layer. This architecture has been widely used as a backbone encoder for various point generation networks [1, 260]. The primary disadvantage of the point cloud representation is that it does not describe a surface and thus not suitable for realistic rendering or augmented reality applications.

2.2.3 Mesh-Based

Other works apply existing mesh parameterization techniques to describe 3D surfaces by warping a 2D plane [212, 159]. These methods depend heavily on the choice of the hand-designed 2D parameterization algorithms. [89, 16] learn the 2D parameterization using a set of 2D planes, but report that the planes are not stitched well together, thereby generating non-watertight surfaces. Recently, graph convolution operations have been adopted to process and generate triangle meshes [58, 233].

2.3 Neural Implicit Representations (NIR)

In 2019, we [182] and [164, 39] have concurrently discovered that, instead of explicitly expressing geometry using voxels or meshes, implicitly expressing geometry using a neural network provides a dramatic improvement of efficiency. Such neural implicit representation methods train a neural network that takes an xyz-coordinate and outputs either occupancy or an SDF value for that particular position. An explicit mesh or voxel representations can be extracted through marching cubes [152]. These NIR techniques have subsequently been used in such applications as robotics [267] or human reconstructions [204].

Radiance Field Modeling More recently, authors of NERF [167] realized that the neural implicit representation could be used to model volumetric radiance fields instead of SDFs to achieve high quality appearance reconstruction. NERF combines the volume rendering, NIR, and positional encoding techniques from the literature to reconstruct geometry and radiance of highly complex, reflective, and refractive scenes from a sequence of calibrated images. Many extensions of NERF have been introduced, ranging from speeding up the training

[147], or enabling real-time rendering [244], to modeling dynamic scenes [185]. SIREN [214] demonstrates that replacing the ReLU activation layers with harmonic functions induces differentiability of the MLP, which enables fitting data in the gradient domain, e.g., fitting a signed distance field only from surface points via solving a relevant differential equation. Pi-GAN [32] applies the SIREN architecture to train a generative model that handles a space of 3D appearance models rather than fitting to a specific scene. Finally, recent works have sought to unify SDF and radiance field modeling using a single MLP, to simultaneously obtain high quality geometry and appearance [261, 238].

Scene-level Reconstruction Recent work aims to enable scene-level NIR reconstruction. [223] applies the implicit representation for tracking and mapping to build a real-time SLAM system. Some NIR methods [30, 115, 147] divide the target scene into a grid of voxels and learn one latent code for each voxel to describe complex large scenes, but limit their applications to reconstructing visible surfaces. [187, 40] apply 3D convolutional layers on the voxel grid to learn high-level scene priors for geometry reconstruction. [262] uses a 2D CNN that takes as input a single image to output a 3D feature volume, which can be converted into the 3D radiance field with an MLP.

2.4 Deep Representation Learning Techniques

Representation learning techniques try to automatically discover low dimensional features that compactly but expressively describe large and complex datasets. For a more comprehensive review of the field, I refer readers to [17].

Auto-Encoders Auto-encoder algorithms [202] project high dimensional input data to a low-dimensional bottleneck (encoder) and then reconstructs the input data using the bottleneck features (decoder). Thus, the bottleneck features are forced to learn compact representations of a dataset. Auto-encoder techniques have been widely used in the 3D learning literature [50, 222, 88], often using variational variants [22, 9] to obtain smooth and complete

latent spaces for ease of sampling and optimization.

Generative Adversarial Networks Generative Adversarial Networks (GANs) and their extensions [83, 110, 120] train a network that takes a randomly sampled latent code from the prior distribution and outputs a 2D image or 3D appearance model, whose realism is scored by a discriminator. The discriminators of the GANs are trained to discern real data from generated ones, where the the generators are optimized to fool the discriminators. This adversarial training allows GANs to realistically model a wide range of data distributions. In 3D vision, GANs have been applied to voxel grids [246], and more recently, NIR-based GANs were introduced to train a generator that can produce realistic 3D appearance model from unlabeled images [32]. For a comprehensive review, refer to [240]

Multi-Code Generative Models Recent GAN-related methods explore using more than one latent code to describe images, as the traditional single code-based GANs are known to be incapable of describing scenes with multiple objects and complex structures. [90] uses multiple copies of a trained StyleGAN [123] network to discover multiple latent variables that, when overlayed together, explain an input image. More recently, the Transformer architecture [232], used in NLP community, has been applied to GANs, successfully generating multi-object scenes using multiple codes [108] (this paper also provides a more comprehensive review of the multi-code generative models). Slot Attention [148] uses an iterative attention mechanism to reduce a given image into slots of latent vectors, each of them representing a salient object. In the 3D domain, GIRAFFE [178] learns a latent space of 3D appearance models of single objects without explicit supervision of object segmentation, and obtains a high quality generative model for sampling compositional scenes.

Chapter 3

APPEARANCE RECONSTRUCTION WITH RGB-D SENSORS

In this chapter, I introduce a hand-held scanning approach for reconstructing highly realistic digital representations of shiny objects. In particular, the proposed system takes input from a hand-held commodity RGB-D imaging system, e.g., Microsoft Kinect or Apple iPhone, to achieve high-quality appearance modeling, via leveraging the infrared laser system that comes with an off-the-shelf RGB-D sensor.

The advent of commodity RGB-D sensing, first mass-produced by Microsoft, has led to great progress in 3D shape reconstruction. The modern real-time dense geometry reconstruction systems provide interactive visual feedback of the scanning process that greatly lowered the barrier for 3D applications. The real-time scanning capabilities have promoted the development of consumer Augmented Reality devices equipped with RGB-D sensors, including Microsoft Hololens and Apple iPhone [48, 175, 243].

Despite the progress in geometry reconstruction, however, the *appearance* of scanned models is often unconvincing, as the textures are often washed out and the specular highlights are baked into the texture. In particular, modeling of the specular highlights had been widely overlooked in the casual 3D reconstruction approaches [216, 175]. This is because the view-dependent reflections (Bidirectional Reflectance Distribution Function, or BRDF) and global light transport (shadows, interreflections, subsurface scattering) are challenging to model.

Rather than attempt to fit BRDFs and invert global light transport, an alternative is just to model the radiance (in every direction) coming from each surface point. This representation, known as a *surface light field* [245], captures the scene as it appears from any viewpoint in its native lighting environment.

I present the first Kinect Fusion-style [175] approach designed to recover surface light

fields. This approach convincingly reproduces view-dependent effects such as specular highlights, while retaining global effects such as diffuse interreflections and shadows that affect object appearance in its native environment. To achieve this capability, I approximate the full surface light field by factoring it into two components: 1) a local specular BRDF model that accounts for view-dependent effects, and 2) a diffuse component that factors in global light transport.

The resulting approach reconstructs both shape and high quality appearance, with about the same effort needed to recover shape alone – I simply require a handful of IR images taken during the usual scanning process. In particular, I exploit the IR sensor in commodity depth cameras to capture not just the shape, but also the view-dependent properties of the BRDF in the IR channel. My renderings under environment lighting capture glossy and shiny objects, and provide limited support for anisotropic BRDFs.

This chapter makes the following contributions. The first is my novel, factored, surface light field-based problem formulation. Unlike traditional BRDF-fitting methods which enable scene relighting, I trade-off relighting for the ability to capture diffuse global illumination effects like shadows and diffuse interreflections that significantly improve realism. Second, I present the first end-to-end system for capturing full (non-planar) surface light field object models with a hand-held scanner. Third, I introduce the first reconstructions of *anisotropic* surfaces using commodity RGBD sensors. Fourth, I introduce a high-resolution texture tracking and modeling approach that better models high frequency details, relative to prior Kinect Fusion [175] approaches, while retaining real-time performance.

3.1 Problem Formulation and Related Work

The input to the system is an RGBD stream and infrared (IR) video from a consumer-grade sensor. The output is a shape reconstruction and a model of scene appearance represented by a surface light field SL, which describes the radiance of each surface point $\boldsymbol{x}$ as a function of local outgoing direction $\boldsymbol{\omega_o}$ and wavelength λ: $SL(\boldsymbol{x}, \boldsymbol{\omega_o}, \lambda)$. I introduce a novel formulation designed to effectively represent and estimate SL. I start by writing SL in terms of the

rendering equation [118]:

$$SL(\boldsymbol{x}, \boldsymbol{\omega_o}, \lambda) = \int_{\Omega} E(\boldsymbol{x}, \boldsymbol{\omega_i}, \lambda) f(\boldsymbol{x}, \boldsymbol{\omega_o}, \boldsymbol{\omega_i}, \lambda)(\boldsymbol{n_x} \cdot \boldsymbol{\omega_i}) d\boldsymbol{\omega_i}, \tag{3.1}$$

where E denotes global incident illumination, f surface BRDF, $\boldsymbol{\omega_i}$ incident direction, and $\boldsymbol{n_x}$ surface normal.

Prior work on modeling SL uses either parametric BRDF estimation methods or non-parametric image-based rendering methods.

BRDF estimation methods [82, 140, 247, 248] solve for an analytical BRDF f by separating out the global lighting E. Accurately modeling the global light transport, however, is extremely challenging; hence most prior art [82, 140, 247, 248] simply ignore global effects such as shadows and interreflections, assuming the scene surface is convex. Occlusions and interreflections cause artifacts under this assumption, so some methods [247, 248] require a black sheet of paper below the target object to minimize these effects.

Image-based rendering techniques [36, 168, 245] avoid this problem by nonparametrically modeling the surface light field SL, via densely captured views on a hemisphere. However, the dense hemispherical capture requirement is laborious, and therefore not suited to casual hand-held scanning scenarios. While the recent neural network-based implicit representation approaches [167, 262, 185, 147] achieves photo-realistic radiance modeling, they still require a wide range of input viewpoints around the hemisphere to model the full appearance model of a scene.

I address these two issues by 1) *empirically* capturing the diffuse component of the appearance that factors in the global lighting, and 2) *analytically* solving for *specular* parameters leveraging the built-in IR projector.

Specifically, I further decompose SL into a diffuse term D, specular term S with *wavelength-*

independent BRDF (as in the Dichromatic model [210]), and residual R:

$$SL(\boldsymbol{x}, \boldsymbol{\omega_o}, \lambda) = \underbrace{\int_\Omega E(\boldsymbol{x}, \boldsymbol{\omega_i}, \lambda) f_d(\boldsymbol{x}, \lambda)(\boldsymbol{n_x} \cdot \boldsymbol{\omega_i}) d\boldsymbol{\omega_i}}_{D(\boldsymbol{x}, \lambda)} + \underbrace{\int_\Omega E_d(\boldsymbol{x}, \boldsymbol{\omega_i}, \lambda) f_s(\boldsymbol{x}, \boldsymbol{\omega_o}, \boldsymbol{\omega_i}; \boldsymbol{\beta})(\boldsymbol{n_x} \cdot \boldsymbol{\omega_i}) d\boldsymbol{\omega_i}}_{S(\boldsymbol{x}, \boldsymbol{\omega_o}, \lambda)} + R(\boldsymbol{x}, \boldsymbol{\omega_o}, \lambda), \tag{3.2}$$

where f_d is diffuse albedo, f_s specular component of BRDF with parameters $\boldsymbol{\beta}$, and E_d direct illumination; I ignore specular interreflection.

Rather than infer diffuse albedo f_d as in [82, 140, 247, 248], I simply capture the diffuse appearance D directly in a texture map (the concept referred to as a "lightmap" in the gaming industry). I thereby make use of the rich global effects like shadows and interreflections that are captured in D but avoid the difficult problem of factoring out reflectance and multi-bounce lighting. In doing so, I give up the ability to relight the scene, in exchange for the ability to realistically capture global illumination effects.

The specular term S is approximated by recovering a parametric BRDF f_s with an active-light IR system, assuming that f_s is wavelength-independent. S captures the specular properties of *dieletric* materials like plastic or wood and non-colored metals like steel or aluminum [210, 226, 265].

I assume the residual R to be negligible - as such, my approach does not accurately model colored metals like bronze or gold, and omits specular interreflections.

Overall, my formulation has two main advantages. First, I realistically capture diffuse global illumination effects in SL without explicitly simulating global light transport. Second, the *specular* BRDF is analytically recovered from only a sparse set of IR images, removing the need for controlled lighting rigs [106, 140, 265] and extensive view sampling [36, 112, 168, 245].

Other Related Works Some authors explicitly model global light transport, as in [150] and, to a lesser extent, [197] that assumes low-frequency lighting and material. However, this

requires solving a much more difficult problem, and the authors note difficulties in recovering highly specular surfaces and limit their results to simple, mostly uniform textured objects.

[112] captures a *planar* surface light field with a monocular camera using feature-based pose tracking. Although [112] removes the need for special gantries used in [142, 245], it still requires a dense sampling of the surface light field, making it hard to scale beyond a small planar surface.

A few other authors utilize IR light and sensors to enhance the estimation of surface properties. [41] observes that many interesting materials exhibit limited texture variations in the IR channel to estimate the geometric details and BRDF. [42] refines a reconstructed mesh with shading cues in the IR channel. [128] leverages an IR sensor to add constraints to the intrinsic image decomposition task.

3.2 Computing the Specular Components

In this section, I assume that the geometry is already reconstructed and focus on estimating S, the view-dependent component of the surface light field. I assume that the material varies over the surface of the object or scene, but that there is a relatively small number of material clusters which estimates can be aggregated. I begin by calibrating the infrared (IR) system attached to the depth sensor and describe how to measure material properties in the IR channel for each material.

3.2.1 IR Projector Modeling

BRDF Estimation in the IR Channel I leverage the IR projector built-in to most depth sensors as a point light source. Because I know the projector location relative to the camera along with the scanned geometry, the surface BRDF can be estimated.

One challenge, however, is that the IR projector is typically not uniform – I use a Primesense structured light sensor that generates complex speckle patterns (Fig. 3.1).

I take a simple approach to select pixels that are lit by the projector: I divide the IR image into a grid structure and keep the brightest pixel within each 5x5 grid (Fig. 3.1). The

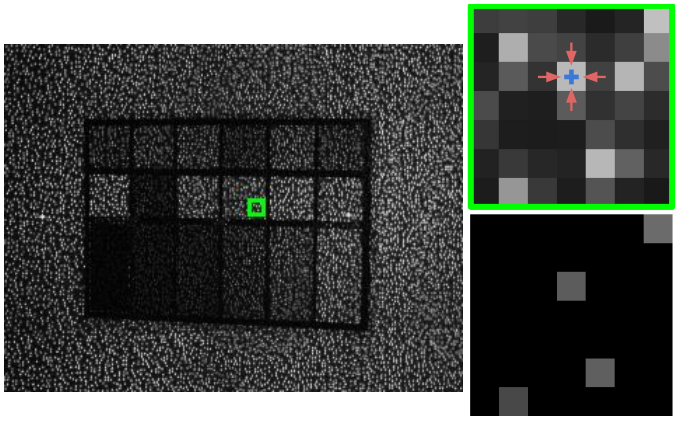


Figure 3.1: IR image (left), zoomed in (top-right) and local maximum subsampled image (bottom-right). Best viewed digitally.

intensity of such pixel is averaged with its four-neighborhood. Throughout the chapter, a *pixel* in the IR image refers to this local maximum intensity. For simplicity, I consider only the central 192x192 region, to minimize the effects of vignetting and radial distortion.

Despite the usage of the structured light sensor, the system can be easily generalized to other depth sensors with a projector and a receiver pair. For example, I refer readers to [128] for a time-of-flight projector model.

IR Projector and Camera Calibration With constant exposure time, the IR projector's light intensity κ and the camera's gamma compression parameter γ is optimized as following by capturing a white piece of paper whose albedo is assumed to be 1:

$$\min_{\kappa,\gamma} \sum_{\boldsymbol{x}\in P} \left(L(\boldsymbol{x}) - \left(\kappa \frac{\boldsymbol{n}_{\boldsymbol{x}} \cdot \boldsymbol{l}_{\boldsymbol{x}}}{\pi d_{\boldsymbol{x}}^2} \right)^{\gamma} \right)^2, \tag{3.3}$$

where $L(\boldsymbol{x})$ is observed IR intensity at a pixel $\boldsymbol{x}$ in a collated pixel set P over multiple images, and $\boldsymbol{n}_{\boldsymbol{x}}$, $\boldsymbol{l}_{\boldsymbol{x}}$ and $d_{\boldsymbol{x}}$ are normal, light direction and distance to the light source of the surface point seen by $\boldsymbol{x}$, respectively. Following [42], I assume indoor ambient IR light is negligible.

3.2.2 Per-Segment BRDF Estimation

Specular Reflection Model Renaming the directions $\boldsymbol{\omega_o}$ and $\boldsymbol{\omega_i}$ in Eq. (3.2) as local view $\boldsymbol{v_x}$ and light directions $\boldsymbol{l_x}$ respectively, I denote the specular component of the BRDF as $f_s(\boldsymbol{v_x}, \boldsymbol{l_x}; \boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is a vector of BRDF parameters that decides the reflectance distribution. In this work, I use the Ward BRDF model [241] for its simplicity and applicability to a variety of materials.

Material Segmentation Accurately fitting BRDF parameters for each individual point requires observing specular highlights for all surface points, which is prohibitively expensive. Therefore, estimating a single BRDF for a region with shared reflectance properties is essential for a practical and robust system. To identify regions of the surface with similar material, I apply an image segmentation method [15] using a Convolutional Neural Network (CNN) for semantic material classification, adapted to operate on a mesh rather than an image (Figure 3.2). I first convert the patch classifier into a Fully Convolutional Network [151] for dense class predictions (as in [15]) but increase the resolution of the output signal using Dilated Convolution [263]. At each input frame, the output probability map from the softmax layer is projected and averaged into the vertices of the mesh $\mathcal{M} = \{\mathcal{V}, E\}$. I consider the mesh connectivity as a Markov Random Field and obtain the final vertex material label $\boldsymbol{y}$ by minimizing the following energy function:

$$\Phi(\boldsymbol{y}) = \sum_{v \in \mathcal{V}} \psi_v(y_v) + \sum_{\{m,n\} \in E} \psi_{m,n}(y_m, y_n), \tag{3.4}$$

where the data term $\psi_v(y_v)$ of the MRF is the standard negative log likelihood at vertex v: $\psi_v(y_v) = -\log(p_i(y_v))$, and $\psi_{m,n}(y_m, y_n)$ is the pairwise term.

The pairwise term incorporates both a photometric and a geometric smoothness prior:

$$\psi_{m,n}(y_m, y_n) = \mathbf{1}_{[y_m \neq y_n]} \left(\lambda_p \exp(-\theta_p ||\boldsymbol{c}_m - \boldsymbol{c}_n||^2) + \lambda_g \frac{(\boldsymbol{g}(m,n) + \epsilon)}{||\mathcal{N}(m) - \mathcal{N}(n)||^2} \right),$$

where λ_p, λ_g, θ_p, ϵ are balancing parameters, and $\mathcal{N}$ and $\boldsymbol{c}$ are the vertex normal and color, respectively. Here, $\boldsymbol{g}(m,n)$ is an indicator function designed to penalize concave surface transitions as in [137]:

$$\boldsymbol{g}(m,n) = (\mathcal{N}(m) - \mathcal{N}(n)) \cdot (\mathcal{V}(m) - \mathcal{V}(n)) > 0,$$

where $\mathcal{V}$ is the vertex location. The optimal label $\boldsymbol{y}$ for $\Phi(\boldsymbol{y})$ is estimated via graph cuts [25].

Specular BRDF Optimization For each material segment, the specular parameters $\boldsymbol{\beta}$ that best explain the IR video are estimated. Each pixel $\boldsymbol{x}$ in an IR frame constrains the shared specular parameters $\boldsymbol{\beta}$ and the IR diffuse albedo $\rho(\boldsymbol{x})$ of a surface point. I thus minimize the difference between the actual intensity $L(\boldsymbol{x})$ and the prediction from my model to obtain the optimal $\boldsymbol{\beta}$ and IR diffuse albedo $\boldsymbol{\rho}$:

$$\min_{\boldsymbol{\beta},\boldsymbol{\rho}} \sum_{\boldsymbol{x}\in P} \left(L(\boldsymbol{x}) - \left(\kappa \frac{\boldsymbol{n_x} \cdot \boldsymbol{l_x}}{d_{\boldsymbol{x}}^2} \left(\frac{\rho(\boldsymbol{x})}{\pi} + f_s(\boldsymbol{v}_x, \boldsymbol{l}_x; \boldsymbol{\beta}) \right) \right)^{\gamma} \right)^2, \tag{3.5}$$

where the set P collates pixels over all frames. The resulting $\boldsymbol{\beta}$ is used to describe f_s, and thus S.

The full optimization involves estimating hundreds of thousands of parameters from millions of pixels. So, in practice, I exploit the interactive nature of the scanning process to have the user choose a small subset of frames over which to optimize. These frames are chosen such that a specular highlight is observed in the IR channel and each material is captured at least once. When a material is captured in only one frame, Eq. 3.5 is solved with spatially constant diffuse albedo. Choosing reference views of convex surface regions improves the specularity estimation by reducing the impact of interreflections. The optimization is solved with Levenberg-Marquardt method [160]. (Fig. 3.3)

Anisotropic Surfaces Many common reflective surfaces (especially wood and metal) reflect light anisotropically. I therefore extend my approach to model anisotropy for a limited class of surfaces (planes and cylinders) that occur frequently in man-made scenes.

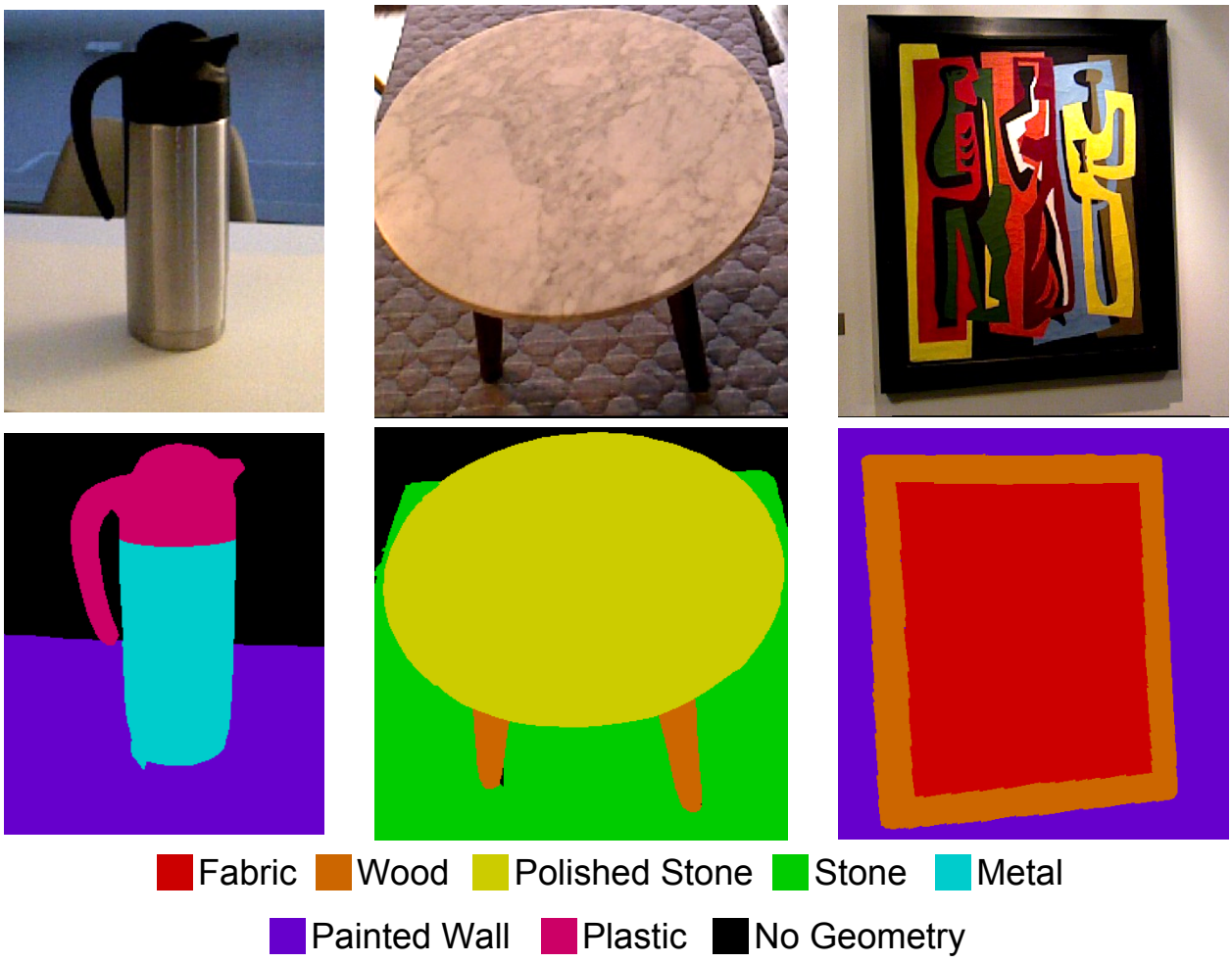
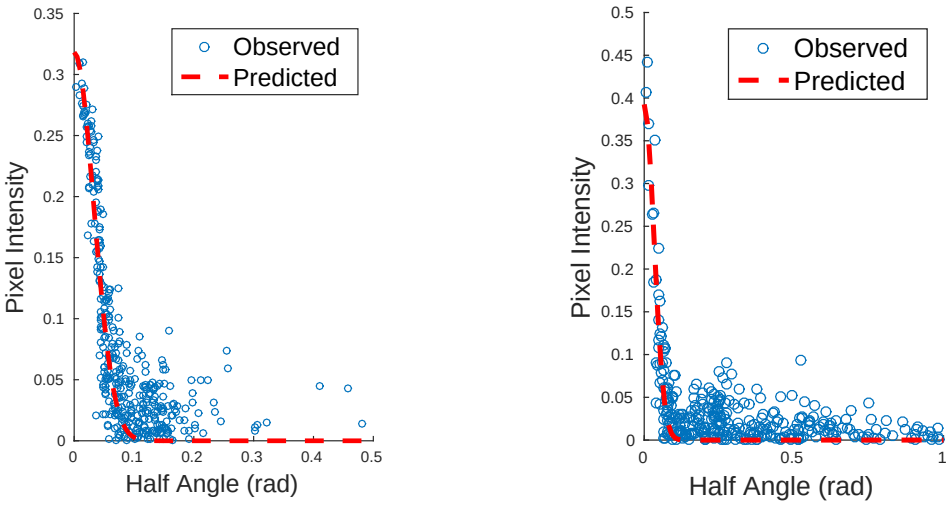


Figure 3.2: Material segmentation results. The first row shows pictures of target scenes, and the second row shows 3D material segmentation results. Note that the segmentation algorithm works on the reconstructed mesh instead of a single image.



(a) *Dog* Scene

(b) *Corncho* Scene

Figure 3.3: Specular BRDF fitting results for two scenes, plotting specular components versus half angles of pixel observations along with the fitted prediction curves.

For an anisotropic surface, solving Eq. (3.5) requires estimating tangent and binormal vectors at each surface point. Among other methods in the literature [78, 237], [106] suggests a photometric approach to estimate the per-point tangent vector by projecting half vectors onto the tangent plane and finding the direction that maximizes symmetry. Applying this technique in our context means densely observing each surface point from many different viewpoints, which is infeasible in a casual hand-held scanning session. I therefore assume that the tangent vector is shared across a material segment, i.e., the surface is planar or cylindrical.

Each pixel $\boldsymbol{x}$ in an IR frame has an associated half-vector $\boldsymbol{h_x}$ and normal vector $\boldsymbol{n_x}$ of the surface point represented in global coordinates and intensity $L(\boldsymbol{x})$. To determine a single representative tangent plane P_R, I choose a pixel with the smallest half-angle (angle between half-vector and local normal) as a reference, with its normal $\boldsymbol{n}_R$. Half-vectors of all other

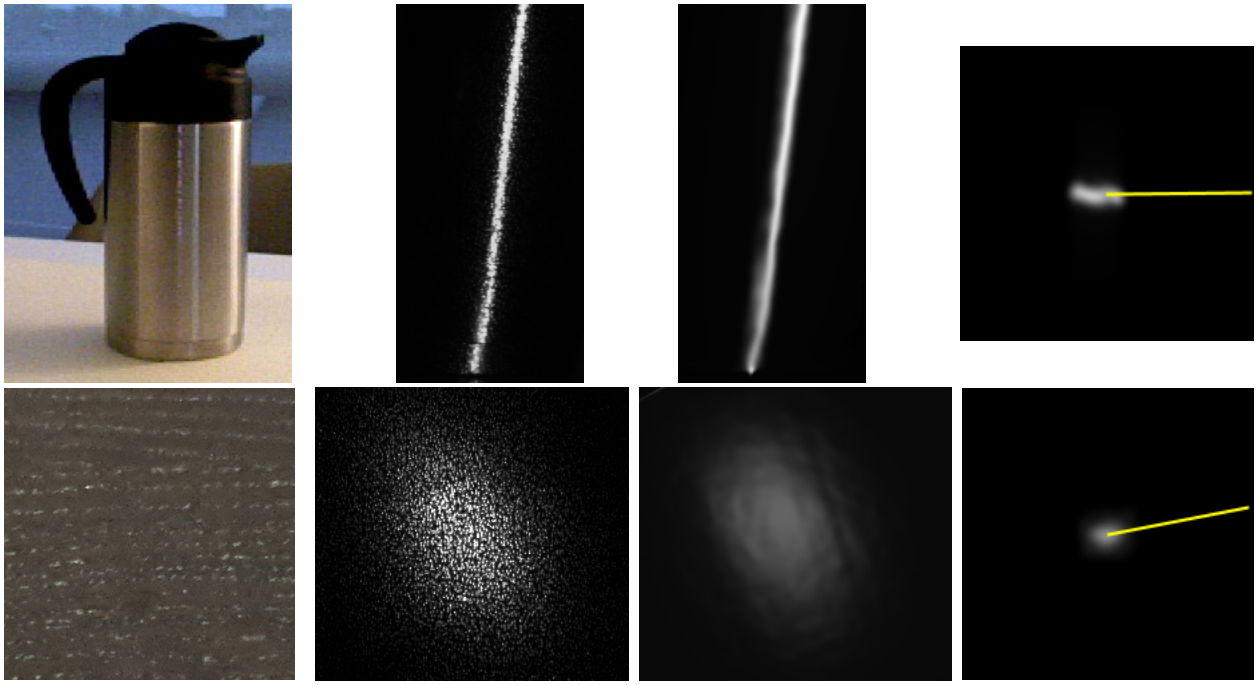


Figure 3.4: Anisotropic surface fitting results. From left to right: Photo of the target surface (brushed metal cylinder and planar wood surface); Image captured from IR camera; our rendering from the same camera pose; Tangent plane of the reference point where half-vectors of observations are projected (the yellow line is the optimal tangent vector that maximizes symmetry). Best viewed digitally.

pixels relative to its local normal are projected onto the tangent plane of the reference point:

$$P_R(\text{Proj}_{\boldsymbol{n}_R}(\boldsymbol{n}_R + \boldsymbol{h}_{\boldsymbol{x}} - \boldsymbol{n}_{\boldsymbol{x}})) \leftarrow L(\boldsymbol{x}). \tag{3.6}$$

The discretized tangent plane is then smoothed with Gaussian filtering to get the dense BRDF slice as in Figure 3.4. I then use the Nelder-Mead Simplex Method [173] to find the tangent and binormal vectors that maximize the symmetry [106]. Given the estimated tangent vector, the BRDF parameters $\boldsymbol{\beta}$ for the anisotropic surface are computed through Eq. (3.5), but with spatially constant diffuse albedo (Fig. 3.4).

3.3 Computing the Diffuse Component

In this section, I describe how to obtain high quality scene texture in real-time and iteratively remove baked-in specular highlights as a post-processing step.

Real-time High Resolution Texture (HRT) Fusion High quality texture is essential to the perceived realism of scene appearance. While interactive RGBD scanning has seen great progress on delivering quality geometry [48, 243], the state-of-the-art systems still produce low-resolution reconstructions with blurring and ghosting artifacts.

Recently, global pose optimization approaches [273] and its patch-based variants [21] have produced impressive texture mapping results. However, these methods work offline and thus cannot provide interactive visual feedback to users on which part of the current model is missing texture or needs close-up scanning.

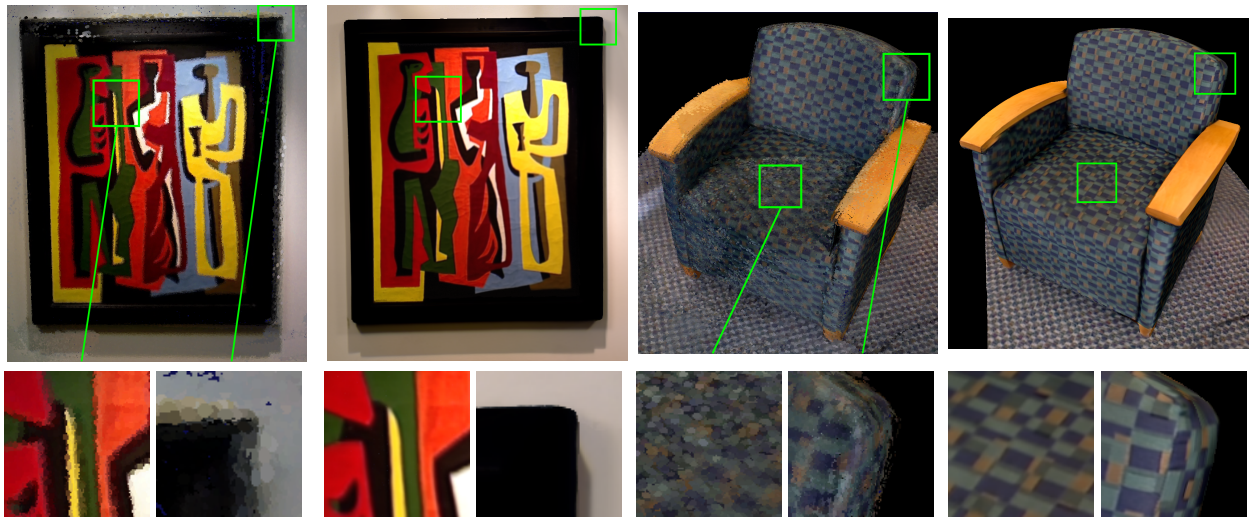
I show that a high resolution texture representation combined with GPU-accelerated dense photometric pose optimization [129, 243] can greatly improve the real-time scanning quality (Figure 3.5). I also propose a gradient-based objective function and generalized specular highlight removal algorithm for handling non-Lambertian surfaces.

3.3.1 Preliminaries

I adopt Kinect Fusion [175] to obtain the scene geometry and refine the extracted mesh using the method of [114]. I denote the final mesh, vertex set, and vertex normal set as $\mathcal{M}$, $\mathcal{V}$, and $\mathcal{N}$, respectively.

The focal length $\boldsymbol{f}$ and principal point $\boldsymbol{c}$ of the camera are estimated to form a calibration matrix K. I define a projection operator $\pi : \mathbb{R}^4 \mapsto \mathbb{R}^2$ from a 3D homogeneous point to a 2D point on image plane: $\pi(\boldsymbol{p}) := \left(\frac{p_x}{p_z} f_x + c_x, \frac{p_y}{p_z} f_y + c_y\right)$. The rigid body transformation of the camera at time i relative to the first frame is:

$$T_i = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} : \quad R \in \mathbb{SO}3, t \in \mathbb{R}^3, \tag{3.7}$$



(a) [243] (b) My Method (c) [243] (d) My Method

Figure 3.5: Comparison of texture quality with a state-of-the-art method [243]. The first and third columns show reconstruction results of [243], while the second and fourth show my results. The second row shows magnified patches of the green boxes. Best viewed digitally.

where a 3D point $\boldsymbol{p}$ in local camera coordinates can be transformed to a point in global coordinates as $\boldsymbol{p}' = T_i\boldsymbol{p}$. I calibrate low polynomial radial and decentering distortion [26] parameters, and each RGB frame is undistorted accordingly. All RGB images are compensated for vignetting after radiometric calibration [131].

3.3.2 Texture Representation

I represent the color texture of the model as a 2D texture atlas [156] A that stores RGB values and associated weights. I parameterize the mesh $\mathcal{M}$ by simply laying each triangle on A without overlap. For each pixel $\boldsymbol{u}$ on the texture plane, its 3D position can be computed through barycentric interpolation, denoted as $\boldsymbol{u}^{3d}$.

3.3.3 Rendering for Appearance Prediction

Using the rasterization pipeline, I can render a high resolution prediction of the scene appearance $\mathcal{R}_i$ along with a depth map Z_i and normal map into a camera with estimated pose T_i. This high resolution rendering provides constraints for pose optimization for the upcoming input frame.

3.3.4 Texture Update

Given its estimated pose T_i, each RGB frame $\mathcal{I}_i$ is densely fused into the texture atlas. I compute the color input of a texture pixel $\boldsymbol{u} \in \Omega$ by projecting the corresponding 3D point $\boldsymbol{u}^{3d}$ onto the bilinearly interpolated image $\mathcal{I}_i$ with depth testing. Specifically, let $\boldsymbol{x}$ be a point on the image plane of $\mathcal{I}_i$ obtained from perspective projection of $\boldsymbol{u}^{3d}$. The color measurement $\mathcal{I}_i(\boldsymbol{x})$ is blended into $A(\boldsymbol{u})$ through weighted averaging with weight $w_i(\boldsymbol{x})$ associated with $\boldsymbol{x}$ at time i. I assign high weights to reliable and important color observations:

$$w_i(\boldsymbol{x}) = m_i z_i(\boldsymbol{x}) s_i(\boldsymbol{x}). \tag{3.8}$$

Here, m_i accounts for motion blur and rolling shutter effects that occur during fast camera motions measured by the L^2 distance between translation and rotation components of T_{i-1}

and T_i. $z_i(\boldsymbol{x})$ rejects pixels on depth discontinuities where estimated geometry is unreliable. Finally, measurements close to the surface and near perpendicular to it provide sharper texture, yielding the importance term $s_i(\boldsymbol{x})$:

$$s_i(\boldsymbol{x}) = \frac{\boldsymbol{n_x} \cdot \boldsymbol{v_x}}{Z_i^2(\boldsymbol{x})}. \tag{3.9}$$

3.3.5 Frame-to-Model Photometric Pose Refinement

To estimate the camera pose T_i, I use a frame-to-model RGBD dense alignment approach, minimizing the photometric error between the live frame $\mathcal{I}_i$ and the previous frame's rendering $\mathcal{R}_{i-1}$ on top of traditional dense point-plane Iterative Closest Point algorithm [175].

Unlike previous real-time RGBD tracking methods [129, 243], my 2D texture atlas can store higher frequency photometric details compared to commonly used surfels or volumetric textures, providing tighter pose constraints.

Dense RGBD alignment techniques aim to find a relative rigid body transformation $T^\circ = T_i^{-1}T_{i-1}$ that minimizes photometric error when $\mathcal{I}_i$ is warped into the reference image $\mathcal{R}_{i-1}$ with the corresponding geometry represented by the reference depth map Z_{i-1}. For a pixel $\boldsymbol{x}$ on image plane Λ of $\mathcal{R}_{i-1}$, I get a 3D point $\boldsymbol{p_x}$ in local coordinates through back-projection: $\boldsymbol{p_x} = Z_{i-1}(\boldsymbol{x})K^{-1}\boldsymbol{x}$.

I can then look up how this point appears in the live input image: $\mathcal{I}_i(\pi(T^\circ \boldsymbol{p_x}))$. Modern dense alignment techniques then minimize greyscale photometric error between $\mathcal{R}_{i-1}(\boldsymbol{x})$ and $\mathcal{I}_i(\pi(T^\circ \boldsymbol{p_x}))$ under a quadratic loss.

For my application, however, maximizing photoconsistency does not necessarily results in accurate camera tracking since specular surfaces can appear significantly different across viewpoints. I find that applying a high-pass filter to images improves tracking accuracy and therefore modify the objective to operate on the gradient of the predicted and live images. The optimal transformation $\hat{T}^\circ$ is then:

$$\hat{T}^\circ = \arg\min_{T^\circ} \sum_{\boldsymbol{x}\in\Lambda} \left(||\nabla \mathcal{R}_{i-1}^{\mathcal{G}}(\boldsymbol{x})|| - ||\nabla \mathcal{I}_i^{\mathcal{G}}(\pi(T^\circ \boldsymbol{p_x}))|| \right)^2, \tag{3.10}$$

where the superscript $\mathcal{G}$ denotes greyscale images.

The intuition is that the high-pass filter reduces each specular highlight (which can be large) to its boundary, and the boundary tends to be lower frequency than the underlying diffuse texture, as the specular BRDF acts as a low-pass filter on the incident illumination [195].

3.3.6 Specular Highlight Removal

The resulting texture has specular highlights "baked-in", which need to be removed to obtain D. This problem can be posed as a layer separation problem [46, 258], to separate the stationary diffuse texture from the view-dependent specular layer. Specifically, [225] proposes the *min-composite*, i.e., minimum intensity values across all viewpoints, to approximate (via least upper bound) the diffuse layer.

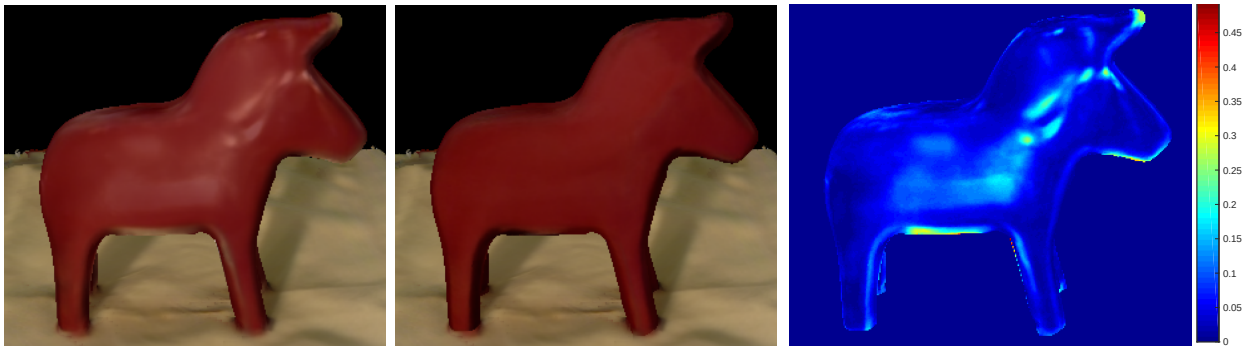
I use Iteratively Reweighted Least Squares (IRLS) [85] to robustly compute the *min-composite*. For a given pixel on the texture map, computing its weighted average $\hat{q}$ as in Section 3.3.4 is equivalent to finding the solution of a weighted least squares problem $\arg\min_{\hat{q}} \sum_i w_i \left(q_i - \hat{q}\right)^2$, where q_i is the ith intensity input, w_i is the weight of the input computed in equation (3.8). IRLS filters out the highlights by down-weighting bright outliers in each iteration:

$$\hat{q}^{t+1} = \arg\min_{\hat{q}} \sum_i \mu_{\tau,\upsilon}(\hat{q}^t, q_i) w_i \left(q_i - \hat{q}\right)^2, \tag{3.11}$$

$$\mu_{\tau,\upsilon}(\hat{q}^t, q_i) = \begin{cases} 1 & q_i \leq \hat{q}^t + \tau \\ \exp(-\frac{(\hat{q}^t + \tau - q_i)^2}{\upsilon^2}) & q_i > \hat{q}^t + \tau. \end{cases} \tag{3.12}$$

In practice, one or two iterations is sufficient. (Figure 3.6)

While this approach is effective for surfaces with significant diffuse components, it fails for metallic surfaces with negligible diffuse reflection. Fortunately, I can leverage the IR BRDF measurements (Sec.3.2) to identify such surfaces; I set D to zero for any surface whose estimated diffuse albedo is less than 0.03 and specular albedo greater than 0.15.



(a) Before (b) After (c) Difference

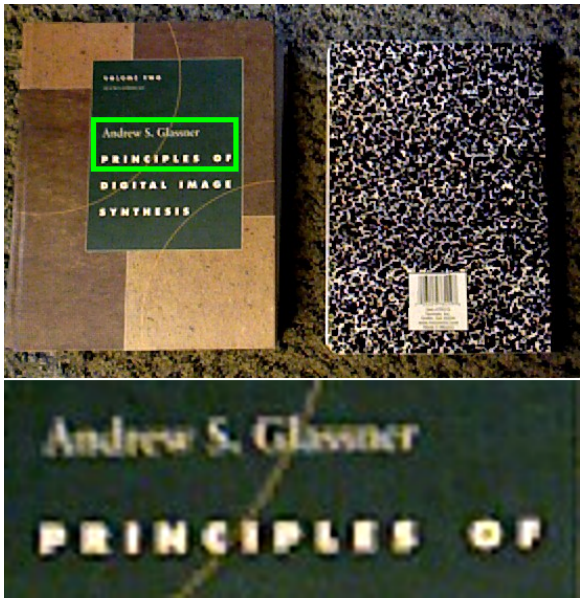
Figure 3.6: Specular highlights are removed from (a) by computing a *min-composite* via Iteratively Reweighted Least Squares. (b) shows the estimated D and (c) the difference heat map.

3.4 Experiments

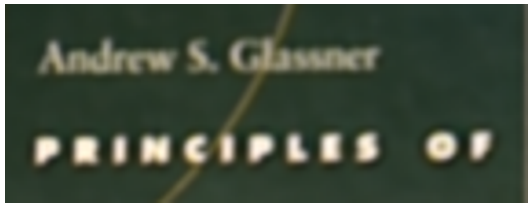
I conducted qualitative and quantitative experiments to assess my method. All experiments use a calibrated Primesense Carmine RGBD sensor capturing 640x480 video at 30Hz for RGB and IR channels, synchronized with the same size depth video. Exposure time was set to be constant within each sequence. The system runs on a laptop with a NVIDIA GTX 980M GPU and an Intel i7 CPU.

3.4.1 HRT Tracking Evaluation

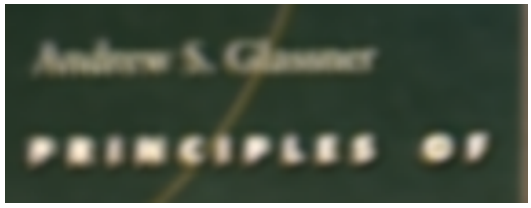
I evaluate the performance of my texture fusion method by measuring photometric error between the rendered and ground truth images for a held-out sample of frames. For comparison, I replace my HRT tracking with existing real-time camera tracking methods and evaluate the photometric error on a video of a highly textured Lambertian scene. I test RGBD tracking methods using a combination of frame-to-model Iterative Closest Point (ICP) [175], frame-to-frame (f2f) RGBD tracking [129, 220], and frame-to-model RGBD tracking (f2m) [243]. I did not implement loop closure techniques introduced in the state-of-the-art approaches such



(a) Ground-truth



(b) Ours



(c) Frame-to-Model [243]

(d) Frame-to-Frame [129]

Figure 3.7: Ground-truth image and predicted rendering of 3D models reconstructed from different tracking methods. The visualization shows magnified patches of the green box.

as that of [48, 243] as my target scenes are of smaller scale where tracking drift is limited.

Figure 3.7 compares the quality of texture fused using camera poses provided by each algorithm. My high resolution frame-to-model tracking yields the sharpest result.

For a quantitative measure, I adopt the evaluation method introduced by [234] and compute the pixel-wise root-mean-square-error (RMSE) and 1-NCC error (for different patch sizes) between the ground-truth and rendered texture reconstruction. I refer to [234] for the precise procedure. Results can be found in Table 3.2.

Error/Method	ICP	f2f	f2m	ours
RMSE	0.1361	0.1177	0.1184	**0.0902**
1-NCC(3x3)	0.7220	0.6124	0.6521	**0.4692**
1-NCC(5x5)	0.6510	0.5430	0.5689	**0.3877**
1-NCC(7x7)	0.5999	0.5043	0.5138	**0.3491**

Table 3.2: Photometric Errors of Tracking Methods

3.4.2 Surface Light Field Rendering Results

To render a reconstructed model, I use D represented as a texture map and implement a custom shader to evaluate S with a HDR environment lighting which is captured from a consumer-grade 360 degree camera by taking several photos with varying exposure times [56]. Figure 3.8 compares the reconstructed surface light field of test scenes with real photographs.

These results demonstrate that my method can successfully recover surface light fields under a wide range of geometry, material, and lighting variations. The *Rooster* scene features challenging high-frequency textures and shiny ceramic surfaces, captured under office florescent lights. My method can accurately reproduce sharp diffuse textures and specular highlights. The *Corncho* scene contains a bumpy specular vinyl surface captured near a large window. Notice the faithful soft-shadows and strong interreflections on the white floor expressed in the renderings. The *Dog* scene has significant self-occlusions. A bright directional lamp illuminates the object from a short distance, inducing strong specularities and sharp shadows. My system faithfully reconstructs these effects. The *Bottle* scene contains a highly anisotropic brushed metal cylinder and a glossy black plastic cap. The results show that my system successfully recovers the vertically elongated specular highlights on the metal surface. Also notice that the relative specular reflectance of the two glossy materials are estimated correctly.

3.5 Limitations and Conclusions

My proposed system is unable to model near perfect mirrors or surfaces with micro-structure, due to inaccurate sensor shape measurement; e.g., tiny bumps on the *Corncho* model were not captured, causing noticeable difference in the sharpness of the reflection. Moreover, my surface light field formulation does not model colored metals such as bronze and omits specular interreflections.

I presented the first end-to-end system for computing surface light field object models using a hand-held, commodity RGBD sensor. I leverage a novel factorization of the surface light field that simplifies capture requirements, yet enables high quality results for a wide range of materials, including anisotropic ones. My approach captures global illumination effects (like shadows and interreflections) and high resolution textures that greatly improve realism relative to prior interactive RGBD scanning methods.

Figure 3.8: Ground truth images and renderings. First column: Ground truth photographs. Second column: Synthetic rendering of the same pose. Third and fourth column: Rendering of different camera poses.

Chapter 4

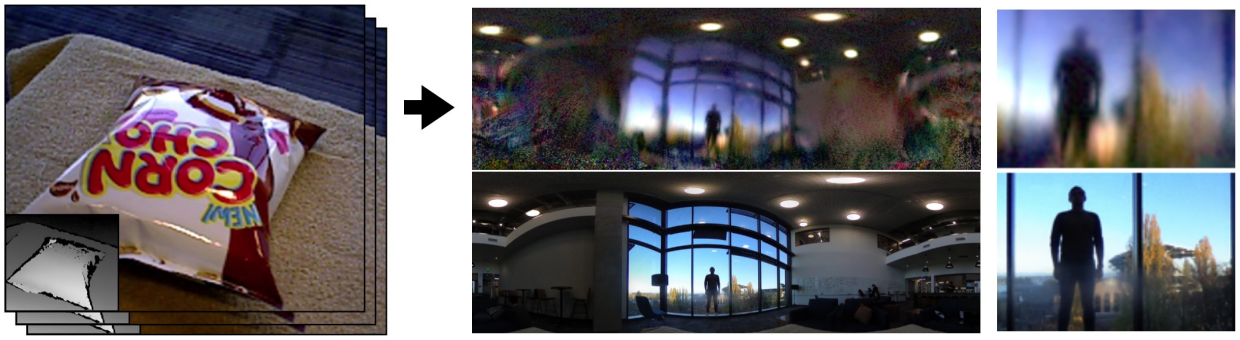
JOINT RECOVERY OF SCENE LIGHTING AND APPEARANCE

In Chapter 3, I introduced a user-friendly system for appearance reconstruction with a hand-held camera. The proposed approach, however, has two limitations. The first is its use of a 360° camera for illumination capture, which could be a a barrier to users who may not have such a specialized device. Second, the system is unable to model near-perfect mirrors due to the measurement inaccuracies. Finally, the system fails to model complex surface scattering effects (e.g., Fresnel Effect [206]) due to the use of a simple parametric BRDF model.

In this chapter, I tackle both of these issues by addressing the dual problems of novel view synthesis and environment reconstruction from hand-held RGBD sensors. Specifically, the proposed approach automatically recovers scene environment from the specular highlights (Figure 5.10), and is able to handle complex surface scattering effects such as mirror-like reflections and interreflections via neural-network-based rendering.

In their *visual microphone* work, Davis et al. [53] showed how sound and even conversations can be reconstructed from the minute vibrations visible in a bag of chips. Inspired by their work, I show that the same bag of chips can be used to reconstruct an image of the environment. Instead of high speed video, however, I operate on RGBD video, as obtained with commodity depth sensors.

Visualizing the environment is closely connected to the problem of modeling the scene that reflects that environment. I solve both problems; beyond visualizing the room, I seek to predict how the objects and scenes appear from any new viewpoint — i.e., to virtually explore the scene as if you were there. As discussed in the earlier chapters, this view synthesis problem



(a) Input images **(b)** Estimated environment (top), ground truth (bottom) **(c)** Zoom-in

Figure 4.1: From a hand-held RGBD sequence of an object (a), I reconstruct an image of the surrounding environment (b, top) that closely resembles the real environment (b, bottom), entirely from the specular reflections. Note the reconstruction of fine details (c) such as a human figure and trees with fall colors through the window. I use the recovered environment for novel view rendering.

has a large literature in computer vision and graphics, but several open problems remain. Chief among them are 1) specular surfaces, 2) inter-reflections, and 3) simple capture. In this chapter I address all three of these problems, based on the framework of *surface light fields* [245].

My environment reconstructions, which I call *specular reflectance maps* (*SRMs*), represent the distant environment map convolved with the object's specular BRDF. In cases where the object has strong mirror-like reflections, this SRM provides sharp, detailed features like the one seen in Fig. 5.10. As most scenes are composed of a mixture of materials, each scene has multiple basis SRMs. I therefore reconstruct a global set of SRMs, together with a weighted material segmentation of scene surfaces. Based on the recovered SRMs, together with additional physically motivated components, I build a neural rendering network capable of faithfully approximating the true surface light field.

Like the method in the last chapter, a major contribution of this approach is the capability of reconstructing a surface light field with the same input needed to compute shape alone [175] using an RGBD camera. Additional contributions of my approach include the ability

to operate on regular (low-dynamic range) imagery, and applicability to general, non-convex, textured scenes containing multiple objects and both diffuse and specular materials.

4.1 Related Work

I review related work in environment lighting estimation and novel-view synthesis approaches for modeling specular surfaces.

4.1.1 Environment Estimation

Single-View Estimation The most straightforward way to capture an environment map (image) is via light probes (e.g., a mirrored ball [55]) or taking photos with a 360° camera [184]. Human eye balls [179] can even serve as light probes when they are present. For many applications, however, light probes are not available and I must rely on existing cues in the scene itself.

Other methods instead study recovering lighting from a photo of a general scene. Because this problem is severely under-constrained, these methods often rely on human inputs [124, 272] or manually designed "intrinsic image" priors on illumination, material, and surface properties [125, 13, 12, 20, 149].

Recent developments in deep learning techniques facilitate data-driven approaches for single view estimation. [76, 75, 218, 139] learn a mapping from a perspective image to a wider-angle panoramic image. Other methods train models specifically tailored for outdoor scenes [105, 104]. Because the single-view problem is severely ill-posed, most results are plausible but often non-veridical. Closely related to my work, Georgoulis [77] reconstruct higher quality environment images, but under very limiting assumptions; textureless painted surfaces and manual specification of materials and segmentation.

Multi-View Estimation For the special case of planar reflectors, layer separation techniques [225, 213, 258, 93, 91, 112, 271] enable high quality reconstructions of reflected environments, e.g., from video of a glass picture frame. Inferring reflections for general, curved

surfaces is dramatically harder, even for humans, as the reflected content depends strongly and nonlinearly on surface shape and spatially-varying material properties,

A number of researchers have sought to recover *low-frequency* lighting from multiple images of curved objects. [278, 180, 155] infer spherical harmonics lighting (following [195]) to refine the surface geometry using principles of shape-from-shading. [197] jointly optimizes low frequency lighting and BRDFs of a reconstructed scene. While suitable for approximating light source directions, these models don't capture detailed images of the environment.

Wu [246], like us, use a hand-held RGBD sensor to recover lighting and reflectance properties. But the method can only reconstruct a single, floating, convex object, and requires a black background. Dong [61] produces high quality environment images from a video of a single rotating object. This method assumes a laboratory setup with a mechanical rotator, and manual registration of an accurate geometry to their video. Similarly, Xia [254] use a robotic arm with calibration patterns to rotate an object. The authors note highly specular surfaces cause trouble, thus limiting their real object samples to mostly rough, glossy materials. In contrast, my method operates with a hand-held camera for a wide-range of multi-object scenes, and is designed to support specularity.

4.1.2 Novel View Synthesis

Here I focus on methods capable of modeling *specular reflections* from new viewpoints.

Image-based Rendering Light field methods [84, 142, 36, 245, 52] enable highly realistic views of specular surfaces at the expense of laborious scene capture from densely sampled viewpoints. Chen [34] regresses surface light fields with neural networks to reduce the number of required views, but requires samples across a full hemisphere captured with a mechanical system. Park [184] avoid dense hemispherical view sampling by applying a parametric BRDF model, but assume known lighting.

Learning-based Rendering Recent work applies convolutional neural networks (CNN) to image-based rendering [72, 165]. Hedman [102] replaced the traditional view blending heuristics of IBR systems with a CNN-learned blending weights. Still, novel views are composed of existing, captured pixels, so unobserved specular highlights cannot be synthesized. More recently, [3, 230] enhance the traditional rendering pipeline by attaching learned features to 2D texture maps [230] or 3D point clouds [3] and achieve high quality view synthesis results. The features are nonetheless specifically optimized to fit the input views and do not extrapolate well to novel views. Recent learning-based methods achieve impressive local (versus hemispherical) light field reconstruction from a small set of images [166, 219, 43, 119, 275].

More recently, the coordinate based continuous implicit representation using neural networks is used for appearance modeling [167, 161]. These methods typically overfit a single scene into a single network by modeling the pixel radiance via the integration of alpha-blended radiance values in the ray direction (the technique referred to as Volume Rendering [141]). Similar to the CNN-based methods, these neural implicit methods are unable to extrapolate to a new viewpoint as it primarily focuses on interpolating between the input viewpoints.

BRDF Estimation Methods Another way to synthesize novel views is to recover intrinsic surface reflection functions, known as BRDFs [177]. In general, recovering the surface BRDFs is a difficult task, as it involves inverting the complex light transport process. Consequently, existing reflectance capture methods place limits on operating range: e.g., an isolated single object [246, 61], known or controlled lighting [184, 57, 140, 276, 257], single view surface (versus a full 3D mesh) [82, 144], flash photography [2, 138, 172], or spatially constant material [163, 130].

Interreflections Very few view synthesis techniques support interreflections. Modeling general multi-object scene requires solving for global illumination (e.g. shadows or interreflections), which is difficult and sensitive to imperfections of real-world inputs [8]. Sim-

ilarly, Lombardi [150] model multi-bounce lighting but with noticeable artifacts and limit their results to mostly uniformly textured objects. Zhang [269] require manual annotations of light types and locations.

4.2 Technical Approach

My proposed system takes a video and 3D mesh of a static scene (obtained via Newcombe [175]) as input and automatically reconstructs an image of the environment along with a scene appearance model that enables novel view synthesis. My approach excels at specular scenes, and accounts for both specular interreflection and Fresnel effects. A key advantage of my approach is the use of easy, casual data capture from a hand-held camera; I reconstruct the environment map and a surface light field with the same input needed to reconstruct the geometry alone, e.g., using [175].

Section 4.2.1 formulates surface light fields [245] and define the specular reflectance map (SRM). Section 4.2.2 shows how, given geometry and diffuse texture as input, I can jointly recover SRMs and material segmentation through an end-to-end optimization approach. Lastly, Section 4.2.3, describes a *scene-specific* neural rendering network that combines recovered SRMs and other rendering components to synthesize realistic novel-view images, with interreflections and Fresnel effects.

4.2.1 Surface Light Field Formulation

I model scene appearance using the concept of a surface light field [245], which defines the color radiance of a surface point in every view direction, given approximate geometry, denoted $\mathcal{G}$ [175].

Formally, the surface light field, denoted SL, assigns an RGB radiance value to a ray coming from surface point $\boldsymbol{x}$ with outgoing direction $\boldsymbol{\omega}$: $SL(\mathbf{x}, \boldsymbol{\omega}) \in \text{RGB}$. As is common [188, 241], I decompose SL into diffuse (view-independent) and specular (view-dependent) components:

$$SL(\mathbf{x}, \boldsymbol{\omega}) \approx D(\mathbf{x}) + S(\mathbf{x}, \boldsymbol{\omega}). \tag{4.1}$$

I compute the diffuse texture D for each surface point as the minimum intensity of across different input views following [225, 184]. Because the diffuse component is view-independent, I can then render it from arbitrary viewpoints using the estimated geometry. However, textured 3D reconstructions typically contain errors (e.g., silhouettes are enlarged, as in Fig. 4.2), so I refine the rendered texture image using a neural network (Sec. 4.2.2).

For the specular component, I define the specular reflectance map (SRM) (also known as *lumisphere* [245]) and denoted SR, as a function that maps a reflection ray direction $\boldsymbol{\omega}_r$, defined as the vector reflection of $\boldsymbol{\omega}$ about surface normal $\boldsymbol{n_x}$ [245] to specular reflectance (i.e., radiance): $SR(\boldsymbol{\omega}_r) : \Omega \mapsto RGB$, where Ω is a unit hemisphere around the scene center. This model assumes distant environment illumination, although I add support for specular interreflection later in Sec. 4.2.3. Note that this model is closely related to prefiltered environment maps [126], used for real-time rendering of specular highlights.

Given a specular reflectance map SR, I can render the specular image S from a virtual camera as follows:

$$S(\boldsymbol{x}, \boldsymbol{\omega}) = V(\boldsymbol{x}, \boldsymbol{\omega}_r; \mathcal{G}) \cdot SR(\boldsymbol{\omega}_r), \tag{4.2}$$

where $V(\boldsymbol{x}, \boldsymbol{\omega}_r; \mathcal{G})$ is a shadow (visibility) term that is 0 when the reflected ray $\boldsymbol{\omega}_r := \boldsymbol{\omega} - 2(\boldsymbol{\omega} \cdot \boldsymbol{n_x})\boldsymbol{n_x}$ from $\boldsymbol{x}$ intersects with known geometry $\mathcal{G}$, and 1 otherwise.

An SRM contains distant environment lighting convolved with a particular specular BRDF. As a result, a single SRM can only accurately describe one surface material. In order to generalize to multiple (and spatially varying) materials, I modify Eq. (4.2) by assuming the material at point $\boldsymbol{x}$ is a linear combination of M basis materials [82, 4, 277]:

$$S(\boldsymbol{x}, \boldsymbol{\omega}) = V(\boldsymbol{x}, \boldsymbol{\omega}_r; \mathcal{G}) \cdot \sum_{i=1}^{M} W_i(\boldsymbol{x}) \cdot SR_i(\boldsymbol{\omega}_r), \tag{4.3}$$

where $W_i(x) \geq 0$, $\sum_{i=1}^{M} W_i(\boldsymbol{x}) = 1$ and M is user-specified. For each surface point $\boldsymbol{x}$, $W_i(\boldsymbol{x})$ defines the weight of material basis i. I use a neural network to approximate these weights in image-space, as described next.

(a) Diffuse image D_P (b) Refined Diffuse image D'_P

Figure 4.2: The role of diffuse network u_{ϕ} to correct geometry and texture errors of RGBD reconstruction. The bottle geometry in image (a) is estimated larger than it actually is, and the background textures exhibit ghosting artifacts (faces). The use of the refinement network corrects these issues (b). Best viewed digitally.

4.2.2 Estimating SRMs and Material Segmentation

Given scene shape $\mathcal{G}$ and photos from known viewpoints as input, I now describe how to recover an optimal set of SRMs and material weights.

Suppose I want to predict a view of the scene from camera P at a pixel $\boldsymbol{u}$ that sees surface point $\boldsymbol{x_u}$, given known SRMs and material weights. I render the diffuse component $D_P(\boldsymbol{u})$ from the known diffuse texture $D(\boldsymbol{x_u})$, and similarly the blending weight map $W_{P,i}$ from W_i for each SRM using standard rasterization. A reflection direction image $R_P(\boldsymbol{u})$ is obtained by computing per-pixel $\boldsymbol{\omega}_r$ values. I then compute the specular component image S_P by looking up the reflected ray directions R_P in each SRM, and then combining the radiance values using $W_{P,i}$:

$$S_P(\boldsymbol{u}) = V(\boldsymbol{u}) \cdot \sum_{i=1}^{M} W_{P,i}(\boldsymbol{u}) \cdot SR_i(R_P(\boldsymbol{u})), \tag{4.4}$$

where $V(\boldsymbol{u})$ is the visibility term of pixel $\boldsymbol{u}$ as used in Eq. (4.3). Each SR_i is stored as a 2D panorama image of resolution 500 x 250 in spherical coordinates.

Now, suppose that SRMs and material weights are unknown; the optimal SRMs and

combination weights minimize the energy $\mathcal{E}$ defined as the sum of differences between the real photos G and the rendered composites of diffuse and specular images D_P, S_P over all input frames $\mathcal{F}$:

$$\mathcal{E} = \sum_{P \in \mathcal{F}} \mathcal{L}_1(G_P, D_P + S_P), \tag{4.5}$$

where $\mathcal{L}_1$ is pixel-wise $L1$ loss.

While Eq. (4.5) could be minimized directly to obtain $W_{P,i}$ and SR_i, two factors introduce practical difficulties. First, specular highlights tend to be sparse and cover a small percentage of specular scene surfaces. Points on specular surfaces that don't see a highlight are difficult to differentiate from diffuse surface points, thus making the problem of assigning material weights to surface points severely under-constrained. Second, captured geometry is seldom perfect, and misalignments in reconstructed diffuse texture can result in incorrect SRMs. In the remainder of this section, I describe my approach to overcome these limiting factors.

Material weight network. To address the problem of material ambiguity, I pose the material assignment problem as a statistical pattern recognition task. I compute the 2D weight maps $W_{P,i}(\boldsymbol{u})$ with a convolutional neural network $w_{\boldsymbol{\theta}}$ that learns to map a diffuse texture image patch to the blending weight of ith material: $W_{P,i} = w_{\boldsymbol{\theta}}(D_P)_i$. This network learns correlations between diffuse texture and material properties (i.e., shininess), and is trained on each scene by jointly optimizing the network weights and SRMs to reproduce the input images.

Since $w_{\boldsymbol{\theta}}$ predicts material weights in image-space, and therefore per view, I introduce a view-consistency regularization function $\mathcal{V}(W_{P_1}, W_{P_2})$ penalizing the pixel-wise $L1$ difference in the predicted materials between a pair of views when cross-projected to each other (i.e., one image is warped to the other using the known geometry and pose).

Diffuse refinement network. Small errors in geometry and calibration, as are typical in scanned models, cause misalignment and ghosting artifacts in the texture reconstruction D_P. Therefore, I introduce a refinement network $u_{\boldsymbol{\phi}}$ to correct these errors (Fig. 4.2). I

replace D_P with the refined texture image: $D'_P = u_{\boldsymbol{\phi}}(D_P)$. Similar to the material weights, I penalize the inconsistency of the refined diffuse images across viewpoints using $\mathcal{V}(D'_{P_1}, D'_{P_2})$. Both networks $w_{\boldsymbol{\theta}}$ and $u_{\boldsymbol{\phi}}$ follow the encoder-decoder architecture with residual connections [116, 100], while $w_{\boldsymbol{\theta}}$ has lower number of parameters. I refer readers to supplementary for more details.

Robust Loss. Because a pixel-wise loss alone is not robust to misalignments, I define the image distance metric $\mathcal{L}$ as a combination of pixel-wise $L1$ loss, perceptual loss $\mathcal{L}_p$ computed from feature activations of a pretrained network [35], and adversarial loss [83, 110]. My total loss, for a pair of images I_1, I_2, is:

$$\begin{aligned}\mathcal{L}(I_1, I_2; d) = \lambda_1 \mathcal{L}_1(I_1, I_2) + \lambda_p \mathcal{L}_p(I_1, I_2) \\ + \lambda_G \mathcal{L}_G(I_1, I_2; d),\end{aligned} \tag{4.6}$$

where d is the discriminator, and $\lambda_1 = 0.01$, $\lambda_p = 1.0$, and $\lambda_G = 0.05$ are balancing coefficients. The neural network-based perceptual and adversarial loss are effective because they are robust to image-space misalignments caused by errors in the estimated geometry and poses.

Finally, I add a sparsity term on the specular image $||S_P||_1$ to regularize the specular component from containing colors from the diffuse texture.

Combining all elements, I get the final loss function:

$$\begin{aligned}SR^*, \boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \arg\min_{SR, \boldsymbol{\theta}, \boldsymbol{\phi}} \max_d \sum_{P \in \mathcal{F}} \mathcal{L}(G_P, D'_P + S_P; d) \\ + \lambda_S ||S_P||_1 + \lambda_V \mathcal{V}(W_P, W_{P_r}) + \lambda_T \mathcal{V}(D'_P, D'_{P_r}),\end{aligned} \tag{4.7}$$

where P_r is a randomly chosen frame in the same batch with P during each stochastic gradient descent step. λ_S, λ_T and λ_V are set to 1e-4. An overview diagram is shown in Fig. 4.3. Fig. 4.4 shows that the optimization discovers coherent material regions and a detailed environment image.

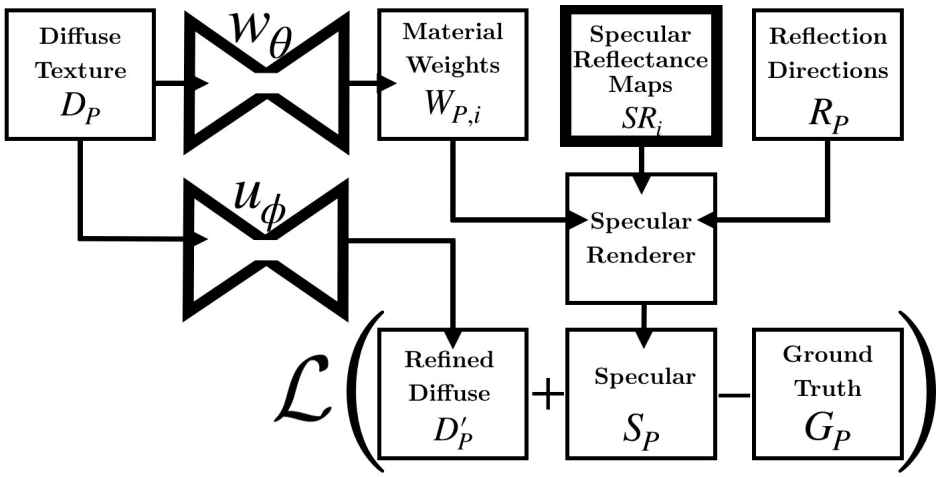
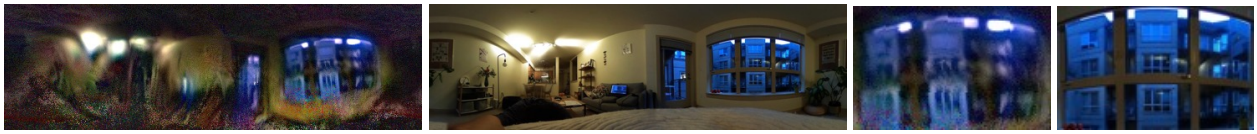


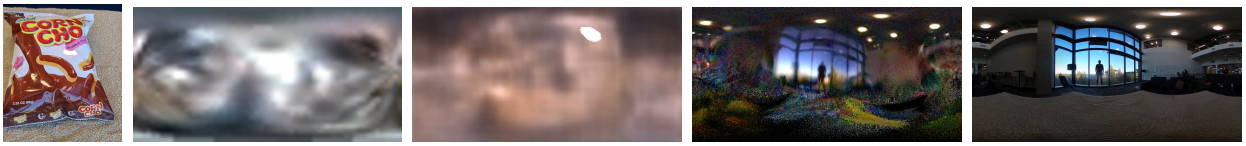
Figure 4.3: The components of my SRM estimation pipeline (optimized parameters shown in bold). I predict a view by adding refined diffuse texture D'_P (Fig. 4.2) and the specular image S_P. S_P is computed, for each pixel, by looking up the basis SRMs (SR_i's) with surface reflection direction R_P and blending them with weights $W_{P,i}$ obtained via network w_θ. The loss between the predicted view and ground truth G_P is backpropagated to jointly optimize the SRM pixels and network weights.

(a) Video (b) Material (c) Recovered SRM (d) Ground Truth



(e) Recovered SRM (f) Ground Truth (g) Zoom-in(ours) (h) Zoom-in(GT)

Figure 4.4: Sample results of recovered SRMs and material weights. Given input video frames (a), I recover global SRMs (c) and their linear combination weights for materials (b) from the optimization of Eq. (4.7). The scenes presented here have two material bases, visualized with red and green channels. Estimated SRMs (c) corresponding to the shiny object surface (green channel) correctly capture the light sources of the scenes, shown in the reference panorama images (d). For both scenes the SRMs corresponding to the red channel is mostly black, thus not shown, as the surface is mostly diffuse. The recovered SRM of (c) overemphasizes blue channel due to oversaturation in input images. Third row shows estimation result from a video of the same bag of chips (first row) under different lighting. Close inspection of the recovered environment (g) reveals many scene details, including floors in a nearby building visible through the window.



(a) Input (b) Legendre [139] (c) Gardner [76] (d) My Result (e) Ground Truth

(f) Input (g) Lombardi [150] (h) My Result (i) Ground Truth

Figure 4.5: Comparisons with existing single-view and multi-view based environment estimation methods. Given a single image (a), Deeplight [139] (b), and Gardner [75] (c), do not produce accurate environment reconstructions, relative to what I obtain from an RGBD video (d) which better matches ground truth (e). Additionally, from a video sequence and noisy geometry of a synthetic scene (f), my method (h) more accurately recovers the surrounding environment (i) compared to Lombardi (g). Best viewed digitally.

4.2.3 Novel-View Neural Rendering

With reconstructed SRMs and material weights, I can synthesize specular appearance from any desired viewpoint via Eq. (4.2). However, while the approach detailed in Sec. 4.2.2 reconstructs high quality SRMs, the renderings often lack realism (shown in supplementary), due to two factors. First, errors in geometry and camera pose can sometimes lead to weaker reconstructed highlights. Second, the SRMs do not model more complex light transport effects such as interreflections or Fresnel reflection. This section describes how I train a network to address these two limitations, yielding more realistic results.

Simulations only go so far, and computer renderings will never be perfect. In principle, you could train a CNN to render images as a function of viewpoint directly, training on actual photos. Indeed, several recent neural rendering methods adapt image translation [110] to learn mappings from projected point clouds [165, 189, 3] or a UV map image [230] to a photo.

However, these methods struggle to extrapolate far away from the input views because their networks don't have built-in physical models of specular light transport.

Rather than treat the rendering problem as a black box, I arm the neural renderer with knowledge of physics – in particular, diffuse, specular, interreflection, and Fresnel reflection, to use in learning how to render images. Formally, I introduce an adversarial neural network-based generator g and discriminator d to render realistic photos. g takes as input our best prediction of diffuse D_P and specular S_P components for the current view (obtained from Eq. (4.7)), along with interreflection and Fresnel terms FBI, R_P, and FCI that will be defined later in this section.

Consequently, the generator g receives $C_P = (D_P, S_P,\ FBI, R_P, FCI)$ as input and outputs a prediction of the view, while the discriminator d scores its realism. I use the combination of pixelwise L_1, perceptual loss L_p [35], and the adversarial loss [110] as described in Sec. 4.2.2:

$$g^* = \arg\min_g \max_d \lambda_G \bar{\mathcal{L}}_G(g,d) + \lambda_p \bar{\mathcal{L}}_p(g) + \lambda_1 \bar{\mathcal{L}}_1(g), \tag{4.8}$$

where $\bar{\mathcal{L}}_p(g) = \frac{1}{|\mathcal{F}|}\sum_{P\in\mathcal{F}} \mathcal{L}_p(g(C_P), G_P)$ is the mean of perceptual loss across all input images, and $\mathcal{L}_G(g,d)$ and $\bar{\mathcal{L}}_1(g)$ are similarly defined as an average loss across frames. Note that this renderer g is *scene specific*, trained only on images of a particular scene to extrapolate new views of that same scene, as commonly done in the neural rendering community [165, 230, 3].

Modeling Interreflections and Fresnel Effects Eq. (4.2) models only the direct illumination of each surface point by the environment, neglecting interreflections. While modeling full, global, diffuse + specular light transport is intractable, I can approximate first order interreflections by ray-tracing a first-bounce image (FBI) as follows. For each pixel $\mathbf{u}$ in the virtual viewpoint to be rendered, cast a ray from the camera center through $\mathbf{u}$. If I pretend for now that every scene surface is a perfect mirror, that ray will bounce potentially multiple times and intersect multiple surfaces. Let $\mathbf{x}_2$ be the second point of intersection of that ray with the scene. Render the pixel at $\mathbf{u}$ in FBI with the diffuse color of $\mathbf{x}_2$, or with black if

there is no second intersection (Fig. 4.6(d)).

Glossy (imperfect mirror) interreflections can be modeled by convolving the FBI with the BRDF. Strictly speaking, however, the interreflected image should be filtered in the *angular domain* [195], rather than image space, i.e., convolution of incoming light following the specular lobe whose center is the reflection ray direction $\boldsymbol{\omega}_r$. Given $\boldsymbol{\omega}_r$, angular domain convolution can be approximated in image space by convolving the FBI image weighted by $\boldsymbol{\omega}_r$. However, because I do not know the specular kernel, I let the network infer the weights using $\boldsymbol{\omega}_r$ as a guide. I encode the $\boldsymbol{\omega}_r$ for each pixel as a three-channel image R_P (Fig. 4.6(e)).

Fresnel effects make highlights stronger at near-glancing view angles and are important for realistic rendering. Fresnel coefficients are approximated following [206]: $R(\alpha) = R_0 + (1 - R_0)(1 - cos\alpha)^5$, where α is the angle between the surface normal and the camera ray, and R_0 is a material-specific constant. I compute a Fresnel coefficient image (FCI), where each pixel contains $(1 - cos\alpha)^5$, and provide it to the network as an additional input, shown in Fig. 4.6(f). Using the additional components I show that in Fig. 4.7 that the neural rendering system correctly models the Fresnel effect.

In total, the rendering components C_P are now composed of five images: diffuse and specular images, FBI image, R_P, and FCI. C_P is then given as input to the neural network, and my network weights are optimized as in Eq. (4.8). Fig. 4.6 shows the effectiveness of the additional three rendering components for modeling interreflections.

4.2.4 Dataset

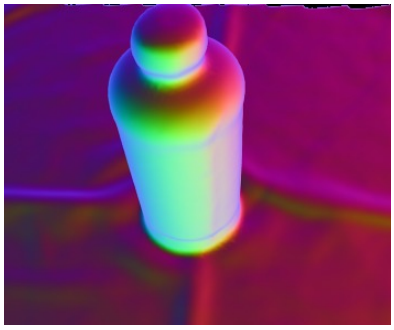
I captured ten sequences of RGBD video with a hand-held Primesense depth camera, featuring a wide range of materials, lighting, objects, environments, and camera paths. The length of each sequence ranges from 1500 to 3000 frames, which are split into train and test frames. Some of the sequences were captured such that the test views are very far from the training views, making them ideal for benchmarking the extrapolation abilities of novel-view synthesis methods. Moreover, many of the sequences come with ground truth HDR environment maps to facilitate future research on environment estimation. Further capture and

(a) W/O Interreflections

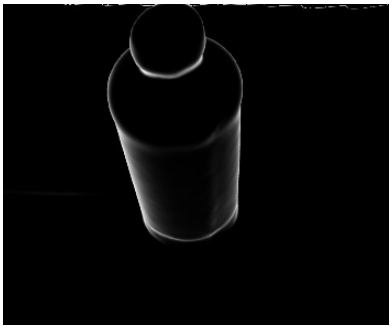
(b) With Interreflections

(c) Ground Truth

(d) FBI



(e) R_P



(f) Fresnel

Figure 4.6: Modeling interreflections. First row shows images of an unseen viewpoint rendered by a network trained with direct (a) and with interreflection + Fresnel models (b), compared to ground truth (c). Note accurate interreflections on the bottom of the green bottle (b). (d), (e), and (f) show first-bounce image (FBI), reflection direction image (R_P), and Fresnel coefficient image (FCI), respectively. Best viewed digitally.

(a) View 1 (b) View 2 (c) View 3 (d) View 1 (e) View 2 (f) View 3

Figure 4.7: Demonstration of the Fresnel effect. The intensity of specular highlights tends to be amplified at slant viewing angles. I show three different views (a,b,c) for a glossy bottle, each of them generated by my neural rendering pipeline and presenting different viewing angles with respect to the bottle. Notice that the neural rendering correctly amplifies the specular highlights as the viewing angle gets closer to perpendicular with the surface normal. Images (d,e,f) show the computed Fresnel coefficient (FCI) (see Sec. 6.1) for the corresponding views. These images are given as input to the neural-renderer that subsequently use them to simulate the Fresnel effect. Best viewed digitally.

data-processing details are in supplementary.

4.3 Experiments

I describe experiments to test my system's ability to estimate images of the environment and synthesize novel viewpoints, and ablation studies to characterize the factors that most contribute to system performance.

I compare my approach to several state-of-the-art methods: recent single view lighting estimation methods (DeepLight [139], Gardner [76]), an RGBD video-based lighting and material reconstruction method [150], an IR-based BRDF estimation method [184] (shown in supplementary), and two leading view synthesis methods capable of handling specular highlights – DeepBlending [102] and Deferred Neural Rendering (DNS) [230].

Finally, I conduct various ablation studies to identify the effects of loss functions, number of material bases, and surface roughness on the quality of the lighting and appearance reconstructions.

4.3.1 Environment Estimation

The computed SRMs demonstrate my system's ability to infer detailed images of the environment from the pattern and motion of specular highlights on an object. For example from 4.4(b), I can see the general layout of the living room, and even count the number of floors in buildings visible through the window. Note that the person capturing the video does not appear in the environment map because he is constantly moving. The shadow of the moving person, however, causes artifacts, e.g. the fluorescent lighting in the first row of Fig. 4.4 is not fully reconstructed.

Compared to state-of-the-art single view estimation methods [139, 76], my method produces a more accurate image of the environment, as shown in Fig. 4.5. Note the reconstruction shows a person standing near the window and autumn colors in a tree visible through the window.

(a) Camera Trajectory (b) Reference Photo (c) Ours (d) DeepBlending [102] (e) Thies [230]

Figure 4.8: View extrapolation to extreme viewpoints. I evaluate novel view synthesis on test views (red frusta) that are furthest from the input views (black frusta) (a). The view predictions of DeepBlending [102] and Thies [230] (d,e) are notably different from the reference photographs (b), e.g., missing highlights on the back of the cat, and incorrect highlights at the bottom of the cans. Thies [230] shows severe artifacts, likely because their learned UV texture features overfits to the input views, and thus cannot generalize to very different viewpoints. My method (c) produces images with highlights appearing at correct locations.

I compare with a multi-view RGBD based method [150] on a synthetic scene containing a red object, obtained from the authors. As in [150], I estimate lighting from the known geometry with added noise and a video of the scene rendering, but produce more accurate results (Fig. 4.5).

4.3.2 Novel View Synthesis

I recover specular reflectance maps and train a generative network for each video sequence. The trained model is then used to generate novel views from held-out views.

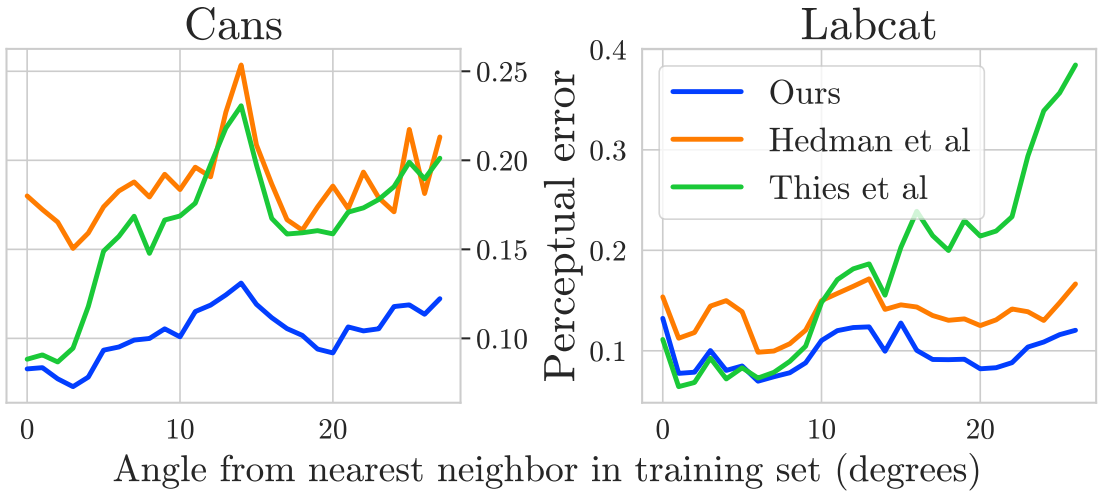
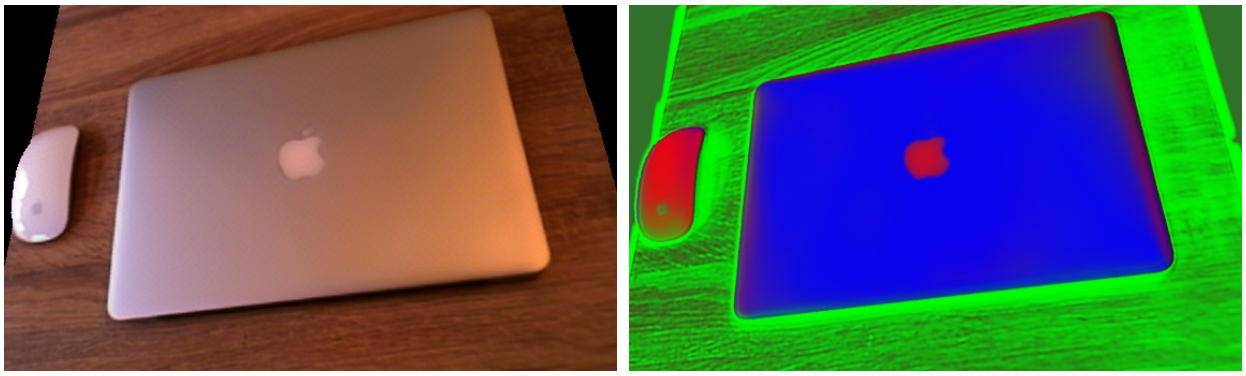


Figure 4.9: Quantitative comparisons for novel view synthesis. I plot the perceptual loss [270] between a novel view rendering and the ground truth test image as a function of its distance to the nearest training view (measured in angle between the view vectors). I compare my method with two leading NVS methods [102, 230] on two scenes. On average, my results have lowest error.

In the supplementary, I show novel view generation results for different scenes, along with the intermediate rendering components and ground truth images. As view synthesis results are better shown in video form, I strongly encourage readers to watch the *supplementary video*.

Novel View Extrapolation Extrapolating novel views far from the input range is particularly challenging for scenes with reflections. To test the operating range of my and other recent view synthesis results, I study how the quality of view prediction degrades as a function of the distance to the nearest input images (in difference of viewing angles) (Fig. 4.9). I measure prediction quality with perceptual loss [270], which is known to be more robust to shifts or misalignments, against the ground truth test image taken from same pose. I use two video sequences both containing highly reflective surfaces and with large differences in



(a) Synthesized Novel-view (b) Material Weights

Figure 4.10: Image (a) shows a synthesized novel view using neural rendering (Sec. 4.2.3) of a scene with multiple glossy materials. The spatially varying materials (SRM blending weights) of the wooden tabletop and the laptop are accurately estimated by my algorithm (Sec. 4.2.2), as visualized in (b).

(a) Ground Truth (b) Synthesized Novel-view

Figure 4.11: Concave surface reconstruction. The appearance of highly concave bowls is realistically reconstructed by my system. The rendered result (b) captures both occlusions and highlights of the ground truth (a).

train and test viewpoints. I focus my attention on parts of the scene which exhibit significant view-dependent effects. That is, I mask out the diffuse backgrounds and measure the loss on only central objects of the scene. I compare my method with DeepBlending [102] and Thies [230]. The quantitative (Fig. 4.9) and qualitative (Fig. 4.8) results show that my method is able to produce more accurate images of the scene from extrapolated viewpoints.

4.3.3 Robustness and Photo-realism

My method is robust to various scene configurations, such as scenes containing multiple objects (Fig. 4.8), spatially varying materials (Fig. 4.10), and concave surfaces (Fig. 4.11).

Fig. 4.12 shows novel-view neural rendering results, together with the estimated components (diffuse and specular images D_P, S_P) provided as input to the renderer. My approach can synthesize photorealistic novel views of a scene with wide range of materials, object compositions, and lighting condition. Note that the featured scenes contain challenging properties such as bumpy surfaces (Fruits), rough reflecting surfaces (Macbook), and concave surfaces (Bowls). Overall, I demonstrate the robustness of my approach for various materials including fabric, metals, plastic, ceramic, fruit, wood, glass, etc.

On a separate note, reconstructing SRMs of planar surfaces could require more views to fully cover the environment hemisphere, because the surface normal variation of each view is very limited for a planar surface. I refer readers to Janick [112] that studies capturing planar surface light field, which reports that it takes about a minute using their real-time, guided capture system.

4.3.4 Ablation Study

Effects of Loss Functions I study how the choice of loss functions affects the quality of environment estimation and novel view synthesis. Specifically, I consider three loss functions between prediction and reference images as follow: (i) pixel-wise $L1$ loss, (ii) neural-network based perceptual loss, and (iii) adversarial loss. I run each of my algorithms (environment estimation and novel-view synthesis) for the three following cases: using (i) only, (i+ii) only,

(a) Ground Truth G_P (b) Rendering $g(C_P)$ (c) Specular Component (d) Diffuse Component

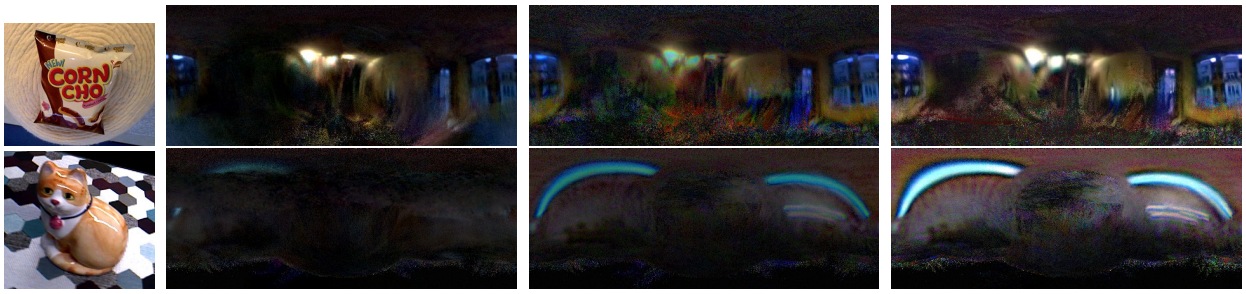
Figure 4.12: Novel view renderings and intermediate rendering components for various scenes. From left to right: (a) reference photograph, (b) my rendering, (c) specular reflectance map image S_P, and (d) diffuse texture image D_P. Note that some of the ground truth reference images have black "background" pixels inserted near the top and left borders where reconstructed geometry is missing, to provide equal visual comparisons to rendered images.

and all loss functions combined (i+ii+iii). For both algorithms I provide visual comparisons for each set of loss functions in Figures 4.14 and 4.13.

I run my joint optimization of SRMs and material weights to recover a visualization of the environment using the set of loss functions described above. As shown in Fig. 4.13, the pixel-wise L1 loss was unable to effectively penalize the view prediction error because it is very sensitive to misalignments due to noisy geometry and camera pose. While the addition of perceptual loss produces better results, one can observe muted specular highlights in the very bright regions. The adversarial loss, in addition to the two other losses, effectively deals with the input errors while simultaneously correctly capturing the light sources.

I similarly train the novel-view neural rendering network in Sec. 4.2.3 using the aforementioned loss functions. Results in Fig. 4.14 shows that while L1 loss fails to capture specularity when significant image misalignments exist, the addition of perceptual loss somewhat addresses the issue. As expected, using adversarial loss, along with all other losses, allows the neural network to fully capture the intensity of specular highlights.

Effects of Surface Roughness The recovered specular reflectance map is environment lighting convolved with the surface's specular BRDF. Thus, the quality of the estimated SRM should depend on the roughness of the surface, e.g. a near Lambertian surface would not provide significant information about its surroundings. To test this claim, I run the SRM estimation algorithm on a synthetic object with varying levels of specular roughness. Specifically, I vary the roughness parameter of the GGX shading model [235] from 0.01 to 1.0, where smaller values correspond to more mirror-like surfaces. I render images of the synthetic object, and provide those rendered images, as well as the geometry (with added noise in both scale and vertex displacements, to simulate a real scanning scenario), to my algorithm. The results show that, as expected, the increasing amounts of surface roughness cause the amount of details in the estimated environments to decrease, but the recovered SRMs still faithfully reproduces the blurred environment as in (Fig. 4.15).



(a) Scene (b) L1 Loss (c) L1+Perceptual Loss (d) L1+Perceptual+GAN

Figure 4.13: Environment estimation using different loss functions. From input video sequences (a), I run my SRM estimation algorithm, varying the final loss function between the view predictions and input images. Because L1 loss (b) is very sensitive to misalignments caused by geometric and calibration errors, it averages out the observed specular highlights, resulting in missing detail for large portions of the environment. While the addition of perceptual loss (c) mitigates this problem, the resulting SRMs often lose the brightness or details of the specular highlights. The adoption of GAN loss produces improved results (d).

(a) GT (b) L1 Loss (c) L1+Percept (d) All Losses

Figure 4.14: Effects of loss functions on neural-rendering. The specular highlights on the forehead of the Labcat is expressed weaker than it actually is when using L1 or perceptual loss, likely due to geometric and calibration errors. The highlight is best expressed when the neural rendering pipeline of Sec. 6 is trained with the combination of L1, perceptual, and adversarial loss.

Effects of Number of Material Bases The joint SRM and segmentation optimization in Eq. 4.7 requires a user to set the number of material bases. In this section, I study how the algorithm is affected by the user specified number. Specifically, for a scene containing two cans, I run my algorithm twice, with number of material bases set to be two and three, respectively. The results of the experiment in Figure 4.16 suggest that the number of material bases does not have a significant effect on the output of my system.

4.4 Additional Results

Fig. 4.18 shows that the naïve addition of diffuse and specular components obtained from the optimization in Sec. 5 does not results in photorealistic novel view synthesis, thus motivating a separate neural rendering step that takes as input the intermediate physically-based rendering components.

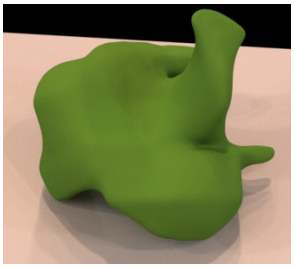
(a) Ground Truth Environment

(b) Input Frame

(c) Recovered SRM (GGX roughness 0.01)

(d) Input Frame

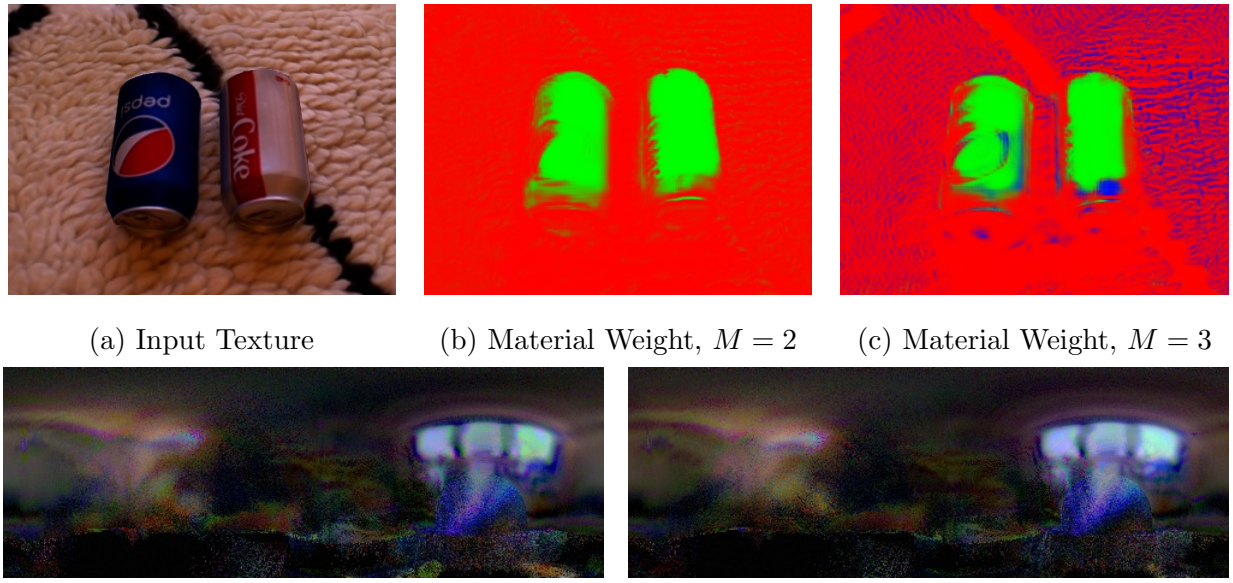
(e) Recovered SRM (GGX roughness 0.1)



(f) Input Frame

(g) Recovered SRM (GGX roughness 0.7)

Figure 4.15: Recovering SRM for different surface roughness. I test the quality of estimated SRMs (c,e,g) for various surface materials (shown in (b,d,f)). The results closely match my expectation that environment estimation through specularity is challenging for glossy (d) and diffuse (f) surfaces, compared to the mirror-like surfaces (c). Note that the input to my system are rendering images and noisy geometry, from which my system reliably estimates the environment.



(a) Input Texture (b) Material Weight, $M = 2$ (c) Material Weight, $M = 3$

(d) Recovered SRM, $M = 2$ (e) Recovered SRM, $M = 3$

Figure 4.16: Sensitivity to the number of material bases M. I run my SRM estimation and material segmentation pipeline twice on a same scene but with different number of material bases M, showing that my system is robust to the choice of M. I show the predicted combination weights of the network trained with two (b) and three (c) material bases. For both cases (b,c), SRMs that correspond to the red and blue channel are mostly black, i.e. diffuse BRDF. Note that my algorithm consistently assigns the specular material (green channel) to the same regions of the image (cans), and that the recovered SRMs corresponding to the green channel (d,e) are almost identical.

(a) Reference (b) This Approach (c) [184]

Figure 4.17: Comparison with [184] from Chapter 3. Note that the sharp specular highlight on the bottom-left of the Corncho bag is poorly reconstructed in the rendering of [184] (c). As shown in Sec. 4.3.4 and Fig. 4.12, these high frequency appearance details are only captured when using neural rendering and robust loss functions (b).

(a) Ground Truth (b) Rendering with SRM

Figure 4.18: Motivation for neural rendering. While the SRM and segmentation obtained from the optimization of Sec. 5 of the main text provides high quality environment reconstruction, the simple addition of the diffuse and specular component does not yield photorealistic rendering (b) compared to the ground truth (a). This motivates the neural rendering network that takes input as the intermediate rendering components and generate photorealistic images (e.g. shown in Fig. 4.12).

Comparison to BRDF Fitting Recovering a parametric analytical BRDF is a popular strategy to model view-dependent effects. I thus compare my neural network-based novel-view synthesis approach against a recent BRDF fitting method of [184] that uses an IR laser and camera to optimize for the surface specular BRDF parameters. As shown in Fig. 4.17, sharp specular BRDF fitting methods are prone to failure when there are calibration errors or misalignments in geometry.

4.5 Implementation Details

I follow [116] for the generator network architecture, use the PatchGAN discriminator [110], and employ the loss of LSGAN [158]. I use ADAM [132] with learning rate 2e-4 to optimize the objectives. Data augmentation was essential for viewpoint generalization, by applying random rotation, translation, flipping, and scaling to each input and output pair.

My pipeline is built using PyTorch [186]. For all of my experiments I used ADAM optimizer with learning rate 2e-4 for the neural networks and 1e-3 for the SRM pixels. For the SRM optimization described in Sec. 5 of the main text the training was run for 40 epochs (i.e. each training frame is processed 40 times), while the neural renderer training was run for 75 epochs.

I find that data augmentation plays a significant role to the view generalization of my algorithm. For training in Sec. 5, I used random rotation (up to 180°), translation (up to 100 pixels), and horizontal and vertical flips. For neural renderer training in Sec. 6, I additionally scale the input images by a random factor between 0.8 and 1.25.

I use Blender [45] for computing the reflection direction image R_P and the first bounce interreflection (FBI) image described in the main text.

Network Architectures Let `C(k,ch_in,ch_out,s)` be a convolution layer with kernel size `k`, input channel size `ch_in`, output channel size `ch_out`, and stride `s`. When the stride `s` is smaller than 1, I first conduct nearest-pixel upsampling on the input feature and then process it with a regular convolution layer. I denote `CNR` and `CR` to be the Convolution-

InstanceNorm-ReLU layer and Convolution-ReLU layer, respectively. A residual block `R(ch)` of channel size `ch` contains convolutional layers of `CNR(3,ch,ch,1)-CN(3,ch,ch,1)`, where the final output is the sum of the outputs of the first and the second layer.

Encoder-Decoder Network Architecture The architecture of the texture refinement network and the neural rendering network in Sec.5 and Sec.6 closely follow the architecture of an encoder-decoder network of Johnson [116]: `CNR(9,ch_in,32,1)-CNR(3,32,64,2)-CNR(3,64,128,2)-R(128)-R(128)-R(128)-R(128)-R(128)-CNR(3,128,64,1/2)-CNR(3,64,32,1/2)-C(3,32,3,1)`, where `c_in` represents a variable input channel size, which is 3 and 13 for the texture refinement network and neural rendering generator, respectively.

Material Weight Network The architecture of the material weight estimation network in Sec. 5 is as follows: `CNR(5,3,64,2)-CNR(3,64,64,2)-R(64)-R(64)-CNR(3,64,32,1/2)-C(3,32,3,1/2)`.

Discriminator Architecture The discriminator network used for the adversarial loss in Eq. 4.7 and Eq. 4.8 both use the same architecture as follows: `CR(4,3,64,2)-CNR(4,64,128,2)-CNR(4,128,256,2)-CNR(4,256,512,2)-C(1,512,1,1)`. For this network, I use a LeakyReLU activation (slope 0.2) instead of the regular ReLU, so CNR used here is a Convolution-InstanceNorn-LeakyReLU layer. Note that the spatial dimension of the discriminator output is larger than 1x1 for the image dimensions (640x480), i.e., the discriminator scores realism of patches rather than the whole image (as in PatchGAN [110]).

Data Capture Details I capture ten videos of objects with varying materials, lighting and compositions. I used a Primesense Carmine RGBD structured light camera. I perform intrinsic and radiometric calibrations, and correct the images for vignetting. During capture, the color and depth streams were hardware-synchronized, and registered to the color camera frame-of-reference. The resolution of both streams are VGA (640x480) and the frame rate was set to 30fps. Camera exposure was manually set and fixed within a scene.

I obtained camera extrinsics by running ORB-SLAM [171] (ICP [175] was alternatively used for feature-poor scenes). Using the estimated pose, I ran volumetric fusion [175] to obtain the geometry reconstruction. Once geometry and rough camera poses are estimated, I ran frame-to-model dense photometric alignment following [184] for more accurate camera positions, which are subsequently used to fuse in the diffuse texture to the geometry. Following [184], I use iteratively reweighted least squares to compute a robust minimum of intensity for each surface point across viewpoints, which provides a good approximation to the diffuse texture.

4.6 Discussion

In this chapter, I introduced a technique to recover a detailed image of an environment from a hand-held RGB-D camera. Moreover, I use a combination of physically-based and learning-based approach to achieve realistic and robust novel-view synthesis, outperforming the state-of-the-art methods.

However, my proposed approach relies on the reconstructed mesh obtained from fusing depth images of consumer-level depth cameras and thus fails for surfaces out of the operating range of these cameras, e.g., thin, transparent, or mirror surfaces. The recovered environment images are filtered by the surface BRDF; separating these two factors is an interesting topic of future work, perhaps via data-driven deconvolution (e.g. [255]). Moreover, reconstructing a room-scale photorealistic appearance model remains a major open challenge to explore

Finally, The ability to reconstruct the reflected scene from images of an object opens up real and valid concerns about privacy. While my method requires a depth sensor, future research may lead to methods that operate on regular photos. In addition to educating people on what's possible, my work could facilitate research on privacy-preserving cameras and security techniques that actively identify and scramble reflections.

Chapter 5

SHAPE MODELING VIA NEURAL IMPLICIT FUNCTIONS

Chapters 3 and 4 introduced techniques for recovering high quality appearance models and scene environments from a hand-held camera. While the above systems achieve photo-realistic reconstruction from casual inputs, they cannot complete the unobserved parts of the target scene. This means that an user needs to scan every surface in a scene for a full reconstruction, which is impractical for real world applications. For example, when scanning a chair, an user needs to kneel down to scan the bottom parts of the chair and obtain a full reconstruction. Therefore, a truly casual reconstruction approach naturally calls for the automatic completion ability, as described in Criteria 3 in Chapter 1.

Plausibly completing partial input data is an ill-posed problem as there are infinitely many ways to fill in the missing data. Therefore, data completion typically requires learning priors and patterns from a large dataset to generate likely content given the partial observation. In the 2D domain, deep neural networks have been successfully adopted for image inpainting or generation tasks [122, 120]. However, the complexity of convolutional neural networks grows quickly in space and time when directly applied to a discretized voxel grid. More classical and compact surface representations such as triangle or quad meshes pose problems in training since we may need to handle an unknown number of vertices and arbitrary topology. These challenges have limited the quality, flexibility and fidelity of deep learning approaches when attempting to either process or generate 3D data [249, 50].

In this chapter, I introduce a novel representation for generative 3D modeling that is efficient, expressive, and fully continuous. My approach uses the concept of a SDF, but unlike common surface reconstruction techniques which discretize this SDF into a regular grid [47, 175], I instead learn a generative model to produce a continuous SDF.

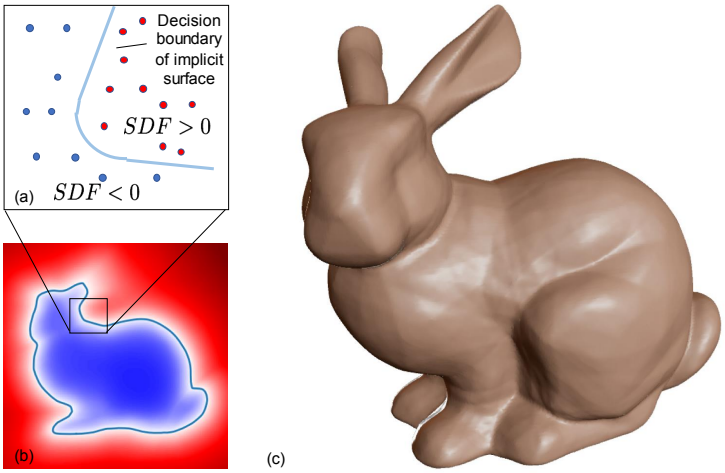


Figure 5.1: My DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

The proposed continuous representation may be intuitively understood as a 3D classifier that predicts whether an arbitrary 3D location is inside or outside of a shape, and thus its decision boundary is the surface of the shape itself, as shown in Fig. 5.1. My approach shares the generative aspect of other works seeking to map a latent space to a distribution of complex shapes in 3D [249], but critically differs in the central representation. While the notion of an implicit surface defined as a SDF is widely known in the computer vision and graphics communities, to my knowledge, this is the first attempt to directly learn continuous, generalizable 3D generative models of SDFs.

The contributions include: (i) the formulation of generative shape-conditioned 3D modeling with a continuous implicit surface, (ii) a learning method for 3D shapes based on a probabilistic auto-decoder, and (iii) the demonstration and application of this formulation to shape modeling and completion. The proposed implicit models produce high quality continuous surfaces with complex topologies, and obtain state-of-the-art results in quantitative

comparisons for shape reconstruction and completion. As an example of the effectiveness of the method, my models use only 7.4 MB (megabytes) of memory to represent entire classes of shapes (for example, thousands of 3D chair models) – this is, for example, less than half the memory footprint (16.8 MB) of a single uncompressed 512^3 3D bitmap.

5.1 Related Work

I review three main areas of related work: 3D representations for shape learning (Sec. 5.1.1), techniques for learning generative models (Sec. 5.1.2), and shape completion (Sec. 5.1.3).

5.1.1 Representations for 3D Shape Learning

Representations for data-driven 3D learning approaches can be largely classified into three categories: point-based, mesh-based, and voxel-based methods. While some applications such as 3D-point-cloud-based object classification are well suited to these representations, I address their limitations in expressing continuous surfaces with complex topologies.

Point-based. A point cloud is a lightweight 3D representation that closely matches the raw data that many sensors (i.e. LiDARs, depth cameras) provide, and hence is a natural fit for applying 3D learning. PointNet [191], for example, uses max-pool operations to extract global shape features, and the technique is widely used as an encoder for point generation networks [259, 1]. There is a sizable list of related works for learning on point clouds [192, 239, 266]. A primary limitation, however, of learning with point clouds is that they do not describe topology and are not suitable for producing watertight surfaces.

Mesh-based. Various approaches represent classes of similarly shaped objects, such as morphable human body parts, with predefined template meshes and some of these models demonstrate high fidelity shape generation results [9, 145]. Other recent works [11] use poly-cube mapping [228] for shape optimization. While the use of template meshes is convenient and naturally provides 3D correspondences, it can only model shapes with fixed mesh topology.

Other mesh-based methods use existing [212, 159] or learned [88, 92] parameterization

techniques to describe 3D surfaces by morphing 2D planes. The quality of such representations depends on parameterization algorithms that are often sensitive to input mesh quality and cutting strategies. To address this, recent data-driven approaches [259, 88] learn the parameterization task with deep networks. They report, however, that (a) multiple planes are required to describe complex topologies but (b) the generated surface patches are not stitched, i.e. the produced shape is not closed. To generate a closed mesh, sphere parameterization may be used [88, 92], but the resulting shape is limited to the topological sphere. Other works related to learning on meshes propose to use new convolution and pooling operations for meshes [58, 233] or general graphs [27].

Voxel-based. Voxels, which non-parametrically describe volumes with 3D grids of values, are perhaps the most natural extension into the 3D domain of the well-known learning paradigms (i.e., convolution) that have excelled in the 2D image domain. The most straightforward variant of voxel-based learning is to use a dense occupancy grid (occupied / not occupied). Due to the cubically growing compute and memory requirements, however, current methods are only able to handle low resolutions (128^3 or below). As such, voxel-based approaches do not preserve fine shape details [252, 44], and additionally voxels visually appear significantly different than high-fidelity shapes, since when rendered their normals are not smooth. Octree-based methods [229, 198, 96] alleviate the compute and memory limitations of dense voxel methods, extending for example the ability to learn at up to 512^3 resolution [229], but even this resolution is far from producing shapes that are visually compelling.

Aside from occupancy grids, and more closely related to my approach, it is also possible to use a 3D grid of voxels to represent a signed distance function. This inherits from the success of fusion approaches that use a truncated SDF (TSDF), pioneered in [47, 175], to combine noisy depth maps into a single 3D model. Voxel-based SDF representations have been extensively used for 3D shape learning [268, 50, 222], but their use of discrete voxels is expensive in memory. As a result, these approaches generally present low resolution shapes. Wavelet transform-based methods [117] and dimensionality reduction techniques [117] for distance fields were reported, but they encode the SDF volume of each individual scene

rather than a dataset of shapes.

Recently, concurrently to my work, binary implicit surface representations were used by [38, 164], where they train deep networks across classes of shapes to classify 3D points as inside or outside of a shape. Note that the binary occupancy function is a special case of SDF, considering only 'sign' of SDF values. As DeepSDF models metric signed distance to the surface, it can be used to raycast against surfaces and compute surface normals with its gradients.

5.1.2 Representation Learning Techniques

Modern representation learning techniques aim at automatically discovering a set of features that compactly but expressively describe data. For a more extensive review of the field, I refer to Bengio et al. [17].

Generative Adversial Networks. GANs [83] and their variants [37, 194, 110, 120] learn deep embeddings of target data, from which realistic images are sampled, by training discriminators adversarially against generators. In the 3D domain, Wu et al. [249] trains a GAN to generate objects in a voxel form, while Hamu et al. [92] uses multiple parameterization planes to generate shapes of topological spheres. On the downside, training for GANs is known to be unstable.

Auto-encoders. The ability of auto-encoders as a representation learning tool has been evidenced by the vast variety of 3D shape learning works in the literature [50, 222, 9, 88, 250] who adopt auto-encoders for representation learning. Recent 3D vision works [22, 9, 145] often adopt a variational auto-encoder (VAE) learning scheme, in which bottleneck features are perturbed with Gaussian noise to encourage smooth and complete latent spaces. The regularization on the latent vectors enables exploring the embedding space with gradient descent or random sampling.

Optimizing Latent Vectors. Instead of using the full auto-encoder, an alternative is to learn compact data representations by training *decoder-only* networks. This idea goes back to at least the work of Tan et al. [227] which simultaneously optimizes the latent vectors assigned

to each data point and the decoder weights through back-propagation. For inference, an optimal latent vector is searched to match the new observation with fixed decoder parameters. Similar approaches have been extensively studied in [196, 24, 193], for applications including noise reduction, missing measurement completions, and fault detections. Recent approaches [23, 69] extend the technique by applying deep architectures. Throughout the chapter I refer to this class of networks as *auto-decoders*, for they are trained with *self*-reconstruction loss on decoder-only architectures.

5.1.3 Shape Completion

3D shape completion related works aim to infer unseen parts of the original shape given sparse or partial input observations. This task is analogous to image-inpainting in 2D computer vision.

Classical surface reconstruction methods complete a point cloud into a dense surface by fitting radial basis function (RBF) [29] to approximate implicit surface functions, or by casting the reconstruction from oriented point clouds as a Poisson problem [127]. These methods only model a single shape rather than a dataset.

Various recent methods use data-driven approaches for the 3D completion task. Most of these methods adopt encoder-decoder architectures to reduce partial inputs of occupancy voxels [252], discrete SDF voxels [50], depth maps [199], RGB images [44, 250] or point clouds [222] into a latent vector and subsequently generate a prediction of full volumetric shape based on learned priors.

5.2 Modeling SDFs with Neural Networks

In this section I present DeepSDF, my continuous shape learning approach. I describe modeling shapes as the zero iso-surface decision boundaries of feed-forward networks trained to represent SDFs. A signed distance function is a continuous function that, for a given spatial point, outputs the point's distance to the closest surface, whose sign encodes whether

the point is inside (negative) or outside (positive) of the watertight surface:

$$SDF(\boldsymbol{x}) = s : \boldsymbol{x} \in \mathbb{R}^3,\, s \in \mathbb{R}\,. \tag{5.1}$$

The underlying surface is implicitly represented by the iso-surface of $SDF(\cdot) = 0$. A view of this implicit surface can be rendered through raycasting or rasterization of a mesh obtained with, for example, Marching Cubes [152].

My key idea is to directly regress the continuous SDF from point samples using deep neural networks. The resulting trained network is able to predict the SDF value of a given query position, from which I can extract the zero level-set surface by evaluating spatial samples. Such surface representation can be intuitively understood as a spatial classifier for which the decision boundary is the surface of the shape itself (Fig. 5.1). As a universal function approximator [107], deep feed-forward networks in theory can learn continuous SDFs with arbitrary precision. Yet, the precision of the approximation in practice is limited by the finite number of point samples that guide the decision boundaries and the finite capacity of the network due to restricted compute power.

The most direct application of this approach is to train a single deep network for a given target shape as depicted in Fig. 5.2a. Given a target shape, I prepare a set of pairs X composed of 3D point samples and their SDF values:

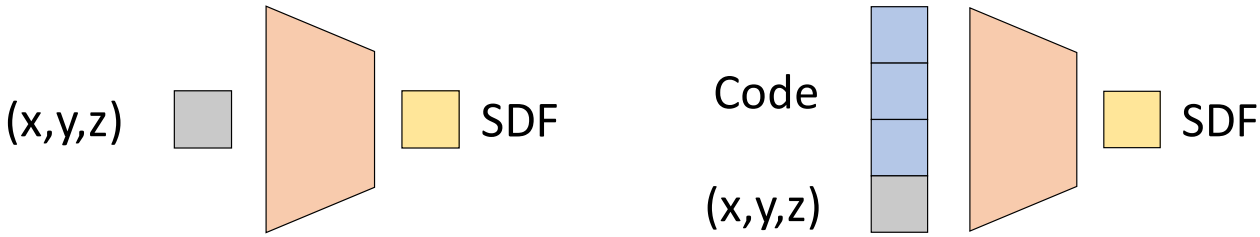
$$X := \{(\boldsymbol{x}, s) : SDF(\boldsymbol{x}) = s\}\,. \tag{5.2}$$

I train the parameters θ of a multi-layer fully-connected neural network f_θ on the training set S to make f_θ a good approximator of the given SDF in the target domain Ω:

$$f_\theta(\boldsymbol{x}) \approx SDF(\boldsymbol{x}),\, \forall \boldsymbol{x} \in \Omega\,. \tag{5.3}$$

The training is done by minimizing the sum over losses between the predicted and real SDF values of points in X under the following L_1 loss function:

$$\mathcal{L}(f_\theta(\boldsymbol{x}), s) = |\,\mathrm{clamp}(f_\theta(\boldsymbol{x}), \delta) - \mathrm{clamp}(s, \delta)\,|, \tag{5.4}$$



(a) Single Shape DeepSDF (b) Coded Shape DeepSDF

Figure 5.2: DeepSDF network outputs SDF value at a 3D query location. While (a) the network can memorize a single shape, (b) conditioning the network with a code vector allows DeepSDF to model a large space of shapes, where the shape information is contained in the code vector that is concatenated with the query point.

where $\text{clamp}(x, \delta) := \min(\delta, \max(-\delta, x))$ introduces the parameter δ to control the distance from the surface over which I expect to maintain a metric SDF. Larger values of δ allow for fast ray-tracing since each sample gives information of safe step sizes [99]. Smaller δ can be used to concentrate network capacity on details near the surface. I used $\delta = 0.1$ in practice.

Once trained, the surface is implicitly represented as the zero iso-surface of $f_\theta(\boldsymbol{x})$, which can be visualized through raycasting or marching cubes. Another nice property of this approach is that accurate normals can be analytically computed by calculating the spatial derivative $\partial f_\theta(\boldsymbol{x})/\partial \boldsymbol{x}$ via back-propogation through the network.

5.3 Learning the Latent Space of Shapes

Training a specific neural network for each shape is neither feasible nor very useful. Instead, we want a model that can represent a wide variety of shapes, discover their common properties, and embed them in a low dimensional latent space. To this end, I introduce a latent vector $\boldsymbol{z}$, which can be thought of as encoding the desired shape, as a second input to the neural network as depicted in Fig. 5.2b. Conceptually, I map this latent vector to a 3D shape represented by a continuous SDF. Formally, for some shape indexed by i, f_θ is now a function

of a latent code $\boldsymbol{z}_i$ and a query 3D location $\boldsymbol{x}$, which outputs the shape's approximate SDF:

$$f_\theta(\boldsymbol{z}_i, \boldsymbol{x}) \approx SDF^i(\boldsymbol{x}). \tag{5.5}$$

By conditioning the network output on a latent vector, this formulation allows modeling multiple SDFs with a single neural network. Given the decoding model f_θ, the continuous surface associated with a latent vector $\boldsymbol{z}$ is similarly represented with the zero iso-surface of $f_\theta(\boldsymbol{z}, \boldsymbol{x})$, and the shape can again be discretized for visualization by, for example, raycasting or Marching Cubes.

Throughout the chapter I use the coded shape DeepSDF model of Fig. 5.2b whose decoder is a feed-forward network composed of eight fully connected layers, each of them applied with dropouts. All internal layers are 512-dimensional and have ReLU non-linearities. The output non-linearity regressing the SDF value is tanh. I found training with batch-normalization [109] to be unstable and applied the weight-normalization technique instead [205]. For training, I use the Adam optimizer [132].

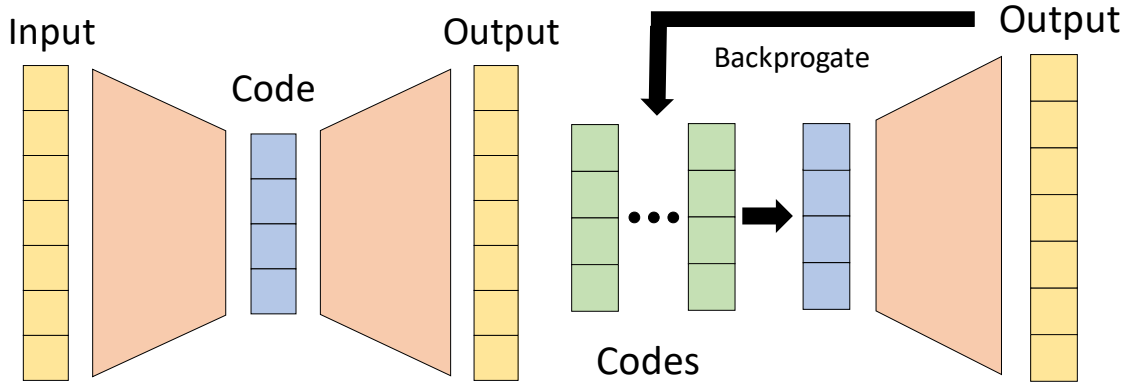
In the next section I explain training the decoding model $f_\theta(\boldsymbol{z}, \boldsymbol{x})$ and introduce the 'auto-decoder' formulation for encoder-less training of shape-coded DeepSDF.

5.3.1 Motivating Encoder-less Learning

Auto-encoders and encoder-decoder networks are widely used for representation learning as their bottleneck features tend to form natural latent variable representations.

Recently, in applications such as modeling depth maps [22], faces [9], and body shapes [145] a full encoder-decoder network is trained but only the decoder is retained for inference, where they search for an optimal latent vector given some input observation. Since the trained encoder is unused at test time, it is unclear whether (1) training encoder is an effective use of computational resources and (2) it is necessary for researchers to design encoders for various 3D input representations (e.g. points, meshes, octrees, etc).

This motivates us to use an *auto*-**decoder** for learning a shape embedding without an encoder as depicted in Fig. 5.3b. I show that learning continuous SDFs with auto-decoder



(a) Auto-encoder (b) Auto-decoder

Figure 5.3: Different from an auto-encoder whose latent code is produced by the encoder, an auto-decoder directly accepts a latent vector as an input. A randomly initialized latent vector is assigned to each data point in the beginning of training, and the latent vectors are optimized along with the decoder weights through standard backpropagation. During inference, decoder weights are fixed, and an optimal latent vector is estimated.

leads to high quality 3D generative models. Further, I develop a probabilistic formulation for training and testing the auto-decoder that naturally introduces latent space regularization for improved generalization. To the best of my knowledge, this work is the first to introduce the auto-decoder learning method to the 3D learning community.

5.3.2 Auto-decoder-based DeepSDF Formulation

To derive the auto-decoder-based shape-coded DeepSDF formulation I adopt a probabilistic perspective. Given a dataset of N shapes represented with signed distance function ${SDF^i}_{i=1}^N$, I prepare a set of K point samples and their signed distance values:

$$X_i = \{(\boldsymbol{x}_j, s_j) : s_j = SDF^i(\boldsymbol{x}_j)\}\,. \tag{5.6}$$

For an auto-decoder, as there is no encoder, each latent code $\boldsymbol{z}_i$ is paired with training shape X_i. The posterior over shape code $\boldsymbol{z}_i$ given the shape SDF samples X_i can be

decomposed as:

$$p_\theta(\boldsymbol{z}_i|X_i) = p(\boldsymbol{z}_i) \prod_{(\boldsymbol{x}_j,\boldsymbol{s}_j)\in X_i} p_\theta(\boldsymbol{s}_j|z_i;\boldsymbol{x}_j)\,, \tag{5.7}$$

where θ parameterizes the SDF likelihood. In the latent shape-code space, I assume the prior distribution over codes $p(\boldsymbol{z_i})$ to be a zero-mean multivariate-Gaussian with a spherical covariance $\sigma^2 I$. This prior encapsulates the notion that the shape codes should be concentrated and I empirically found it was needed to infer a compact shape manifold and to help converge to good solutions.

For an auto-decoder, as there is no encoder, each latent code $\boldsymbol{z}_i$ is paired with training shape data X_i and randomly initialized from a zero-mean Gaussian. I use $\mathcal{N}(0, 0.001^2)$. The latent vectors $\{\boldsymbol{z}_i\}_{i=1}^N$ are then jointly optimized during training along with the decoder parameters θ.

I assume that each shape in the given dataset $\boldsymbol{X} = \{X_i\}_{i=1}^N$ follows the joint distribution of shapes:

$$p_\theta(X_i, \boldsymbol{z}_i) = p_\theta(X_i|\boldsymbol{z}_i)p(\boldsymbol{z}_i)\,, \tag{5.8}$$

where θ parameterizes the data likelihood. For a given θ a shape code $\boldsymbol{z}_i$ can be estimated via Maximum-a-Posterior (MAP) estimation:

$$\hat{\boldsymbol{z}}_i = \arg\max_{\boldsymbol{z}_i} p_\theta(\boldsymbol{z}_i|X_i) = \arg\max_{\boldsymbol{z}_i} \log p_\theta(\boldsymbol{z}_i|X_i)\,. \tag{5.9}$$

I estimate θ as the parameters that maximizes the posterior across all shapes:

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \sum_{X_i\in\boldsymbol{X}} \max_{\boldsymbol{z}_i} \log p_\theta(\boldsymbol{z}_i|X_i) \\ &= \arg\max_{\theta} \sum_{X_i\in\boldsymbol{X}} \max_{\boldsymbol{z}_i} (\log p_\theta(X_i|\boldsymbol{z}_i) + \log p(\boldsymbol{z}_i))\,,\end{aligned} \tag{5.10}$$

where the second equality follows from Bayes Law.

For each shape X_i defined via point and SDF samples $(\boldsymbol{x}_j, \boldsymbol{s}_j)$ as defined in Eq. 5.6 I make a conditional independence assumption given the code z_i to arrive at the decomposition of the posterior $p_\theta(X_i|z_i)$:

$$p_\theta(X_i|z_i) = \prod_{(\boldsymbol{x}_j,\boldsymbol{s}_j)\in X_i} p_\theta(\boldsymbol{s}_j|z_i;\boldsymbol{x}_j)\,. \tag{5.11}$$

Note that the individual SDF likelihoods $p_\theta(\boldsymbol{s}_j|z_i;\boldsymbol{x}_j)$ are parameterized by the sampling location $\boldsymbol{x}_j$.

To derive the proposed auto-decoder-based DeepSDF approach I express the SDF likelihood via a deep feed-forward network $f_\theta(\boldsymbol{z}_i,\boldsymbol{x}_j)$ and, without loss of generality, assume that the likelihood takes the form:

$$p_\theta(\boldsymbol{s}_j|z_i;\boldsymbol{x}_j) = \exp(-\mathcal{L}(f_\theta(\boldsymbol{z}_i,\boldsymbol{x}_j), s_j))\,. \tag{5.12}$$

The SDF prediction $\tilde{s}_j = f_\theta(\boldsymbol{z}_i,\boldsymbol{x}_j)$ is represented using a fully-connected network and $\mathcal{L}(\tilde{s}_j, s_j)$ is a loss function penalizing the deviation of the network prediction from the actual SDF value s_j. One example for the cost function is the standard L_2 loss function which amounts to assuming Gaussian noise on the SDF values. In practice I use the clamped L_1 cost from Eq. 5.4 for reasons outlined previously.

In the latent shape-code space, I assume the prior distribution over codes $p(\boldsymbol{z_i})$ to be a zero-mean multivariate-Gaussian with a spherical covariance $\sigma^2 I$. Note that other more complex priors could be assumed. This leads to the final cost function via Eq. 5.10 which I jointly minimize with respect to the network parameters θ and the shape codes $\{z_i\}_{i=1}^N$:

$$\underset{\theta,\{\boldsymbol{z}_i\}_{i=1}^N}{\arg\min} \sum_{i=1}^{N} \left(\sum_{j=1}^{K} \mathcal{L}(f_\theta(\boldsymbol{z}_i,\boldsymbol{x}_j), s_j) + \frac{1}{\sigma^2}||\boldsymbol{z}_i||_2^2 \right). \tag{5.13}$$

At inference time, after training and fixing θ, a shape code $\boldsymbol{z}_i$ for shape X_i can be estimated via Maximum-a-Posterior (MAP) estimation as:

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\arg\min} \sum_{(\boldsymbol{x}_j,\boldsymbol{s}_j)\in X} \mathcal{L}(f_\theta(\boldsymbol{z},\boldsymbol{x}_j), s_j) + \frac{1}{\sigma^2}||\boldsymbol{z}||_2^2\,. \tag{5.14}$$

Crucially, this formulation is valid for SDF samples X of arbitrary size and distribution because the gradient of the loss with respect to $\boldsymbol{z}$ can be computed separately for each SDF sample. This implies that DeepSDF can handle any form of partial observations such as depth maps. This is a major advantage over the auto-encoder framework whose encoder expects a test input similar to the training data, e.g. shape completion networks of [50, 266] require preparing training data of partial shapes.

To incorporate the latent shape code, I stack the code vector and the sample location as depicted in Fig. 5.2b and feed it into the same fully-connected NN described previously at the input layer and additionally at the 4th layer. I again use the Adam optimizer [132]. The latent vector $\boldsymbol{z}$ is initialized randomly from $\mathcal{N}(0, 0.01^2)$.

Note that while both VAE and the proposed auto-decoder formulation share the zero-mean Gaussian prior on the latent codes, I found that the the stochastic nature of the VAE optimization did not lead to good training results.

Training and Testing Details

For training, I find it important to initialize the latent vectors quite small, so that similar shapes do not diverge in the latent vector space – I used $\mathcal{N}(0, 0.01^2)$. Another crucial point is balancing the positive and negative samples both for training and testing: for each batch used for gradient descent, I set half of the SDF point samples positive and the other half negative.

Learning rate for the decoder parameters was set to be 1e-5 * B, where B is number of shapes in one batch. For each shape in a batch I randomly subsampled 16384 SDF samples (out of 500K available points). Learning rate for the latent vectors was set to be 1e-3. Also, I set the regularization parameter $\sigma = 10^{-2}$. I trained my models on 8 Nvidia GPUs approximately for 8 hours for 1000 epochs. For reconstruction experiments the latent vector size was set to be 256, and for the shape completion task I used models with 128 dimensional latent vectors.

Fig. 5.4 depicts the overall architecture of DeepSDF. For all experiments in this chapter I used a network composed of 8 fully connected layers each of which are applied with weight-normalization, and each intermediate vectors are processed with RELU activation and 0.2 dropout except for the final layer. A skip connection is included at the fourth layer.

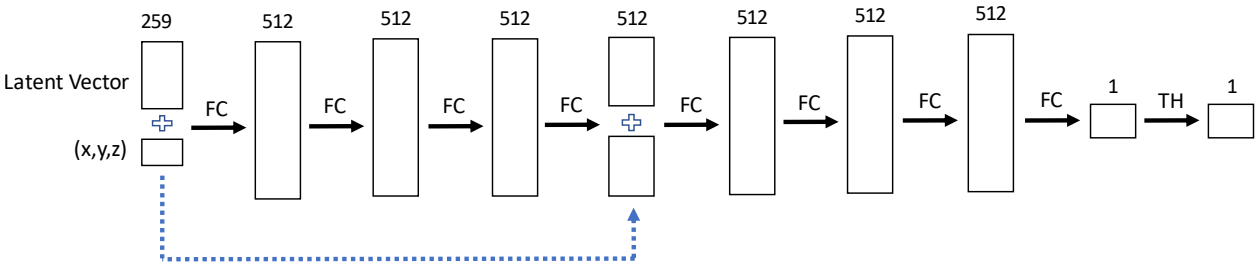


Figure 5.4: DeepSDF architecture used for experiments. Boxes represent vectors while arrows represent operations. The feed-forward network is composed of 8 fully connected layers, denoted as "FC" on the diagram. Each of the "FC" layers except for the last one is processed with weight-normalization, ReLU activation and Dropout. I used 256 and 128 dimensional latent vectors for reconstruction and shape completion experiments, respectively. The latent vector is concatenated, denoted "+", with the xyz query, making 259 length vector, and is given as input to the first layer. I find that inserting the latent vector again to the middle layers significantly improves the learning, denoted as dotted arrow in the diagram: the 259 vector is concatenated with the output of fourth fully connected layer to make a 512 vector. Final SDF value is obtained with hypberbolic tangent non-linear activation denoted as "TH".

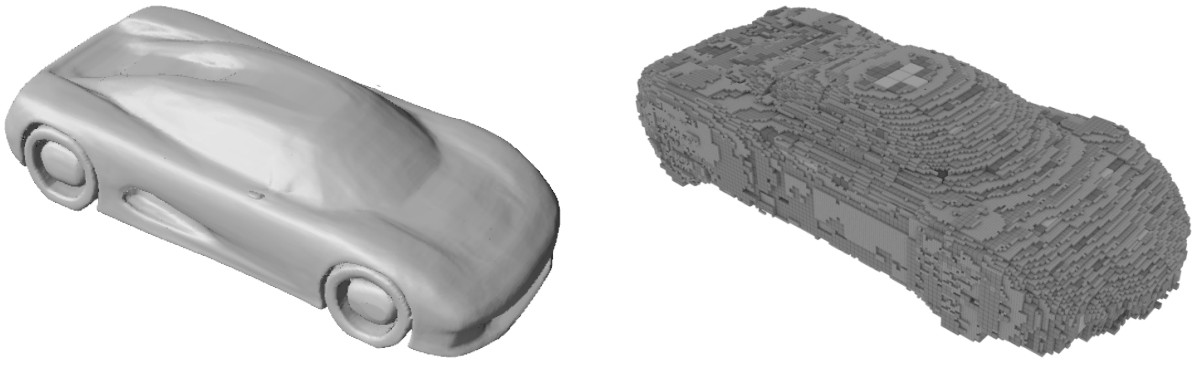


Figure 5.5: Compared to car shapes memorized using OGN [229] (right), our models (left) preserve details and render visually pleasing results as DeepSDF provides oriented surace normals.

Method	Type	Discretization	Complex topologies	Closed surfaces	Surface normals	Model size (GB)	Inf. time (s)	Eval. tasks
3D-EPN [50]	Voxel SDF	32^3 voxels	✓	✓	✓	0.42	-	C
OGN [229]	Octree	256^3 voxels	✓	✓		0.54	0.32	K
AtlasNet -Sphere [88]	Parametric mesh	1 patch		✓		0.015	**0.01**	K, U
AtlasNet -25 [88]	Parametric mesh	25 patches	✓			0.172	0.32	K, U
DeepSDF (ours)	Continuous SDF	**none**	✓	✓	✓	**0.0074**	9.72	K, U, C

Table 5.1: Overview of the benchmarked methods. AtlasNet-Sphere can only describe topological-spheres, voxel/octree occupancy methods (i.e. OGN) only provide 8 directions for normals, and AtlasNet does not provide oriented normals. Our tasks evaluated are: (K) representing known shapes, (U) representing unknown shapes, and (C) shape completion.

5.4 Data Preparation

To train my continuous SDF model, I prepare the SDF samples X (Eq. 5.2) for each mesh, which consists of 3D points and their SDF values. While SDF can be computed through a distance transform for any watertight shapes from real or synthetic data, I train with synthetic objects, (e.g. ShapeNet [33]), for which I are provided complete 3D shape meshes. To prepare data, I start by normalizing each mesh to a unit sphere and sampling 500,000 spatial points $\boldsymbol{x}$'s: I sample more aggressively near the surface of the object as I want to capture a more detailed SDF near the surface. For an ideal oriented watertight mesh, computing the signed distance value of $\boldsymbol{x}$ would only involve finding the closest triangle, but I find that human designed meshes are commonly not watertight and contain undesired internal structures. To obtain the *shell* of a mesh with proper orientation, I set up equally spaced virtual cameras around the object, and densely sample surface points, denoted P_s,

with surface normals oriented towards the camera. Double sided triangles visible from both orientations (indicating that the shape is not closed) cause problems in this case, so I discard mesh objects containing too many of such faces. Then, for each $\boldsymbol{x}$, I find the closest point in P_s, from which the $SDF(\boldsymbol{x})$ can be computed.

More specifically, I begin the data preparation by normalizing each shape so that the shape model fits into a unit sphere with some margin (in practice fit to sphere radius of 1/1.03). Then, I virtually render the mesh from 100 virtual cameras regularly sampled on the surface of the unit sphere. Next, I gather the surface points by back-projecting the depth pixels from the virtual renderings, and the points' normals are assigned from the triangle to which it belongs. Triangle surface orientations are set such that they are towards the camera. When a triangle is visible from both orientations, however, the given mesh is not watertight, making true SDF values hard to calculate, so I discard a mesh with more than 2% of its triangles being double-sided. For a valid mesh, I construct a KD-tree for the oriented surface points.

I want to emphasize that it is important that I sample more aggressively near the surface of the mesh as I want to accurately model the zero-crossings. Specifically, I sample around 250,000 points randomly on the surface of the mesh, weighted by triangle areas. Then, I perturb each surface point along all xyz axes with mean-zero Gaussian noise with variance 0.0025 and 0.00025 to generate two spatial samples per surface point. For around 25,000 points I uniformly sample within the unit sphere. For each collected spatial samples, I find the nearest surface point from the KD-tree, measure the distance, and decide the sign from the dot product between the normal and their vector difference.

5.5 Results

I conduct a number of experiments to show the representational power of DeepSDF, both in terms of its ability to describe geometric details and its generalization capability to learn a desirable shape embedding space. Largely, I propose four main experiments designed to test its ability to 1) represent training data, 2) use learned feature representation to reconstruct

Method \metric	CD, mean	CD, median	EMD, mean	EMD, median
OGN	0.167	0.127	**0.043**	**0.042**
AtlasNet-Sph.	0.210	0.185	0.046	0.045
AtlasNet-25	0.157	0.140	0.060	0.060
DeepSDF	**0.084**	**0.058**	**0.043**	**0.042**

Table 5.2: Comparison for representing known shapes (K) for cars trained on ShapeNet. CD = Chamfer Distance ($30,000$ points) multiplied by 10^3, EMD = Earth Mover's Distance (500 points).

unseen shapes, 3) apply shape priors to complete partial shapes, and 4) learn smooth and complete shape embedding space from which I can sample new shapes. For all experiments I use the popular ShapeNet [33] dataset.

I select a representative set of 3D learning approaches to comparatively evaluate aforementioned criteria: a recent octree-based method (OGN) [229], a mesh-based method (AtlasNet) [88], and a volumetric SDF-based shape completion method (3D-EPN) [50] (Table 5.1). These works show state-of-the-art performance in their respective representations and tasks, so I omit comparisons with the works that have already been compared: e.g. OGN's octree model outperforms regular voxel approaches, while AtlasNet compares itself with various points, mesh, or voxel based methods and 3D-EPN with various completion methods.

5.5.1 Representing Known 3D Shapes

First, I evaluate the capacity of the model to represent *known* shapes, i.e. shapes that were in the training set, from only a restricted-size latent code — testing the limit of expressive capability of the representations.

Quantitative comparison in Table 5.2 shows that the proposed DeepSDF significantly beats OGN and AtlasNet in Chamfer distance against the true shape computed with a large

number of points (30,000). The difference in earth mover distance (EMD) is smaller because 500 points do not well capture the additional precision. Figure 5.5 shows a qualitative comparison of DeepSDF to OGN.

5.5.2 Representing Test 3D Shapes (auto-encoding)

For encoding *unknown* shapes, i.e. shapes in the held-out test set, DeepSDF again significantly outperforms AtlasNet on a wide variety of shape classes and metrics as shown in Table 5.3. Note that AtlasNet performs reasonably well at classes of shapes that have mostly consistent topology without holes (like planes) but struggles more on classes that commonly have holes, like chairs. This is shown in Fig. 5.6 where AtlasNet fails to represent the fine detail of the back of the chair. Figure 5.7 shows more examples of detailed reconstructions on test data from DeepSDF as well as two example failure cases.

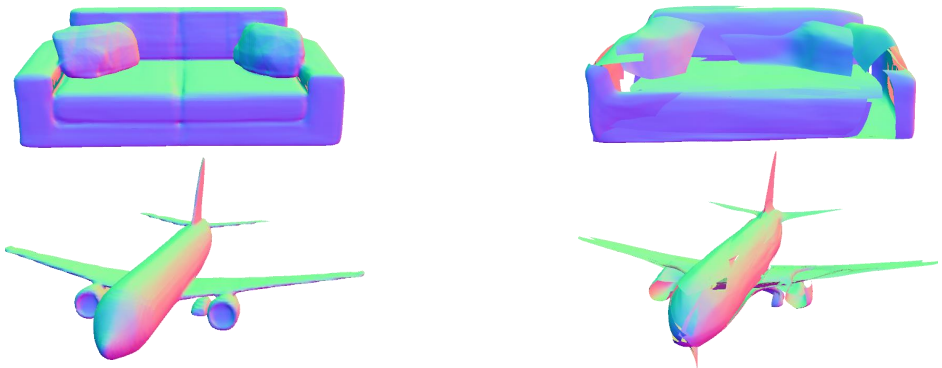
5.5.3 Shape Completion

A major advantage of the proposed DeepSDF approach for representation learning is that inference can be performed from an arbitrary number of SDF samples. In the DeepSDF framework, shape completion amounts to solving for the shape code that best explains a partial shape observation via Eq. 5.14. Given the shape-code a complete shape can be rendered using the priors encoded in the decoder.

I test my completion scheme using single view depth observations which is a common use-case and maps well to my architecture without modification. Note that the completion algorithm currently require the depth observations in the canonical shape frame of reference.

To generate SDF point samples from the depth image observation, I sample two points for each depth observation, each of them located η distance away from the measured surface point (along surface normal estimate). With small η I approximate the signed distance value of those points to be η and $-\eta$, respectively. I solve for Eq. 5.14 with loss function of Eq. 5.4 using clamp value of η. Additionally, I incorporate free-space observations, (i.e. empty-space

(a) Ground-truth (b) Our Result (c) [88]-25 patch (d) [88]-sphere



(e) Our Result (f) [88]-25 patch

Figure 5.6: Reconstruction comparison between DeepSDF and AtlasNet [88] (with 25-plane and sphere parameterization) for test shapes. Note that AtlasNet fails to capture the fine details of the chair, and that (f) shows holes on the surface of sofa and the plane.

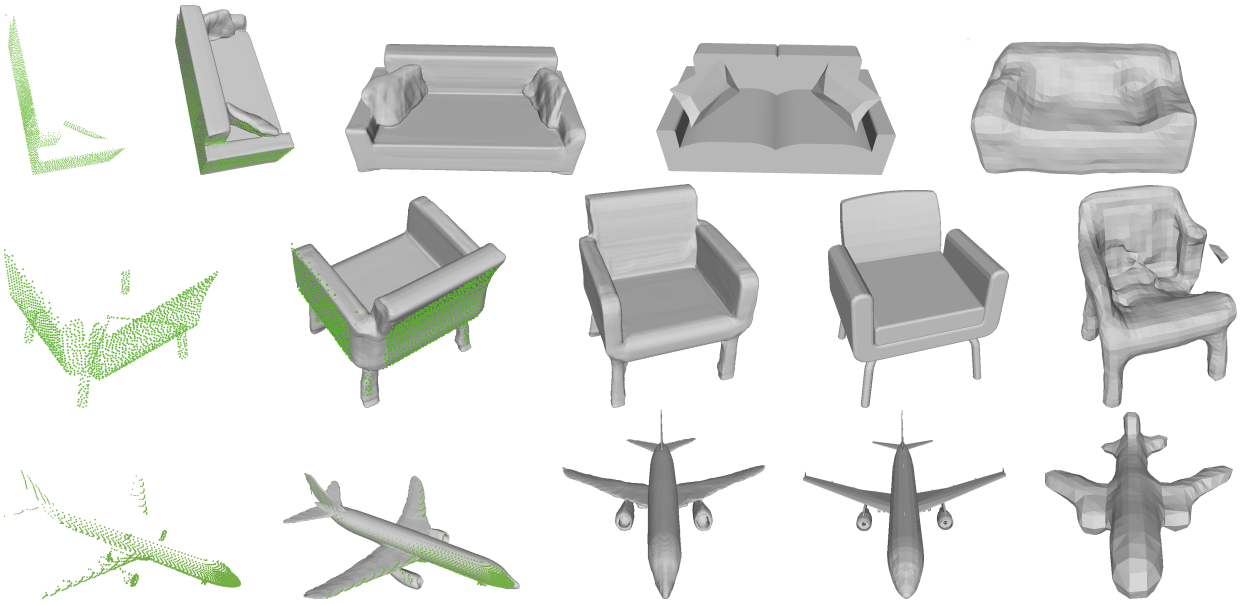
Figure 5.7: Reconstruction of test shapes. From left to right alternating: ground truth shape and our reconstruction. The two bottom right images show failure modes of DeepSDF. These failures are likely due to lack of training data and failure of minimization convergence.

between surface and camera), by sampling points along the freespace-direction and enforce larger-than-zero constraints. The freespace loss is $|f_\theta(\boldsymbol{z}, \boldsymbol{x}_j)|$ if $f_\theta(\boldsymbol{z}, \boldsymbol{x}_j) < 0$ and 0 otherwise.

Given the SDF point samples and empty space points, I similarly optimize the latent vector using MAP estimation. Tab. 5.4 and Figs. (5.13, 5.9) respectively shows quantitative and qualitative shape completion results. Compared to one of the most recent completion approaches [50] using volumetric shape representation, my continuous SDF approach produces more visually pleasing and accurate shape reconstructions. While a few recent shape completion methods were presented [94, 250], I could not find the code to run the comparisons, and their underlying 3D representation is voxel grid which I extensively compare against.

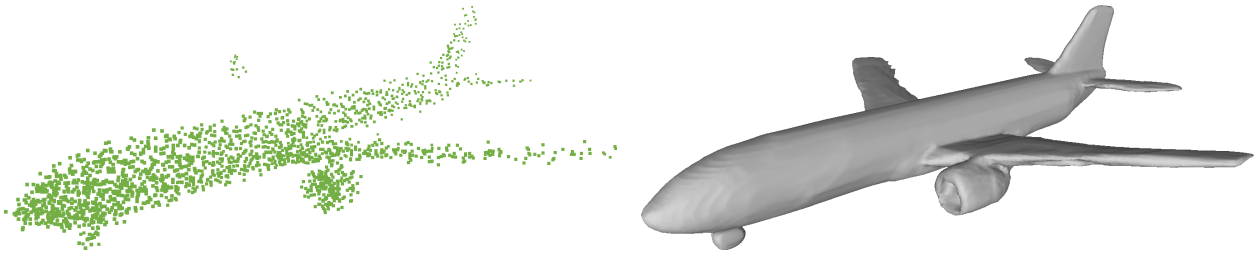
5.5.4 Details on Quantitative Metrics

The first two metrics, Chamfer and Earth Mover's, are easily applicable to points, meshes (by sampling points from the surface) and voxels (by sampling surface voxels and treating



(a) Input Depth (b) Completion (c) Second View (d) Ground truth (e) 3D-EPN

Figure 5.8: For a given depth image visualized as a green point cloud, we show a comparison of shape completions from our DeepSDF approach against the true shape and 3D-EPN. Left to right: Input depth map, our completion result from the input, rotated view of the completed result, ground truth of the second view, comparison with 3D-EPN [50].



(a) Noisy Input Point Cloud (b) Shape Completion

Figure 5.9: Demonstration of DeepSDF shape completion from a partial noisy point cloud. Input here is generated by perturbing the 3D point cloud positions generated by the ground truth depth map by 1.5% of the plane length.

their centers as points). When meshes are available, we also can compute metrics suited particularly for meshes: mesh accuracy, mesh completion, and mesh cosine similarity.

Chamfer distance is a popular metric for evaluating shapes, perhaps due to its simplicity [68]. Given two point sets S_1 and S_2, the metric is simply the sum of the nearest-neighbor distances for each point to the nearest point in the other point set.

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2$$

Note that while sometimes the metric is only defined one-way (i.e., just $\sum_{x \in S_1} \underset{y \in S_2}{min} ||x - y||_2^2$) and this is not symmetric, the sum of both directions, as defined above, is symmetric: $d_{CD}(S_1, S_2) = d_{CD}(S_2, S_1)$. In all of my experiments I report the Chamfer distance for 30,000 points for both $|S_1|$ and $|S_2|$, which can be efficiently computed by use of a KD-tree, and akin to prior work [88] I normalize by the number of points: I report $\frac{d_{CD}(S_1,S_2)}{30{,}000}$.

Earth Mover's distance [201], also known as the Wasserstein distance, is another popular metric for measuring the difference between two discrete distributions. Unlike the Chamfer distance, which does not require any constraints on the correspondences between evaluated points, for the Earth Mover's distance a bijection $\phi : S_1 \rightarrow S_2$, i.e. a one-to-one correspondence, is formed. Formally, for two point sets S_1 and S_2 of equal size $|S_1| = |S_2|$, the metric is defined via the optimal bijection [68]:

$$d_{EMD}(S_1, S_2) = \min_{\phi : S_1 \rightarrow S_2} \sum_{x \in S_1} ||x - \phi(x)||_2$$

Although the metric is commonly approximated in the deep learning literature [68] by distributed approximation schemes [18] for speed during training, I compute the metric accurately for evaluation using a more modest number of point samples (500) using [62].

Mesh accuracy, as defined in [207], is the minimum distance d such that 90% of generated points are within d of the ground truth mesh. I used 1,000 points sampled evenly from

the generated mesh surface, and computed the minimum distances to the *full* ground truth mesh.

Mesh completion, also as defined in [207], is the fraction of points sampled from the ground truth mesh that are within some distance Δ (a parameter of the metric) to the generated mesh. I used $\Delta = 0.01$, which well measured the differences in mesh completion between the different methods. With this metric the *full* generated mesh is used, and points (I used 1,000) are sampled from the ground truth mesh (mesh accuracy is vice versa). Ideal mesh completion is 1.0, minimum is 0.0.

Mesh cosine similarity is a metric I introduce to measure the accuracy of mesh normals. I define the metric as the mean cosine similarity between the normals of points sampled from the ground truth mesh, and the normals of the nearest faces of the generated mesh. More precisely, given the generated mesh M_{gen} and a set of points with normals S_{gt} sampled from the ground truth mesh, for each point x_i in S_{gt} I look up the closest face F_i in M_{gen}, and then compute the average cosine similarity between the normals associated with x_i and F_i,

$$\text{Cos. sim}(M_{gen}, S_{gt}) = \frac{1}{|S_{gt}|} \sum_{x_i \in S_{gt}} \hat{n}_{F_i} \cdot \hat{n}_{x_i} \ ,$$

where each $\hat{n} \in \mathbb{R}^3$ is a unit-norm normal vector. I use $|S_{gt}| = 2,500$ and in order to allow for [88] which does not provide oriented normals, I compute the $\min(\cdot)$ over both the generated mesh normal and its flipped normal: $\min(\hat{n}_{F_i} \cdot \hat{n}_{x_i}, -\hat{n}_{F_i} \cdot \hat{n}_{x_i})$. Ideal cosine similarity is 1.0, minimum (given the allowed flip of the normal) is 0.0.

5.5.5 Latent Space Shape Interpolation

To show that my learned shape embedding is complete and continuous, I render the results of the decoder when a pair of shapes are interpolated in the latent vector space (Fig. 5.10). The results suggests that the embedded continuous SDF's are of meaningful shapes and that my representation extracts common interpretable shape features, such as the arms of a chair, that interpolate linearly in the latent space.

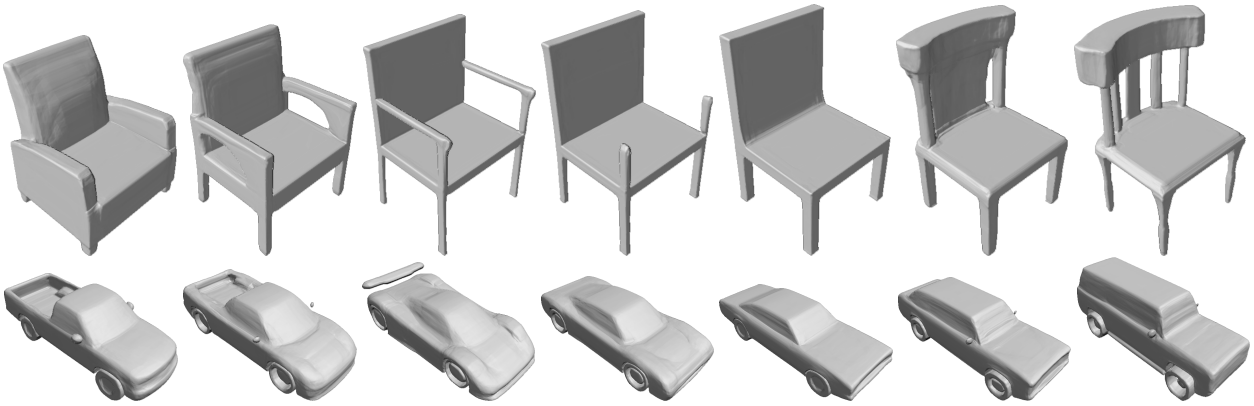


Figure 5.10: DeepSDF interpolation in the latent space. DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

5.5.6 Additional Results

I provide additional results on representing test objects with trained DeepSDF (Fig. 5.11, 5.12). The success of this task for DeepSDF implies that 1) high quality shapes similar to the test shapes exist in the embedding space, and 2) the codes for the shapes can be found through simple gradient descent.

Finally I show additional shape completion results on unperturbed depth images of synthetic ShapeNet dataset (Fig. 5.13), demonstrating the quality of the auto-decoder learning scheme and the new shape representation.

5.6 Conclusion & Future Work

DeepSDF significantly outperforms the applicable benchmarked methods across shape representation and completion tasks and simultaneously addresses the goals of representing complex topologies, closed surfaces, while providing high quality surface normals of the shape. However, while point-wise forward sampling of a shape's SDF is efficient, shape completion

Figure 5.11: Additional test shape reconstruction results. Left to right alternatingly: DeepSDF reconstruction and ground truth.

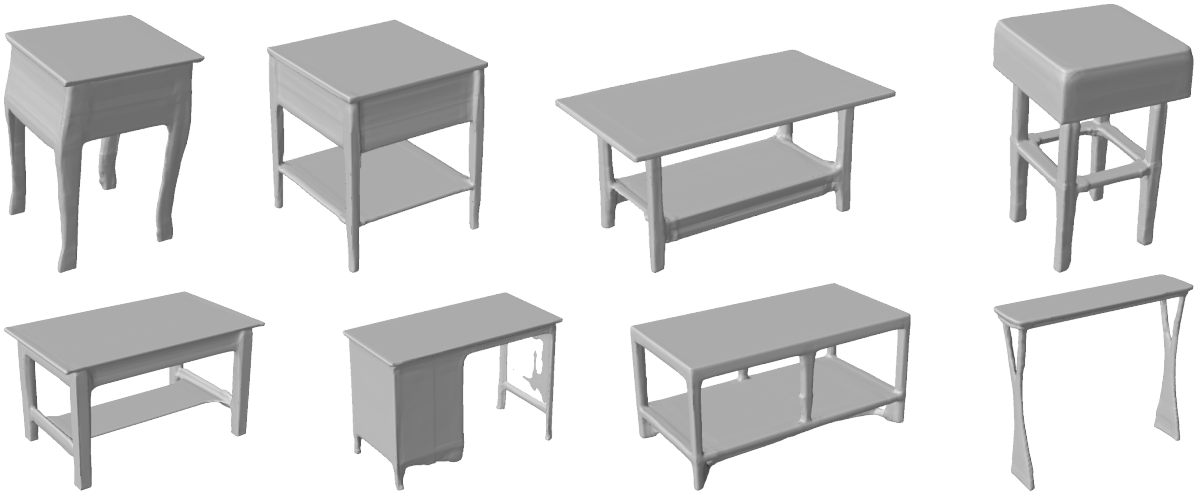


Figure 5.12: Additional test shape reconstruction results for Table ShapeNet class. All of the above images are test shapes represented with our DeepSDF network during inference time, showing the accuracy and expressiveness of the shape embedding.

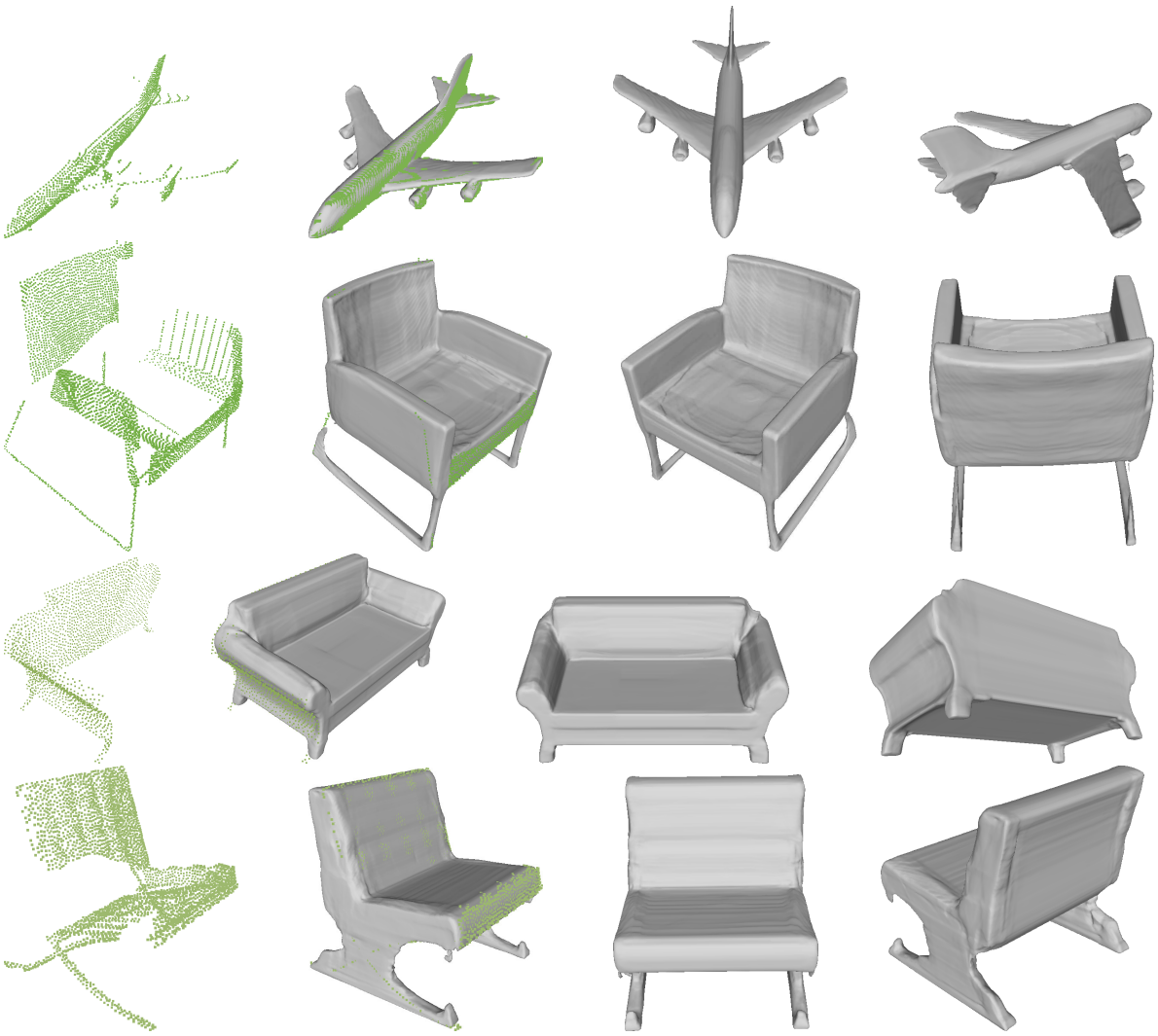


Figure 5.13: Additional shape completion results. Left to Right: input depth point cloud, shape completion using DeepSDF, second view, and third view.

(auto-decoding) takes considerably more time during inference due to the need for explicit optimization over the latent vector. I look to increase performance by replacing ADAM optimization with more efficient Gauss-Newton or similar methods that make use of the analytic derivatives of the model.

DeepSDF models enable representation of more complex shapes without discretization errors with significantly less memory than previous state-of-the-art results as shown in Table 5.1, demonstrating an exciting route ahead for 3D shape learning. The clear ability to produce quality latent shape space interpolation opens the door to reconstruction algorithms operating over scenes built up of such efficient encodings. However, DeepSDF currently assumes models are in a canonical pose and as such completion in-the-wild requires explicit optimization over a $SE(3)$ transformation space increasing inference time. Finally, to represent the true *space-of-possible-scenes* including dynamics and textures in a single embedding remains a major challenge, one which I continue to explore.

CD, mean	chair	plane	table	lamp	sofa
AtlasNet-Sph.	0.752	0.188	0.725	2.381	0.445
AtlasNet-25	0.368	0.216	**0.328**	1.182	0.411
DeepSDF	**0.204**	**0.143**	0.553	**0.832**	**0.132**
CD, median					
AtlasNet-Sph.	0.511	0.079	0.389	2.180	0.330
AtlasNet-25	0.276	0.065	0.195	0.993	0.311
DeepSDF	**0.072**	**0.036**	**0.068**	**0.219**	**0.088**
EMD, mean					
AtlasNet-Sph.	0.071	0.038	0.060	0.085	0.050
AtlasNet-25	0.064	0.041	0.073	0.062	0.063
DeepSDF	**0.049**	**0.033**	**0.050**	**0.059**	**0.047**
Mesh acc., mean					
AtlasNet-Sph.	0.033	0.013	0.032	0.054	0.017
AtlasNet-25	0.018	0.013	0.014	0.042	0.017
DeepSDF	**0.009**	**0.004**	**0.012**	**0.013**	**0.004**
Mesh comp., mean	chair	plane	table	lamp	sofa
AtlasNet-Sph.	0.668	0.862	0.755	0.281	0.641
AtlasNet-25	0.723	0.887	0.785	0.528	0.681
DeepSDF	**0.947**	**0.943**	**0.959**	**0.877**	**0.931**
Mesh comp., median					
AtlasNet-Sph.	0.686	0.930	0.795	0.257	0.666
AtlasNet-25	0.736	0.944	0.825	0.533	0.702
DeepSDF	**0.970**	**0.970**	**0.982**	**0.930**	**0.941**
Cosine sim., mean					
AtlasNet-Sph.	0.790	0.840	0.826	0.719	0.847
AtlasNet-25	0.797	0.858	0.835	0.725	0.826
DeepSDF	**0.896**	**0.907**	**0.916**	**0.862**	**0.917**

Table 5.3: Comparison for representing unknown shapes (U) for various classes of ShapeNet. Lower is better for the first four metrics and higher is better for the rest. Please refer to Sec. 5.5.4 for details on the metrics used.

	lower is better				*higher is better*	
Method / Metric	CD Median	CD Mean	EMD	Mesh acc.	Mesh comp.	Cos sim.
chair						
3D-EPN	2.25	2.83	0.084	0.059	0.209	0.752
DeepSDF	**1.28**	**2.11**	**0.071**	**0.049**	**0.500**	**0.766**
plane						
3D-EPN	1.63	2.19	0.063	0.040	0.165	0.710
DeepSDF	**0.37**	**1.16**	**0.049**	**0.032**	**0.722**	**0.823**
sofa						
3D-EPN	2.03	2.18	0.071	0.049	0.254	0.742
DeepSDF	**0.82**	**1.59**	**0.059**	**0.041**	**0.541**	**0.810**

Table 5.4: Comparison for shape completion (C) from partial range scans of unknown shapes from ShapeNet. For the evaluation metric, CD = Chamfer Distance, EMD = Earth Movers Distance.

Chapter 6

MODELING TRANSFORMATIONS FOR NEURAL IMPLICIT REPRESENTATION

In Chapter 5, I introduced a novel 3D representation, DeepSDF, that presents state-of-the-art results in 3D shape modeling. In particular, I proposed a representation and techniques that could complete unseen parts of a shape in a plausible way, reducing the user efforts during scanning, as motivated in Sec. 1 (Criteria 3).

In this chapter, I propose an extention to DeepSDF by introducing a novel generative model, which I call *SceneSDF*. SceneSDF is able to handle objects in non-canonical poses, i.e., the latent space of this model contains objects in various rotations and translations. As such, this approach is able to reconstruct scenes with multiple objects from noisy and partial observations, an important capability for applying the new implicit representations for casual 3D reconstruction.

Since DeepSDF is trained on a dataset of aligned objects in a canonical pose, the learned shape representation cannot be directly used for reconstructing real-world objects in arbitrary poses. Therefore, approaches for modeling multi-object scenes using DeepSDF depend on a series of steps including object detection, shape code estimation, and pose optimization [203, 157].

Directly applying DeepSDF to be trained on an unaligned object dataset, however, would be ineffective, because the DeepSDF network takes the absolute coordinates as input. For example, a local shape feature learned at $x = 0$ cannot be reused to describe a similar feature at coordinate $x = 1$. Thus the network has to waste a significant amount of expressive power on separately representing an identical shape in different locations. This limitation has prevented the original DeepSDF formulation from being used for modeling non-aligned,

large-scale scenes with more than one object [115, 30, 262].

Motivated by this observation, I divide the target scene volume into a grid of voxels, where each voxel contains a *local* feature vector that encodes the local SDF within the voxel via DeepSDF. The use of a discrete feature volume limits the space of shapes that a single MLP network needs to model and promotes reusability of the local features across scenes. Note that, as voxels are tiled translationally, the use of a grid naturally promotes translational but not rotational equivariance. An interesting future direction is to introduce explicit equivariance to rotation into the model.

However, when the generative model using the grid structure is used for inference reconstruction, I observe that the optimization in the latent space often fails to converge to a desirable state. I hypothesize that the few thousand training scene is not enough compactly cover the latent space for smooth optimization convergence during inference. While it is possible to use more training data, preparing and processing large amounts of training data is expensive, so I seek to *virtually* augment the data by adopting adversarial training. That is, I densely sample the latent space and ensure that the sampled latent codes are mapped to a plausible shapes via an adversarial loss. Experiments show that the effectiveness of adversarial training, as shown in Sec. 6.4.1. Using the trained generative model (SceneSDF), reconstructing a multi-object scene amounts to simply optimizing for the randomly initialized latent vectors using a Expectation-Maximization-style algorithm [169] (describe in Section 6.3.2), replacing more complex methods [203, 157].

The main contribution of this Chapter are 1) a novel method of training a generative adversarial network (GAN) in the space of voxelized local features for simultaneously modeling an object shape and its spatial transformation, and 2) an EM-style algorithm [169] to jointly optimize multiple latent codes to reconstruct scenes with more than one objects.

6.1 Related Work

I review two main areas of related work: modeling object transformations and multi-object scenes.

6.1.1 Modeling Transformation of objects

3D entities in the world come with various translations, rotations, and scales with respect to their surroundings. Since the space of transformed objects is much larger than the space of objects itself, it is efficient to separately represent an object and its transformation. Spatial Transformer Networks [113] predict affine transformation parameters to obtain an image in a canonical form. A classification network that operates on the aligned images then only needs to learn how to recognize entities in canonical pose. PointNet [191] adopts a similar strategy to predict a 3D transformation matrix to transform a point cloud in canonical pose. Alias-free GAN [121] learns to sample a 2D rotation and translation parameters to generate realistic images. Similarly, GIRAFFE [178] independently samples 3D transformation and object appearance to generate compositional scenes without supervision.

6.1.2 Learning-based Multi-Object and Scene Modeling

2D Domain

Learning to represent scenes with many entities (e.g., objects) is a challenging task, as the space of possible scenes grows combinatorially with the number of possible entities and their spatial arrangements. In the 2D domain, several works have explored modeling an image with a set of latent vectors to decompose the space of scenes into manageable pieces [108, 90].

Slot Attention [148] and IODINE [86] learn, without supervision, to decompose a multi-object scene image into a set of latent vectors, each of them representing one object. GANsformer [108]takes a different approach where the latent vectors collaborate each other, instead of independently take charge of one object, to generate a compositional image.

3D Domain

In the 3D domain, [264] applies the Slot Attention approach to model compositional scenes using multiple latent vectors. [30] uses a grid of features to describe large scenes, where each voxel feature describes local SDFs via the neural implicit representaion. [187] applies

convolution layers on the voxel features to learn geometric scene priors, while [60] uses the same voxelized local features to learn a space of scene appearance.

6.2 Learning the Latent Space of Shapes and Transformations

This section describes the proposed technical approach for training a generative model for simultaneously modeling object shapes and transformations. In Sec. 6.2.1, I motivate the use of a grid structure for modeling local SDFs. Next, I describe learning the latent space of transformed objects from surface points via an auto-decoder training scheme (Sec. 6.2.2). To enforce that the whole latent space generates plausible shapes, in Sec. 6.2.3, I adopt an adversarial training scheme where a discriminator encourages the generative model to produce realistic feature volumes.

6.2.1 Representing Local SDFs with a Grid Structure

Chapter 5 introduces a global signed distance field (DeepSDF) to represent a shape as a function of global 3D coordinate $\boldsymbol{x}$:

$$f_\theta(\boldsymbol{c}, \boldsymbol{x}) = s : \boldsymbol{x} \in \mathbb{R}^3, s \in \mathbb{R}, \boldsymbol{c} \in \mathbb{R}^D, \tag{6.1}$$

where f_θ is a neural network parameterized by θ, and $\boldsymbol{c}$ is a D dimensional latent code controlling the shape.

While this formulation works well in the absence of global transformations (i.e., when the target objects are aligned), it does not efficiently model objects in arbitrary poses, as the dependence of the global coordinate prevents reusing local shape features in different locations (as shown in Fig. 6.2). Recent approaches in NIR address this problem by dividing the target volume into a grid of features, where each feature represents local SDFs or radiance fields [30, 262, 60]. The use of the grid structure allows the MLP network to focus on the local features that could be shared within and across scenes. Motivated by this finding, I adopt a 3D generator that learns to produce a feature volume, which is a combination of smaller local geometric pieces.

The 3D CNN generator g_κ with parameter κ produces a regular grid of voxels F that covers the target volume:

$$g_\kappa(\boldsymbol{z}) = F, \boldsymbol{z} \in \mathbb{R}^Z, F \in \mathbb{R}^{L \times 32 \times 32 \times 32}, \tag{6.2}$$

where $\boldsymbol{z}$ is a Z-dimensional global latent code, whose distribution will be learned via auto-decoder (Sec. 6.2.2) or adversarial training (Sec. 6.2.3). Each voxel is associated with a local feature vector $\boldsymbol{l} \in \mathbb{R}^L$ that encodes the local geometry using a shared DeepSDF network f_θ operating on the local coordinate system:

$$f_\theta(\boldsymbol{l}, \boldsymbol{x}^l) = s : \boldsymbol{x}^l \in \mathbb{R}^3, s \in \mathbb{R}, \boldsymbol{l} \in \mathbb{R}^L, \tag{6.3}$$

where $\boldsymbol{x}^l$ is the local xyz coordinate around the center of the voxel. To enforce that the predicted SDF value is consistent between neighbors, I let the volume of influence of each voxel feature to overlap with its neighbors', and encourage (during training) that the SDF predictions are consistent across neighboring feature vectors. Specifically, each $\boldsymbol{l}$ is trained to cover a symmetrically extended volume of size $2p \times 2p \times 2p$, where p is the length of the voxel edge. During training, I introduce a consistency loss (defined in Eq. 6.7) to ensure that SDF predictions are consistent across the overlapping voxel features.

Even with the consistency loss, when visualizing the global SDFs via raycasting or mesh extraction, there could be remaining discontinuities at the voxel borders where the feature vector association abruptly changes. To prevent undesirable artifacts, I blend the SDF predictions of neighboring voxels via trilinear interpolation [10]:

$$SDF(\boldsymbol{x}) = \sum_{t \in \mathcal{N}^8} \frac{V_t}{V} \cdot f_\theta(\boldsymbol{l}_t, \boldsymbol{x} - \boldsymbol{c}_t), \tag{6.4}$$

where $\mathcal{N}^8$ is a set of 8 closest voxels for the point $\boldsymbol{x}$, and $\boldsymbol{l}_t$ and $\boldsymbol{c}_t$ are respectively the latent code and center of the neighbor voxel t. The voxel weights V_t in the equation is the volume of the opposite rectangular cuboid formed between $\boldsymbol{x}$ and $\boldsymbol{c}_t$'s, following trilinear interpolation [10]. $V = \sum_{t \in \mathcal{N}} V_t$ is a normalization factor.

Overall, the discretized local SDF representation allows reusing the learned local geometric features. Furthermore, the translational invariance of the convolutional architecture for

the generator network is suited for generating objects at arbitrary locations. Finally, I note that because the local features represent *continuous* SDFs via DeepSDF, we only need to have a relatively sparse resolution voxel grid (e.g. 32 x 32 x 32) to represent high quality shapes.

6.2.2 Auto-Decoder Training from Surface Points

Generator Architecture

Inspired by the success of the 2D generators that use discrete convolution layers to produce realistic images [122], I design a 3D CNN architecture for the 3D generator g_κ. I adapt the StyleGAN architecture to have only 10 convolutional layers because the target output resolution is 32 x 32 x 32, and use SWISH non-activation functions instead of ReLU because SWISH is continuous everywhere (the continuity property enables computing second-order gradient in Sec.6.2.2). Overall, as described in Eq. 6.2, $g_\kappa(\boldsymbol{z})$ takes a latent code $\boldsymbol{z}$ and generates a volume of L dimensional features.

Auto-Decoder Formulation

In Sec. 6.2.1, I described representing unaligned object SDFs with a grid of local features that encode local SDFs. These feature volumes are produced by a 3D CNN generator given a global latent code. Now the important question is: how do we learn the latent space of such feature volumes and the object SDFs produced by them?

As in Chapter 5, I adopt the auto-decoder algorithm to learn a latent space of unaligned shapes. Different from Chapter 5, however, I do not precompute the SDF values for training, but instead indirectly obtain the scene SDFs from surface point samples and their normal directions via solving a differential equation. I adopt this strategy because directly computing SDFs is computationally expensive due to the need to find the nearest surface. During training, I randomly choose a subset of an object's surface points (1024 points out of 1 million) for each batch iteration. Formally, similar to Eq. 5.13, I jointly optimize a set of

latent codes $\boldsymbol{z}$'s (one for each transformed object) along with the generator g and the local DeepSDF network f:

$$\underset{\theta,\kappa,\{\boldsymbol{z}_i\}_{i=1}^{N}}{\arg\min} \sum_{i=1}^{N} \left(\sum_{j=1}^{K} \mathcal{L}_{AD}(\boldsymbol{x}_j, \boldsymbol{n}_j, \boldsymbol{z}_i; g_\kappa, f_\theta) + \frac{1}{\sigma^2} ||\boldsymbol{z}_i||_2^2 \right), \tag{6.5}$$

where $\mathcal{L}_{AD}$ is the loss function that ensures SDF value is zero on the surface, and surface normal is close to the ground truth $\boldsymbol{n}_j$. As a result of the auto-decoder training, we are left with the trained generator g and DeepSDF network f along with the latent code $\boldsymbol{z}_i$ for each scene in the dataset.

For inference, I optimize a randomly initialized latent code to fit the observed surface points while keeping the network weights fixed:

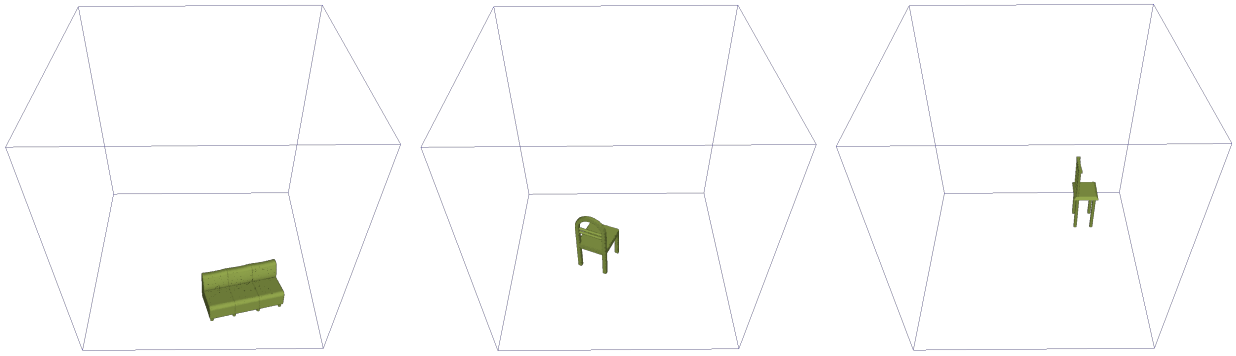
$$\underset{\boldsymbol{z}}{\arg\min} \sum_{j=1}^{K} \mathcal{L}_{AD}(g_\kappa, f_\theta, \boldsymbol{z}, \boldsymbol{x}_j, \boldsymbol{n}_j) + \frac{1}{\sigma^2} ||\boldsymbol{z}||_2^2. \tag{6.6}$$

I sample the initial code near zero (i.e., low standard deviation Guassian) following Chapter 5, and minimize Eq. 6.6 using the ADAM optimizer [132].

Training Data

I sample oriented surface points from the ShapeNet chair dataset using the procedure explained in Section 5.4 (I find it sufficiently expressive to sample one million surface points per object). Then, I place each object (i.e., the object's sampled surface points) onto a 5m x 5m x 5m canvas, such that the bottom of the object is on the XY plane. I uniformly sample the translation of the object within the 5m x 5m XY plan, and apply random rotation along the Z axis to simulate most real-world scenes where the objects are standing upright. Then, I normalize the 5m volume to a length-one cuboid such that each axis ranges from 0 to 1. Finally, I divide the ShapeNet chair dataset into 2,000 training and 500 test sets for all the experiments. Figure 6.1 shows examples of the prepared dataset.

In Chapter 5, the training of the DeepSDF network requires a set of point samples and their ground truth SDF values for each scene. However, computing the SDFs is computationally expensive as we need to find the closest surface points for each point sample. Therefore, I



(a) (b) (c)

Figure 6.1: Example training scenes. Each scene contains a ShapeNet [33] chair object, which is randomly translated and rotated on the XY plane. The length of the edge of the target cube volume is 5 meter and is normalized to unit length 1 after placing the object. The blue lines describe the boundary of the target volume.

adopt the derivative based techniques from [87, 5, 214] to indirectly learn SDFs from surface point samples only. These methods leverage the fact that the derivative of the SDF with respect to the *xyz* coordinate has magnitude of 1 and amounts to the surface normal for a surface point. This approach is shown to increase the quality of the generated surfaces [87]. Therefore, I prepare surface point samples along with their surface normals for a dataset of unaligned shapes.

Loss Function

I learn the latent space of transformed shapes from oriented surface points using the auto-decoder formulation of Eq. 6.5. The loss function $\mathcal{L}_{AD}$ for a surface point $\boldsymbol{x}$ from a scene represented by a latent code $\boldsymbol{z}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{AD}(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{z}; \kappa, \theta) = \lambda_S \psi_{\kappa,\theta}(\boldsymbol{x}, \boldsymbol{z}) + \lambda_N \|\nabla_{\boldsymbol{x}} \psi_{\kappa,\theta}(\boldsymbol{x}, \boldsymbol{z}) - \boldsymbol{n}\| + \lambda_E \|\,|\nabla_{\boldsymbol{x}'} \psi_{\kappa,\theta}(\boldsymbol{x}', \boldsymbol{z})| - 1\| \\ + \lambda_M M(\psi_{\kappa,\theta}(\boldsymbol{x}'', \boldsymbol{z})) + \lambda_C C_{\kappa,\theta}(\boldsymbol{x}'', \boldsymbol{z})) + \lambda_T T(\psi_{\kappa,\theta}(\boldsymbol{x}'', \boldsymbol{z})),\end{aligned} \tag{6.7}$$

where $\psi_{\kappa,\theta}$ is an operator that evaluates SDF at a point by selecting the nearest voxel feature from $g_\kappa(\boldsymbol{z})$ and subsequently computing the local SDF using f_θ as described in Eq. 6.3.

At a surface point $\boldsymbol{x}$, the first two terms of Eq. 6.7 respectively encourage that the SDF value is zero and the surface normal (analytically computed as the gradient of SDF with respect to $\boldsymbol{x}$) is equal to the ground truth value $\boldsymbol{n}$. Furthermore, for each surface point $\boldsymbol{x}$, I randomly sample another point $\boldsymbol{x}'$ near $\boldsymbol{x}$ to encourage the modeled SDF to be correct, i.e., the magnitude of the gradient of SDF is equal to 1. More specifically, I model the truncated version of SDFs by enforcing the third term in the loss function (referred to as the Eikonal term in [87]) for randomly sampled $\boldsymbol{x}'$ near the surface such that $\|\boldsymbol{x} - \boldsymbol{x}'\| < \tau$, for some truncation value τ. That is, I encourage modeling proper SDF within the truncation distance from the surface.

In the above equation, $M(s) = \exp(-\alpha \cdot |s|), \alpha \gg 1$ in the fourth term penalizes non-surface points from having SDF values close to zero (i.e., prevents spurious surfaces). Here, $\boldsymbol{x}''$ is another random point uniformly sampled from the target volume. $C_{\kappa,\theta}$ is a consistency loss that measures the absolute difference between $\psi_{\kappa,\theta}(\boldsymbol{x}'', \boldsymbol{z})$ and the SDF predicted by associating $\boldsymbol{x}''$ with a voxel randomly chosen from its 8 nearest voxels. This term ensures that the solution of the differential equation is globally consistent. Finally, $T(s) = \max(\tau, s)$ in the last term encourages SDF values to not go above the truncation value τ. Note that λ's in the equation are balancing scalar parameters for the loss function.

6.2.3 Adversarial Training in the Latent Space

In the previous section, I adopted auto-decoder algorithm introduced in Chapter 5 to learn a latent space of transformed shapes. However, the space of objects under arbitrary transformations is much larger than the space of aligned objects used in Chapter 5. Therefore, I hypothesize that the same auto-decoder algorithm requires significantly more training scenes to obtain a complete and smooth latent space through gradient-based optimization. In fact, while I was able to successfully reconstruct *aligned* test shapes during inference time in Chapter 5, inference optimization for non-aligned shapes in this chapter fails when using the

networks trained via the auto-decoder training (Eq. 6.5), as shown in Figure 6.2.

While we might increase the number of training scenes to address this issue, it comes with significantly more computational cost, as the auto-decoder training involves jointly optimizing for the scenes' latent vectors. Therefore, I take a different approach of virtually "augmenting" the data via adversarial training. To this end, I introduce a discriminator 3D CNN that processes 3D feature volumes. I note that the feature volumes obtained with the auto-decoder training (Sec. 6.2.2) can represent a distribution of real shapes. Similar to typical GAN training approaches [83], I randomly sample latent codes to generate feature volumes via my generator. Then, the discriminator takes the feature volumes as input and outputs the real/fake labels that serve as a training signal to penalize unrealisitc feature volumes. The adversarial training encourages the latent space to be densely packed with plausible feature volumes, which contrasts with the auto-decoder training that only uses a few thousand latent space samples.

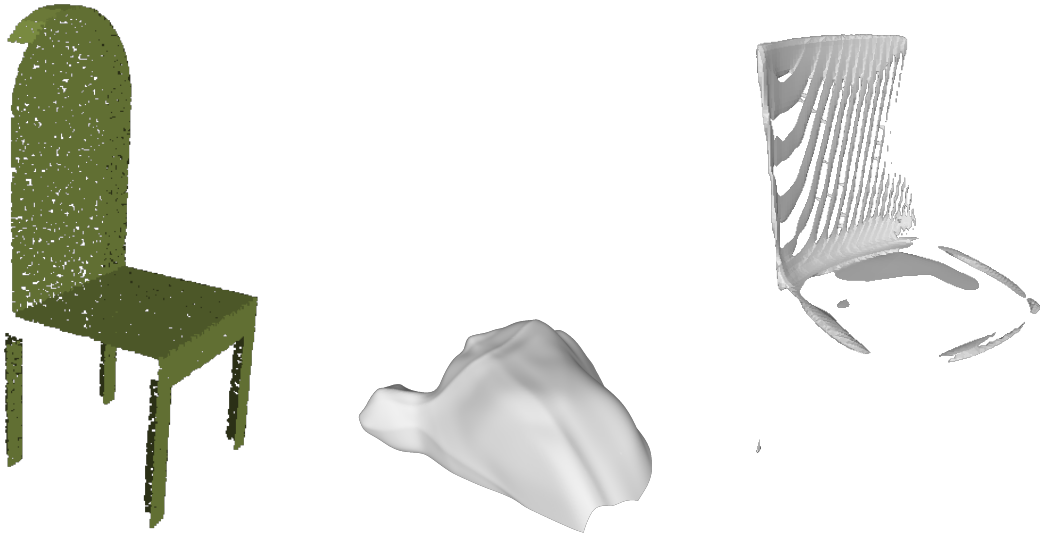
Formally, let $\{F\}_{i=1}^{N}$ be the set of feature volumes obtained from the auto-decoder training of Sec. 6.2.2, and let d_δ be the discriminator with parameter δ that processes F's and predicts real/fake labels. We can then jointly train the generator and discriminator using the combination of auto-decoder and the adversarial loss:

$$\arg\min_{\kappa,\{\boldsymbol{z}_i\}_{i=1}^{N}} \max_{\delta} \sum_{i=1}^{N} \|g_\kappa(\boldsymbol{z}_i) - F_i\|_1 + \frac{1}{\sigma^2}||\boldsymbol{z}_i||_2^2 + \mathcal{L}_{GAN}(g_\kappa, d_\delta). \tag{6.8}$$

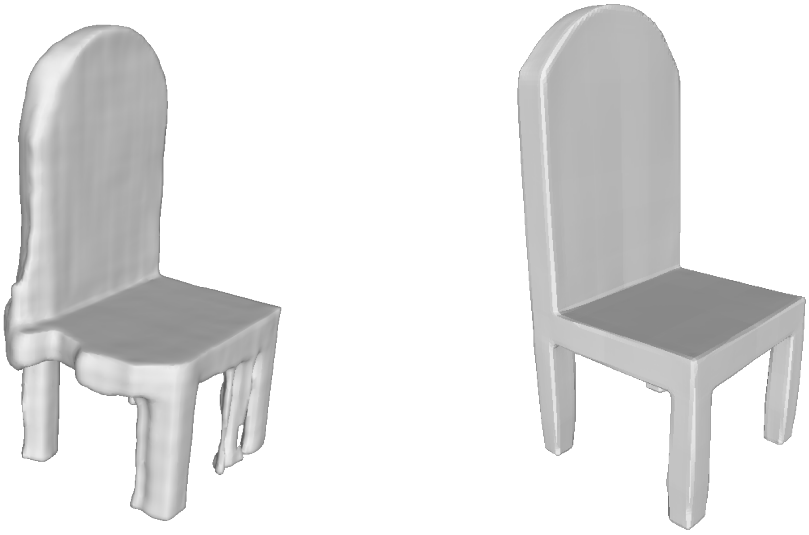
Here, $\mathcal{L}_G$ defines the adversarial loss of the well-known *min-max game* optimization [83] as follows:

$$\mathcal{L}_{GAN}(g_\kappa, d_\delta) = \mathbb{E}_F[d_\delta(F)] + \mathbb{E}_{\boldsymbol{\zeta}}[1 - d_\delta(g_\kappa(\boldsymbol{\zeta}))], \tag{6.9}$$

where the latent code $\boldsymbol{\zeta}$ is sampled from a Gaussian distribution with mean 0 and variance empirically computed from the $\boldsymbol{z}_i$'s obtained from the auto-decoder training of Eq. 6.5. The L_1 reconstruction term and the regularization term (second term) of Eq.6.8 are auto-decoder losses which ensure that the training scenes are incorporated in the latent space. I find the adversarial training to be unstable without these auto-decoder terms. I name my proposed learning method of Eq. 6.8 as *adversarial auto-decoder* algorithm, due to the existence of



(a) Depth Map (b) DeepSDF (c) Grid+Auto-decoder



(d) DeepSDF-PE (e) Grid + Adversarial

Figure 6.2: Given a depth map (a) of an object under a random transformation, original DeepSDF (b) and the voxel grid generator trained on pure auto-decoder loss (c) was unable to reconstruct the object through optimization. While positional encoding on DeepSDF improves optimization convergence (d), the shape's surface quality is much worse than that of the full SceneSDF model (e) trained with the adversarial auto-decoder algorithm.

both auto-decoder and the adversarial terms. Overall, the adversarial auto-decoder learning scheme allows training with dense samples of the latent space, which enables successful inference optimization as I will show in the experiment section.

6.3 Multi-Object Scene Reconstruction

In the previous section, I introduced a generative model for single objects in arbitrary poses, leveraging my new proposed learning algorithm. Now the question to ask is: How can we use this to model scenes with multiple objects?

Since each latent vector for the trained generative model produces a single object in non-canonical pose, it is reasonable to combine multiple latent vectors to describe a scene with multiple objects. What is not straightforward, however, is optimizing for the multiple latent vectors. In this case, it is ambiguous which latent vector should be associated with a point for optimization. When a latent vector is associated with points from multiple objects, the optimization of the latent code to model more than one object could result in erroneous shape completion.

6.3.1 Min-Composite Approach

Perhaps the simplest solution to this optimization problem would be to represent a multi-code scene via min-composite of multiple SDFs. That is, I associate a point to a latent vector that predicts the smallest SDF value at that position. The validity of the minimum operation comes from the definition of SDF, which measures the *closest* distance to all existing surfaces from a point, and thus, associating a point with the latent vector that generates the closest surface to that point is theoretically sound. Formally, when we have M randomly initialized global latent codes to describe a scene with M or fewer objects, we can write the min-composite optimization to fit the given point observations as follows:

$$\arg\min_{\{\boldsymbol{z}_i\}} \sum_{i=1}^{M} \sum_{j=1}^{K} \mathbb{1}_{i,j} \mathcal{L}_{AD}(g_\kappa, f_\theta, \boldsymbol{z}_i, \boldsymbol{x}_j, \boldsymbol{n}_j) + \frac{1}{\sigma^2} ||\boldsymbol{z}_i||_2^2, \tag{6.10}$$

where $\mathbb{1}_{i,j}$ is an indicator function that is 1 when the ith latent code produces the smallest SDF at $\boldsymbol{x}_j$, and 0 otherwise.

The above approach of labeling each point using the min-operation, however, often leaves some objects unreconstructed or results in optimization failure. This is because the use of hard (all-or-nothing) assignment of the input points makes the algorithm susceptible to being trapped in local minima depending on the initial assignments.

6.3.2 Expectation-Maximization Style Approach

Rather than assigning hard labels to each point, we compute the probability of each point belonging to a latent code. This problem of fitting multiple generative models to data of unknown labels is analogous to the well known problem of finding mixture model parameters given unclustered data points. While numerous methods have been proposed for fitting mixture models to data (see [162] for a comprehensive review), many of these approaches separately solve for the hidden variables (point labels in our problem) and the model parameters in an alternating fashion, following the pioneering Expectation-Maximization (EM) algorithm [169]. Similarly, I adopt the strategy of computing the soft point labels and optimizing the latent vectors in an alternating fashion.

Setup

I initialize M number of latent vectors $\{\boldsymbol{z}_i\}$ randomly drawn from a Gaussian distribution with mean 0 and variance close to 0 (in practice I use 0.01). The input to the algorithm is partial surface observations of a multi-object scene, e.g., a depth map. I assume that the actual number of objects in the target scene is less than or equal to M, but the exact number is unknown to the algorithm. Lastly, I assume we have the trained networks g_κ and f_θ from the adversarial training of Eq. 6.8, which can generate a scene's SDF given a latent vector.

E-Step

In this step, I estimate the soft assignment of each observed point to the latent vectors. Specifically, I compute the probability that an observed surface point $\boldsymbol{x}_i$ is assigned to the latent vector $\boldsymbol{z}_j$: $P(\boldsymbol{x}_i \in L_j|\boldsymbol{x}_i)$, where L_j is a set of all points explained by $\boldsymbol{z}_j$, i.e., the closest surface point for any point in L_j belongs to the surface generated by $\boldsymbol{z}_j$. To compute this probability, I use the Bayes' Law:

$$P(\boldsymbol{x}_i \in L_j|\boldsymbol{x}_i) = \frac{P(\boldsymbol{x}_i|\boldsymbol{x}_i \in L_j)P(L_j)}{P(\boldsymbol{x}_i)}. \tag{6.11}$$

Here, $P(\boldsymbol{x}_i|\boldsymbol{x}_i \in L_j)$ is the probability that $\boldsymbol{x}_i$ is observed as surface (i.e., SDF at $\boldsymbol{x}_i$ is 0) given that it belongs to L_j. Given that the SDF prediction at $\boldsymbol{x}_i$ is $\psi_{\kappa,\theta}(\boldsymbol{x}_i, \boldsymbol{z}_j)$ (as defined in Eq. 6.7), I model $P(\boldsymbol{x}_i|\boldsymbol{x}_i \in L_j)$ using a Gaussian distribution with mean $\psi_{\kappa,\theta}(\boldsymbol{x}_i, \boldsymbol{z}_j)$ and standard deviation σ_E:

$$P(\boldsymbol{x}_i|\boldsymbol{x}_i \in L_j) = \mathcal{N}_{pdf}(0; \psi_{\kappa,\theta}(\boldsymbol{x}_i, \boldsymbol{z}_j), \sigma_E^2), \tag{6.12}$$

where $\mathcal{N}_{pdf}$ outputs the probability density function and σ_E could be estimated empirically from the auto-decoder training (Eq. 6.5).

$P(L_j)$ in Eq 6.11 is the label prior which is the probability that a random point belongs to L_j, which will be estimated in next step. The probability that $\boldsymbol{x}_i$ is a surface point without knowing its label $P(\boldsymbol{x}_i)$ can be computed via marginalizing the joint probability as follows:

$$P(\boldsymbol{x}_i) = \sum_j P(\boldsymbol{x}_i, \boldsymbol{x}_i \in L_j) = \sum_j P(L_j)P(\boldsymbol{x}_i|\boldsymbol{x}_i \in L_j). \tag{6.13}$$

Finally, then, we can rewrite Eq. 6.11 as follows:

$$P(\boldsymbol{x}_i \in L_j|\boldsymbol{x}_i) = \frac{P(L_j)\mathcal{N}_{pdf}(0; \psi_{\kappa,\theta}(\boldsymbol{x}_i, \boldsymbol{z}_j), \sigma_E^2)}{\sum_j P(L_j)\mathcal{N}_{pdf}(0; \psi_{\kappa,\theta}(\boldsymbol{x}_i, \boldsymbol{z}_j), \sigma_E^2)}. \tag{6.14}$$

The soft label $P(\boldsymbol{x}_i \in L_j|\boldsymbol{x}_i)$ is computed for each surface observation $\boldsymbol{x}_i$ in the E-step. Note that label prior $P(L_j)$ is initialized as $\frac{1}{M}$ and updated in the M-step described below.

M-Step

In this step, I optimize by model parameters (i.e., the latent vector $\boldsymbol{z}_i$'s) and update the label priors. Using the soft labels computed in the E-step, in Eq. 6.14, I take a gradient step to minimize the following weighted loss:

$$\arg\min_{\{\boldsymbol{z}_i\}} \sum_{j=1}^{M} \sum_{i=1}^{K} P(\boldsymbol{x}_i \in L_j | \boldsymbol{x}_i) \mathcal{L}_{AD}(g_\kappa, f_\theta, \boldsymbol{z}_j, \boldsymbol{x}_i, \boldsymbol{n}_i) + \frac{1}{\sigma^2} ||\boldsymbol{z}_j||_2^2, \tag{6.15}$$

where $\mathcal{L}_{AD}$ is the auto-decoder loss on the oriented point cloud observations ($\boldsymbol{x}_i$'s and $\boldsymbol{n}_i$'s) as defined in Eq. 6.7.

Finally, I update the label prior by summing up the soft label for each latent vector:

$$P(L_j) = \frac{\sum_j P(\boldsymbol{x}_i \in L_j | \boldsymbol{x}_i)}{M}. \tag{6.16}$$

In order to prevent some of the $P(L_j)$'s from quickly converging to zero in the beginning of the optimization (because their surfaces are initialized far from any of the points), I adopt a deferred update strategy by updating the label priors once every ten iterations. The E-step and M-step are executed alternatingly for each iteration.

Note that in the special case of $M = 1$, the EM-style algorithm reverts back to the auto-decoder inference of Eq. 6.6, as the label $P(\boldsymbol{x}_i \in L_0 | \boldsymbol{x}_i) = 1$ for all points.

6.4 Experiments

In this experiment, I conduct experiments to show the ability of the SceneSDF model to reconstruct unaligned single objects and multi-object scenes. For these experiments, I use the popular ShapeNet [33] chair dataset – the chair class presents sufficiently complex and diverse structures for 3D shape and appearance research [231, 264].

6.4.1 Representing Unaligned Single Objects

I compare SceneSDF against a variant of DeepSDF on reconstructing *unknown* shapes with arbitrary poses, i.e., shapes in the held-out test set. Given a partial point cloud of an aligned object, I run the inference algorithm of Sec. 6.3.2.

(a) Depth Map (b) SceneSDF (c) DeepSDF PE

Figure 6.3: Given a depth map (a) of an object under a random transformation, the shape completion result of SceneSDF algorithm (b) is of higher quality than the result of DeepSDF with positional encoding (c).

As shown in Figure 6.2, notice that the original DeepSDF formulation fails to reconstruct an unaligned object via optimization, so I cannot compare the SceneSDF against the original DeepSDF. To present a reasonable baseline, I follow the recent works on NIR [167, 214] and apply the positional encoding to the *xyz* coordinate inputs to improve the expressive power and optimizability of DeepSDF. Formally, I adopt this positional encoding to augment each coordinate into a $\mathcal{R}^{2L}$ dimensional feature using a set of $2L$ harmonic functions [167]:

$$\gamma(x) = (\sin(2^0\pi x), \cos(2^0\pi x), \ldots, \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x)). \tag{6.17}$$

Interestingly, the positional encoding formulation is closely related to the local coordinate system used in this chapter, as they both introduce periodic functions. Applying a sine function with frequency, for instance, 64 to the *xyz* coordinate is similar to representing the target volume with a 32-resolution voxel grid. However, the positional encoding version of DeepSDF (from now I call this PE-DeepSDF) critically differs from my generative model as it lacks the convolution layers that promote interactions between coordinate features.

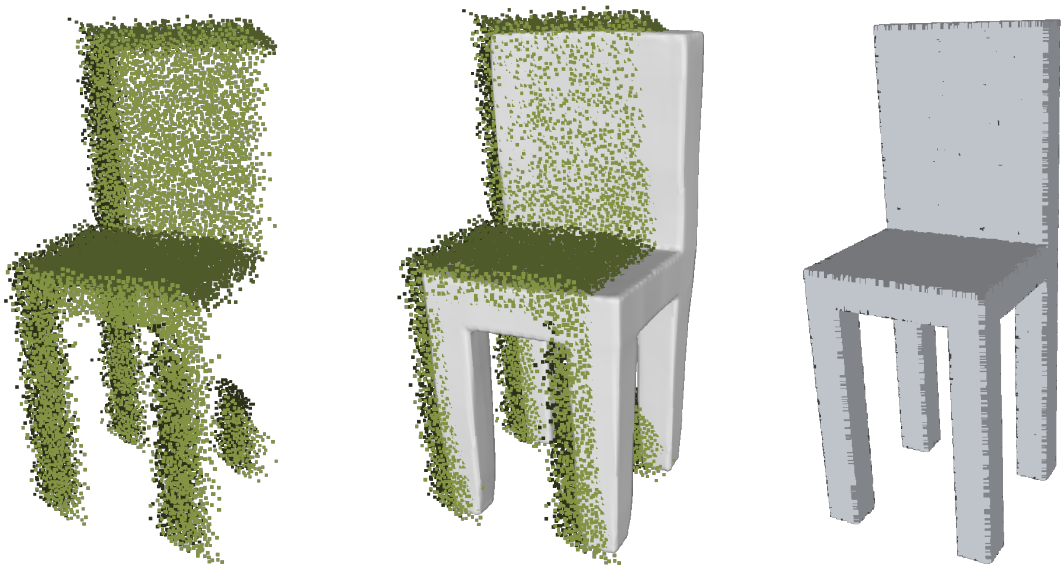
To compare SceneSDF against PE-DeepSDF, I conduct the EM optimization of Eq. 6.15 with $M = 2$, given depth maps of 200 held-out test objects. Similarly, I run the auto-decoder inference algorithm of Eq. 5.14 to optimize the latent vector given the oriented point cloud. Because I optimize two latent vectors for SceneSDF, I ran the PE-DeepSDF experiment twice for each scene and used the better result. As shown in Fig. 6.3, the SceneSDF generative model clearly produces a higher quality shape compared to PE-DeepSDF. The quantitative results show that the proposed generative model significantly outperforms in terms of Chamfer distance defined in Chapter 5: the average and median Chamfer distance for SceneSDF were 1.578 and 0.4007, respectively, compared to the mean of 3.3655 and the median of 1.1498 for PE-DeepSDF.

Finally, to show the robustness of the algorithm to noisy input data, I run the inference optimization on a noisy depth point cloud. I follow the procedure described in Chapter 5, to simulate the noise pattern common to structure-light depth sensors. The result shown in Fig. 6.4 demonstrates the robustness of the generative model for fitting noisy, partial data.

6.4.2 Representing Multi-Object Scenes

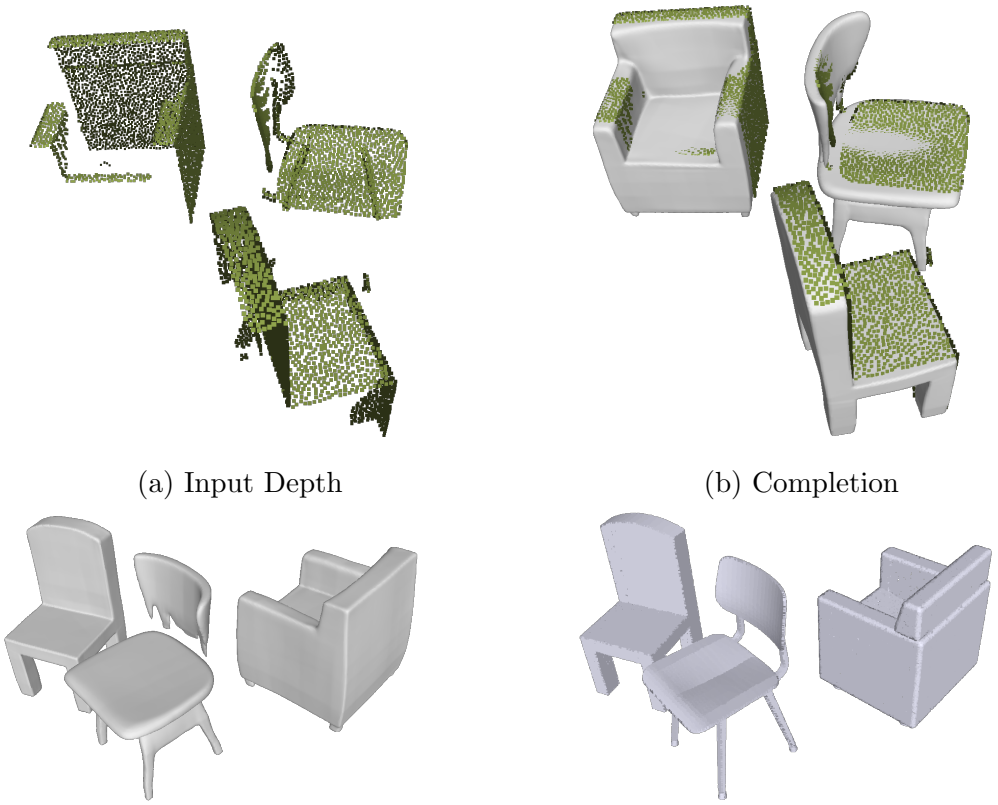
In this experiment, I demonstrate SceneSDF's ability to reconstruct multi-object scenes using the EM-style algorithm described in Sec. 6.3.2. Fig 6.5 visualizes an example input point cloud and shape completion result for the scene. As can be seen in the figure, the depth map for a multi-object scene is highly sparse because the objects occlude themselves (self-occlusion) and each other, making the shape completion even more challenging.

Given a scene with up to three non-overlapping shapes, I initialize six latent vectors sampled from a Gaussian distribution with mean zero and standard deviation sampled from a uniform distribution: $\sigma \sim U(0.4\sigma_E, 0.6\sigma_E)$, where σ_E is the standard deviation computed from the resulting latent vectors of the auto-decoder training in Eq. 6.5. Given the initialized latent vectors and the partial point cloud, I jointly optimize the six latent vectors to minimize the loss of Eq. 6.15, whose soft-labels are updated after each gradient step according to Eq. 6.14. The results in Fig. 6.5 and Fig. 6.7 present the reconstruction result, demonstrating



(a) Noisy Depth Map (b) Reconstruction (c) Ground Truth

Figure 6.4: Noisy depth completion result. (a) shows a depth map with added Gaussian noise (on the depth value) with standard deviation 4% of the radius of the circumscribed sphere of the object. (b) shows the reconstruction using the SceneSDF model compared to the ground truth (c).



(a) Input Depth (b) Completion

(c) Second View (d) Ground truth

Figure 6.5: Multi-object completion result. For a given depth image of a scene with three objects visualized as a green point cloud (a), I run my shape completion algorithm of Sec. 6.3.2. The completion result (b,c) is of high quality and close to the ground truth shapes (d).

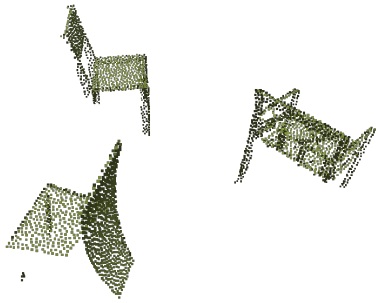
the strong shape priors learned by the generative model to plausibly complete the unobserved regions of the scene. That is, the reconstruction results correctly capture the overall shapes of the chairs, such as the existence of arms and curved backs (Fig. 6.5).

However, note that the reconstructions miss the fine details of the point clouds, such as the holes on the back of the chairs in Fig. 6.6. These artifacts are likely due to the lack of expressive power of the generator network, which might be addressed by using a network with more number of free parameters. Another possible remedy to these artifacts is to boost the empty space loss on the important regions (e.g., holes on the back of the chairs). As described in Eq. 6.5, the empty space loss is currently applied to random points that are uniformly sampled from the target volume. Instead, sampling more aggressively in the perceptually important regions could help modeling the fine details.

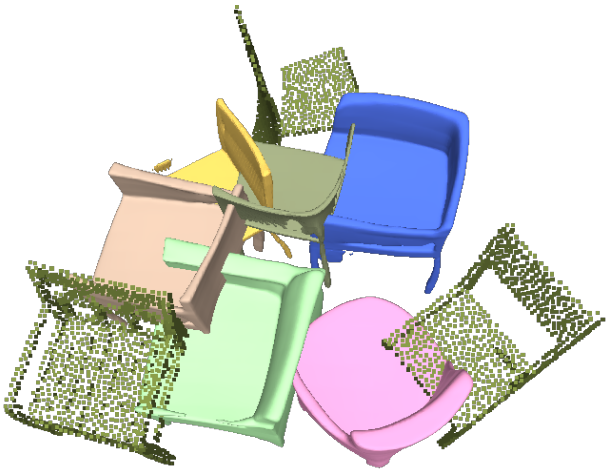
In Figure 6.6, I visualize the shapes generated by the latent vectors before and after the optimization, where the corresponding shape pairs are displayed with the same colors. Note how the initial shapes near target objects are snapped into the partial points and generate completed shapes. The label priors $P(L_j)$'s, which are the byproducts of the EM-style optimization, provide a convenient way of rejecting the unused shapes that are not snapped into the target point clouds in the image. That is, I reject shapes generated from the latent vectors with $P(L_j) < 0.01$ (less than one percent of points are associated with this latent code)

6.5 Conclusion

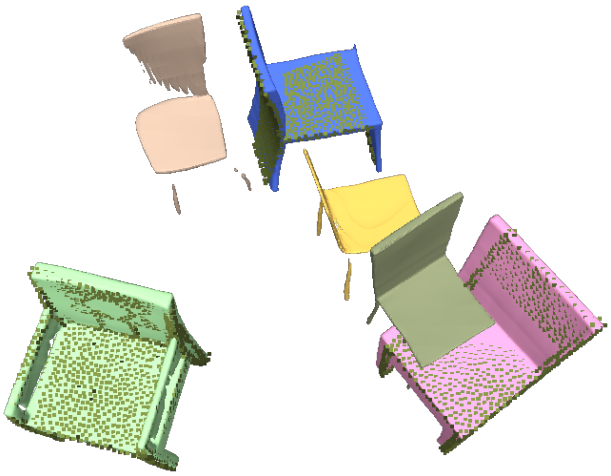
In this chapter, I introduced a generative model (ScenceSDF) for unaligned objects and an algorithm to fit the model to a multi-object scene. SceneSDF is trained using the *adversarial auto-decoder* algorithm, which simultaneously minimizes the training data reconstruction loss (auto-decoder) and the adversarial loss in the feature space. The adoption of adversarial loss induces a generative model with a smooth latent space, enabling reconstruction of unaligned objects. I also propose an EM-style algorithm to reconstruct multiple shapes from an unsegmented partial point cloud.



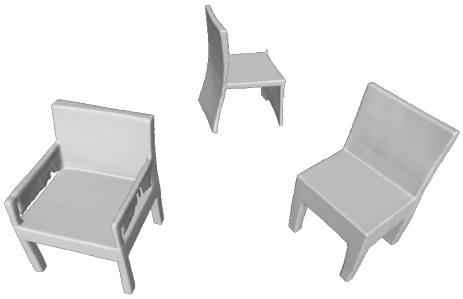
(a) Input Depth Points



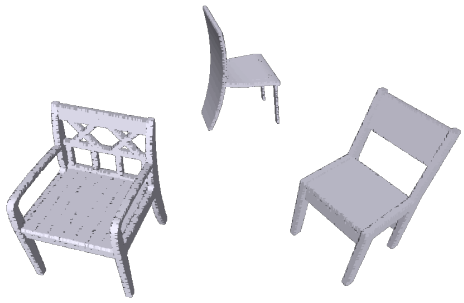
(b) Initial State + Input Points



(c) Final State + Input Points

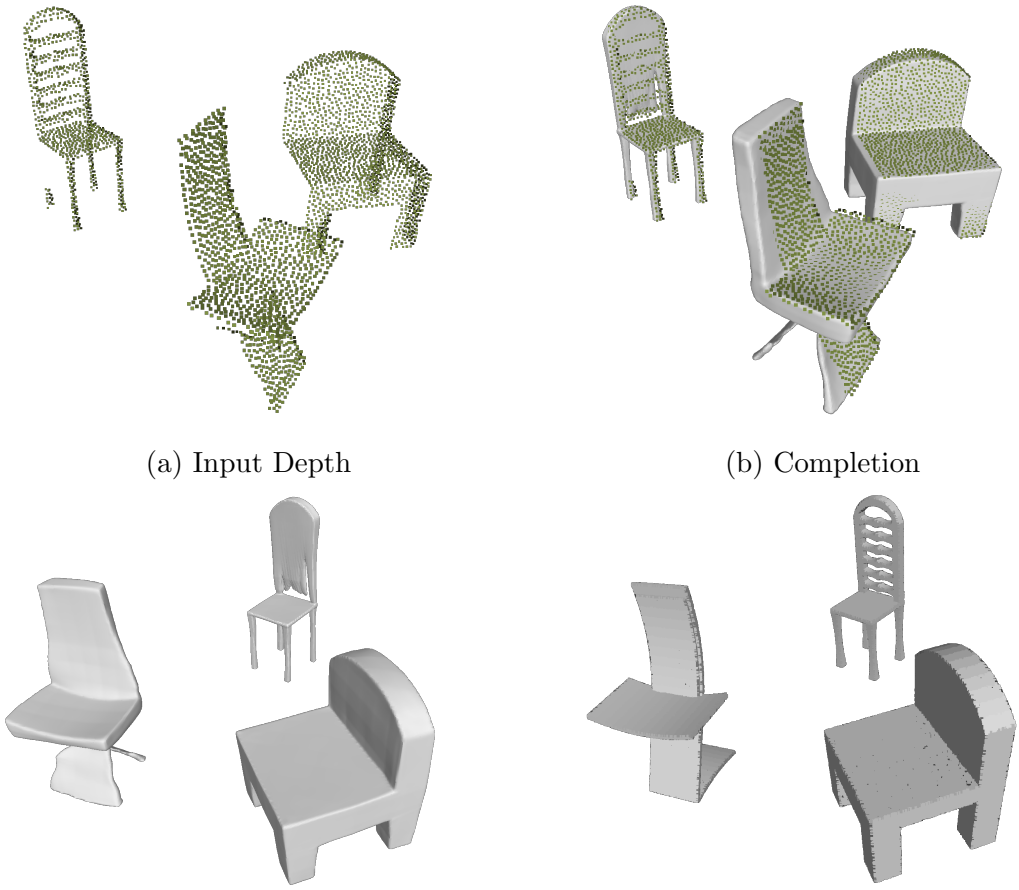


(d) Reconstruction (excess removed)



(e) Ground Truth

Figure 6.6: Visualizing initial and final states of multi-object completion. Given a depth map points of a multi-object scene shown as the green point cloud (a), I randomly initialize 6 latent vectors (b) and optimize them to fit to the input points (c). The shapes' colors of (b,c) visualize the initial and final states of each latent vector. We can identify and remove the objects unused for reconstruction using the label priors of Eq. 6.16. (d) shows a rendering of the valid reconstruction from a viewpoint compared against the ground truth (c).



(a) Input Depth (b) Completion

(c) Completion (Second View) (d) Ground truth

Figure 6.7: Additional multi-object completion result. For a given depth image of a scene with three objects visualized as a green point cloud (a), I run my shape completion algorithm of Sec. 6.3.2. The completion result (b,c) is of high quality and close to the ground truth shapes (d).

The computer vision, graphics, and robotics communities have proposed various approaches [274, 181] for registering existing shape models to partial data such as point clouds or RGB images. These methods typically involve a complex pipeline of detecting target shapes, estimating object poses (relative to camera or to some canonical pose), and retrieving nearest neighbor shapes from a data bank [6, 7, 111, 236]. My proposed generative model and inference algorithm suggests a new possibility of reconstructing objects via random initialization and optimization of latent vectors, which could significantly simplify the reconstruction effort. Note that my EM-style inference algorithm is closely related to joint semantic segmentation and 3D reconstruction approaches [97, 95] that formulate a feedback loop between the two tasks. On the contrary, my use of object-level generative model provides much stronger geometric priors that allow completion of unobserved regions.

Moreover, training a scene/object completion network typically requires artificially corrupting the input data or generating scenes with "realistic" object distributions to faithfully simulate the real-world, noisy inputs [50, 49, 187]. My algorithm reconstructs test scenes via an optimization in the latent space of a generative model that is trained on uncorrupted synthetic data. In contrast, typical completion networks [50, 49, 187] are discriminative models that directly process noisy, partial input data, and thus they require the training data to be similar to the real-world inputs.

One limitation is that the inference optimization currently takes a few minutes to fully converge, which is much slower than existing feed-forward-based approaches. Furthermore, the use of grid-structure and convolution layers naturally induce translational invariance, but not rotational invariance. An interesting future direction is to apply a learned, rigid transformation to each object as explored in Alias-Free GAN [121]. Finally, the proposed multi-object reconstruction via SceneSDF has not yet been shown to work for real-world scenes. Real scene experiments would require additional steps for identifying and removing non-object elements (e.g., floors and walls), and a generative model trained on multiple object classes. I leave this experiment as a future work.

Chapter 7
CONCLUSION AND FUTURE WORK

My thesis presented new techniques, systems, concepts, and algorithms that push the boundary of 3D reconstruction in two directions that trade-off photo-realism and ease of capture. Specifically, my work has made the following contributions towards the goal of photorealistic 3D reconstruction from casual scanning:

- **Casual surface light field reconstruction:** different from existing methods that require costly laboratory setups [84, 245, 34, 246], my proposed system, described in Chapter 3, reconstructs photo-realistic surface light fields of a scene using a hand-held RGB-D camera. This is possible via my theoretical formulation of factoring scene appearance into view-independent and wavelength-independent components, which enables recovering scene specular highlights using the IR laser that comes with a commodity RGB-D sensor.

- **Recovering detailed environment image from a video:** in Chapter 4, I introduced an approach to automatically reconstruct a detailed image of an environment from a video taken using a hand-held camera. This approach is made possible via jointly optimizing per-material specular reflection maps and material segmentation neural network. Due to the exceptional quality of recovered environment images, I believe that the technique can be used for forensic studies.

- **Faithful and robust appearance modeling via neural rendering:** in Chapter 4, I introduced a neural-network-based rendering technique that achieves photo-realistic quality, robust view extrapolation, and robust modeling of interreflections, Fresnel, and concave surfaces. A key to these capabilities is the combination of physically-based and

learning-based technologies, where the rendering network takes physically-motivated input layers to produce realistic view prediction.

- **Effective 3D representation for deep-learning:** in Chapter 5, I proposed a new 3D representation suitable for deep learning. Different from existing 3D representations including triangle meshes, point clouds, or voxels, this approach classifies whether an *xyz* point is inside or outside of an object shape, representing the surface as the decision boundary of this classifier. Together with adopting an encoder-less learning algorithm (*auto-decoder*), I show that the new representation promotes highly effective learning of 3D shapes from large datasets, evidenced by enabling high quality object completion.

- **Multi-object scene modeling using generative models:** in Chapter 6, I extended the approach in Chapter 5 to apply to multi-object scenes. I proposed a method of training a generative model for simultaneously modeling object shape and its transformation, and an EM [169] style approach for reconstructing scenes with more than one object using multiple latent vectors. These two contributions allow shape completion of multi-object scenes via simple gradient-based optimization in the latent space of a generative model.

7.1 Future Work

In this section, I discuss future research directions by answering the following three questions: 1) What is keeping us from using existing technologies to easily scan real scenes? 2) How can we improve the current NIR approaches? 3) What is an important research topic beyond photo-realistic reconstruction?

7.1.1 Casual 3D Reconstruction

Computer vision community has introduced a number of approaches for reconstructing realistic 3D models using hand-held cameras (e.g., multi-view stereo [74, 79, 73] or RGB-D

camera-based methods [183, 242, 273]). Notably, [242] demonstrated large-scale indoor reconstruction with highly accurate geometry and texture, in addition to the ability to model mirrors and glass surfaces. The biggest drawback of these approaches regarding casual scanning would be the lack of scene completion. Without the completion ability, an user has to scan every corner of a scene (as in [242]), which is prohibitively expensive for non-professionals.

Despite the progress in object-level shape and appearance modeling [182, 256], scene-level completion approaches typically produce unfaithful, low-quality results [49, 187]. This is likely because the space of possible scenes is exponentially larger than the space of objects. An interesting future direction would be to automatically decompose the complex scenes into more manageable pieces (e.g., objects and parts) and model the relationships between these entities via, for example, Natural Language Processing techniques that study relationships between discrete words and phrases. Training a generative model of scenes, perhaps by leveraging the decomposition, is another promising direction, as having a scene generative model could reduce the reconstruction task into a simple gradient-based optimization in the latent space.

Faithful scene completion would require learning scene priors from large datasets. As shown in Chapter 5, the choice of 3D representation plays an important role on the performance of machine learning algorithms in 3D. While my proposed NIR has gained popularity due to its differentiability, expressiveness, and efficiency (in space and time), the NIR-related approaches still face many challenges, as will be discussed in the next section. Therefore, a crucial future work is to explore new representations or improve existing representations to further develop scene-scale 3D deep learning.

7.1.2 Neural Implicit Representation

The methods I introduced in this thesis enable machines to infer beyond what's recorded in traditional media, such as recovering detailed scene environment from a video, rendering a scene from uncaptured viewpoints, or completing invisible parts of objects. I believe that

these technologies will contribute towards making high quality 3D capture accessible to a wide range of users. My neural network-based implicit representation (NIR) (Chapter 5,6) has inspired a large amount of follow-up research in 3D reconstruction [167, 178, 185, 5, 204]. Some of these methods improved the accuracy and robustness of appearance [167, 244, 185] and geometry [238] reconstruction to the level previously unseen in the research community. However, many of these methods fit a neural network to a set of images from a single scene, and thus they do not generalize well to viewpoints far from the input range. NIR approaches that do generalize across scenes, including that of Chapter 5, either focus on single object scenes [32, 182] or tend to produce lower quality results [262, 187, 60]. These gaps could be due to the lack of expressive power of the latent space (i.e., representing a scene with a single latent vector [60, 182, 32]). To increase the expressive power, we need to consider more complex latent representations, e.g., multi-slot representations of Chapter 6 or others [148, 108]. Another problem of NIR is that the memory and computational footprint of the volume rendering used in [32, 60, 262] is too high to train high quality generalizable models. This problem might be addressed by devising smarter sampling strategy, e.g., via combining radiance and SDF representations as in [238, 261]. Approaches that use discrete voxel grids to model local NIRs, including [262, 187] and Chapter 6, could suffer from the inherent difficulty of the convolution operators in modeling rotations, which could result in wasting representation power to model same features in different orientations. An interesting future direction is an effective modeling of transformations within a neural network, as tried in the 2D domain [113, 51, 121]. Finally, the lack of interactions between the point features of NIR (each point is independently processed via an MLP) might be fundamentally limiting the learning of higher-order logic within scenes, an important open problem for the research community.

7.1.3 Beyond photo-realism

While my proposed systems and techniques in Chapter 3 and 4 can produce faithful appearance modeling of target scenes, the resulting reconstructions remain static, meaning that we

(a) Original Scene (b) Rearranged Scene

Figure 7.1: Example of virtual scene modification. Given a 3D reconstruction of scene (a), changing the object arrangement as in (b) would require faithful scene completion and appearance modeling.

cannot move the objects around within the scene or turn off the lights, etc. In fact, most existing approaches in multi-view stereo [74, 79, 73], NIR [187, 167, 182], or dense SLAM [175, 184, 243] create *frozen* scenes that are difficult to interact with.

Beyond photo-realism, I argue that the ability to modify and interact with digital reconstructions will open up a diverse set of interesting applications. Take, for example, furniture shopping for non-professionals. The current way of planning a furniture and interior design change of your existing home involves "imagining" how it would look like when you paint this wall in some color or buy this sofa and place it by the wall. If we have a modifiable, realistic 3D reconstruction of your physical home, we could realistically simulate and visualize how it would look like when you apply the new paint, furniture, or lamps. Such application could greatly help the decision making process for interior change for non-professionals.

Interactive 3D reconstructions could also promote robotics research. Recent reinforcement learning algorithms typically train their models in an virtual environment, where ex-

periments for gradient steps could be run at much lower cost compared to real-world robotics experiments. However, these virtual environments are usually manually designed and come with a low level of visual realism (e.g., Gibson environment [253], House3D [251]), or AI2-THOR [59]. More realistic scanned datasets including Replica [221], on the other hand, do not allow scene interactions such as knocking over a chair or picking up an object as the reconstructions are composed of static triangle meshes. A dataset of dynamic, interactive, and photo-realistic 3D reconstructions, which we currently lack, will be invaluable to modern robotics research.

However, obtaining a modifiable 3D scanned scene is much more challenging than reconstructing a static model. From a simple mental exercise shown in Figure 7.1, we can think of what it takes to photo-realistically modify a scene. Changing the location of a kettle from the scene in the left image to that of the right involves inpainting unobserved parts of the kettle and the tabletop, re-rendering the specular highlights of the kettle based on the scene illumination, removing the incorrect shadows, and recomputing the new shadow cast by the kettle, etc. These tasks require not only a highly reliable scene completion algorithm but also accurate recovery of intrinsic material parameters (e.g., albedo or specularity), lighting conditions, and a way of photo-realistically rendering the modified scene in interactive time, the challenges that our community should continue to explore.

BIBLIOGRAPHY

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[2] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM Transactions on Graphics (TOG)*, 34(4):110, 2015.

[3] Kara-Ali Aliev, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

[4] Neil Alldrin, Todd Zickler, and David Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[5] Matan Atzmon and Yaron Lipman. Sald: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020.

[6] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2614–2623, 2019.

[7] Armen Avetisyan, Angela Dai, and Matthias Nießner. End-to-end cad model retrieval and 9dof alignment in 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2551–2560, 2019.

[8] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation. *arXiv preprint arXiv:1903.07145*, 2019.

[9] Timur Bagautdinov, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. Modeling facial geometry using compositional vaes. 1:1.

[10] Ying Bai and Dali Wang. On the comparison of trilinear, cubic spline, and fuzzy interpolation methods in the high-accuracy measurements. *IEEE Transactions on fuzzy Systems*, 18(5):1016–1022, 2010.

[11] Pierre Baqué, Edoardo Remelli, François Fleuret, and Pascal Fua. Geodesic convolutional shape optimization. *arXiv preprint arXiv:1802.04016*, 2018.

[12] Jonathan T Barron and Jitendra Malik. Shape, albedo, and illumination from a single image of an unknown object. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 334–341. IEEE, 2012.

[13] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1670–1687, 2014.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[15] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3479–3487, 2015.

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

[17] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[18] Dimitri P Bertsekas. A distributed asynchronous relaxation algorithm for the assignment problem. In *Decision and Control, 1985 24th IEEE Conference on*, pages 1703–1704. IEEE, 1985.

[19] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[20] Sai Bi, Xiaoguang Han, and Yizhou Yu. An l 1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics (TOG)*, 34(4):78, 2015.

[21] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*, 36(4), 2017.

[22] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam-learning a compact, optimisable representation for dense visual slam. *arXiv preprint arXiv:1804.00874*, 2018.

[23] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 600–609. PMLR, 10–15 Jul 2018.

[24] Messaoud Bouakkaz and Mohamed-Faouzi Harkat. Combined input training and radial basis function neural networks based nonlinear principal components analysis model applied for process monitoring. In *IJCCI*, pages 483–492, 2012.

[25] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[26] Duane C Brown. Close-range camera calibration. In *PHOTOGRAMMETRIC ENGINEERING*, 1971.

[27] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[28] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[29] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH*, pages 67–76. ACM, 2001.

[30] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020.

[31] Emilie Chalmin, F Farges, C Vignaud, J Susini, M Menu, and GE Brown Jr. Discovery of unusual minerals in paleolithic black pigments from lascaux (france) and ekain (spain). In *AIP Conference Proceedings*, volume 882, pages 220–222. American Institute of Physics, 2007.

[32] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *arXiv preprint arXiv:2012.00926*, 2020.

[33] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[34] Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. Deep surface light fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–17, 2018.

[35] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1520, 2017.

[36] Wei-Chao Chen, Jean-Yves Bouguet, Michael H Chu, and Radek Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields. *ACM Transactions on Graphics (TOG)*, 21(3):447–456, 2002.

[37] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, pages 2172–2180, 2016.

[38] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *arXiv preprint arXiv:1812.02822*, 2018.

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

[40] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[41] Gyeongmin Choe, Srinivasa G Narasimhan, and In So Kweon. Simultaneous estimation of near ir brdf and fine-scale surface geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2452–2460, 2016.

[42] Gyeongmin Choe, Jaesik Park, Yu-Wing Tai, and In So Kweon. Exploiting shading cues in kinect ir images for geometry refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3922–3929, 2014.

[43] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7781–7790, 2019.

[44] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[45] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[46] Antonio Criminisi, Sing Bing Kang, Rahul Swaminathan, Richard Szeliski, and P Anandan. Extracting layers and analyzing their specular properties using epipolar-plane-image analysis. *Computer vision and image understanding*, 97(1):51–85, 2005.

[47] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[48] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017.

[49] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018.

[50] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[51] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[52] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012.

[53] Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham Mysore, Fredo Durand, and William T. Freeman. The visual microphone: Passive recovery of sound from video. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 33(4):79:1–79:10, 2014.

[54] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[55] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[56] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 classes*, page 31. ACM, 2008.

[57] Paul Ernest Debevec. *Modeling and rendering architecture from photographs*. University of California, Berkeley, 1996.

[58] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[59] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020.

[60] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. *arXiv preprint arXiv:2104.00670*, 2021.

[61] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Transactions on Graphics (TOG)*, 33(6):193, 2014.

[62] Gary Doran. PyEMD: Earth mover's distance for Python, 2014–. [Online; accessed ¡today¿].

[63] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced global illumination*. AK Peters/CRC Press, 2018.

[64] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[65] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[66] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[67] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[68] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image.

[69] Jicong Fan and Jieyu Cheng. Matrix completion by deep matrix factorization. *Neural Networks*, 98:34–41, 2018.

[70] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[71] Andrew W Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *European conference on computer vision*, pages 311–326. Springer, 1998.

[72] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.

[73] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[74] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.

[75] Marc-André Gardner, Yannick Hold-Geoffroy, Kalyan Sunkavalli, Christian Gagné, and Jean-François Lalonde. Deep parametric indoor lighting estimation. *arXiv preprint arXiv:1910.08812*, 2019.

[76] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017.

[77] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Tinne Tuytelaars, and Luc Van Gool. What is around the camera? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5170–5178, 2017.

[78] Abhijeet Ghosh, Tongbo Chen, Pieter Peers, Cyrus A Wilson, and Paul Debevec. Estimating specular roughness and anisotropy from second order spherical gradient illumination. In *Computer Graphics Forum*, volume 28, pages 1161–1170. Wiley Online Library, 2009.

[79] Michael Goesele, Brian Curless, and Steven M Seitz. Multi-view stereo revisited. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2402–2409. IEEE, 2006.

[80] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[81] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2009.

[82] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2010.

[83] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[84] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996.

[85] Peter J Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 149–192, 1984.

[86] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019.

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

[88] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-m\^ ach\'e approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018.

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

[90] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3012–3021, 2020.

[91] Xiaojie Guo, Xiaochun Cao, and Yi Ma. Robust separation of reflection from multiple images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2187–2194, 2014.

[92] Heli Ben Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *arXiv preprint arXiv:1806.02143*, 2018.

[93] Byeong-Ju Han and Jae-Young Sim. Reflection removal using low-rank matrix completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5438–5446, 2017.

[94] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference.

[95] Christian Häne and Marc Pollefeys. An overview of recent progress in volumetric semantic 3d reconstruction. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3294–3307. IEEE, 2016.

[96] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *3D Vision (3DV), 2017 International Conference on*, pages 412–420. IEEE, 2017.

[97] Christian Hane, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3d scene reconstruction and class segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 97–104, 2013.

[98] Christopher G Harris and JM Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[99] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[100] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[101] Peter Hedman and Johannes Kopf. Instant 3d photography. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[102] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. In *SIGGRAPH Asia 2018 Technical Papers*, page 257. ACM, 2018.

[103] Carlos Hernández, George Vogiatzis, and Roberto Cipolla. Probabilistic visibility for multi-view stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[104] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. *arXiv preprint arXiv:1905.03897*, 2019.

[105] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7312–7321, 2017.

[106] Michael Holroyd, Jason Lawrence, Greg Humphreys, and Todd Zickler. A photometric approach for estimating normals and tangents. *ACM Transactions on Graphics (TOG)*, 27(5):133, 2008.

[107] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[108] Drew A Hudson and C Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209*, 2021.

[109] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[110] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[111] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5134–5143, 2017.

[112] Jan Jachnik, Richard A Newcombe, and Andrew J Davison. Real-time surface light-field capture for augmentation of planar specular surfaces. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 91–97. IEEE, 2012.

[113] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015.

[114] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Transactions on Graphics (TOG)*, 34(6):189, 2015.

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

[116] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[117] Mark W. Jones. Distance field compression. *Journal of WSCG*, 12(2):199–204, 2004.

[118] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[119] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):193, 2016.

[120] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[121] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *arXiv preprint arXiv:2106.12423*, 2021.

[122] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[123] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[124] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30, page 157. ACM, 2011.

[125] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics (TOG)*, 33(3):32, 2014.

[126] Jan Kautz, Pere-Pau Vázquez, Wolfgang Heidrich, and Hans-Peter Seidel. A unified approach to prefiltered environment maps. In *Rendering Techniques 2000*, pages 185–196. Springer, 2000.

[127] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM TOG*, 32(3):29, 2013.

[128] Christian Kerl, Mohamed Souiai, Jürgen Sturm, and Daniel Cremers. Towards illumination-invariant 3d reconstruction using tof rgb-d cameras. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 39–46. IEEE, 2014.

[129] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106. IEEE, 2013.

[130] Kihwan Kim, Jinwei Gu, Stephen Tyree, Pavlo Molchanov, Matthias Nießner, and Jan Kautz. A lightweight approach for on-the-fly reflectance estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 20–28, 2017.

[131] Seon Joo Kim and Marc Pollefeys. Robust radiometric calibration and vignetting correction. *IEEE transactions on pattern analysis and machine intelligence*, 30(4):562–576, 2008.

[132] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[133] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[134] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. One shot 3d photography. *ACM Transactions on Graphics (TOG)*, 39(4):76–1, 2020.

[135] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[136] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000.

[137] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3d scenes. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1330–1337. IEEE, 2012.

[138] Joo Ho Lee, Adrian Jarabo, Daniel S Jeon, Diego Gutierrez, and Min H Kim. Practical multiple scattering for rough surfaces. In *SIGGRAPH Asia 2018 Technical Papers*, page 275. ACM, 2018.

[139] Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5918–5928, 2019.

[140] Hendrik PA Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics (TOG)*, 22(2):234–257, 2003.

[141] Marc Levoy. Display of surfaces from volume data. *IEEE Computer graphics and Applications*, 8(3):29–37, 1988.

[142] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[143] Jianguo Li, Eric Li, Yurong Chen, Lin Xu, and Yimin Zhang. Bundled depth-map merging for multi-view stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2769–2776. IEEE, 2010.

[144] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. In *SIGGRAPH Asia 2018 Technical Papers*, page 269. ACM, 2018.

[145] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. *CVPR*, 2017.

[146] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[147] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020.

[148] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

[149] Stephen Lombardi and Ko Nishino. Reflectance and natural illumination from a single image. In *European Conference on Computer Vision*, pages 582–595. Springer, 2012.

[150] Stephen Lombardi and Ko Nishino. Radiometric scene decomposition: Scene reflectance, illumination, and geometry from rgb-d images. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 305–313. IEEE, 2016.

[151] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[152] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, volume 21, pages 163–169. ACM, 1987.

[153] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[154] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (TOG)*, 39(4):71–1, 2020.

[155] Robert Maier, Kihwan Kim, Daniel Cremers, Jan Kautz, and Matthias Nießner. Intrinsic3d: High-quality 3d reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3114–3122, 2017.

[156] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 27–34. ACM, 1993.

[157] Kevis-Kokitsi Maninis, Stefan Popov, Matthias Nießner, and Vittorio Ferrari. Vid2cad: Cad model alignment using multi-view constraints from videos. *arXiv preprint arXiv:2012.04641*, 2020.

[158] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[159] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. 2017.

[160] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[161] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[162] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6:355–378, 2019.

[163] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6315–6324, 2018.

[164] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018.

[165] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. *CoRR*, abs/1904.04290, 2019.

[166] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *arXiv preprint arXiv:1905.00889*, 2019.

[167] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[168] Gavin Miller, Steven Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *Rendering Techniques' 98*, pages 281–292. Springer, 1998.

[169] T.K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

[170] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[171] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[172] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical svbrdf acquisition of 3d objects with unstructured flash photography. In *SIGGRAPH Asia 2018 Technical Papers*, page 267. ACM, 2018.

[173] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[174] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

[175] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011.

[176] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[177] Fred E Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied optics*, 4(7):767–775, 1965.

[178] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.

[179] Ko Nishino and Shree Nayar. Eyes for relighting. *ACM Trans. Graph.*, 23:704–711, 08 2004.

[180] Roy Or-El, Guy Rosman, Aaron Wetzler, Ron Kimmel, and Alfred M Bruckstein. Rgbd-fusion: Real-time high precision depth recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5407–5416, 2015.

[181] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 143–152, 2017.

[182] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[183] Jeong Joon Park, Aleksander Holynski, and Steven M Seitz. Seeing the world in a bag of chips. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1417–1427, 2020.

[184] Jeong Joon Park, Richard Newcombe, and Steve Seitz. Surface light field fusion. In *2018 International Conference on 3D Vision (3DV)*, pages 12–21. IEEE, 2018.

[185] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[186] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[187] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *arXiv preprint arXiv:2003.04618*, 2, 2020.

[188] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[189] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 145–154, 2019.

[190] Mark Pupilli and Andrew Calway. Real-time camera tracking using a particle filter. In *BMVC*, 2005.

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

[192] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[193] Zhu Qunxiong and Li Chengfei. Dimensionality reduction with input training neural network and its application in chemical process modelling. *Chinese Journal of Chemical Engineering*, 14(5):597–603, 2006.

[194] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[195] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 117–128. ACM, 2001.

[196] VN Reddy and ML Mavrovouniotis. An input-training neural network approach for gross error detection and sensor replacement. *Chemical Engineering Research and Design*, 76(4):478–489, 1998.

[197] Thomas Richter-Trummer, Denis Kalkofen, Jinwoo Park, and Dieter Schmalstieg. Instant mixed reality lighting from casual scanning. In *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, pages 27–36. IEEE, 2016.

[198] Gernot Riegler, Ali O Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, pages 6620–6629. IEEE, 2017.

[199] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3d object shape from one depth image. In *CVPR*, pages 2484–2493, 2015.

[200] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[201] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[202] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[203] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, et al. Frodo: From detections to 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14720–14729, 2020.

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

[205] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, pages 901–909, 2016.

[206] Christophe Schlick. An inexpensive brdf model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Wiley Online Library, 1994.

[207] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. pages 519–528. IEEE, 2006.

[208] Steven M Seitz and Charles R Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30, 1996.

[209] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[210] Steven A Shafer. Using color to separate reflection components. *Color Research & Application*, 10(4):210–218, 1985.

[211] Qi Shan, Riley Adams, Brian Curless, Yasutaka Furukawa, and Steven M Seitz. The visual turing test for scene reconstruction. In *2013 International Conference on 3D Vision-3DV 2013*, pages 25–32. IEEE, 2013.

[212] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European conference on computer vision*, pages 223–240. Springer, 2016.

[213] Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. Image-based rendering for scenes with reflections. *ACM Trans. Graph.*, 31(4):100–1, 2012.

[214] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[215] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[216] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006.

[217] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International journal of computer vision*, 80(2):189–210, 2008.

[218] Shuran Song and Thomas Funkhouser. Neural illumination: Lighting prediction for indoor environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6918–6926, 2019.

[219] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2243–2251, 2017.

[220] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *ICCV Workshops*, pages 719–722. IEEE, 2011.

[221] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[222] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *CVPR*, pages 1955–1964, 2018.

[223] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. *arXiv preprint arXiv:2103.12352*, 2021.

[224] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[225] Richard Szeliski, Shai Avidan, and P Anandan. Layer extraction from multiple images containing reflections and transparency. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 246–253. IEEE, 2000.

[226] Robby T Tan and Katsushi Ikeuchi. Separating reflection components of textured surfaces using a single image. *IEEE transactions on pattern analysis and machine intelligence*, 27(2):178–193, 2005.

[227] Shufeng Tan and Michael L Mayrovouniotis. Reducing data dimensionality through optimizing neural network inputs. *AIChE Journal*, 41(6):1471–1480, 1995.

[228] Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. Polycube-maps. In *ACM TOG*, volume 23, pages 853–860. ACM, 2004.

[229] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV*, 2017.

[230] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *arXiv preprint arXiv:1904.12356*, 2019.

[231] Mikaela Angelina Uy, Vladimir G Kim, Minhyuk Sung, Noam Aigerman, Siddhartha Chaudhuri, and Leonidas J Guibas. Joint learning of 3d shape retrieval and deformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11722, 2021.

[232] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

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

[234] Michael Waechter, Mate Beljan, Simon Fuhrmann, Nils Moehrle, Johannes Kopf, and Michael Goesele. Virtual rephotography: Novel view prediction error for 3d reconstruction. *ACM Trans. Graph.*, 36(1):8:1–8:11, January 2017.

[235] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 195–206. Eurographics Association, 2007.

[236] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas 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.

[237] Jiaping Wang, Shuang Zhao, Xin Tong, John Snyder, and Baining Guo. Modeling anisotropic surface reflectance with example-based microfacet synthesis. In *ACM Transactions on Graphics (TOG)*, volume 27, page 41. ACM, 2008.

[238] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[239] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[240] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[241] Gregory J Ward. Measuring and modeling anisotropic reflection. *ACM SIGGRAPH Computer Graphics*, 26(2):265–272, 1992.

[242] Thomas Whelan, Michael Goesele, Steven J Lovegrove, Julian Straub, Simon Green, Richard Szeliski, Steven Butterfield, Shobhit Verma, Richard A Newcombe, et al. Reconstructing scenes with mirror and glass surfaces. 2018.

[243] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems.

[244] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021.

[245] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000.

[246] Hongzhi Wu, Zhaotian Wang, and Kun Zhou. Simultaneous localization and appearance estimation with a consumer rgb-d camera. *IEEE transactions on visualization and computer graphics*, 22(8):2012–2023, 2015.

[247] Hongzhi Wu, Zhaotian Wang, and Kun Zhou. Simultaneous localization and appearance estimation with a consumer rgb-d camera. *IEEE transactions on visualization and computer graphics*, 22(8):2012–2023, 2016.

[248] Hongzhi Wu and Kun Zhou. Appfusion: Interactive appearance acquisition using a kinect sensor. In *Computer Graphics Forum*, volume 34, pages 289–298. Wiley Online Library, 2015.

[249] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, pages 82–90, 2016.

[250] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman, and Joshua B Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. *arXiv preprint arXiv:1809.05068*, 2018.

[251] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[252] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[253] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[254] Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Transactions on Graphics (TOG)*, 35(6):187, 2016.

[255] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in neural information processing systems*, pages 1790–1798, 2014.

[256] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *arXiv preprint arXiv:1905.10711*, 2019.

[257] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)*, 38(4):76, 2019.

[258] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T Freeman. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)*, 34(4):79, 2015.

[259] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *arXiv preprint arXiv:1712.07262*, 2017.

[260] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[261] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv preprint arXiv:2106.12052*, 2021.

[262] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020.

[263] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[264] Hong-Xing Yu, Leonidas J Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. *arXiv preprint arXiv:2107.07905*, 2021.

[265] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 215–224. ACM Press/Addison-Wesley Publishing Co., 1999.

[266] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *3DV*, 2018.

[267] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12224–12233, 2020.

[268] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811, 2017.

[269] Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):1–14, 2016.

[270] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[271] Xuaner Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4786–4794, 2018.

[272] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J Mitra. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph.*, 31(4):99–1, 2012.

[273] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014.

[274] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European conference on computer vision*, pages 766–782. Springer, 2016.

[275] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[276] Zhiming Zhou, Guojun Chen, Yue Dong, David Wipf, Yong Yu, John Snyder, and Xin Tong. Sparse-as-possible svbrdf acquisition. *ACM Transactions on Graphics (TOG)*, 35(6):189, 2016.

[277] Todd Zickler, Sebastian Enrique, Ravi Ramamoorthi, and Peter Belhumeur. Reflectance Sharing: Image-based Rendering from a Sparse Set of Images. In Kavita Bala and Philip Dutre, editors, *Eurographics Symposium on Rendering (2005)*. The Eurographics Association, 2005.

[278] Michael Zollhöfer, Angela Dai, Matthias Innmann, Chenglei Wu, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, 34(4):96, 2015.