Doctoral Thesis

Learning 3D Registration and Reconstruction from the Visual World

Chen-Hsuan Lin

The Robotics Institute
School of Computer Science
Carnegie Mellon University

June 28, 2021

Doctoral Thesis Committee:

Simon Lucey	*Carnegie Mellon University* (chair)
Deva Ramanan	*Carnegie Mellon University*
Abhinav Gupta	*Carnegie Mellon University*
Andrea Vedaldi	*University of Oxford*

Technical Report Number: CMU-RI-TR-21-13

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Keywords: registration, image alignment, dense 3D reconstruction, self-supervised learning, structure from motion, multi-view geometry, photometric optimization, neural rendering, neural scene representations

For my friends and family.

Abstract

Humans learn to develop strong senses for 3D geometry by looking around in the visual world. Through pure visual perception, not only can we recover a mental 3D representation of *what* we are looking at, but meanwhile we can also recognize *where* we are looking at the scene from. Finding the 3D scene representation from RGB images (*i.e.* **3D reconstruction**) and localizing the camera frames (*i.e.* **registration**) are two long-standing problems in computer vision. Simultaneously solving *both* tasks makes it an even more challenging chicken-and-egg problem — recovering the 3D structure requires observations with accurate camera poses, while localizing the cameras requires reliable correspondences from the reconstruction.

In this thesis, we explore the problem of learning geometric alignment and dense 3D reconstruction from images and videos using self-supervised learning techniques. Toward this end, we discuss the general importance of factorizing geometric information from visual data. First, we build up the theoretical foundations from learning-based planar registration for images. We show that incorporating geometric priors in learning models increases learning efficacy for alignment algorithms, and we demonstrate both discriminate and generative applications of image registration with modern deep neural networks. Second, we explore more complex 3D shape priors parametrized by neural networks, which we train from images without 3D supervision by utilizing differentiable rendering techniques. We develop methods for learning from multi-view depth observations and even single-view supervision from static RGB images. Finally, we investigate the challenging problem of joint optimization of 3D registration and reconstruction. Given a video sequence, we demonstrate how one can exploit pretrained 3D shape priors to register and refine the shape reconstruction to the video sequences, as well as a more generic rendering prior for learning neural 3D scene representations from unknown camera poses.

Images and videos contain very rich and detailed information about the 3D world. Baking in suitable geometric priors allows learning models to effectively recover both the dense 3D scene structures and the corresponding camera poses using image synthesis as the proxy objective. We believe this is an essential ingredient towards scalable learning of in-the-wild spatial 3D understanding for future AI systems.

Acknowledgments

First and foremost, I would like to thank my advisor Simon Lucey for all the guidance and support throughout my seven years at CMU. When I first joined CMU, I knew very little about computer vision and machine learning, and I did not have too much experience in doing research. Simon welcomed me into the lab with open arms. He cultivated me with full patience, even at times when I had no idea what I was working on in my first few years. There are countless things I have learned from Simon, from choosing the research topics, designing experiments, to presentation skills. I have always thought Simon is more than a teacher — he is also an amazing educator. Becoming what I am today is unimaginable seven years ago when I first met Simon, and I cannot thank Simon enough for always putting full faith in me. In my very honest opinion, I cannot imagine a better PhD advisor than Simon who would be as brilliant, caring, tolerating, motivating, and supporting.

I would also like to thank my PhD thesis committee members Deva Ramanan, Abhinav Gupta, and Andrea Vedaldi for the time and effort to discuss my research and critically evaluating my thesis. I have learned so much from the valuable and critical feedback they have given for my research, and their insights have also prompted me to think realistically without being overambitious. I also thank Martial Hebert for evaluating my PhD speaking qualifier, as well as Fernando De la Torre and Wen-Sheng Chu for evaluating my MS thesis. They have provided great feedback for my earlier stages of research works, and I am very thankful for them.

I am extremely fortunate to have worked with many incredible researchers through internships. I would like to thank all my mentors Oliver Wang, Eli Shechtman, Bryan Russell, Vladimir Kim, Matthew Fisher, and Ersin Yumer from Adobe Research, as well as Kaiming He, Georgia Gkioxari, and Justin Johnson from Facebook AI Research. I also thank the other great researchers I have met during the summers including Connelly Barnes, Jitendra Malik, Saining Xie, which has sparked new inspirations for me. I have learned how one should always set high standards for ourselves, think far and beyond, manage research works in a meticulous and disciplined way, and collaborate with others by effectively communicating with everyone. These valuable lessons will be in my heart for the rest of my research career.

I would also like to thank Yaser Sheikh, Katerina Fragkiadaki, and David Fouhey for taking time off to talk with me in the past about career decisions and research life in general. I also thank Kris Kitani for selflessly helping advertise our research works. I am so grateful to meet the kind researchers that have helped me without conditions during my years at CMU, and I have learned to always be warmhearted and caring for others, as research is really not an easy career that goes smoothly all

the time. In addition, I'd like to thank Suzanne Lyons Muth for helping take care of all the program-related work to make my PhD studies worry-free.

I would also like to thank my undergraduate research advisor Homer Chen for building up my fundamentals and teaching me to do principled research. Homer was strict and it was not easy to meet his expectations, but the training from the undergraduate years has played a huge role and I am really grateful for that.

My PhD life would not have been complete without my awesome labmates from the CI2CV lab. I would like to thank Chen Kong, Chaoyang Wang, Nathaniel Chodosh, Ming-Fang Chang, Calvin Murdock, Rui Zhu, Hamed Kiani, Hatem Alismail, Hilton Bristow, Jack Valmadre, Christopher Ham, Ashton Fagg, Ziyan Wang, Iman Abbasnejad, Shubham Agrawal, Anuj Pahuja, Kushal Vyas, Brendan Miller, Hunter Goforth, Xueqian Li, Jianqiao Zheng for all the helpful discussion and wonderful times we have spent together. Thank you for sitting through the numerous talk rehearsals, discussing our research and brainstorming bold ideas that never really works, and all the encouragement throughout the years.

I also thank all the friends I have met through my research life. I would like to thank Wei-Chiu Ma, who shares multiple important roles since my undergraduate years: an amazing collaborator, motivator, and true friend who never trash-talks enough. I also thank Hsiao-Yu Fish Tung, also a long-time friend to strive together through the PhD years. I thank the great fellow researchers from Smith Hall that have inspired me: Aayush Bansal, Minh Vo, Shubham Tulsiani, Jun-Yan Zhu, David Held, De-An Huang, Yuxiong Wang, Hanbyul Joo, Shih-En Wei, Mengtian Li, Chao Liu, Xiaofang Wang, Minyoung Huh, Leonid Keselman, Peiyun Hu, Yufei Ye, Senthil Purushwalkam, Rohit Girdhar, Xiaolong Wang, Kenneth Marino, Gunnar Sigurdsson, Nadine Chang, Sheng-Yu Wang, Zhe Cao, Chia Dai, and Yi-Ting Chen. I also thank the fine fellows outside Smith Hall: Chao-Yuan Wu, Ming Hsiao, Guan-Horng Liu, Po-Wei Chou, Chia-Yin Tsai, Jen-Hao Chang, Yen-Chi Chen, and Yen-Chia Hsu. I really enjoyed all the research discussions and our chatters that might never really make sense, and I am so lucky to have such awesome companions.

I am also grateful to have interacted with amazing researchers outside the CMU network that have also sparked great insiprations: Alyosha Efros, Antonio Torralba, Noah Snavely, James Hays, Tsung-Yi Lin, Richard Zhang, Yijun Li, Chenxi Liu, Carlos Esteves, Olivia Wiles, Yen-Chen Lin, Weicheng Kuo, and many more. I am grateful to have befriended everyone that has also shaped a big part of my inspirations.

I am deeply thankful for all the friends that have largely supported me outside my research life. First, I thank my 2320 Eldridge roommates Roger Lo, Wei-Chiu Ma, and YiuChang Lin, who are the most important machies since coming to Pittsburgh. I cannot imagine what life would have been without all the support from housing, everyday living, the ball games, the road trips, and the great times together even to today. I also thank my past roommates Wei-Cheng Chang, Wei-Yu Chen, Yu-Hsuan Wang, Yutong Zheng, and Yongshuo Jiang for taking care of each other throughout

the years, as well as Roger's cat Lena for spending countless purring nights with cuteness overloaded. I thank all my Pittsburgh friends Rushane Hua, Hao-Ping Ho, Yu-Hsin Kuo, Ting-Yu Chen, Chieh-Han Wu, Hsu-Chieh Hu, Chung-Yao Chuang, Hsin Miao, Ling-Wan Chen, Po-Hsun Su, and many more for all the love and support. I would also like to thank Emily Huang, Leslie Chen, and Melody Huang for hosting me at their place during the COVID-19 pandemic, so I could do job interviews and finish this dissertation but still full of laughter every day.

Finally, none of this would be possible without the full unconditional support from my family. I thank my mother and father, who are academic researchers themselves, for cultivating my curiosity for everything ever since I was born, raising me to become proactive to problem solving, and for all the encouragement and motivation to make smart decisions and never give up. I thank my sister for all the home chitchats and comfort stays during visits. Last but not least, I thank Emily Huang for all the love and understanding throughout my PhD life.

This dissertation is partially supported by the NVIDIA Graduate Fellowship and the CMU Argo AI Center for Autonomous Vehicle Research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans have amazing capabilities of reasoning about the 3D world through pure visual perception. When we move and look around, we are able to immediately infer the 3D structures of the visual world by sensing how our observations change over time, even within the slightest instant. In addition, our cognition can go further beyond ego-motion: we can perceive the 3D structures of objects and scenes even when looking and static images and photographs. What also naturally follows is our "sense of direction": we can also immediately infer where we are looking at the objects or scenes from. Such capabilities in turn allow us to understand and answer more complex questions about our visual perception, such as "this office chair has five legs" or "the car on the right is parked sideways on the road".

The incredible thing is that we as humans develop such ability of 3D geometric reasoning and sense of direction mostly just by looking and learning from the visual world, from as early as when we were infants. For example, we can instantly identify the underlying shapes and structures of famous landmarks around the world just by looking at photographs, even though we might have never physically been to these places (not to say having actually touched or "felt" such geometric structures). Furthermore, we could even also determine the location the photograph was taken at. Through pure visual perception, not only can we recover a mental 3D representation of *what* we are looking at, but meanwhile we can also recognize *where* we are looking at the scene from.

We thus ask the question: can we also allow *machines* to do the same? Is it possible to enable AI systems to perceive dense 3D geometric structures by emulating our human learning process — with "visual self-supervision", *i.e.* learning from abundant visual data like images and videos?

The ability for machines to perceive and understand the world in 3D has a tremendous range of practical applications (Fig. 1.1). Autonomous driving systems need to build accurate 3D maps of the surrounding environment and geo-localize itself with respect to the maps, and they need to be able to reason about nearby vehicles and pedestrians for safety control. Similarly, it is necessary for robotic agents to perceive the world in 3D in order to navigate and interact with the environment without colliding with its surroundings. For augmented/virtual/mixed reality applications, it is also essential to successfully track the ego-motion of mobile displays relative to

Figure 1.1: The problems of **3D reconstruction** — recovering the 3D geometric representation from the images, and **registration** — localizing the corresponding camera frames, are fundamental problems in computer vision which concerns a wide variety of tasks, including autonomous driving, robot navigation, computer graphics, augmented/virtual reality, and medical imaging.

the 3D world in order to recreate a partial or full virtual world. Computer graphics applications are also highly related as realistic 3D content could be much more easily created through simplified processes without requiring laborious manual efforts. This even has future potentials for medical applications such as medical 3D imaging to facilitate more accurate diagnoses for doctors and surgeons. While far from an exhaustive list, these are some of the key applications that highlight the importance of such ability of 3D perception to AI systems in general.

Solving for the dense 3D geometric structures from RGB images (*i.e.* **3D reconstruction**) and localizing the camera (*i.e.* **registration**) have been two long-standing problems in computer vision. Simultaneously solving *both* tasks makes it an even more challenging chicken-and-egg problem — recovering the 3D structure requires observations with accurate camera poses, while localizing the cameras requires reliable correspondences from the reconstruction. This dissertation is centered around utilizing learning-based methods, more specifically using deep neural networks, to solve for *both* problems together by incorporating prior knowledge of the 3D geometry. In contrast to classical 3D reconstruction methods that could only recover the 3D world as sparse 3D point clouds, data-driven priors further allows us to recover *dense* 3D structures (*i.e.* surfaces) in different forms of 3D representations, so as to enable dense, pixel-level understanding of the underlying 3D geometry within images and videos.

1.1 Registration

A famous quote by celebrated computer vision scientist Prof. Takeo Kanade goes that the three most important problems in computer vision are "registration, registration, and registration". It is indeed true that registration has been the core fundamental for all computer vision algorithms to work more effectively and efficiently. This still stands true even in the modern era of deep learning, where computer vision problems are largely tackled by deep neural networks. For example, to track an object within a video sequence, one aims to locate the relative change of position in the video frames; to analyze the expression of a face in an image, it is more desirable to locate the facial landmarks for further analysis; to understand human actions in visual data, the first essential step would be to extract the human body pose. While far from an exhaustive list, these problems are examples of a very general problem in computer vision — extracting the *geometry* that can accurately describe the image data, as well as finding the *transformations* between such extracted geometric representations so that the computer vision algorithms can understand the image data much more efficiently.

The essential step of achieving geometric alignment is to establish pixel correspondences between images. Correspondences can either refer to (a) physical correspondences that indicate the same physical point of the same object, or (b) semantic correspondences that indicate two points on different objects that carry the same semantic meaning, usually encoded as some feature representation. Established correspondences between two images carry information about the underlying 3D geometry, which can be more compactly represented by a set of parameters that describes the 3D structure; in the simpler case of planar objects (*e.g.* paintings), the geometry can be represented by a homography transformation. This also builds the foundation of classical multi-view 3D reconstruction algorithms, which attempt to solve for 3D correspondences by matching interest points that describe the same physical 3D points.

In this dissertation, we explore learning-based direct methods for registration, where we focus on homography transformations on images. We show that (a) structured geometric priors can amount to using less training examples to learn registration, (b) geometric misalignment within an image dataset can be resolved via a discriminative end-goal objective, and (c) spatial alignment of objects can be discovered by comparing against an unpaired image dataset via an adversarial objective. These methods give hint to extensions to more sophisticated warp functions like 3D geometric shapes so as to register the images together in the actual 3D sense; these will also be discussed in later chapters of this dissertation.

1.2 3D Reconstruction

Images are a result of 2D projections of the 3D world, but how do we recover what the original 3D structures are from the images? Classical 3D reconstruction methods, such as Structure from Motion (S*f*M) and Simultaneous Localization and Mapping (SLAM), utilize core concepts of multiple view geometry [58] to constrain the possible solutions of 3D structures that can be

explained by different viewpoints of the 3D object or scene. When only a single view of the same object or scene is available, however, 3D reconstruction becomes an ill-posed problem, since there can potentially be multiple solutions to the 3D structure occluded by the camera viewpoint. In this case, learning-based approaches can come to the rescue, especially methods utilizing the power of deep neural networks. The most straightforward way is to supervise a 3D prediction network with the 3D ground truth data associated with the image samples. Albeit simple, however, such paired datasets require carefully crafted 3D models and manual alignment with the images, which are extremely difficult to come by. Many works have resorted to 3D object datasets such as ShapeNet [16], in which case image data can be created through rendering engines for free; however, such 3D object datasets are usually limited in the variations of shapes and appearances, which typically cannot sufficiently represent the actual distributions in the real world.

Compared to ground-truth 3D annotations such as crafted CAD models, images and videos come in as more abundant sources of data to learn from. Therefore, instead of directly learning from available 3D geometry, it is of major interest to learn 3D reconstruction indirectly by learning to establish pixel correspondences between 2D image observations, physically or semantically. In order to utilize the images and videos as a source of self-supervision for neural networks, there has been various efforts on making the rendering operation differentiable [83, 109, 131, 177]. This allows the shape prediction networks to be trained end-to-end using more scalable supervisory signals. A very common assumption of learning 3D reconstruction without 3D supervision is that multiple observations of the same instance are available during training, so that multi-view supervision can be applied during training by enforcing the 3D shape prediction to be explainable from all the viewpoints. For learning object shapes, however, it is still somewhat impractical to obtain such multi-view silhouette annotations, which is especially difficult for real-world data.

In this dissertation, we explore learning-based 3D shape reconstruction methods without the use of explicit 3D supervision. We also investigate into various differentiable and neural rendering methods for different 3D representations. We show that (a) generalizable 3D shape reconstruction can be trained from a dataset of multi-view depth images, and (b) in a more advanced setting, 3D shape reconstruction can be trained from single-view static images using a clever design of neural rendering to discover semantic correspondences for resolving shape ambiguities, thus lifting the ill-posedness of the problem. These methods allow us to learn 3D shape priors parametrized by neural networks that can be trained at scale from practical image datasets, such as ImageNet [31].

1.3 3D Registration & Reconstruction

Simultaneously solving for the 3D geometric structures *and* localizing the camera frames has been a long-standing chicken-and-egg problem in computer vision. In order to recover the 3D structures, accurate camera poses that underlie need to be known a priori; to localize the cameras, on the other hand, one needs to establish reliable correspondences with respect to the 3D structures so as to optimize for the 6-DoF transformation in the 3D world. Structure from Motion (S*f*M) [153] and Simultaneous Localization and Mapping (SLAM) [4, 13, 27, 36, 204] are *indirect* methods for the problem. At a high level, S*f*M and SLAM methods associate multiple image observations together

by establishing reliable pixel correspondences [58]. This is achieved by detecting salient interest points in the images (*e.g.* with SIFT descriptors [115]) and matched with k-nearest neighbors. Once the point correspondences are established, one can recover the relative camera poses from the essential matrix and triangulate the sparse 3D structures from the pixel correspondences; if the 3D correspondences are given, solving for the poses becomes a Perspective-n-point problem [95]. As this alternation process is sensitive to the quality of local registration and easily falls into suboptimal solutions, a geometric bundle adjustment (BA) is typically required to jointly optimize for the 3D structures and the camera poses.

S*f*M and SLAM are indirect methods in the sense that correspondences are indirectly established from salient interest points in the images. Modern pipelines following such route have achieved tremendous success; however, they often suffer at textureless regions and repetitive patterns, where distinctive keypoints cannot be reliably detected. *Direct* methods, on the other hand, do not rely on such distinctive keypoints — every pixel can contribute to maximizing photometric consistency, leading to improved robustness in sparsely textured environments [187]. One major class of direct methods is photometric bundle adjustment [4, 55], which aims to maximize the reprojection consistency between different frames by directly comparing pixel intensities. These also allow dense image descriptors to be adopted for increased robustness at a pixel level [3].

In this dissertation, we explore using neural networks for photometric bundle adjustment frameworks of objects and scenes. We show that (a) given an object-centric video sequence, one can use a pretrained shape prior as a learned constraint to refine and register 3D shapes at test time, and (b) given a video sequence of an arbitrary scene, one can use a generic 3D volume rendering prior to learn a 3D scene representation while also optimizing for the camera poses. Not only can we recover the relative poses, but we can also recover *dense* 3D structures of object and scenes, in contrast to classical S*f*M and SLAM methods that only return *sparse* 3D structures.

1.4 Dissertation Organization

Fig. 1.2 provides a visual overview of the dissertation and the specific topics to be discussed. We break down the dissertation into three parts: learning-based methods for image registration (Part I), dense 3D reconstruction (Part II), and finally the joint problem of 3D registration and reconstruction (Part III). In detail, we organize the thesis as follows.

Learning image registration (Part I). In Chapter 2, we revisit the renowned Lucas-Kanade (LK) algorithm [116] for image alignment and establish a theoretical connection with Supervised Descent Method (SDM) [199], a learning-based alignment algorithm. We identify that the objective function of LK comes with a built-in geometric prior that results in a reduction of variable parameters. Following SDM to learn such parameters through a conditional objective *with* the geometric prior in LK, our learning-based alignment method, which we term Conditional LK [100], becomes more effective with less training data. This shows the importance of factorizing known geometry in learning problems and the benefits of optimizing through a structured objective.

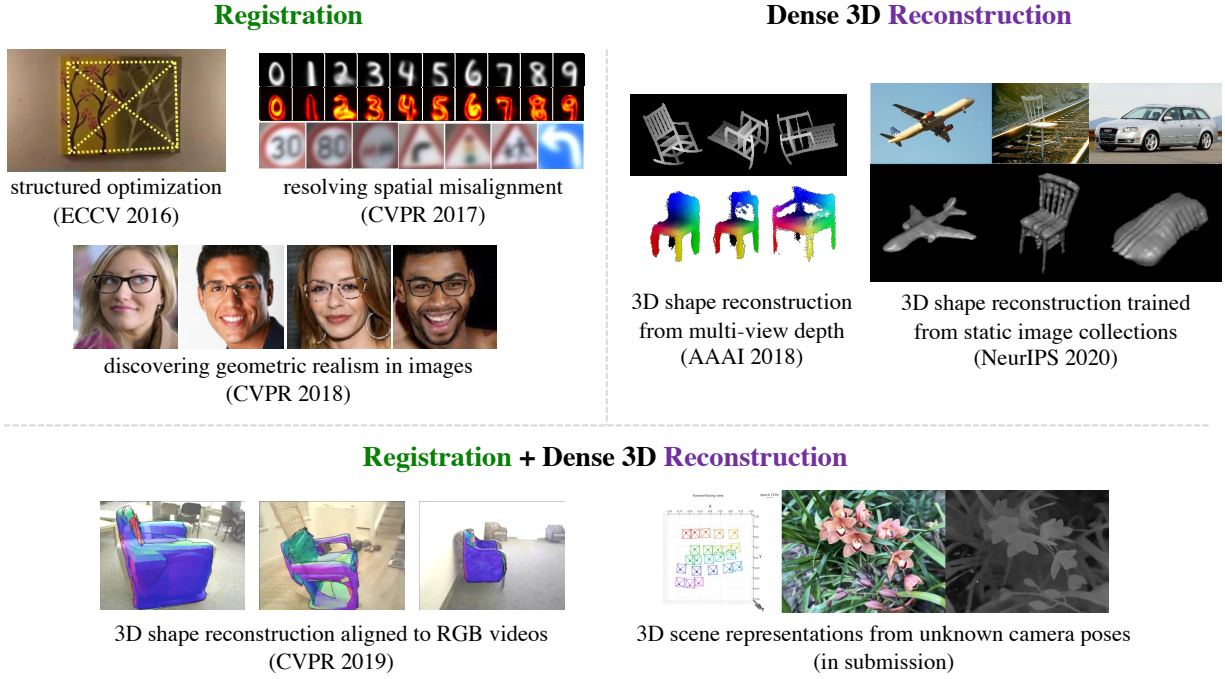


Figure 1.2: A visual overview of this dissertation.

To add to the evidence, in Chapter 3, we show that spatial alignment within image data can be automatically discovered end-to-end through a discriminative objective (*e.g.* classification). We make another theoretical connection of the LK algorithm, this time with Spatial Transformer Networks (STNs) [71]. We show that visual recognition can be learned more effectively if the spatial misalignment were explicitly resolved instead of tolerated as in most ConvNets. Termed Inverse Compositional STN [99], our proposed method automatically learns the optimal recurrent spatial transformations with boosted recognition performance with the same network capacities. This again shows that factorizing known geometry to achieve spatial invariance increases learning efficiency of deep networks.

In Chapter 4, we further show that geometric alignment can be learned in an unsupervised, generative setting. We utilize the power of Generative Adversarial Networks (GANs) [48] to learn a sequence of spatial transformations, resulting in geometric configurations that match a realistic distribution of image data. We refer to this as Spatial Transformer Generative Adversarial Networks (ST-GAN) [102] and demonstrate the effects of our approach on the application of image compositing. More importantly, it give hints to how one could potentially extend the geometric transformations to 3D reconstruction, such that dense 3D shapes could learn to be reconstructed in an adversarial fashion so as to match the given set of 2D image observations.

The following is the relevant publication list for each chapter.

1. Chapter 2 — **Lin** *et al.*, "The Conditional Lucas & Kanade Algorithm", ECCV 2016 [100]. (`https://chenhsuanlin.bitbucket.io/conditional-LK`)

2. Chapter 3 — **Lin** *et al.*, "Inverse Compositional Spatial Transformer Networks", CVPR 2017 [99].
(https://chenhsuanlin.bitbucket.io/inverse-compositional-STN)
3. Chapter 4 — **Lin** *et al.*, "ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing", CVPR 2018 [102].
(https://chenhsuanlin.bitbucket.io/spatial-transformer-GAN)

Learning dense 3D reconstruction (Part II). We turn to the problem of dense 3D reconstruction in Chapter 5, where we focus on object shape reconstruction. We discuss a simple case of 3D reconstruction as dense point cloud generation learned without 3D supervision [101]. The problem is posed as a novel-view synthesis problem of depth, such that the observed depth from different viewpoints have to agree consistently. We show that compared to naïve depth regression as a function of viewpoints, learning is more efficient by explicitly transforming and rendering the reconstruction with a novel differentiable point cloud renderer. This again highlights the importance of exploiting known geometry, as well as an example of how differentiable rendering can play a key role in multi-view reconstruction.

In Chapter 6, we take a further step of eliminating geometric supervision (*e.g.* depth observations). We make advances on learning dense 3D object reconstruction from single images and silhouettes, without the knowledge of the underlying shape structure or topology. To this end, we derive a formulation to learn signed distance functions (SDF) as the shape representation from silhouettes as a source of rich geometric supervision. In addition, we show that using differentiable ray marching [161] for rendering allows discovery of semantic correspondences between different static images, allowing SDF-SRN to train from only single observations of each object instance.

The following is the relevant publication list for each chapter.

4. Chapter 5 — **Lin** *et al.*, "Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction", AAAI 2018 [101].
(https://chenhsuanlin.bitbucket.io/3D-point-cloud-generation)
5. Chapter 6 — **Lin** *et al.*, "SDF-SRN: Learning Signed Distance 3D Object Reconstruction from Static Images", NeurIPS 2020 [104].
(https://chenhsuanlin.bitbucket.io/signed-distance-SRN)

Learning 3D registration and reconstruction (Part III). In Chapter 7, we take a further step of eliminating geometric supervision (*e.g.* depth observations). We bridge the gap between dense 3D object reconstruction with planar image alignment and show that, using meshes as the 3D representation, the problem can be posed as a piece-wise 2D image alignment problem from multiple views in a video sequence [103]. By combining multi-view geometric knowledge and a learned 3D shape prior, we show that accurate 3D reconstruction of objects is possible by optimizing for photometric consistency between video frames, without the need of additional geometric constraints. Enabling explicit 3D reconstruction from RGB-only images is another step towards alleviating the need of impractical supervision sources.

Finally, we show in Chapter 8 that we can go beyond shape priors and to incorporate a more generic rendering prior for arbitrary scenes. We show that we can learn Neural Radiance Fields (NeRF) [126] from imperfect or even unknown camera poses —– the joint problem of reconstructing the 3D scene and registering the camera poses. We draw inspiration from the success of classical image alignment methods and establish a theoretical connection, showing that coarse-to-fine registration is also critical to NeRF. To this end, we present Bundle-Adjusting NeRF (BARF), a simple yet effective strategy for coarse-to-fine registration on coordinate-based scene representations. BARF can be regarded as a type of photometric bundle adjustment (BA) using view synthesis as the proxy objective. Unlike traditional BA, however, BARF can learn the scene representation from scratch, lifting the reliance of local registration subprocedures and allowing for more generic applications.

The following is the relevant publication list for each chapter.

6. Chapter 7 — **Lin** *et al.*, "Photometric Mesh Optimization for Video-Aligned 3D Object Reconstruction", CVPR 2019 [103].
   (https://chenhsuanlin.bitbucket.io/photometric-mesh-optim)
7. Chapter 8 — **Lin** *et al.*, "BARF: Bundle-Adjusting Neural Radiance Fields", in submission [105].
   (https://chenhsuanlin.bitbucket.io/bundle-adjusting-NeRF)

Part I

Learning-based Image Registration

Chapter 2

Structured Optimization for Efficient Registration

2.1 Introduction

The Lucas-Kanade (LK) algorithm [116] has been a popular approach for tackling dense alignment problems for images and objects. At the heart of the algorithm is the assumption that an approximate linear relationship exists between pixel appearance and geometric displacement. Such a relationship is seldom exactly linear, so a linearization process is typically repeated until convergence. Pixel intensities are not deterministically differentiable with respect to geometric displacement; instead, the linear relationship must be established stochastically through a learning process. One of the most notable properties of the LK algorithm is how efficiently this linear relationship can be estimated. This efficiency stems from the assumption of independence across pixel coordinates — the parameters describing this linear relationship are classically referred to as image gradients. In practice, these image gradients are estimated through finite differencing operations. Numerous extensions and variations upon the LK algorithm have subsequently been explored in literature [9], and recent work has also demonstrated the utility LK [3, 5, 15] using classical dense descriptors such as dense SIFT [115], HOG [25], and LBP [133].

A drawback to the LK algorithm and its variants, however, is its generative nature. Specifically, it attempts to synthesize how appearance changes as a function of geometric displacement through a linear model, even though its end goal is the inverse problem. Recently, Xiong & De la Torre [199, 200, 201] proposed a new approach to image alignment known as the Supervised Descent Method (SDM). SDM shares similar properties with the LK algorithm as it also attempts to establish the relationship between appearance and geometric displacement using a sequence of linear models. One marked difference, however, is that SDM directly learns how geometric displacement changes as a function of appearance. This can be viewed as estimating the conditional likelihood function $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{y}$ and $\mathbf{x}$ are geometric displacement and appearance respectively. As reported in literature [73] (and also confirmed by our own experiments), this can lead to substantially improved performance over classical LK as the learning algorithm is focused

directly on the end goal (*i.e.* estimating geometric displacement from appearance).

Although it exhibits many favorable properties, SDM also comes with disadvantages. Specifically, due to its non-generative nature, SDM cannot take advantage of the pixel independence assumption enjoyed through classical LK (see Section 2.4 for a full treatment on this asymmetric property). Instead, it needs to model full dependence across all pixels, which requires: (i) a large amount of training data, and (ii) the requirement of adhoc regularization strategies in order to avoid a poorly conditioned linear system. Furthermore, SDM does not utilize prior knowledge of the type of geometric warp function being employed (*e.g.* similarity, affine, homography, point distribution model, etc.), which further simplifies the learning problem in classical LK.

In this chapter, we propose a novel approach which, like SDM, attempts to learn a linear relationship between geometric displacement directly as a function of appearance. However, unlike SDM, we enforce that the pseudo-inverse of this linear relationship enjoys the generative independence assumption across pixels while utilizing prior knowledge of the parametric form of the geometric warp. We refer to our proposed approach as the Conditional LK algorithm. Experiments demonstrate that our approach achieves comparable, and in many cases better, performance to SDM across a myriad of tasks with substantially less training examples. We also show that our approach does not require any adhoc regularization term, and it exhibits a unique property of being able to "swap" the type of warp function being modeled (*e.g.* replace a homography with an affine warp function) without the need to retrain. Finally, our approach offers some unique theoretical insights into the redundancies that exist when attempting to learn efficient object/image aligners through a conditional paradigm.

Notations. We define our notations throughout the paper as follows: lowercase boldface symbols (*e.g.* $\mathbf{x}$) denote vectors, uppercase boldface symbols (*e.g.* $\mathbf{R}$) denote matrices, and uppercase calligraphic symbols (*e.g.* $\mathcal{I}$) denote functions. We treat images as a function of the warp parameters, and we use the notations $\mathcal{I}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^K$ to indicate sampling of the K-channel image representation at subpixel location $\mathbf{x} = [x, y]^\top$. Common examples of multi-channel image representations include descriptors such as dense SIFT, HOG and LBP. We assume $K = 1$ when dealing with raw grayscale images.

2.2 The Lucas-Kanade Algorithm

At its heart, the Lucas-Kanade (LK) algorithm utilizes the assumption that,

$$\mathcal{I}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathcal{I}(\mathbf{x}) + \nabla\mathcal{I}(\mathbf{x})\Delta\mathbf{x} \; . \tag{2.1}$$

where $\mathcal{I}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^K$ is the image function representation and $\nabla\mathcal{I}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^{K\times 2}$ is the image gradient function at pixel coordinate $\mathbf{x} = [x, y]$. In most instances, a useful image gradient function $\nabla\mathcal{I}(\mathbf{x})$ can be efficiently estimated through finite differencing operations. An alternative strategy is to treat the problem of gradient estimation as a per-pixel linear regression problem, where pixel intensities are samples around a neighborhood in order to "learn" the

image gradients [15]. A focus of this chapter is to explore this idea further by examining more sophisticated conditional learning objectives for learning image gradients.

For a given geometric warp function $\mathcal{W}(\mathbf{x};\mathbf{p}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ parameterized by the warp parameters $\mathbf{p} \in \mathbb{R}^P$, one can thus express the classic LK algorithm as minimizing the sum of squared differences (SSD) objective,

$$\min_{\Delta\mathbf{p}} \sum_{d=1}^{D} \left\| \mathcal{I}(\mathcal{W}(\mathbf{x}_d;\mathbf{p})) + \nabla\mathcal{I}(\mathcal{W}(\mathbf{x}_d;\mathbf{p})) \frac{\partial \mathcal{W}(\mathbf{x}_d;\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} - \mathcal{T}(\mathbf{x}_d) \right\|_2^2 , \tag{2.2}$$

which can be viewed as a quasi-Newton update. The parameter $\mathbf{p}$ is the initial warp estimate, $\Delta\mathbf{p}$ is the warp update being estimated, and $\mathcal{T}$ is the template image we desire to align the source image $\mathcal{I}$ against. The pixel coordinates $\{\mathbf{x}_d\}_{d=1}^{D}$ are taken with respect to the template image's coordinate frame, and $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}} : \mathbb{R}^2 \rightarrow \mathbb{R}^{2\times P}$ is the warp Jacobian. After solving Equation 2.2, the current warp estimate has the following additive update,

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p} \, . \tag{2.3}$$

As the relationship between appearance and geometric deformation is not solely linear, Equations 2.2 and 2.3 must be applied iteratively until convergence is achieved.

Inverse compositional fitting. The canonical LK formulation presented in the previous section is sometimes referred to as the forwards additive (FA) algorithm [9]. A fundamental problem with the forwards additive approach is that it requires recomputing the image gradient and warp Jacobian in each iteration, greatly impacting computational efficiency. Baker and Matthews [9] devised a computationally efficient extension to forwards additive LK, namely the inverse compositional (IC) algorithm. The IC-LK algorithm attempts to iteratively solve the objective

$$\min_{\Delta\mathbf{p}} \sum_{d=1}^{D} \left\| \mathcal{I}(\mathcal{W}(\mathbf{x}_d;\mathbf{p})) - \mathcal{T}(\mathbf{x}_d) - \nabla\mathcal{T}(\mathbf{x}_d) \frac{\partial \mathcal{W}(\mathbf{x}_d;\mathbf{0})}{\partial \mathbf{p}} \Delta\mathbf{p} \right\|_2^2 , \tag{2.4}$$

followed by the inverse compositional update

$$\mathbf{p} \leftarrow \mathbf{p} \circ (\Delta\mathbf{p})^{-1} \, , \tag{2.5}$$

where we have abbreviated the notation $\circ$ to be the composition of warp functions parametrized by $\mathbf{p}$, and $(\Delta\mathbf{p})^{-1}$ to be the parameters of the inverse warp function parametrized by $\Delta\mathbf{p}$. We can express Equation 2.4 in vector form as

$$\min_{\Delta\mathbf{p}} \|\mathcal{I}(\mathbf{p}) - \mathcal{T}(\mathbf{0}) - \mathcal{W}\Delta\mathbf{p}\|_2^2 , \tag{2.6}$$

where

$$\mathbf{W} = \begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{x}_1;\mathbf{0})}{\partial \mathbf{p}} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{x}_D;\mathbf{0})}{\partial \mathbf{p}} \end{bmatrix}$$

and

$$\mathcal{I}(\mathbf{p}) = \begin{bmatrix} \mathcal{I}(\mathcal{W}(\mathbf{x}_1; \mathbf{p})) \\ \vdots \\ \mathcal{I}(\mathcal{W}(\mathbf{x}_D; \mathbf{p})) \end{bmatrix}, \quad \mathcal{T}(\mathbf{0}) = \begin{bmatrix} \mathcal{T}(\mathcal{W}(\mathbf{x}_1; \mathbf{0})) \\ \vdots \\ \mathcal{T}(\mathcal{W}(\mathbf{x}_D; \mathbf{0})) \end{bmatrix}.$$

Here, $\mathbf{p} = \mathbf{0}$ is considered the identity warp (*i.e.* $\mathcal{W}(\mathbf{x}; \mathbf{0}\} = \mathbf{x}$). It is easy to show that the solution to Equation 2.6 is given by

$$\Delta \mathbf{p} = \mathbf{R}[\mathcal{I}(\mathbf{p}) - \mathcal{T}(\mathbf{0})], \tag{2.7}$$

where $\mathbf{R} = \mathbf{W}^{\dagger}$. The superscript $\dagger$ denotes the Moore-Penrose pseudo-inverse operator. The IC form of the LK algorithm comes with a great advantage: the gradients $\nabla \mathcal{T}(\mathbf{x})$ and warp Jacobian $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{0})}{\partial \mathbf{p}}$ are evaluated at the identity warp $\mathbf{p} = \mathbf{0}$, regardless of the iterations and the current state of $\mathbf{p}$. This means that $\mathbf{R}$ remains constant across all iterations, making it advantageous over other variants in terms of computational complexity. For the rest of this chapter, we shall focus on the IC form of the LK algorithm.

2.3 Supervised Descent Method

Despite exhibiting good performance on many image alignment tasks, the LK algorithm can be problematic to use when there is no specific template image $\mathcal{T}$ to align against. For many applications, one may be given just an ensemble of M ground-truth images and warps $\{\mathcal{I}_m, \mathbf{p}_m\}_{m=1}^{M}$ of the object of interest. If one has prior knowledge of the distribution of warp displacements to be encountered, one can synthetically generate N examples to form a much larger set $\mathcal{S} = \{\Delta \mathbf{p}_n, \mathcal{I}_n(\mathbf{p}_n \circ \Delta \mathbf{p}_n)\}_{n=1}^{N}$ to learn from, where $N \gg M$. In these circumstances, a strategy recently put forward known as the Supervised Descent Method (SDM) [199] has exhibited state-of-the-art performance across a number of alignment tasks, most notably facial landmark alignment. The approach attempts to directly learn a regression matrix that minimizes the following SSD objective,

$$\min_{\mathbf{R}} \sum_{n \in \mathcal{S}} \|\Delta \mathbf{p}_n - \mathbf{R}[\mathcal{I}_n(\mathbf{p}_n \circ \Delta \mathbf{p}_n) - \mathcal{T}(\mathbf{0})]\|_2^2 + \Omega(\mathbf{R}) \ . \tag{2.8}$$

The template image $\mathcal{T}(\mathbf{0})$ can be learned either with $\mathbf{R}$ directly or by taking it to be $\frac{1}{N} \sum_{n \in \mathcal{S}} \mathcal{I}(\mathbf{p}_n)$, the average of ground-truth images [201].

Regularization. Ω is a regularization function used to ensure that the solution to $\mathbf{R}$ is unique. To understand the need for this regularization, one can reform Equation 2.8 in matrix form as

$$\min_{\mathbf{R}} \|\mathbf{Y} - \mathbf{R}\mathbf{X}\|_F^2 + \Omega(\mathbf{R}), \tag{2.9}$$

where

$$\mathbf{Y} = \left[\Delta\mathbf{p}_1, \ldots, \Delta\mathbf{p}_N\right], \text{and}$$
$$\mathbf{X} = \left[\mathcal{I}(\mathbf{p}_1 \circ \Delta\mathbf{p}_1) - \mathcal{T}(\mathbf{0}), \ldots, \mathcal{I}(\mathbf{p}_N \circ \Delta\mathbf{p}_N) - \mathcal{T}(\mathbf{0})\right] .$$

Here, $\|\cdot\|_F$ indicates the matrix Frobenius norm. Without the regularization term $\Omega(\mathbf{R})$, the solution to Equation 2.9 is $\mathbf{R} = \mathbf{Y}\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}$. It is understood that raw pixel representations of natural images stem from certain frequency spectrums [159] that leads to an auto-covariance matrix $\mathbf{X}\mathbf{X}^\top$ which is poorly conditioned in nearly all circumstances. It has been demonstrated [159] that this property stems from the fact that image intensities in natural images are highly correlated in close spatial proximity, but this dependence drops off as a function of spatial distance.

In our experiments, we have found that $\mathbf{X}\mathbf{X}^\top$ is always poorly conditioned even when utilizing other image representations such as dense SIFT, HOG, and LBP descriptors. As such, it is clear that some sort of regularization term is crucial for effective SDM performance. As commonly advocated and practiced, we employed a weighted Tikhonov penalty term $\Omega(\mathbf{R}) = \lambda \|\mathbf{R}\|_F^2$, where λ controls the weight of the regularizer. We found this choice to work well in our experiments.

Iteration-specific regressors. Unlike the IC-LK approach, which employs a single regressor/template pair $\{\mathbf{R}, \mathcal{T}(\mathbf{0})\}$ to be applied iteratively until convergence, SDM learns a set of regressor/template pairs $\{\mathbf{R}^{(l)}, \mathcal{T}^{(l)}(\mathbf{0})\}_{l=1}^{L}$ for each iteration $l = 1 : L$ (sometimes referred to as layers). On the other hand, like the IC-LK algorithm, these regressors are precomputed in advance and thus are independent of the current image and warp estimate. As a result, SDM is computationally efficient just like IC-LK. The regressor/template pair $\{\mathbf{R}^{(l)}, \mathcal{T}^{(l)}(\mathbf{0})\}$ is learned from the synthetically generated set $\mathcal{S}^{(l)}$ within Equation 2.8, which we define to be

$$\mathcal{S}^{(l)} = \{\Delta\mathbf{p}_n^{(l)}, \mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n^{(l)})\}_{n=1}^{N}, \tag{2.10}$$

where

$$\Delta\mathbf{p}^{(l+1)} \leftarrow \mathbf{R}^{(l)}\left[\mathcal{I}\left(\mathbf{p} \circ (\Delta\mathbf{p}^{(l)})^{-1}\right) - \mathcal{T}(\mathbf{0})\right] . \tag{2.11}$$

For the first iteration ($l = 1$), the warp perturbations are generated from a pre-determined random distribution; for every subsequent iteration, the warp perturbations are re-sampled from the same distribution to ensure each iteration's regressor does not overfit. Once learned, SDM is applied by employing Equation 2.11 in practice.

Inverse compositional warps. It should be noted that there is nothing in the original treatment [199] on SDM that limits it to compositional warps. In fact, the original work employing facial landmark alignment advocated an additive update strategy. Here, we have chosen to employ inverse compositional warp updates as: (i) we obtained better results for our experiments with planar warp functions, (ii) we observed almost no difference in performance for non-planar warp functions such as those involved in face alignment, and (iii) it is only through the employment of inverse compositional warps within the LK framework that a firm theoretical motivation for fixed regressors can be entertained. Furthermore, we have found that keeping a close mathematical relationship to the IC-LK algorithm is essential for the motivation of our proposed approach.

2.4 The Conditional Lucas-Kanade Algorithm

Although enjoying impressive results across a myriad of image alignment tasks, SDM does have disadvantages when compared to IC-LK. First, it requires large amounts of synthetically warped image data. Second, it requires the utilization of an adhoc regularization strategy to ensure good condition of the linear system. Third, the mathematical properties of the warp function parameters being predicted is ignored. Finally, it reveals little about the actual degrees of freedom necessary in the set of regressor matrices being learned through the SDM process.

Here, we propose an alternative strategy to directly learn a set of iteration-specific regressors,

$$\min_{\nabla\mathcal{T}(\mathbf{0})} \sum_{n\in\mathcal{S}} \|\Delta\mathbf{p}_n - \mathbf{R}[\mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n) - \mathcal{T}(\mathbf{0})]\|_2^2 \tag{2.12}$$

$$\text{such that} \quad \mathbf{R} = \left(\begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{W}(\mathbf{x}_1;\mathbf{0})}{\partial\mathbf{p}} \\ \vdots \\ \frac{\partial\mathcal{W}(\mathbf{x}_D;\mathbf{0})}{\partial\mathbf{p}} \end{bmatrix} \right)^{\dagger},$$

where

$$\nabla\mathcal{T}(\mathbf{0}) = \begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) \\ \vdots \\ \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix}.$$

At first glance, this objective may seem strange, as we are proposing to learn template "image gradients" $\nabla\mathcal{T}(\mathbf{0})$ within a conditional objective. As previously discussed in [15], this idea deviates from the traditional view of what image gradients are — parameters that are derived from heuristic finite differencing operations. Here, we prefer to subscribe to the alternate view that image gradients are simply weights that can be, and should be, learned from data. The central motivation for this objective is to enforce the parametric form of the generative IC-LK form through a conditional objective.

An advantage of the Conditional LK approach is the reduced number of model parameters. Comparing the model parameters of Conditional LK ($\nabla\mathcal{T}(\mathbf{0}) \in \mathbb{R}^{KD\times 2}$) against SDM ($\mathbf{R} \in \mathbb{R}^{P\times KD}$), there is a reduction in the degrees of freedom needing to be learned for most warp functions where $P > 2$. More fundamentally, however, is the employment of the generative pixel independence assumption described originally in Equation 2.1. This independence assumption is useful as it ensures that a unique $\mathbf{R}$ can be found in Equation 2.12 without any extra penalty terms such as Tikhonov regularization. In fact, we propose that the sparse matrix structure of image gradients within the pseudo-inverse of $\mathbf{R}$ acts as a much more principled form of regularization than those commonly employed within the SDM framework.

A further advantage of our approach is that, like the IC-LK framework, it utilizes prior knowledge of the warp Jacobian function $\frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}$ during the estimation of the regression matrix $\mathbf{R}$. Our insight here is that the estimation of the regression matrix $\mathbf{R}$ using a conditional learning

objective should be simplified (in terms of the degrees of freedom to learn) if one had prior knowledge of the deterministic form of the geometric warp function.

A drawback to the approach, in comparison to both the SDM and IC-LK frameworks, is the non-linear form of the objective in Equation 2.12. This requires us to resort to non-linear optimization methods, which are not as straightforward as linear regression solutions. However, as we discuss in more detail in the experimental portion of this chapter, we demonstrate that a Levenberg-Marquardt optimization strategy obtains good results in nearly all circumstances. Furthermore, compared to SDM, we demonstrate good solutions can be obtained with significantly smaller numbers of training samples.

Iteration-specific regressors. As with SDM, we assume we have an ensemble of images and ground-truth warps $\{\mathcal{I}_m, \mathbf{p}_m\}_{m=1}^{M}$ from which a much larger set of synthetic examples can be generated $\mathcal{S} = \{\Delta\mathbf{p}_n, \mathcal{I}_n(\mathbf{p}_n \circ \Delta\mathbf{p}_n)\}_{n=1}^{N}$, where $N \gg M$. Like SDM, we attempt to learn a set of regressor/template pairs $\{\mathbf{R}^{(l)}, \mathcal{T}^{(l)}(\mathbf{0})\}_{l=1}^{L}$ for each iteration $l = 1 : L$. The set $\mathcal{S}^{(l)}$ of training samples is derived from Equations 2.10 and 2.11 for each iteration. Once learned, the application of these iteration-specific regressors is identical to SDM.

Pixel independence assumption. A major advantage of the IC-LK framework is that it assumes generative independence across pixel coordinates (see Equation 2.1). A natural question to ask is: could not one predict geometric displacement (instead of appearance) directly across independent pixel coordinates?

The major drawback to employing such strategy is its ignorance of the well-known "aperture problem" [119] in computer vision (*e.g.* the motion of an image patch containing a sole edge cannot be uniquely determined due to the ambiguity of motion along the edge). As such, it is impossible to ask any predictor (linear or otherwise) to determine the geometric displacement of all pixels within an image while entertaining an independence assumption. The essence of our proposed approach is that it circumvents this issue by enforcing global knowledge of the template's appearance across all pixel coordinates, while entertaining the generative pixel independence assumption that has served the LK algorithm so well over the last three decades.

Generative LK. For completeness, we will also entertain a generative form of our objective in Equation 2.12, where we instead learn "image gradients" that predict generative appearance as a function of geometric displacement, formulated as

$$\min_{\nabla\mathcal{T}(\mathbf{0})} \sum_{n \in \mathcal{S}} \|\mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n) - \mathcal{T}(\mathbf{0}) - \mathbf{W}\Delta\mathbf{p}_n\|_2^2 \tag{2.13}$$

$$\text{s.t.} \mathbf{W} = \begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{W}(\mathbf{x}_1;\mathbf{0})}{\partial\mathbf{p}} \\ \vdots \\ \frac{\partial\mathcal{W}(\mathbf{x}_D;\mathbf{0})}{\partial\mathbf{p}} \end{bmatrix}.$$

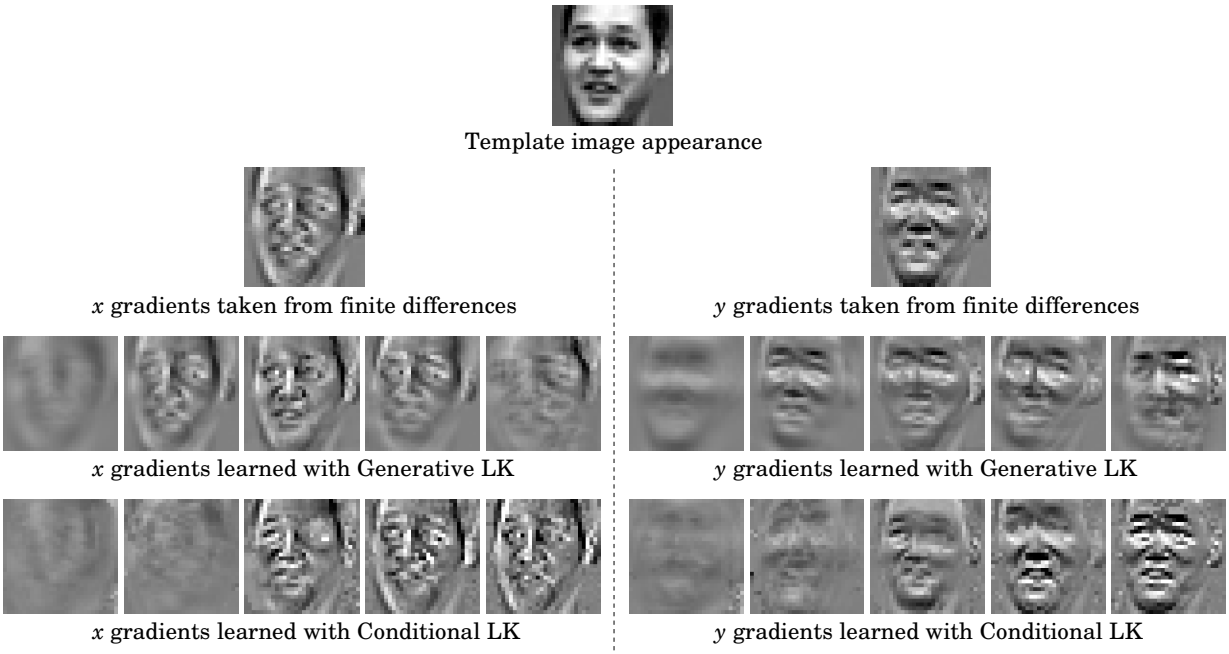


Figure 2.1: Visualization of the learned image gradients for LK from layers 1 (left) to 5 (right).

Unlike our proposed Conditional LK, the objective in Equation 2.13 is linear and directly solvable. Furthermore, due to the generative pixel independence assumption, the problem can be broken down into D independent sub-problems. The Generative LK approach is trained in an identical way to SDM and Conditional LK, where iteration-specific regressors are learned from a set of synthetic examples $\mathcal{S} = \{\Delta\mathbf{p}_n, \mathcal{I}_n(\mathbf{p}_n \circ \Delta\mathbf{p}_n)\}_{n=1}^{N}$.

Figure 2.1 provides an example of visualizing the gradients learned from the Conditional LK and Generative LK approaches. Note that the Conditional LK gradients get sharper over regression iterations, while it is not necessarily the case for Generative LK. The rationale for including the Generative LK form is to highlight the importance of a conditional learning approach, and to therefore justify the added non-linear complexity of the objective in Equation (2.12).

2.5 Experiments

In this section, we present results for our approach across three diverse tasks: (i) planar image alignment, (ii) planar template tracking, and (iii) facial model fitting. We also investigate the utility of our approach across different image representations such as raw pixel intensities and dense LBP descriptors.

Figure 2.2: Visualization of the perturbed samples $\mathcal{S} = \{\Delta\mathbf{p}_n, \mathcal{I}_n(\mathbf{p}_n \circ \Delta\mathbf{p}_n)\}_{n=1}^N$ used for training the SDM, Conditional LK, and Generative LK methods. Left: the original source image, where the red box is the ground truth and the green boxes are perturbed for training. Right: examples of the synthesized training samples.

2.5.1 Planar Image Alignment

Experimental settings. In this experiment, we utilize a subsection of the Multi-PIE [50] dataset. For each image, we denote a 20×20 image $\mathcal{I}(\mathbf{p})$ with ground-truth warp $\mathbf{p}$ rotated, scaled and translated around hand-labeled locations. For the IC-LK approach, this image is then employed as the template $\mathcal{T}(\mathbf{0})$. For the SDM, Conditional LK and Generative LK methods, a synthetic set of geometrically perturbed samples $\mathcal{S}$ are generated $\mathcal{S} = \{\Delta\mathbf{p}_n, \mathcal{I}_n(\mathbf{p}_n \circ \Delta\mathbf{p}_n)\}_{n=1}^N$.

We generate the perturbed samples by adding i.i.d. Gaussian noise of standard deviation σ to the four corners of the ground-truth bounding box as well as an additional translational noise from the same distribution, and then finally fitting the perturbed box to the warp parameters $\Delta\mathbf{p}$. In our experiments, we choose $\sigma = 1.2$ pixels. Figure 2.2 shows an example visualization of the training procedure as well as the generated samples. For SDM, a Tikhonov regularization term is added to the training objective as described in Section 2.3, and the penalty factor λ is chosen by evaluating on a separate validation set; for Conditional LK, we use Levenberg-Marquardt to optimize the non-linear objective where the parameters are initialized through the Generative LK solution.

Frequency of Convergence. We compare the alignment performance of the four types of aligners in our discussion: (i) IC-LK, (ii) SDM, (iii) Generative LK, and (iv) Conditional LK. We state that convergence is reached when the point RMSE of the four corners of the bounding box is less than one pixel.

Figure 2.3 shows the frequency of convergence tested with both a 2D affine and homography warp function. Irrespective of the planar warping function, our results indicate that Conditional LK has superior convergence properties over the others. This result holds even when the approach is initialized with a warp perturbation that is larger than the distribution it was trained under. The

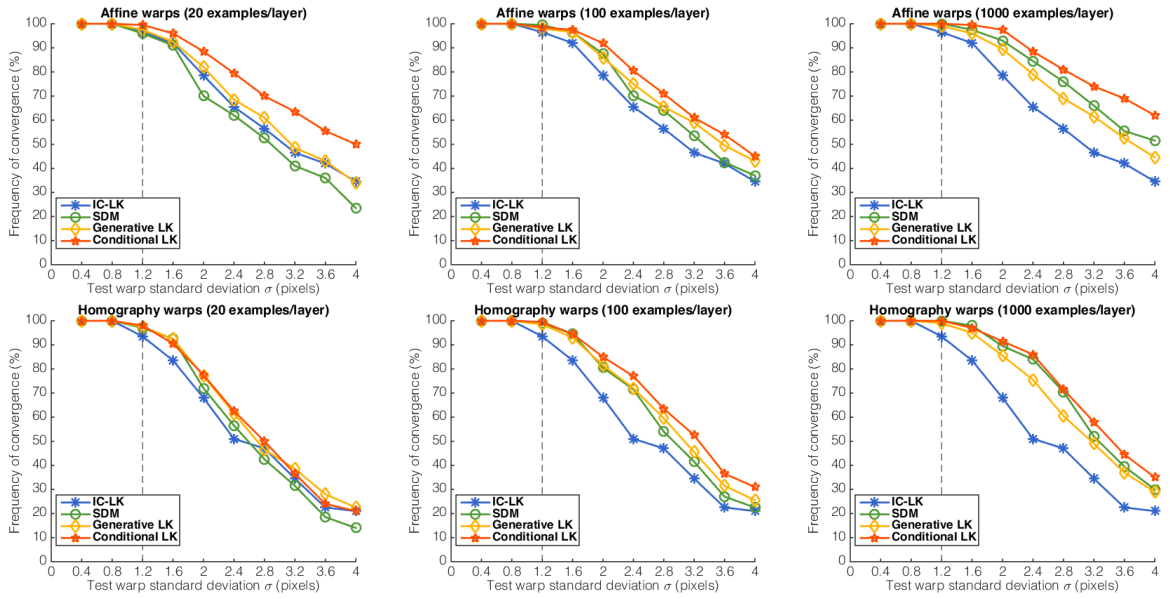


Figure 2.3: Frequency of convergence comparison between IC-LK, SDM, Generative LK, and Conditional LK. The vertical dotted line indicates σ that they were trained with.

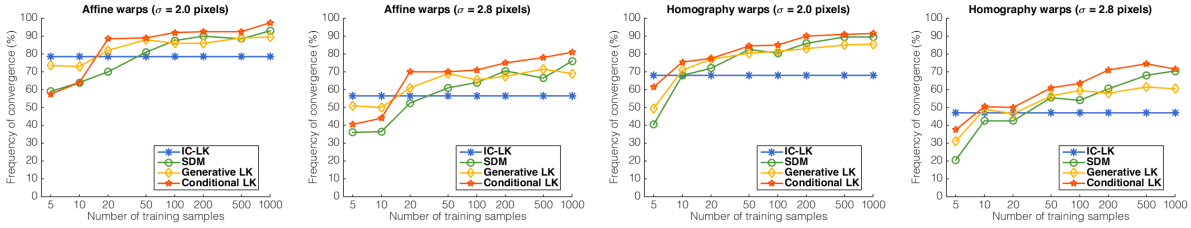


Figure 2.4: Frequency of convergence comparison between SDM, Generative LK, and Conditional LK in terms of number of samples trained with.

alignment performance of Conditional LK is consistently better in all circumstances, although the advantage of the approach is most noticeable when training with just a few training samples.

Figure 2.4 provides another comparison with respect to the amount of training data. It can be observed that SDM is highly dependent on the amount of training data available, but it is still not able to generalize as well as Conditional LK. This is also empirical proof that incorporating principled priors in Conditional LK is more desirable than adhoc regularizations in SDM.

Convergence Rate. We also provide some analysis on the convergence speed. To make a fair comparison, we take the average of only those test runs where all regressors converged. Figure 2.5 illustrates the convergence rates of different regressors learned from different amounts of training data. The improvement of Conditional LK in convergence speed is clear, especially when little training data is provided. SDM starts to exhibit faster convergence rate when learned from over 100 examples per layer; however, Conditional LK still surpasses SDM in term of the frequency of final convergence.

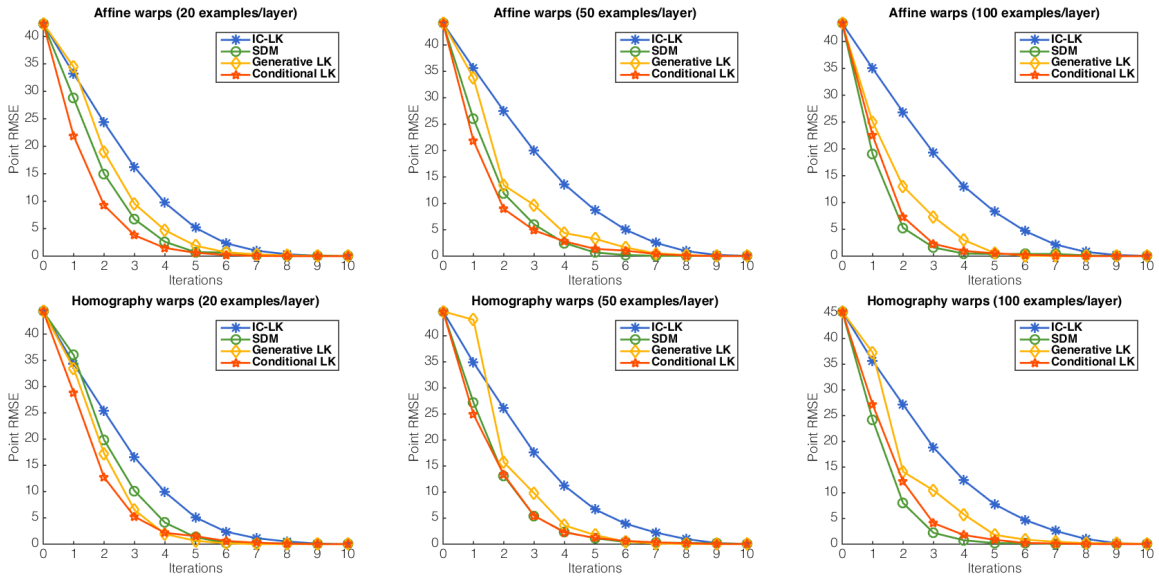


Figure 2.5: Convergence rate comparison between IC-LK, SDM, Generative LK, and Conditional LK, averaged from the tests ($\sigma = 2.8$) where all four converged in the end.

Swapping Warp Functions. A unique property of Conditional LK in relation to SDM is its ability to interchange between warp functions after training. Since we are learning image gradients $\nabla\mathcal{T}(\mathbf{0})$ for the Conditional LK algorithm, one can essentially choose which warp Jacobian to be employed before forming the regressor $\mathbf{R}$. Figure 2.6 illustrates the effect of Conditional LK learning the gradient with one type of warp function and swapping it with another during testing. We see that whichever warp function Conditional LK is learned with, the learned conditional gradients are also effective on the other and still outperforms IC-LK and SDM.

It is interesting to note that when we learn the Conditional LK gradients using either 2D planar similarity warps ($P = 4$) or homography warps ($P = 8$), the performance on 2D planar affine warps ($P = 6$) is as effective. This outcome leads to an important insight: it is possible to learn the conditional gradients with a simple warp function and replace it with a more complex one afterwards; this can be especially useful when certain types of warp functions (*e.g.* 3D warp functions) are harder to come by.

2.5.2 Planar Tracking with LBP Features

In this section, we show how Conditional LK can be effectively employed with dense multi-channel LBP descriptors where $K = 8$. First we analyze the convergence properties of Conditional LK on the dense LBP descriptors, as we did similarly in the previous section, and then we present an application to robust planar tracking. A full description of the multi-channel LBP descriptors we used in our approach can be found in [3].

Figure 2.7 provides a comparison of robustness by evaluating the frequency of convergence

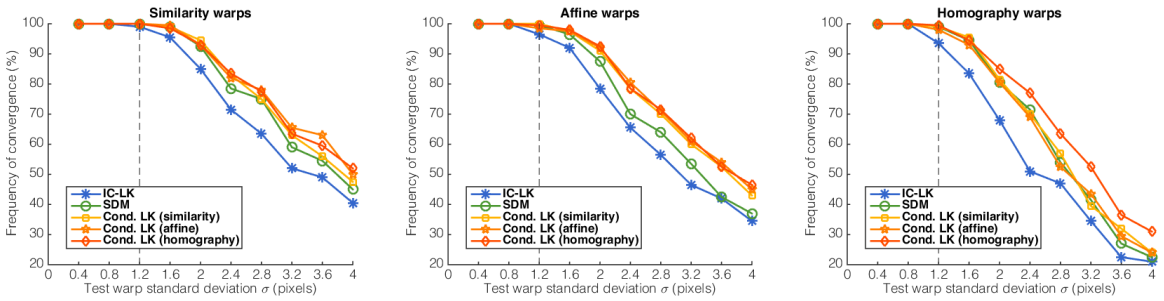


Figure 2.6: Frequency of convergence comparison between IC-LK, SDM, and Conditional LK trained with 100 examples per layer and tested with swapped warp functions. The parentheses indicate the type of warp function trained with.

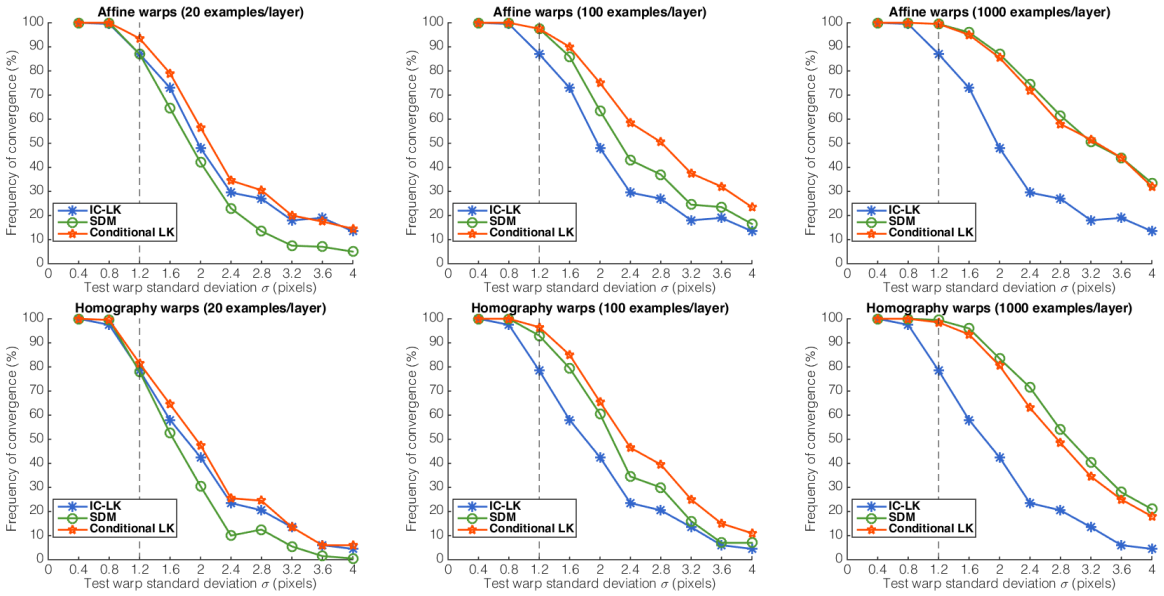


Figure 2.7: Frequency of convergence comparison between IC-LK, SDM and Conditional LK with dense binary descriptors. The vertical dotted line indicates σ that they were trained with.

with respect to the scale of test warps σ. This suggests that Conditional LK is as effective in the LK framework with multi-channel descriptors: in addition to increasing alignment robustness (which is already a well-understood property of descriptor image alignment), Conditional LK is able to improve upon the sensitivity to initialization with larger warps.

Figure 2.8 illustrates alignment performance as a function of the number of training samples. We can see the Conditional LK only requires as few as 20 examples per layer to train a better multi-channel aligner than IC-LK, whereas SDM needs more than 50 examples per iteration-specific regressor. This result again speaks to the efficiency of learning with Conditional LK.

Low Frame-rate Template Tracking. In this experiment, we evaluate the advantage of our proposed approach for the task of low frame-rate template tracking. Specifically, we borrow

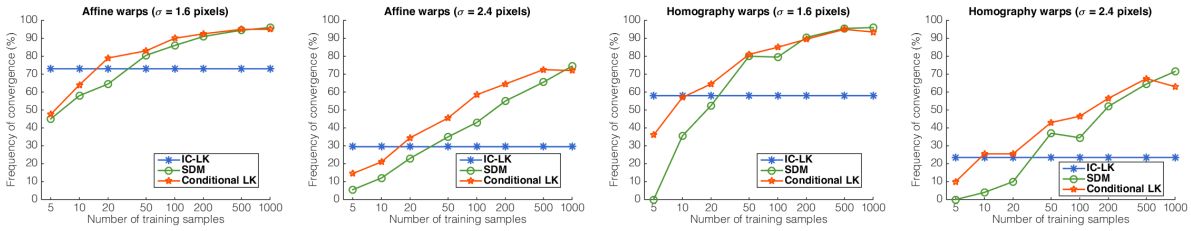


Figure 2.8: Frequency of convergence comparison between SDM and Conditional LK with dense binary descriptors in terms of number of samples trained with.

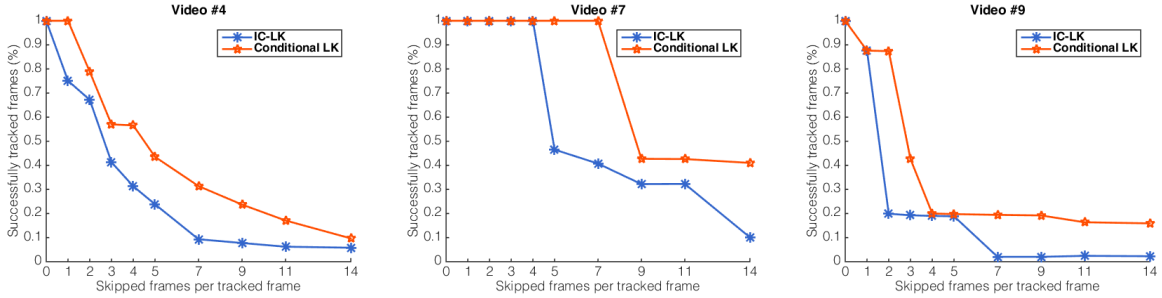


Figure 2.9: Tracking performance using IC-LK and Conditional LK with dense LBP descriptors for three videos under low frame-rate conditions, with and without lighting variations.

a similar experimental setup to Bit-Planes [3]. LBP-style dense descriptors are ideal for this type of task as their computation is computationally feasible in real-time across a number of computational platforms (unlike HOG or dense SIFT). Further computational speedups can be entertained if we start to skip frames to track.

We compare Conditional LK against IC-LK and run the experiments on the videos collected in [3]. We train the Conditional LK tracker on the first frame with 20 synthetic examples. During tracking, we skip every k frames to simulate low frame-rate videos. Figure 2.9 illustrates the percentage of successfully tracked frames over the number of skipped frames k. It is clear that the Conditional LK tracker is more stable and tolerant to larger displacements between frames.

Figure 2.10 shows some video snapshots, including the frames where the IC-LK tracker starts to fail but the Conditional LK tracker remains. This further demonstrates that the Conditional LK tracker maintains the same robustness to brightness variations by entertaining dense descriptors, but meanwhile improves upon convergence. Enhanced susceptibility to noises both in motion and brightness also suggests possible extensions to a wide variety of tracking applications.

2.5.3 Facial Model Fitting

In this experiment, we show how Conditional LK is applicable not only to 2D planar warps like affine or homography, but also to more complex warps that requires heavier parametrization. Specifically, we investigate the performance of our approach with a point distribution model

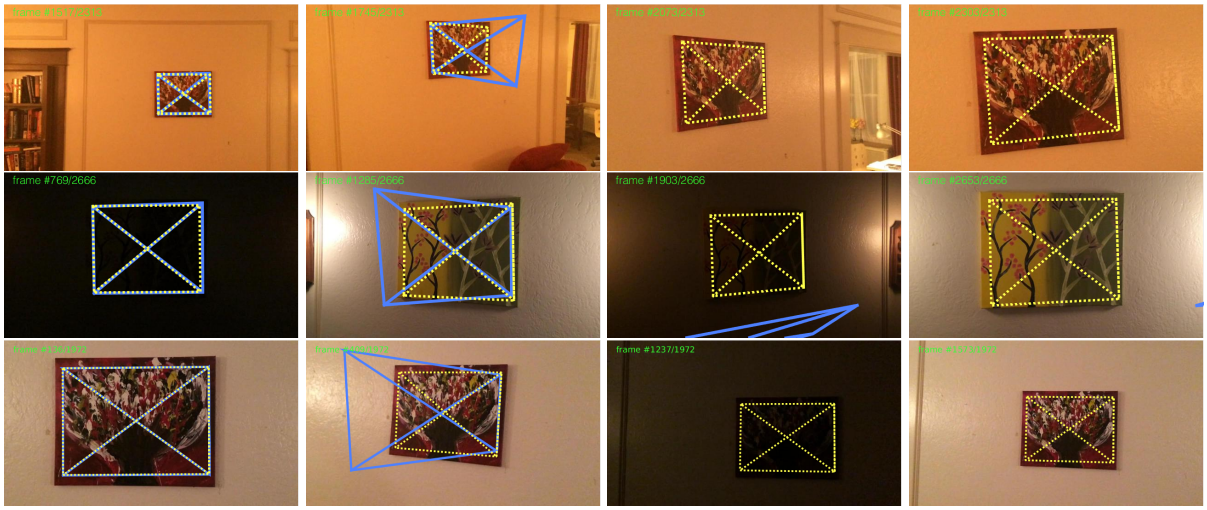


Figure 2.10: Snapshots of tracking results. Blue: IC-LK; yellow: Conditional LK. The second image of each row shows where IC-LK fails but Conditional LK still holds.

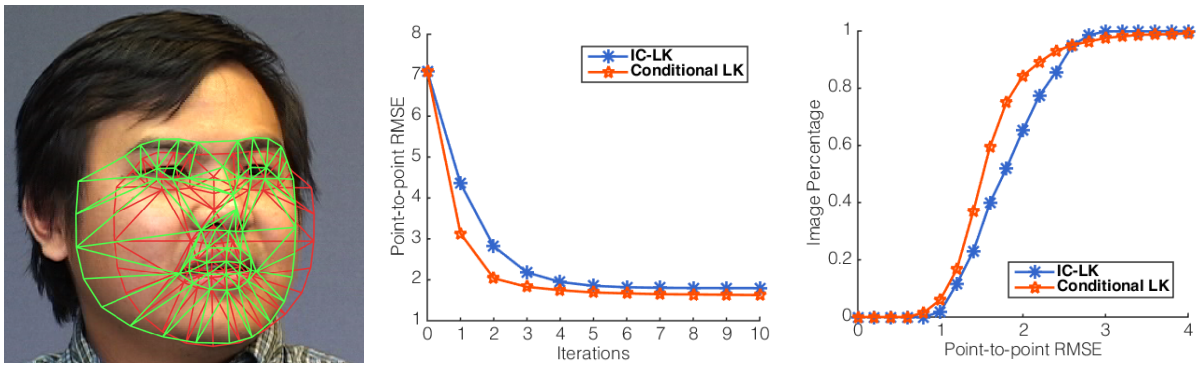


Figure 2.11: (a) An example of facial model fitting. The red shape indicates the initialization, and the green shape is the final fitting result. (b) Convergence rate comparison between IC-LK and Conditional LK. (c) Comparison of fitting accuracy.

(PDM) [120] on the IJAGS dataset [120], which contains an assortment of videos with hand-labeled facial landmarks. We utilize a pretrained 2D PDM learned from all labeled data as the warp Jacobian and compare the Conditional LK approach against IC-LK (it has been shown that there is an IC formulation to facial model fitting [120]). For Conditional LK, we learn a series of regressor/template pairs with 5 examples per layer; for IC-LK, the template image is taken by the mean appearance.

Figure 2.11 shows the results of fitting accuracy and convergence rate of subject-specific alignment measured in terms of the point-to-point RMSE of the facial landmarks; it is clear that Conditional LK outperforms IC-LK in convergence speed and fitting accuracy. This experiment highlights the possibility of extending our proposed Conditional LK to more sophisticated warps. We would like to note that it is possible to take advantage of the Conditional LK warp swapping

property to incorporate a 3D PDM as to introduce 3D shape modeling; this is beyond the scope of discussion of this dissertation.

2.6 Conclusion

In this chapter, we discuss the advantages and drawbacks of the LK algorithm in comparison to SDMs. We argue that by enforcing the pixel independence assumption into a conditional learning strategy we can devise a method that: (i) utilizes substantially less training examples, (ii) offers a principled strategy for regularization, and (iii) offers unique properties for adapting and modifying the warp function after learning. Experimental results demonstrate that the Conditional LK algorithm outperforms both the LK and SDM algorithms in terms of convergence. We also demonstrate that Conditional LK can be integrated with a variety of applications that potentially leads to other exciting avenues for investigation.

2.A Appendix: Math Derivations

We describe the derivation and a few optimization details of the proposed Conditional LK algorithm. For convenience, we repeat the objective here,

$$\min_{\nabla\mathcal{T}(\mathbf{0})} \sum_{n\in\mathcal{S}} \|\Delta\mathbf{p}_n - \mathbf{R}[\mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n) - \mathcal{T}(\mathbf{0})]\|_2^2 \tag{2.14}$$

$$\text{s.t.}\mathbf{R} = \left(\begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{W}(\mathbf{x}_1;\mathbf{0})}{\partial\mathbf{p}} \\ \vdots \\ \frac{\partial\mathcal{W}(\mathbf{x}_D;\mathbf{0})}{\partial\mathbf{p}} \end{bmatrix} \right)^{\dagger},$$

where

$$\nabla\mathcal{T}(\mathbf{0}) = \begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) \\ \vdots \\ \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix}$$

is the compact form of the template "image gradients" we want to learn. For simplicity, we further denote $\mathbf{g} = \text{vec}(\nabla\mathcal{T}(\mathbf{0})) \in \mathbb{R}^{2KD}$ to be the vectorized form of $\nabla\mathcal{T}(\mathbf{0})$, and we use $\mathbf{R}(\mathbf{g})$ here instead of $\mathbf{R}$ to emphasize it is a function of $\mathbf{g}$. Thus we can rewrite Equation 2.14 as

$$\min_{\mathbf{g}} \sum_{n\in\mathcal{S}} \|s_n - \mathbf{R}(\mathbf{g})[\mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n) - \mathcal{T}(\mathbf{0})]\|_2^2 \tag{2.15}$$

$$\text{s.t.}\mathbf{R}(\mathbf{g}) = \left(\mathbf{G}(\mathbf{g}) \frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}} \right)^{\dagger},$$

where

$$\mathbf{G}(\mathbf{g}) = \mathbf{G}\left(\nabla\mathcal{T}(\mathbf{0})\right) = \begin{bmatrix} \nabla\mathcal{T}(\mathbf{x}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla\mathcal{T}(\mathbf{x}_D) \end{bmatrix} .$$

We can expand the pseudo-inverse form of $\mathbf{R}(\mathbf{g})$ to be

$$\mathbf{R}(\mathbf{g}) = \left(\mathbf{H}(\mathbf{g})\right)^{-1} \left(\frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}\right)^{\top} \mathbf{G}(\mathbf{g})^{\top}, \tag{2.16}$$

where

$$\mathbf{H}(\mathbf{g}) = \left(\frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}\right)^{\top} \mathbf{G}(\mathbf{g})^{\top}\mathbf{G}(\mathbf{g}) \frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}$$

is the pseudo-Hessian matrix. By the product rule, the derivative of $\mathbf{R}(\mathbf{g})$ with respect to the jth element of $\mathbf{g}$, denoted as g_j, becomes

$$\frac{\partial\mathbf{R}(\mathbf{g})}{\partial g_j} = \frac{\partial\left(\mathbf{H}(\mathbf{g})\right)^{-1}}{\partial g_j} \left(\frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}\right)^{\top} \mathbf{G}(\mathbf{g})^{\top} + \mathbf{H}(\mathbf{g})^{-1} \left(\frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}\right)^{\top} \mathbf{\Lambda}_j^{\top} , \tag{2.17}$$

where $\mathbf{\Lambda}_j = \frac{\partial\mathbf{G}(\mathbf{g})}{\partial g_j}$ is an indicator matrix with only the element in $\mathbf{G}(\mathbf{g})$ corresponding to g_j being active. The derivative of $\left(\mathbf{H}(\mathbf{g})\right)^{-1}$ with respect to g_j is readily given as

$$\frac{\partial\left(\mathbf{H}(\mathbf{g})\right)^{-1}}{\partial g_j} = -\left(\mathbf{H}(\mathbf{g})\right)^{-1} \frac{\partial\mathbf{H}(\mathbf{g})}{\partial g_j} \left(\mathbf{H}(\mathbf{g})\right)^{-1} , \tag{2.18}$$

where

$$\frac{\partial\mathbf{H}(\mathbf{g})}{\partial g_j} = \left(\frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}}\right)^{\top} \left(\mathbf{G}(\mathbf{g})^{\top}\mathbf{\Lambda}_j + \mathbf{\Lambda}_j^{\top}\mathbf{G}(\mathbf{g})\right) \frac{\partial\mathcal{W}(\mathbf{x};\mathbf{0})}{\partial\mathbf{p}} . \tag{2.19}$$

Now that we have obtained explicit expression of $\frac{\partial\mathbf{R}(\mathbf{g})}{\partial\mathbf{g}}$, we can optimize $\mathbf{g}$ through gradient-based optimization methods by iteratively solving for $\Delta\mathbf{g}$, the updates to $\mathbf{g}$. One can choose to use first-order methods (batch/stochastic gradient descent) or second-order methods (Gauss-Newton or Levenberg-Marquardt). In the second-order method case, for examples, we can first rewrite Equation 2.15 in the vectorized form as

$$\min_{\mathbf{g}} \sum_{n\in\mathcal{S}} \left\| \Delta\mathbf{p}_n - \left[\left(\mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n) - \mathcal{T}(\mathbf{0})\right)^{\top} \otimes \mathbf{I}_P\right] \mathrm{vec}(\mathbf{R}(\mathbf{g})) \right\|_2^2 , \tag{2.20}$$

where $\mathbf{I}_P$ is the identity matrix of size P. Then the iterative update $\Delta\mathbf{g}$ is obtained by solving the least-squares problem

$$\min_{\Delta\mathbf{g}} \sum_{n\in\mathcal{S}} \left\| \Delta\mathbf{p}_n - \left[\left(\mathcal{I}(\mathbf{p}_n \circ \Delta\mathbf{p}_n) - \mathcal{T}(\mathbf{0})\right)^{\top} \otimes \mathbf{I}_P\right] \mathrm{vec}(\mathbf{R}\left(\mathbf{g} + \Delta\mathbf{g}\right)) \right\|_2^2 ,$$

where $\text{vec}(\mathbf{R}(\mathbf{g}+\Delta\mathbf{g}))$ is linearized around $\mathbf{g}$ to be

$$\text{vec}(\mathbf{R}(\mathbf{g}+\Delta\mathbf{g})) \approx \text{vec}(\mathbf{R}(\mathbf{g})) + \frac{\partial \text{vec}(\mathbf{R}(\mathbf{g}))}{\partial \mathbf{g}} \Delta\mathbf{g} \ .$$

Finally, the Conditional LK regressors $\mathbf{R}$ are formed to be

$$\mathbf{R} = \mathbf{R}(\mathbf{g}) = \left(\mathbf{G}(\mathbf{g}) \frac{\partial \mathcal{W}(\mathbf{x};\mathbf{0})}{\partial \mathbf{p}} \right)^{\dagger} \ . \tag{2.21}$$

Chapter 3

Resolving Misalignment in Image Datasets

3.1 Introduction

Rapid advances in deep learning have allowed for the learning of complex functions through convolutional neural networks (CNNs), which have achieved state-of-the-art performances in a plethora of computer vision tasks [59, 90, 160]. Most networks learn to tolerate spatial variations through: (a) spatial pooling layers and/or (b) data augmentation techniques [158]; however, these approaches come with several drawbacks. Data augmentation (*i.e.* the synthetic generation of new training samples through geometric perturbations according to a known noise model) is probably the best known strategy for increasing spatial tolerance within a visual learning system. This is problematic as it can often require an exponential increase in the number of training samples and thus the capacity of the model to be learned. Spatial pooling operations can partially alleviate this problem as they naturally encode spatial invariance within the network architecture and uses sub-sampling to reduce the capacity of the model. However, they have an intrinsic limited range of tolerance to geometric variation they can provide; furthermore, such pooling operations destroy spatial details within the images that could be crucial to the performance of subsequent tasks.

Instead of designing a network to solely give tolerance to spatial variation, another option is to have the network solve for some of the geometric misalignment in the input images [67, 113]. Such a strategy only makes sense, however, if it has lower capacity and computational cost as well as better performance than traditional spatially invariant CNNs. Spatial Transformer Networks (STNs) [71] are one of the first notable attempts to integrate low capacity and computationally efficient strategies for *resolving* — instead of *tolerating* — misalignment with classical CNNs. Jaderberg *et al.* presented a novel strategy for integrating image warping within a neural network and showed that such operations are (sub-)differentiable, allowing for the application of canonical backpropagation to an image warping framework.

The problem of learning a low-capacity relationship between image appearance and geometric distortion is not new in computer vision. Over three and a half decades ago, Lucas-Kanade (LK) [116] proposed the seminal algorithm for gradient descent image alignment. The LK

algorithm can be interpreted as a feed forward network of multiple alignment modules; specifically, each alignment module contains a low-capacity predictor (typically linear) for predicting geometric distortion from relative image appearance, followed by an image resampling/warp operation. The LK algorithm differs fundamentally, however, to STNs in their application: image/object alignment instead of classification.

Putting applications to one side, the LK and STN frameworks share quite similar characteristics however with a criticial exception. In an STN with multiple feed-forward alignment modules, the output image of the previous alignment module is directly fed into the next. As we will demonstate in this chapter, this is problematic as it can create unwanted boundary effects as the number of geometric prediction layers increase. The LK algorithm does not suffer from such problems; instead, it feeds the warp parameters through the network (instead of the warped image) such that each subsequent alignment module in the network resamples the original input source image. Furthermore, the Inverse Compositional (IC) variant of the LK algorithm [9] has demonstrated to achieve equivalently effective alignment by reusing the same geometric predictor in a compositional update form.

Inspired by the IC-LK algorithm, we advocate an improved extension to the STN framework that (a) propagates *warp parameters*, rather than *image intensities*, through the network, and (b) employs the same geometric predictor that could be reapplied for all alignment modules. We propose Inverse Compositional Spatial Transformer Networks (IC-STNs) and show its superior performance over the original STNs across a myriad of tasks, including pure image alignment and joint alignment/classification problems.

We organize the chapter as follows: we give a general review of efficient image/object alignment in Sec. 3.2 and an overview of Spatial Transformer Networks in Sec. 3.3. We describe our proposed IC-STNs in detail in Sec. 3.4 and show experimental results for different applications in Sec. 3.5. Finally, we draw to our conclusion in Sec. 3.6.

3.2 Efficient Image & Object Alignment

In this section, we review the nominal approaches to efficient image/object alignment.

3.2.1 The Lucas-Kanade Algorithm

The Lucas-Kanade (LK) algorithm [116] has been a popular approach for tackling dense alignment problems for images and objects. For a given geometric warp function parameterized by the warp parameters $\mathbf{p}$, one can express the LK algorithm as minimizing the sum of squared differences (SSD) objective in the image space,

$$\min_{\Delta \mathbf{p}} \|\mathcal{I}(\mathbf{p} + \Delta \mathbf{p}) - \mathcal{T}(\mathbf{0})\|_2^2 \ , \tag{3.1}$$

where $\mathcal{I}$ is the source image, $\mathcal{T}$ is the template image to align against, and $\Delta\mathbf{p}$ is the warp update being estimated. Here, we denote $\mathcal{I}(\mathbf{p})$ as the image $\mathcal{I}$ warped with the parameters $\mathbf{p}$. The LK algorithm assumes a approximate linear relationship between appearance and geometric displacements; specifically, it linearizes (3.1) by taking the first-order Taylor approximation as

$$\min_{\Delta\mathbf{p}} \left\| \mathcal{I}(\mathbf{p}) + \frac{\partial \mathcal{I}(\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} - \mathcal{T}(\mathbf{0}) \right\|_2^2 . \tag{3.2}$$

The warp parameters are thus additively updated through $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$, which can be regarded as a quasi-Newton update. The term $\frac{\partial \mathcal{I}(\mathbf{p})}{\partial \mathbf{p}}$, known as the steepest descent image, is the composition of image gradients and the predefined warp Jacobian, where the image gradients are typically estimated through finite differences. As the true relationship between appearance and geometry is seldom linear, the warp update $\Delta\mathbf{p}$ is iteratively estimated and applied until convergence.

A fundamental problem with the canonical LK formulation, which employs addtive updates of the warp parameters, is that $\frac{\partial \mathcal{I}(\mathbf{p})}{\partial \mathbf{p}}$ must be recomputed on the rewarped images for each iteration, greatly impacting computational efficiency. Baker and Matthews [9] devised a computationally efficient variant of the LK algorithm, which they referred to as the Inverse Compositional (IC) algorithm. The IC-LK algorithm reformulates (3.1) to predict the warp update to the template image instead, written as

$$\min_{\Delta\mathbf{p}} \|\mathcal{I}(\mathbf{p}) - \mathcal{T}(\Delta\mathbf{p})\|_2^2 , \tag{3.3}$$

and the linearized least-squares objective is thus formed as

$$\min_{\Delta\mathbf{p}} \left\| \mathcal{I}(\mathbf{p}) - \mathcal{T}(\mathbf{0}) - \frac{\partial \mathcal{T}(\mathbf{0})}{\partial \mathbf{p}} \Delta\mathbf{p} \right\|_2^2 . \tag{3.4}$$

The least-squares solution is given by

$$\Delta\mathbf{p} = \left(\frac{\partial \mathcal{T}(\mathbf{0})}{\partial \mathbf{p}} \right)^{\dagger} (\mathcal{I}(\mathbf{p}) - \mathcal{T}(\mathbf{0})) , \tag{3.5}$$

where the superscript $\dagger$ denotes the Moore-Penrose pseudo-inverse operator. This is followed by the inverse compositional update $\mathbf{p} \leftarrow \mathbf{p} \circ (\Delta\mathbf{p})^{-1}$, where we abbreviate the notation $\circ$ to be the composition of warp functions parameterized by $\mathbf{p}$, and $(\Delta\mathbf{p})^{-1}$ is the parameters of the inverse warp function parameterized by $\Delta\mathbf{p}$.

The solutions of (3.2) and (3.4) are in the form of linear regression, which can be more generically expressed as

$$\Delta\mathbf{p} = \mathbf{R} \cdot \mathcal{I}(\mathbf{p}) + \mathbf{b}, \tag{3.6}$$

where $\mathbf{R}$ is a linear regressor establishing the linear relationship between appearance and geometry, and $\mathbf{b}$ is the bias term. Therefore, LK and IC-LK can be interpreted as belonging to the category of cascaded linear regression approaches for image alignment.

It has been shown [9] that the IC form of LK is effectively equivalent to the original form; the advantage of the IC form lies in its efficiency of computing the fixed steepest descent image $\frac{\partial \mathcal{T}(\mathbf{0})}{\partial \mathbf{p}}$ in the least-squares objective. Specifically, it is evaluated on the static template image $\mathcal{T}$ at the identity warp $\mathbf{p} = \mathbf{0}$ and remains constant across iterations, and thus so is the resulting linear regressor $\mathbf{R}$. This gives an important theoretical proof of concept that a fixed predictor of geometric updates can be successfully employed within an iterative image/object alignment strategy, further reducing unnecessary model capacities.

3.2.2 Learning Alignment from Data

More generally, cascaded regression approaches for alignment can be learned from data given that the distribution of warp displacements is known a priori. A notable example of this kind of approach is the Supervised Descent Method (SDM) [199], which aims to learn the series of linear geometric predictors $\{\mathbf{R}, \mathbf{b}\}$ from data. The formulation of SDM's learning objective is

$$\min_{\mathbf{R},\mathbf{b}} \sum_{n=1}^{N} \sum_{j=1}^{M} \|\delta\mathbf{p}_{n,j} - \mathbf{R} \cdot \mathcal{I}_n(\mathbf{p}_n \circ \delta\mathbf{p}_{n,j}) - \mathbf{b}\|_2^2 , \tag{3.7}$$

where $\delta\mathbf{p}$ is the geometric displacement drawn from a known generating distribution using Monte Carlo sampling, and M is the number of synthetically created examples for each image. Here, the image appearance $\mathcal{I}$ is often replaced with a predefined feature extraction function (*e.g.* SIFT [115] or HOG [25]) of the image. This least-squares objective is typically solved with added regularization (*e.g.* ridge regression) to ensure good matrix condition.

SDM is learned in a sequential manner, *i.e.* the training data for learning the next linear model is drawn from the same generating distribution and applied through the previously learned regressors. This has been a popular approach for its simplicity and effectiveness across various alignment tasks, leading to a large number of variants [7, 100, 147] of similar frameworks. Like the LK and IC-LK algorithms, SDM is another example of employing multiple low-capacity models to establish the nonlinear relationship between appearance and geometry. We draw the readers' attention to [100] for a more formally established link between LK and SDM.

It is a widely agreed that computer vision problems can be solved much more efficiently if misalignment among data is eliminated. Although SDM learns alignment from data and guarantees optimal solutions after each applied linear model, it is not clear whether such alignment learned in a greedy fashion is optimal for the subsequent tasks at hand, *e.g.* classification. In order to optimize in terms of the *final* objective, it would be more favorable to paramterize the model as a deep neural network and optimize the entire model using backpropagation.

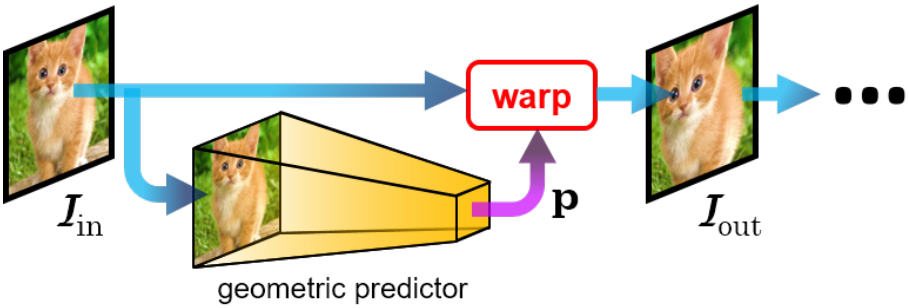


Figure 3.1: Network module of Spatial Transformers [71]. The blue arrows indicate information passing of appearance, and the purple one indicate that of geometry. The yellow 3D trapezoid denotes the geometric predictor, which contains the learnable parameters.

3.3 Spatial Transformer Networks

In the rapidly emerging field of deep learning among with the explosion of available collected data, deep neural networks have enjoyed huge success in various vision problems. Nevertheless, there had not been a principled way of resolving geometric variations in the given data. The recently proposed Spatial Transformer Networks [71] performs spatial transformations on images or feature maps with a (sub-)differentiable module. It has the effects of reducing geometric variations inside the data and has brought great attention to the deep learning community.

In the feed-forward sense, a Spatial Transformer warps an image conditioned on the input. This can be mathematically written as

$$\mathcal{I}_{\text{out}}(\mathbf{0}) = \mathcal{I}_{\text{in}}(\mathbf{p}), \quad \text{where} \quad \mathbf{p} = f(\mathcal{I}_{\text{in}}(\mathbf{0})) \ . \tag{3.8}$$

Here, the nonlinear function f is parametrized as a learnable geometric predictor (termed the localization network in the original paper), which predicts the warp parameters from the input image. We note that the "grid generator" and the "sampler" from the original paper can be combined to be a single warp function. We can see that for the special case where the geometric predictor consists of a single linear layer, f would consists of a linear regressor $\mathbf{R}$ as well as a bias term $\mathbf{b}$, resulting the geometric predictor in an equivalent form of (3.6). This insight elegantly links the STN and LK/SDM frameworks together.

Fig. 3.1 shows the basic architecture of STNs. STNs are of great interest in that transformation predictions can be learned while also showing that grid sampling functions can be (sub-)differentiable, allowing for backpropagation within an end-to-end learning framework.

Despite the similarities STNs have with classic alignment algorithms, there exist some fundamental drawbacks in comparison to LK/SDM. For one, it attempts to directly predict the optimal geometric transformation with a single geometric predictor and does not take advantage of the employment of multiple lower-capacity models to achieve more efficient alignment before classification. Although it has been demonstrated that multiple Spatial Transformers can be inserted

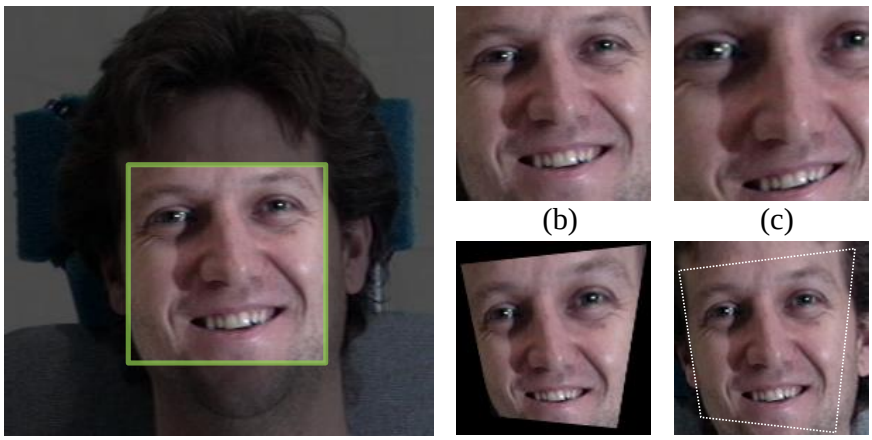


Figure 3.2: Boundary effect of Spatial Transformers on real images. (a) Original image, where the green box indicates the cropped region. (b) Cropped image as the input of the Spatial Transformer. (c) Zoom-in transformation: sampling occurs within the range of the input image. (d)(e) Zoom-out transformation: discarding the information outside the input image introduces a boundary effect (STNs), while it is not the case with geometry preservation (c-STNs). The white dotted box indicates the warp from the original image.

between feature maps, the effectiveness of such employment has on improving performance is not well-understood. In addition, we can observe from (3.8) that no information of the geometric warp $\mathbf{p}$ is preserved after the output image; this leads to a boundary effect when resampling outside the input source image. A detailed treatment on this part is provided in Sec. 3.4.1.

In this work, we aim to improve upon STNs by theoretically connecting it to the LK algorithm. We show that employing multiple low-capacity models as in LK/SDM for learning spatial transformation within a deep network yields substantial improvement on the subsequent task at hand. We further demonstrate the effectiveness of learning a single geometric predictor for recurrent transformation and propose the Inverse Compositional Spatial Transformer Networks (IC-STNs), which exhibit significant improvements over the original STN on various problems.

3.4 Inverse Compositional STNs

3.4.1 Geometry Preservation

One of the major drawbacks of the original Spatial Transformer architecture (Fig. 3.1) is that the output image samples only from the cropped input image; pixel information outside the cropped region is discarded, introducing a boundary effect. Fig. 3.2 illustrates the phenomenon.

We can see from Fig. 3.2(d) that such effect is visible for STNs in zoom-out transformations where pixel information outside the bounding box is required. This is due to the fact that geometric

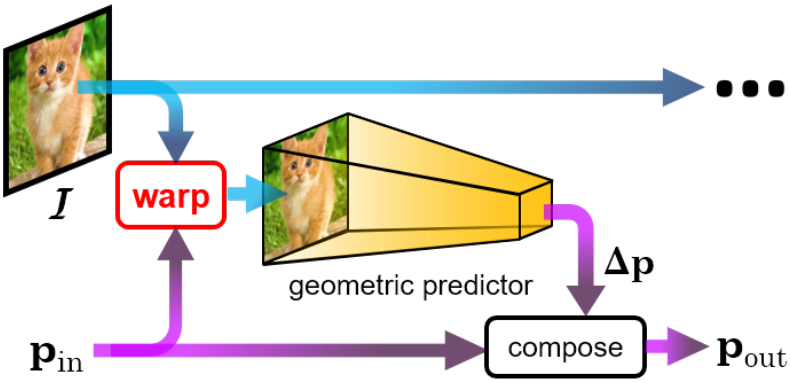


Figure 3.3: A learnable warping module with geometry preserved, termed as c-STNs. The *warp parameters* are passed through the network instead of the *warped images*.

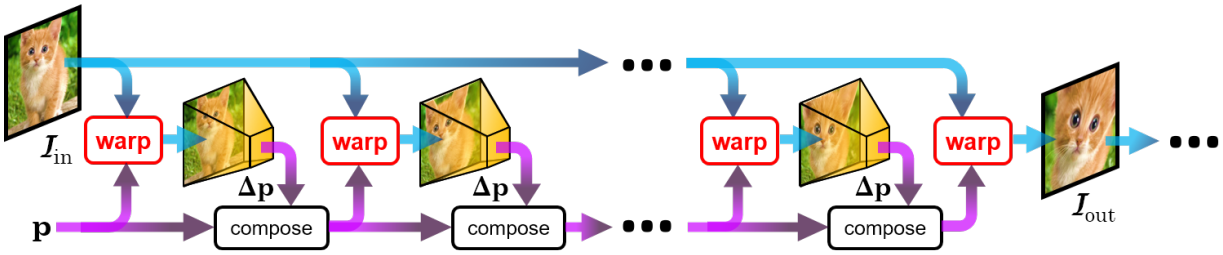


Figure 3.4: Multiple concatenation of c-STNs for an iterative alignment framework.

information is not preserved after the spatial transformations. In the scenario of iterative alignment, boundary effects are accumulated for each zoom-out transformations. Although this is less of an issue with images with clean background, this is problematic with real images.

A series of spatial transformations, however, can be composed and described with exact expressions. Fig. 3.3 illustrates an improved alignment module, which we refer to as compositional STNs (c-STNs). Here, the geometric transformation is also predicted from a geometric predictor, but the *warp parameters* $\mathbf{p}$ are kept track of, composed, and passed through the network instead of the *warped images*. It is important to note that if one were to incorporate a cascade of multiple Spatial Transformers, the geometric transformations are implicitly composed through multiple resampling of the images. We advocate that these transformations are able to be and should be explicitly defined and composed. Unlike the Spatial Transformer module in Fig. 3.1, the geometry is preserved in $\mathbf{p}$ instead of being absorbed into the output image. Furthermore, c-STNs allows repeated concatenation, illustrated in Fig. 3.4, where updates to the warp can be iteratively predicted. This eliminates the boundary effect because pixel information outside the cropped image is also preserved until the final transformation.

The derivative of warp compositions can also be mathematically expressed in closed forms. Consider the input and output warp parameters $\mathbf{p}_{\text{in}}$ and $\mathbf{p}_{\text{out}}$ in Fig. 3.3. Taking the case of affine

warps for example, the parameters $\mathbf{p} = [p_1 \;\; p_2 \;\; p_3 \;\; p_4 \;\; p_5 \;\; p_6]^\top$ are relatable to transformation matrices in the homogeneous coordinates as

$$\mathbf{M}(\mathbf{p}) = \begin{bmatrix} 1+p_1 & p_2 & p_3 \\ p_4 & 1+p_5 & p_6 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.9}$$

From the definition of warp composition, the warp parameters are related to the transformation matrices through

$$\mathbf{M}(\mathbf{p}_{\text{out}}) = \mathbf{M}(\Delta\mathbf{p}) \cdot \mathbf{M}(\mathbf{p}_{\text{in}}). \tag{3.10}$$

We can thus derive the derivative to be

$$\begin{aligned} \frac{\partial \mathbf{p}_{\text{out}}}{\partial \mathbf{p}_{\text{in}}} &= \mathbf{I} + \begin{bmatrix} \Delta p_1 & 0 & 0 & \Delta p_2 & 0 & 0 \\ 0 & \Delta p_1 & 0 & 0 & \Delta p_2 & 0 \\ 0 & 0 & \Delta p_1 & 0 & 0 & \Delta p_2 \\ \Delta p_4 & 0 & 0 & \Delta p_5 & 0 & 0 \\ 0 & \Delta p_4 & 0 & 0 & \Delta p_5 & 0 \\ 0 & 0 & \Delta p_4 & 0 & 0 & \Delta p_5 \end{bmatrix} \\ \frac{\partial \mathbf{p}_{\text{out}}}{\partial \Delta\mathbf{p}} &= \mathbf{I} + \begin{bmatrix} p_{\text{in},1} & p_{\text{in},4} & 0 & 0 & 0 & 0 \\ p_{\text{in},2} & p_{\text{in},5} & 0 & 0 & 0 & 0 \\ p_{\text{in},3} & p_{\text{in},6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{\text{in},1} & p_{\text{in},4} & 0 \\ 0 & 0 & 0 & p_{\text{in},2} & p_{\text{in},5} & 0 \\ 0 & 0 & 0 & p_{\text{in},3} & p_{\text{in},6} & 0 \end{bmatrix}, \end{aligned} \tag{3.11}$$

where $\mathbf{I}$ is the identity matrix. This allows gradients to backpropagate into the geometric predictor.

It is interesting to note that the expression of $\frac{\partial \mathbf{p}_{\text{out}}}{\partial \mathbf{p}_{\text{in}}}$ in (3.11) has a very similar expression as in Residual Networks [59, 60], where the gradients contains the identity matrix $\mathbf{I}$ and "residual components". This suggests that the warp parameters from c-STNs are generally insensitive to the vanishing gradient phenomenon given the predicted warp parameters $\Delta\mathbf{p}$ is small, and that it is possible to repeat the warp/composition operation by a large number of times.

We also note that c-STNs are highly analogous to classic alignment algorithms. If each geometric predictor consists of a single linear layer, *i.e.* the appearance-geometry relationship is assumed to be linearly approximated, then it performs equivalent operations as the compositional LK algorithm. It is also related to SDM, where heuristic features such as SIFT are extracted before each regression layer. Therefore, c-STNs can be regarded as a generalization of LK and SDM, differing that the features for predicting the warp updates can be learned from data and incorporated into an end-to-end learning framework.

3.4.2 Recurrent Spatial Transformations

Of all variants of the LK algorithm, the IC form [9] has a very special property in that the linear regressor remains constant across iterations. The steepest descent image $\frac{\partial \mathcal{T}(\mathbf{0})}{\partial \mathbf{p}}$ in (3.5)

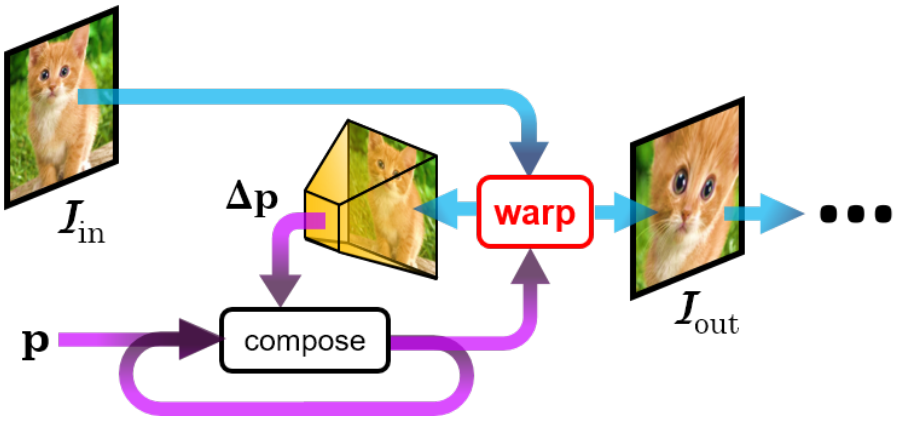


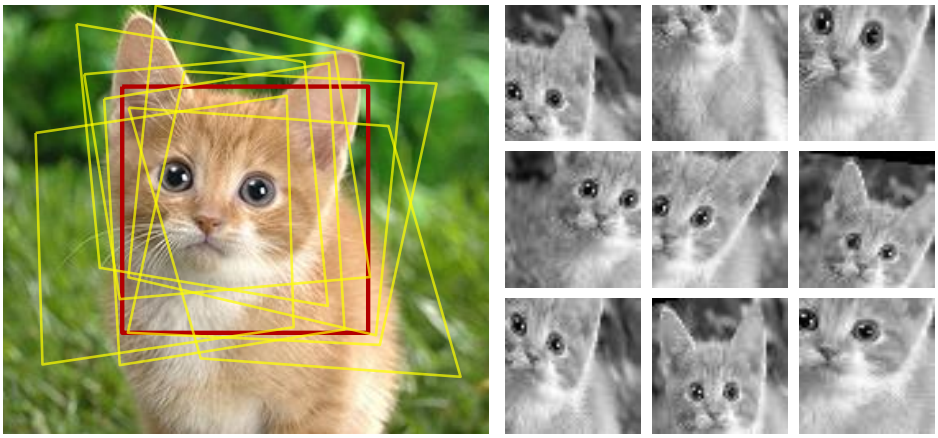
Figure 3.5: Illustration of the proposed Inverse Compositional Spatial Transformer Network (IC-STN). The same geometric predictor is learned to predict recurrent spatial transformations that are composed together to warp the input image.

is independent of the input image and the current estimate of $\mathbf{p}$; therefore, it is only needed to be computed once. In terms of model capacity, IC-LK further reduces the necessary learnable parameters compared to canonical LK, for the same regressor can be applied repeatedly and converges provided a good initialization. The main difference from canonical LK and IC-LK lies in that the warp update $\Delta\mathbf{p}$ should be compositionally applied in the inverse form. We redirect the readers to [9] for a full treatment of IC-LK, which is out of scope of this dissertation.

This inspires us to propose the Inverse Compositional Spatial Transformer Network (IC-STN). Fig. 3.5 illustrates the recurrent module of IC-STN: the warp parameters $\mathbf{p}$ is iteratively updated by $\Delta\mathbf{p}$, which is predicted from the current warped image with the same geometric predictors. This allows one to recurrently predict spatial transformations on the input image. It is possible due to the close spatial proximity of pixel intensities within natural images: there exists high correlation between pixels in close distances.

In the IC-LK algorithm, the predicted warp parameters are inversely composed. Since the IC-STN geometric predictor is optimized in an end-to-end learning framework, we can absorb the inversion operation into the geometric predictor without explicitly defining it; in other words, IC-STNs are able to directly predict the inverse parameters. In our experiments, we find that there is negligible difference to explicitly perform an additional inverse operation on the predicted forward parameters, and that implicitly predicting the inverse parameters fits more elegantly in an end-to-end learning framework using backpropagation. We name our proposed method Inverse Compositional nevertheless as IC-LK is where our inspirations are drawn from.

In practice, IC-STNs can be trained by unfolding the architecture in Fig. 3.5 multiple times into the form of c-STNs (Fig. 3.4), sharing the learnable parameters across all geometric predictors, and backpropagating the gradients as described in Sec. 3.4.1. This results in a single effective geometric predictor that can be applied multiple times before performing the final warp operation that suits subsequent tasks such as classification.



(a) (b)

Figure 3.6: Visualization of the image and perturbed training samples for the planar image alignment experiment. (a) Original image, where the red box indicates the ground-truth warp and the yellow boxes indicate example generated warps. (b) Examples of the perturbed images (affine warps with $\sigma = 7.5$ in this case).

Model	$\sigma = 2.5$	$\sigma = 5$	$\sigma = 7.5$	$\sigma = 10$
c-STN-1	2.699	5.576	9.491	9.218
IC-STN-2	0.615	2.268	5.283	5.502
IC-STN-3	0.434	1.092	2.877	3.020
IC-STN-4	0.292	0.481	1.476	2.287
IC-STN-6	0.027	0.125	0.245	1.305

Table 3.1: Test error for the planar image alignment experiment under different extents of initial perturbations. The number following the model names indicate the number of warp operations unfolded from IC-STN during training.

3.5 Experiments

3.5.1 Planar Image Alignment

To start with, we explore the efficacy of IC-STN for planar alignment of a single image. We took an example image from the Caffe library [75] and generated perturbed images with affine warps around the hand-labeled ground truth, shown in Fig. 3.6. We used image samples of size $50\times$ 50 pixels. The perturbed boxes are generated by adding i.i.d. Gaussian noise of standard deviation σ (in pixels) to the four corners of the ground-truth box plus an additional translational noise from the same Gaussian distribution, and finally fitting the box to the initial warp parameters $\mathbf{p}$.

To demonstrate the effectiveness of iterative alignment under different amount of noise, we consider IC-STNs that consist of a single linear layer with different numbers of learned recurrent transformations. We optimize all networks in terms of L_2 error between warp parameters with

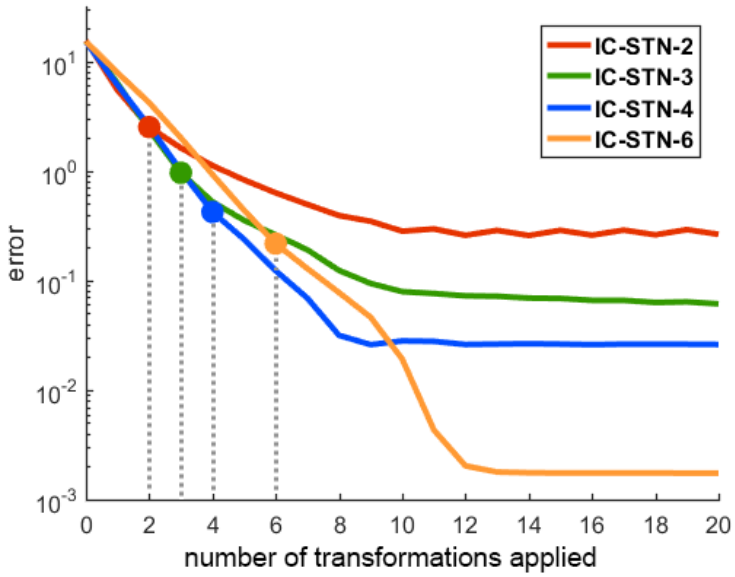


Figure 3.7: Evaluation on trained IC-STNs, where the dot on each curve corresponds to the number of recurrent transformations unfolded during training.

stochastic gradient descent and a batch size of 100 perturbed training samples generated on the fly.

The test error is illustrated in Table 3.1. We see from c-STN-1 (which is equivalent to IC-STN-1 with only one warp operation unfolded) that a single geometric warp predictor has limited ability to directly predict the optimal geometric transformation. Reusing the geometric predictor to incorporating multiple spatial transformations yields better alignment performance given the same model capacity.

Fig. 3.7 shows the test error over the number of warp operations applied to the learned alignment module. We can see that even when the recurrent spatial transformation is applied more times than trained with, the error continues to decrease until some of point of saturation, which typically does not hold true for classical recurrent neural networks. This implies that IC-STN is able to capture the correlation between appearance and geometry to perform gradient descent on a learned cost surface for successful alignment.

3.5.2 MNIST Classification

In this section, we demonstrate how IC-STNs can be utilized in joint alignment/classfication tasks. We choose the MNIST handwritten digit dataset [94], and we use a homography warp noise model to perturb the four corners of the image and translate them with Gaussian noise, both with a standard deviation of 3.5 pixels. We train all networks for 200K iterations with a batch size of 100 perturbed samples generated on the fly. We choose a constant learning rate of 0.01 for the classification subnetworks and 0.0001 for the geometric predictors as we find the geometric predictor sensitive to large changes. We evaluate the classification accuracy on the test set using the same warp noise model.

Model	Error	#Param.	Architecture
CNN(a)	6.597 %	39079	conv(3,3)-conv(36)-P-conv(3,9)-conv(3,12)-fc(48)-fc(10)
STN(a)	4.944 %	39048	[conv(7,4)-conv(7,8)-P-fc(48)-fc(8)]×**1** → conv(9,3)-fc(10)
c-STN-1(a)	3.687 %	39048	[conv(7,4)-conv(7,8)-P-fc(48)-fc(8)]×**1** → conv(9,3)-fc(10)
c-STN-2(a)	2.060 %	38528	[conv(9,4)-fc(8)]×**2** → conv(9×9, 3)-fc(10)
c-STN-4(a)	1.476 %	37376	[fc(8)]×**4** → conv(9,3)-fc(10)
IC-STN-2(a)	**1.905** %	39048	[conv(7,4)-conv(7,8)-P-fc(48)-fc(8)]×**2** → conv(9,3)-fc(10)
IC-STN-4(a)	**1.230** %	39048	[conv(7,4)-conv(7,8)-P-fc(48)-fc(8)]×**4** → conv(9,3)-fc(10)
CNN(b)	19.065 %	19610	conv(9,2)-conv(9,4)-fc(32)-fc(10)
STN(b)	9.325 %	18536	[fc(8)]×**1** → conv(9,3)-fc(10)
c-STN-1(b)	8.545 %	18536	[fc(8)]×**1** → conv(9,3)-fc(10)
IC-STN-2(b)	**3.717** %	18536	[fc(8)]×**2** → conv(9,3)-fc(10)
IC-STN-4(b)	**1.703** %	18536	[fc(8)]×**4** → conv(9,3)-fc(10)

Table 3.2: Classification error on the perturbed MNIST test set. The non-recurrent networks have similar numbers of layers and learnable parameters but different numbers of warp operations (bold-faced). The filter dimensions are shown in parentheses as (kernel size, output channels), where those of the geometric predictor(s) are in green and those of the subsequent classification network are in blue (P denotes a 2×2 max-pooling operation). Best viewed in color.

Figure 3.8: Sample alignment results of IC-STN-4(a) on the MNIST test set with homography warp perturbations. The first row of each column shows the initial perturbation; the middle three rows illustrates the alignment process (iterations 1 to 3); the second last row shows the final alignment before feeding into the classification network. The last row shows the alignment from the original STN: the cropped digits are the results of the boundary effect.

We compare IC-STN to several network architectures, including a baseline CNN with no spatial transformations, the original STN from Jaderberg *et al.*, and c-STNs. All networks with spatial transformations employ the same classification network. The results as well as the architectural details are listed in Table 3.2. We can see that classical CNNs do not handle large spatial variations efficiently with data augmentation. In the case where the digits may be occluded, however, trading off capacity for a single deep predictor of geometric transformation also results in poor performance. Incorporating multiple transformers lead to a significant improvement in classification accuracy; further comparing c-STN-4(a) and IC-STN-4(b), IC-STNs are able to trade little accuracy off for a large reduction of capacity compared to its non-recurrent counterpart.

Fig. 3.8 shows how IC-STNs learns alignment for classification. In many cases where the

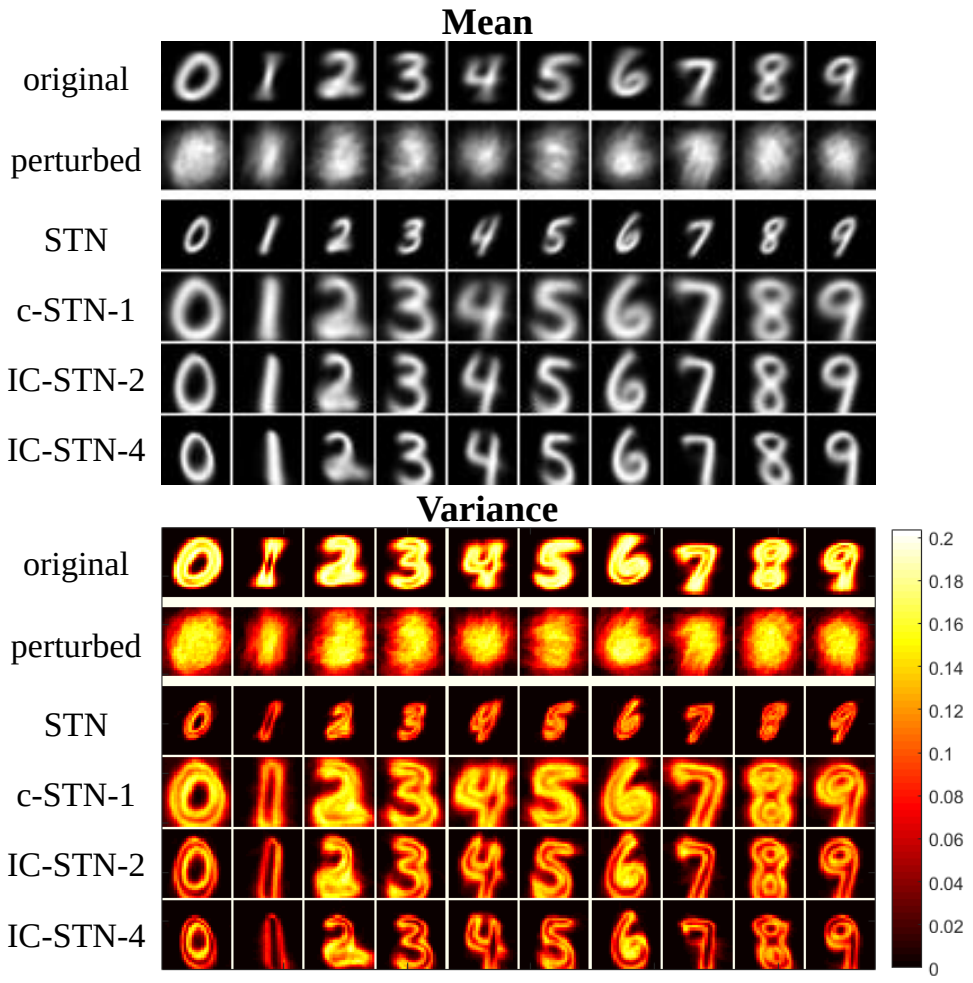


Figure 3.9: Mean/variance of the aligned appearances from the 10 classes of the test set (homography perturbations).

handwritten digits are occluded, IC-STN is able to automatically warp the image and reveal the occluded information from the original image. There also exists smooth transitions during the alignment, which confirms with the recurrent spatial transformation concept IC-STN learns. Furthermore, we see that the outcome of the original STN becomes cropped digits due to the boundary effect described in Sec. 3.4.1.

We visualize the overall final alignment performance by taking the mean and variance on the test set appearance before classification, shown in Fig. 3.9. The mean/variance results of the original STN becomes a down-scaled version of the original digits, reducing information necessary for better classification. From c-STN-1, a single geometric predictor is poor in directly predicting geometric transformations. The variance among all aligned samples is dramatically decreased when more warp operations are introduced in IC-STN. This suggests that elimination of spatial variations within data is crucial to boosting the performance of downstream tasks.

Model	Error	#Param.	Architecture
CNN	8.287 %	200207	conv(7,6)-conv(7,12)-P-conv(7,24)-fc(200)-fc(43)
STN	6.495 %	197343	[conv(7,6)-conv(7,24)-fc(8)]×**1** → conv(7,6)-conv(7,12)-P-fc(43)
c-STN-1	5.011 %	197343	[conv(7,6)-conv(7,24)-fc(8)]×**1** → conv(7,6)-conv(7,12)-P-fc(43)
IC-STN-2	4.122 %	197343	[conv(7,6)-conv(7,24)-fc(8)]×**2** → conv(7,6)-conv(7,12)-P-fc(43)
IC-STN-4	**3.184** %	197343	[conv(7,6)-conv(7,24)-fc(8)]×**4** → conv(7,6)-conv(7,12)-P-fc(43)

Table 3.3: Classification error on the perturbed GTSRB test set. The architectural descriptions follow that in Table 3.2.

Figure 3.10: Sample alignment results of IC-STN-4 on the GTSRB test set in comparison to the original STN.

3.5.3 Traffic Sign Classification

Here, we show how IC-STNs can be applied to real-world classification problems such as traffic sign recognition. We evaluate our proposed method with the German Traffic Sign Recognition Benchmark [168], which consists of 39,209 training and 12,630 test images from 43 classes taken under various conditions. We consider this as a challenging task since many of the images are taken with motion blurs and/or of resolution as low as 15×15 pixels. We rescale all images and generate perturbed samples of size 36×36 pixels with the same homography warp noise model described in Sec. 3.5.2. The learning rate is set to be 0.001 for the classification subnetworks and 0.00001 for the geometric predictors.

We set the controlled model capacities to around 200K learnable parameters and perform similar comparisons to the MNIST experiment. Table 3.3 shows the classification error on the perturbed GTSRB test set. Once again, we see a considerable amount of classification improvement of IC-STN from learning to reuse the same geometric predictor.

Fig. 3.10 compares the aligned images from IC-STN and the original STN before the classification networks. Again, IC-STNs are able to recover occluded appearances from the input image. Although STN still attempts to center the perturbed images, the missing information from occlusion degrades its subsequent classification performance.

We also visualize the aligned mean appearances from each network in Fig. 3.11, and it can be observed that the mean appearance of IC-STN becomes sharper as the number of warp operations increase, once again indicating that good alignment is crucial to the subsequent target tasks. It is also interesting to note that not all traffic signs are aligned to be fit exactly inside the bounding

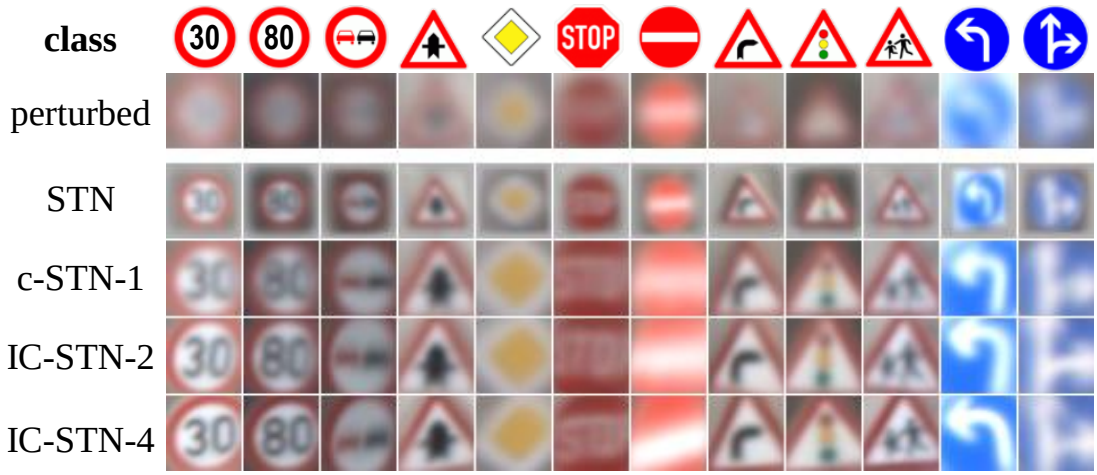


Figure 3.11: Mean aligned appearances for classification on the GTSRB test set.

boxes, *e.g.* the networks finds the optimal alignment for stop signs to be zoomed-in images while excluding the background information outside the octagonal shapes. This suggests that in certain cases, only the pixel information inside the sign shapes are necessary to achieve good alignment for classification.

3.6 Conclusion

In this chapter, we theoretically connect the core idea of the Lucas-Kanade algorithm with Spatial Transformer Networks. We show that geometric variations within data can be eliminated more efficiently through multiple spatial transformations within an alignment framework. We propose Inverse Compositional Spatial Transformer Networks for predicting recurrent spatial transformations and demonstrate superior alignment and classification results compared to baseline CNNs and the original STN.

Chapter 4

Discovering Realism in Geometric Alignment

4.1 Introduction

Generative image modeling has progressed remarkably with the advent of convolutional neural networks (CNNs). Most approaches constrain the possible appearance variations within an image by learning a low-dimensional embedding as an encoding of the natural image subspace and making predictions from this at the pixel level. We refer to these approaches here as *direct image generation*. Generative Adversarial Networks (GANs) [48], in particular, have demonstrated to be an especially powerful tool for realistic image generation. They consist of a generator network ($\mathcal{G}$) that produces images from codes, and a discriminator network ($\mathcal{D}$) that distinguishes real images from fake ones. These two networks play a minimax game that results in $\mathcal{G}$ generating realistic looking images and $\mathcal{D}$ being unable to distinguish between the two when equilibrium is reached.

Direct image generation, however, has its limitations. As the space of images is high-dimensional and image generation methods are limited by finite network capacity, direct image generation methods work better only on restricted domains (*e.g.* faces) or at low resolutions.

In this chapter, we leverage Spatial Transformer Networks (STNs) [71], a special type of CNNs capable of performing geometric transformations on images, to provide a simpler way to generate realistic looking images — by restricting the space of possible outputs to a well-defined *low-dimensional geometric transformation* of real images. We propose Spatial Transformer Generative Adversarial Networks (ST-GANs), which learn Spatial Transformer generators within a GAN framework. The adversarial loss enables us to learn geometric corrections resulting in a warped image that lies at the *intersection* of the natural image manifold and the geometric manifold – the space of geometric manipulations specific to the target image (Fig. 4.1). To achieve this, we advocate a sequential adversarial training strategy to learn iterative spatial transformations that serve to break large transformations down into smaller ones.

We evaluate ST-GANs in the context image compositing, where a source foreground image

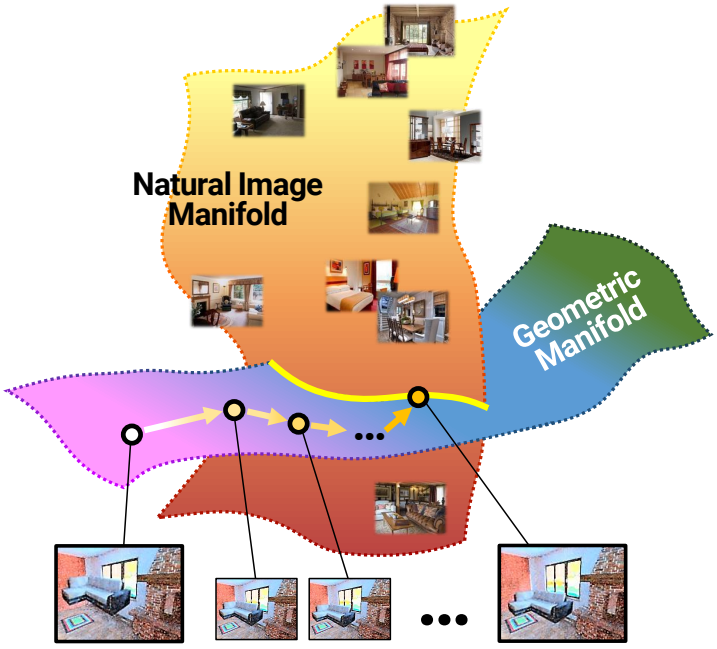


Figure 4.1: Composite images easily fall outside the natural image manifold due to appearance and geometric discrepancies. We seek to learn *geometric corrections* that sequentially warp composite images towards the intersection of the geometric and natural image manifolds.

and its mask are warped by the Spatial Transformer generator $\mathcal{G}$, and the resulting composite is assessed by the discriminator $\mathcal{D}$. In this setup, $\mathcal{D}$ tries to distinguish warped composites from real images, while $\mathcal{G}$ tries to fool $\mathcal{D}$ by generating as realistic looking as possible composites. To our knowledge, we are the first to address the problem of realistic image generation through geometric transformations in a GAN framework. We demonstrate this method on the application of compositing furniture into indoor scenes, which gives a preview of, for example, how purchased items would look in a house. To evaluate in this domain, we created a synthetic dataset of indoor scene images as the background with masked objects as the foreground. We also demonstrate ST-GANs in a fully unpaired setting for the task of compositing glasses on portrait images. A large-scale user study shows that our approach improves the realism of image composites.

Our main contributions are as follows:

- We integrate the STN and GAN frameworks and introduce ST-GAN, a novel GAN framework for finding realistic-looking geometric warps.
- We design a multi-stage architecture and training strategy that improves warping convergence of ST-GANs.
- We demonstrate compelling results in image compositing tasks in both paired and unpaired settings as well as its applicability to high-resolution images.

4.2 Related Work

Image compositing. Image compositing refers to the process of overlaying a masked foreground image on top of a background image. One of the main challenges of image compositing is that the foreground object usually comes from a different scene than the background, and therefore it is not likely to match the background scene in a number of ways that negatively effects the realism of the composite. These can be both appearance differences (lighting, white balance, and shading differences) and geometric differences (changes in camera viewpoint and object positioning).

Existing photo-editing software features various image appearance adjustment operations for that allows users to create realistic composites. Prior work has attempted to automate appearance corrections (*e.g.* contrast, saturation) through Poisson blending [140] or more recent deep learning approaches [175, 219]. In this chapter, we focus on the second challenge: correcting for *geometric* inconsistencies between source and target images.

Spatial Transformer Networks (STNs) [71]. STNs are one way to incorporate learnable image warping within a deep learning framework. A Spatial Transformer module consists of a subnetwork predicting a set of warp parameters followed by a (differentiable) warp function.

STNs have been shown effective in resolving geometric variations for discriminative tasks as well as a wide range of extended applications such as robust filter learning [24, 74], image/view synthesis [45, 136, 207, 216], and 3D representation learning [77, 203, 217]. More recently, Inverse Compositional STNs (IC-STNs) [99] advocated an iterative alignment framework. We borrow the concept of iterative warping but do not enforce recurrence in geometric prediction; instead, we add different generators at each warping step with a sequential training scheme.

Generative Adversarial Networks (GANs) [48]. GANs are a class of generative models that are learned by playing a minimax optimization game between a generator network $\mathcal{G}$ and a discriminator network $\mathcal{D}$. Through this adversarial process, GANs are shown capable of learning a generative distribution that matches the empirical distribution of a given data collection. One advantage of GANs is that the loss function is essentially learned by the discriminator network, allowing for training in cases where ground-truth data with strong supervision is unavailable.

GANs are utilized for data generation in various domains, including images [142], videos [181], and 3D voxelized data [193]. For images in particular, it has been shown to generate compelling results in a vast variety of conditional image generation problems such as super-resolution [92], in-painting [139], image-to-image translation [70, 107, 221], and image editing/manipulation [220].

Recently, STNs were also sought to be adversarially trained for object detection [189], where adversarial examples with feature deformations are generated to robustify object detectors. LR-GAN [205] approached direct image generation problems with additional STNs onto the (directly) generated images to factorize shape variations. We explore the context of STNs with GANs in the space of *conditional* image generation from given inputs, which is a more direct integration of the two frameworks.

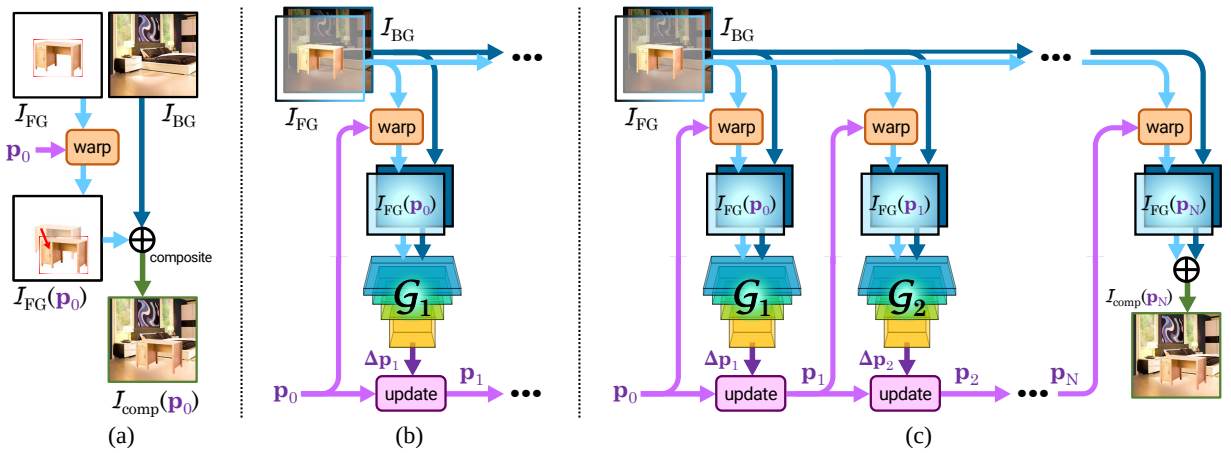


Figure 4.2: Background. (a) Given an initial composite transformation $\mathbf{p}_0$, the foreground image and mask is composited onto the background image using (4.1). (b) Using **Spatial Transformer Networks** (STNs), a geometric prediction network $\mathcal{G}_1$ predicts an update $\Delta\mathbf{p}_1$ conditioned on the foreground and background images, resulting in the new parameters $\mathbf{p}_1$. The update is performed with warp composition (4.3). (c) Our final form is an *iterative* STN to predict a series of accumulative warp updates on the foreground such that the resulting composite image falls closer to the natural image manifold.

4.3 Approach

Our goal is *realistic geometric correction* for image compositing given a background image $\mathcal{I}_{\text{bg}}$ and foreground object $\mathcal{I}_{\text{fg}}$ with a corresponding mask $\mathbf{M}_{\text{fg}}$. We aim to correct the camera perspective, position and orientation of the foreground object such that the resulting composite looks natural. The compositing process can be expressed as:

$$\begin{aligned}\mathcal{I}_{\text{comp}} &= \mathcal{I}_{\text{fg}} \odot \mathbf{M}_{\text{fg}} + \mathcal{I}_{\text{bg}} \odot (1 - \mathbf{M}_{\text{fg}}) \\ &= \mathcal{I}_{\text{fg}} \oplus \mathcal{I}_{\text{bg}} \ . \end{aligned} \tag{4.1}$$

For simplicity, we further introduce the notation $\oplus$ to represent compositing (with $\mathbf{M}_{\text{fg}}$ implied within $\mathcal{I}_{\text{fg}}$). Given the composite parameters $\mathbf{p}_0$ (defining an initial warp state) of $\mathcal{I}_{\text{fg}}$, we can rewrite (4.1) as

$$\mathcal{I}_{\text{comp}}(\mathbf{p}_0) = \mathcal{I}_{\text{fg}}(\mathbf{p}_0) \oplus \mathcal{I}_{\text{bg}} \ , \tag{4.2}$$

where images are written as functions of the warp parameters. This operator is shown in Fig. 4.2(a).

Here, we restrict our geometric warp function to homography transformations, which can represent approximate 3D geometric rectifications for objects that are mostly planar or with small perturbations. As a result, we are making an assumption that the perspective of the foreground object is *close* to the correct perspective; this is often the case when people are choosing similar, but not identical, images from which to composite the foreground object.

The core module of our network design is an STN (Fig. 4.2(b)), where the geometric prediction network $\mathcal{G}$ predicts a correcting update $\Delta\mathbf{p}_1$. We condition $\mathcal{G}$ on both the background and

foreground images, since knowing how an object should be transformed to fit a background scene requires knowledge of the complex interaction between the two. This includes geometry of the object and the background scene, the relative camera position, and semantic understanding of realistic object layouts (*e.g.* having a window in the middle of the room would not make sense).

4.3.1 Iterative Geometric Corrections

Predicting large displacement warp parameters from image pixels is extremely challenging, so most prior work on image alignment predict local geometric transformations in an iterative fashion [9, 65, 100, 116, 199]. Similarly, we propose to use iterative STNs to predict a series of warp updates, shown in Fig. 4.2(c). At the ith iteration, given the input image $\mathcal{I}$ and the previous warp state $\mathbf{p}_{i-1}$, the correcting warp update $\Delta\mathbf{p}_i$ and the new warp state $\mathbf{p}_i$ can be written as

$$\begin{aligned} \Delta\mathbf{p}_i &= \mathcal{G}_i\big(\mathcal{I}_{\text{fg}}(\mathbf{p}_{i-1}), \mathcal{I}_{\text{bg}}\big) \\ \mathbf{p}_i &= \mathbf{p}_{i-1} \circ \Delta\mathbf{p}_i \ , \end{aligned} \tag{4.3}$$

where $\mathcal{G}_i(\cdot)$ is the geometric prediction network and $\circ$ denotes composition of warp parameters. This family of iterative STNs preserves the original images from loss of information due to multiple warping operations [99].

4.3.2 Sequential Adversarial Training

In order for STNs to learn geometric warps that map images closer to the natural image manifold, we integrate them into a GAN framework, which we refer to as ST-GANs. The motivation for this is two-fold. First, learning a realistic geometric correction is a multi-modal problem (*e.g.* a bed can reasonably exist in multiple places in a room); second, supervision for these warp parameters are typically not available. The main difference of ST-GANs from conventional GANs is that (1) $\mathcal{G}$ generates a set of *low-dimensional* warp parameter updates instead of images (the whole set of pixel values); and (2) $\mathcal{D}$ gets as input the warped foreground image composited with the background.

To learn gradual geometric improvements toward the natural image manifold, we adopt a sequential adversarial training strategy for iterative STNs (Fig. 4.3), where the geometric predictor $\mathcal{G}$ corresponds to the stack of generators $\mathcal{G}_i$. We start by training a single $\mathcal{G}_1$, and each subsequent new generator $\mathcal{G}_i$ is added and trained by fixing the weights of all previous generators $\{\mathcal{G}_j\}_{j=1\cdots i-1}$. As a result, we train only $\mathcal{G}_i$ and $\mathcal{D}$ by feeding the resulting composite image at warp state $\mathcal{I}_{\text{comp}}(\mathbf{p}_i)$ into the discriminator $\mathcal{D}$ and matching it against the real data distribution. This learning philosophy shares commonalities with the Supervised Descent Method [199], where a series of linear regressors are solved greedily, and we found it makes the overall training faster and more robust. Finally, we fine-tune the entire network end-to-end to achieve our final result. Note that we use the same discriminator $\mathcal{D}$ for all stages of the generator $\mathcal{G}_i$, as the fundamental measure of "geometric fakeness" does not change over iterations.

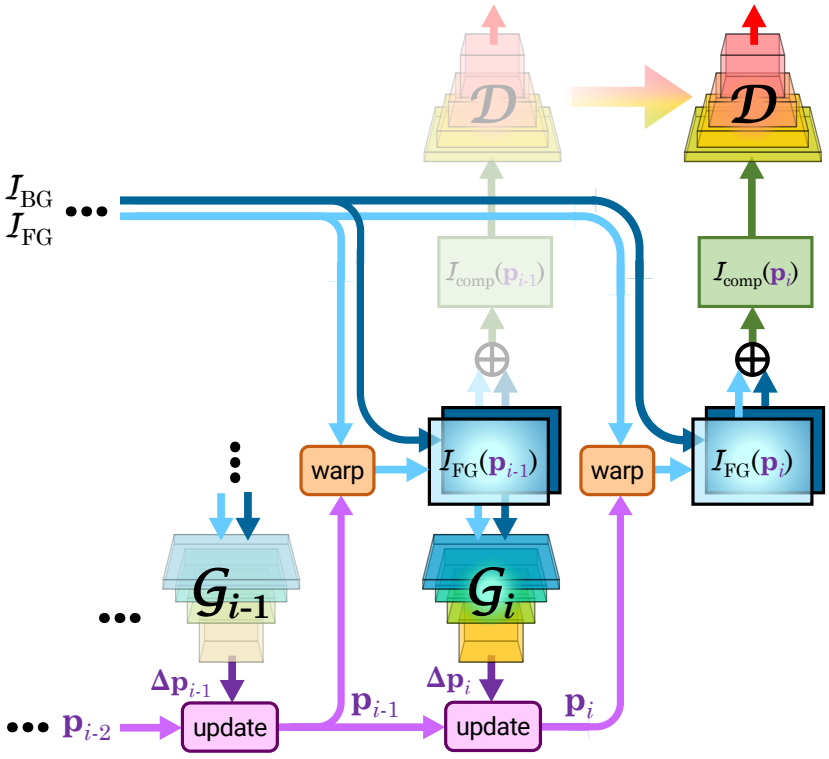


Figure 4.3: **Sequential adversarial training** of ST-GAN. When learning a new warp state $\mathbf{p}_i$, only the new generator $\mathcal{G}_i$ is updated while the previous ones are kept fixed. A single discriminator (learned from all stages) is continuously improved during the sequential learning process.

4.3.3 Adversarial Objective

We optimize the Wasserstein GAN (WGAN) [6] objective for our adversarial game. We note that ST-GAN is amenable to any other GAN variants [10, 118, 215], and that the choice of GAN architecture is orthogonal to this work.

The WGAN minimax objective at the ith stage is

$$\min_{\mathcal{G}_i} \max_{\mathcal{D}\in\mathbb{D}} \mathop{\mathbb{E}}_{\substack{x\sim\mathbb{P}_{\text{fake}} \\ \mathbf{p}_i\sim\mathbb{P}_{\mathbf{p}_i|\mathbf{p}_{i-1}}}} \big[\mathcal{D}\big(x(\mathbf{p}_i)\big)\big] - \mathop{\mathbb{E}}_{y\sim\mathbb{P}_{\text{real}}} \big[\mathcal{D}(y)\big] \,, \tag{4.4}$$

where $y = \mathcal{I}_{\text{real}}$ and $x = \mathcal{I}_{\text{comp}}$ are drawn from the real data and fake composite distributions, and $\mathbb{D}$ is the set of 1-Lipschitz functions enforced by adding a gradient penalty term $\mathcal{L}_{\text{grad}}$ [53]. Here, $\mathbf{p}_i$ (where $\mathcal{G}_i$ is implied, defined in (4.3)) is drawn from the posterior distribution conditioned on $\mathbf{p}_{i-1}$ (recursively implied). When $i = 1$, the initial warp $\mathbf{p}_0$ is drawn from $\mathbb{P}_{\text{pert}}$, a predefined distribution for geometric data augmentation.

We also constrain the warp update $\Delta\mathbf{p}_i$ to lie within a trust region by introducing an additional penalty $\mathcal{L}_{\text{update}} = \|\Delta\mathbf{p}_i\|_2^2$. This is essential since ST-GAN may learn trivial solutions to remove the foreground (*e.g.* by translating it outside the image or shrinking it into nothing), leaving behind only the background image and in turn making the composite image realistic already.

When training ST-GAN sequentially, we update $\mathcal{D}$ and $\mathcal{G}_i$ alternating the respective loss functions:

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{x,\mathbf{p}_i}\big[\mathcal{D}\big(x(\mathbf{p}_i)\big)\big] - \mathbb{E}_y\big[\mathcal{D}(y)\big] + \lambda_{\text{grad}} \cdot \mathcal{L}_{\text{grad}} \tag{4.5}$$

$$\mathcal{L}_{\mathcal{G}_i} = -\mathbb{E}_{x,\mathbf{p}_i}\big[\mathcal{D}\big(x(\mathbf{p}_i)\big)\big] + \lambda_{\text{update}} \cdot \mathcal{L}_{\text{update}} \ , \tag{4.6}$$

where λ_{grad} and λ_{update} are the penalty weights for the $\mathcal{D}$ gradient and the warp update $\Delta\mathbf{p}_i$ respectively, and $\mathcal{G}_i$ and $\Delta\mathbf{p}_i$ are again implied through (4.3). When fine-tuning ST-GAN with N learned updates end-to-end, the generator objective is the sum of that from each $\mathcal{G}_i$, *i.e.* $\mathcal{L}_{\mathcal{G}} = \sum_{i=1}^{N} \mathcal{L}_{\mathcal{G}_i}$.

4.4 Experiments

We begin by describing the basic experimental settings.

Warp parameterizations. We parameterize a homography with the $\mathfrak{sl}(3)$ Lie algebra [122], *i.e.* the warp parameters $\mathbf{p} \in \mathfrak{sl}(3)$ and homography matrices $\mathbf{H} \in \mathbb{SL}(3)$ are related through the exponential map. Under this parameterization, warp composition can be expressed as the addition of parameters, *i.e.* $\mathbf{p}_a \circ \mathbf{p}_b \equiv \mathbf{p}_a + \mathbf{p}_b \quad \forall \mathbf{p}_a, \mathbf{p}_b \in \mathfrak{sl}(3)$.

Model architecture. We denote the following: $\mathbf{C}(k)$ is a 2D convolutional layer with k filters of size 4×4 and stride 2 (halving the feature map resolution) and $\mathbf{L}(k)$ is a fully-connected layer with k output nodes. The input of the generators $\mathcal{G}_i$ has 7 channels: RGBA for foreground and RGB for background, and the input to the discriminator $\mathcal{D}$ is the composite image with 3 channels (RGB). All images are rescaled to 120×160, but we note that the parameterized warp can be applied to full-resolution images at test time.

The architecture of $\mathcal{G}$ is $\mathbf{C}(32)$-$\mathbf{C}(64)$-$\mathbf{C}(128)$-$\mathbf{C}(256)$-$\mathbf{C}(512)$-$\mathbf{L}(256)$-$\mathbf{L}(8)$, where the output is the 8-dimensional (in the case of a homography) warp parameter update $\Delta\mathbf{p}$. For each convolutional layer in $\mathcal{G}$, we concatenate a downsampled version of the original image (using average pooling) with the input feature map. For $\mathcal{D}$, we use a PatchGAN architecture [70], with layout $\mathbf{C}(32)$-$\mathbf{C}(64)$-$\mathbf{C}(128)$-$\mathbf{C}(256)$-$\mathbf{C}(512)$-$\mathbf{C}(1)$. Nonlinearity activations are inserted between all layers, where they are ReLU for $\mathcal{G}$ and LeakyReLU with slope 0.2 for $\mathcal{D}$. We omit all normalization layers as we found them to deteriorate training performance.

4.4.1 3D Cubes

To begin with, we validate whether ST-GANs can make geometric corrections in a simple, artificial setting. We create a synthetic dataset consisting of a 3D rectangular room, an axis-aligned cube inside the room, and a perspective camera (Fig. 4.4(a)). We apply random 3-DoF translations to

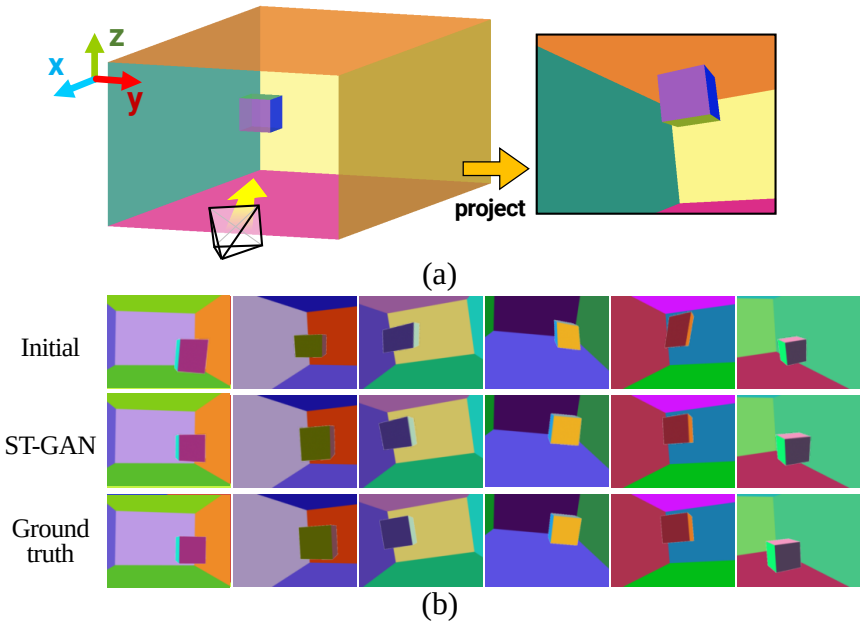


Figure 4.4: (a) We create a synthetic dataset of 3D cube renderings and validate the efficacy of ST-GAN by attempting to correct randomly generated geometric perturbations. (b) ST-GAN is able to correct the cubes to a right perspective, albeit a possible translational offset from the ground truth.

the cube and 6-DoF perturbations to the camera, and render the cube/room pair separately as the foreground/background (of resolution 120×160). We color the cube and the room randomly.

We perturb the rendered foreground cubes with random homography transformations as the initial warp $\mathbf{p}_0$ and train ST-GAN by pairing the original cube as the ground-truth counterpart for $\mathcal{D}$. As shown in Fig. 4.4(b), ST-GAN is able to correct the perturbed cubes scale and perspective distortion with respect to the underlying scene geometry. In addition, ST-GAN is sometimes able to discover other realistic solutions (*e.g.* not necessarily aligning back to the ground-truth location), indicating ST-GAN's ability to learn the multi-modal distribution of correct cube placements.

4.4.2 Indoor Objects

Next, we show how ST-GANs can be applied to practical image compositing domains. We choose the application of compositing furniture in indoor scenes and demonstrate its efficacy on both simulated and real-world images. To collect training data, we create a synthetic dataset consisting of rendered background scenes and foreground objects with masks. We evaluate on the synthetic test set as well as high-resolution real world photographs to validate whether ST-GAN also generalizes to real images.

Category	Training set		Test set	
	# 3D inst.	# pert.	# 3D inst.	# pert.
Bed	3924	11829	414	1281
Bookshelf	508	1280	58	137
Cabinet	9335	31174	1067	3518
Chair	196	609	22	60
Desk	64	1674	73	214
Dresser	285	808	31	84
Refrigerator	3802	15407	415	1692
Sofa	3604	11165	397	1144
Total	22303	73946	2477	8130

Table 4.1: Dataset statistics for the indoor object experiment, reporting the number of object instances chosen for perturbation, and the final number of rendered perturbed samples.

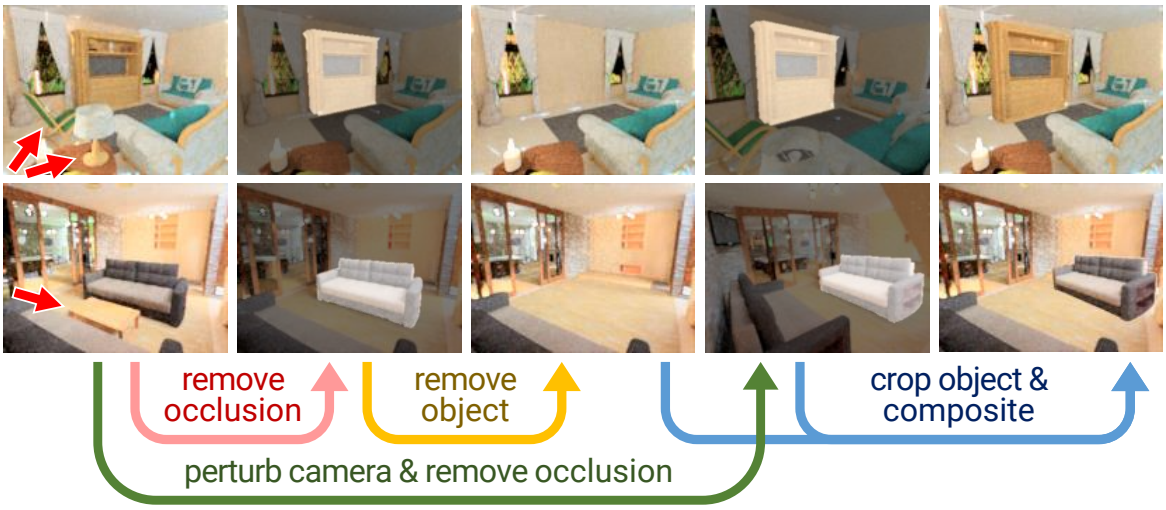


Figure 4.5: **Rendering pipeline.** Given an indoor scene and a candidate object, we remove occluding objects to create an occlusion-free scenario, which we do the same at another perturbed camera pose. We remove the object to create a training sample pair with mismatched perspectives.

Data preparation. We render synthetic indoor scene images from the SUNCG dataset [165], consisting of 45,622 indoor scenes with over 5M 3D object instances from 37 categories [157]. We use the selected 41,499 scene models and the 568,749 camera viewpoints from Zhang *et al.* [214] and utilize Mitsuba [72] to render photo-realistic images with global illumination. We keep a list of candidate 3D objects consisting of all instances visible from the camera viewpoints and belonging to the categories listed in Table 4.1.

The rendering pipeline is shown in Fig. 4.5. During the process, we randomly sample an object from the candidate list, with an associated camera viewpoint. To emulate an occlusion-free compositing scenario, occlusions are automatically removed by detecting overlapping object

Category	Initial warp	SDM [199]	HomographyNet [32]	ST-GAN (non-seq.)	ST-GAN (warp 1)	ST-GAN (warp 2)	ST-GAN (warp 4)	**ST-GAN (end-to-end)**	Ground truth
Bed	35.5 %	30.5 %	30.2 %	32.8 %	32.8 %	**46.8 %**	32.8 %	32.2 %	75.0 %
Bookshelf	21.1 %	33.9 %	35.1 %	16.7 %	26.4 %	26.2 %	39.5 %	**42.6 %**	68.9 %
Cabinet	20.9 %	19.8 %	35.0 %	36.6 %	14.3 %	31.2 %	44.4 %	**50.0 %**	74.3 %
Chair	32.8 %	36.8 %	47.6 %	50.9 %	**62.3%**	42.7 %	50.0 %	58.6 %	68.7 %
Desk	18.9 %	13.1 %	36.1 %	35.4 %	29.2 %	29.0 %	39.4 %	**40.7 %**	65.1 %
Dresser	14.9 %	18.6 %	20.7 %	16.7 %	24.6 %	27.4 %	29.7 %	**48.4 %**	66.1 %
Refrigerator	37.1 %	21.4 %	50.0 %	37.7 %	28.6 %	47.1 %	39.7 %	**51.7 %**	81.6 %
Sofa	15.9 %	31.0 %	42.4 %	28.9 %	37.0 %	54.9 %	**56.1 %**	51.8 %	78.2 %
Average	24.6 %	25.6 %	37.1 %	31.9 %	31.9 %	38.2 %	41.5 %	**47.0 %**	72.6 %

Table 4.2: **AMT user studies** for the indoor objects experiment. Percentages represent the how often the images in each category were classified as "real" by Turkers. Our final model, ST-GAN (end-to-end), substantially improves over geometric realism when averaged across all classes. Our realism performance improves with the number of warps trained as well as after the end-to-end fine-tuning. The ground truth numbers serve as a theoretical upper bound for all methods.

masks. We render one image with the candidate object present (as the "real" sample) and one with it removed (as the background image). In addition, we perturb the 6-DoF camera pose and render the object with its mask (as the foreground image) for compositing. We thus obtain a rendered object as viewed from a *different camera perspective*; this simulates the image compositing task where the foreground and background perspectives mismatch. We note that a homography correction can only approximate these 3D perturbations, so there is *no planar ground-truth warp* to use for supervision. We report the statistics of our rendered dataset in Table 4.1. All images are rendered at 120×160 resolution.

Settings. Similar to the prior work by Lin & Lucey [99], we train ST-GAN for $N = 4$ sequential warps During adversarial training, we rescale the foreground object randomly from $\text{Unif}(0.9, 1.1)$ and augment the initial warp $\mathbf{p}_0$ with a translation sampled from $\mathcal{N}(0, 0.05)$ scaled by the image dimensions. We set $\lambda_{\text{update}} = 0.3$ for all methods.

Baselines. One major advantage of ST-GAN is that it can learn from "realism" comparisons *without ground-truth warp parameters* for supervision. However, prior approaches require supervision directly on the warp parameters. Therefore, we compare against self-supervised approaches trained with random homography perturbations on foreground objects as input, yielding warp parameters as self-supervision. We reemphasize that such direct supervision is insufficient in this application as we aim to find the closest point on a manifold of realistic looking composites rather than fitting a specific paired model. Our baselines are (1) HomographyNet [32], a CNN-based approach that learns direct regression on the warp parameters, and (2) Supervised Descent Method (SDM) [199], which greedily learns the parameters through cascaded linear regression. We train the SDM baseline for 4 sequential warps as well.

Figure 4.6: **Qualitative evaluation** on the indoor rendering test set. Compared to the baselines trained with direct homography supervision, ST-GAN creates more realistic composites. We find that ST-GAN is able to learn common object-room relationships in the dataset, such as beds being against walls. Note that ST-GANs corrects the perspectives but not necessarily scale, as objects often exist at multiple scales in the real data. We observe that ST-GAN occasionally performs worse for unusual objects (*e.g.* with peculiar colors, last column).

Quantitative evaluation. As with most image generation tasks where the goal is realism, there is no natural quantitative evaluation possible. Therefore, we carry out a perceptual study on Amazon Mechanical Turk (AMT) to assess geometric realism of the warped composites. We randomly chose 50 test images from each category and gather data from 225 participants. Each participant was shown a composite image from a randomly selected algorithm (Table 4.2), and was asked whether they saw any objects whose shape does not look natural in the presented image.

We report the AMT assessment results in Table 4.2. On average, ST-GAN shows a large improvement of geometric realism, and quality improves over the sequential warps. When considering that the warp is restricted to homography transformations, these results are promising, as we are not correcting for more complicated view synthesis effects for out-of-plane rotations such as occlusions. Additionally, ST-GAN, which does not require ground truth warp parameters during training, greatly outperforms other baselines, while SDM yields no improvement and HomographyNet increases realism, but to a lesser degree.

Ablation studies. We found that learning iterative warps is advantageous: compared with a non-iterative version with the same training iterations (non-seq. in Table 4.2), ST-GAN (with multiple generators) approaches geometric realism more effectively with iterative warp updates. In addition, we trained an iterative HomographyNet [32] using the same sequential training strategy as ST-GAN but found little visual improvement over the non-iterative version; we thus focus our comparison against the original [32].

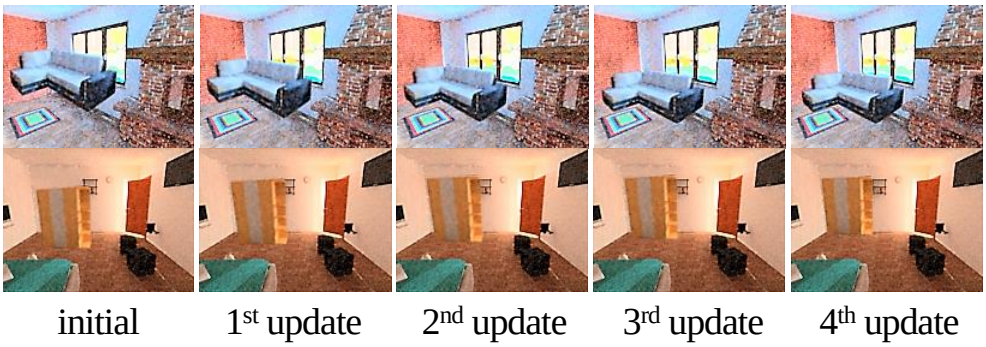


Figure 4.7: Visualization of iterative updates in ST-GAN, where objects make gradual improvements that reaches closer to realism in an incremental fashion.

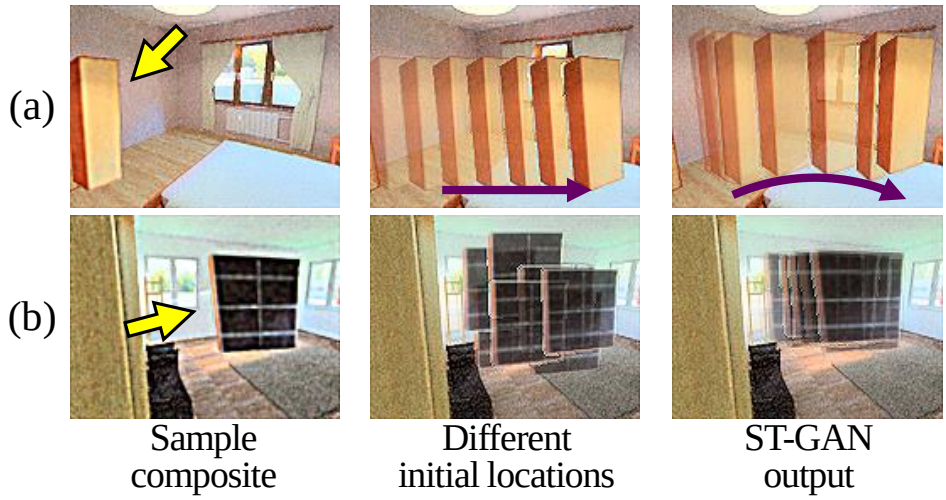


Figure 4.8: **Dragging and snapping.** (a) When an object is dragged across the scene, the perspective changes with the composite location to match the camera. (b) ST-GAN "snaps" objects to where it would likely be composited (*e.g.* a bookshelf is usually laid against the wall).

Qualitative evaluation. We present qualitative results in Fig. 4.6. ST-GAN visually outperforms both baselines trained with direct homography parameter supervision, which is also reflected in the AMT assessment results. Fig. 4.7 shows how ST-GAN updates the homography warp with each of its generators; we see that it learns gradual updates that makes a realism improvement at each step. In addition, we illustrates in Fig. 4.8 the effects ST-GAN learns, including gradual changes of the object perspective at different composite locations inside the room, as well as a "snapping" effect that predicts a most likely composite location given a neighborhood of initial locations. These features are automatically learned from the data, and they can be useful when implemented in interactive settings.

Finally, to test whether ST-GAN extends to real images, we provide a qualitative evaluation on photographic, high-resolution test images gathered from the Internet and manually masked (Fig 4.9). This is feasible since the warp parameters predicted from the low-resolution network

Figure 4.9: **Real world high-resolution test results.** Here we show our method applied to real images. The inputs are scaled down and fed to the network and then the warp parameters are applied at full resolution.

input are transferable to high-resolution images. As a consequence, ST-GAN is indirectly applicable to various image resolutions and not strictly limited as with conventional GAN frameworks. Our results demonstrates the utilization of ST-GAN for high-quality image generation and editing.

4.4.3 Glasses

Finally, we demonstrate results in an entirely unpaired setting where we learn warping corrections for compositing glasses on human faces. The lack of paired data means that we do not necessarily have pictures of the same people both with and without glasses (ground truth).

Data preparation. We use the CelebA dataset [111] and follow the provided training/test split. We then use the "eyeglasses" annotation to separate the training set into two groups. The first group of people with glasses serve as the real data to be matched against in our adversarial settings, and the group of people without glasses serves as the background. This results in 152249 training and 18673 test images without glasses, and 10521 training images with glasses. We hand-crafted 10 pairs of frontal-facing glasses as the foreground source (Fig. 4.10). We note that there are no annotations about where or how the faces are placed, and we do not have any information where the different parts of the glasses are in the foreground images.

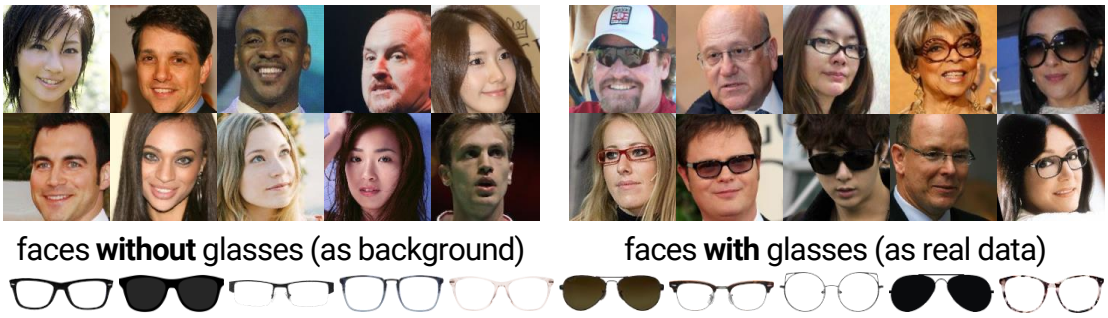


Figure 4.10: The split of CelebA for the background and the real images, as well as the crafted glasses as the foreground.

In this experiment, we train ST-GAN with $N = 5$ sequential warps. We crop the aligned faces into 144×144 images and resize the glasses to widths of 120 pixels initialized at the center. During training, we add geometric data augmentation by randomly perturbing the faces with random similarity transformations and the glasses with random homographies.

Results. The results are shown in Fig. 4.11. As with the previous experiments, ST-GAN learns to warp the foreground glasses in a gradual fashion that improves upon realism at each step. Our method can correctly align glasses onto the people's faces, even with a certain amount of in-plane rotations. However, ST-GAN does a poorer job on faces with too much out-of-plane rotation.

While such an effect is possible to achieve by taking advantage of facial landmarks, our results are encouraging as no information was given about the structure of either domain, and we only had access to unpaired images of people with and without glasses. Nonetheless, ST-GAN was able to learn a realism manifold that drove the Spatial Transformer generators. We believe this demonstrates great potential to extend ST-GANs to other image alignment tasks where acquiring paired data is very challenging.

4.5 Conclusion

We have introduced ST-GANs as a class of methods to model *geometric realism*. We have demonstrated the potential of ST-GANs on the task of image compositing, showing improved realism in a large-scale rendered dataset, and results on fully unpaired real-world image data. It is our hope that this work will open up new revenues to the research community to continue to explore in this direction.

Despite the encouraging results ST-GAN achieves, there are still some limitations. We find that ST-GAN suffers more when presented imbalanced data, particularly rare examples (*e.g.* white, thick-framed glasses in the glasses experiment). In addition, we also find convergence of ST-GAN to fail with more extreme translation or in-plane rotation of objects. We believe a future analysis of the convergence properties of classical image alignment methods with GAN frameworks is

worthy of investigation in improving the robustness of ST-GANs.

4.A Appendix

4.A.1 Indoor Object Experiment: Rendering Details

We describe additional details regarding the rendering of the SUNCG dataset [165] for our experiment. In addition to Mitsuba [72] for rendering photo-realistic textures, we also utilize the OpenGL toolbox provided by Song *et al.* [165], which supports rendering of instance segmentation.

Candidate object selection. For each of the provided camera viewpoints from Zhang *et al.* [214], we render an instance segmentation of all objects visible in the camera viewpoint. For each of these objects, we also separately render a binary object mask by removing all other existing objects (including the floor/ceiling/walls).

We use these information to exclude objects that are not ideal for our compositing experiment, including those that are too tiny or only partially visible in the camera view. Therefore, we include objects into the candidate selection list that match the criteria:

- The entire object mask is visible within the camera.
- The object mask occupies at least 10% of all pixels.
- At least 50% of the object mask is visible within the instance segmentation mask.
- The object belongs to one of the NYUv2 [157] categories of refrigerators, desks, bookshelves, cabinets, beds, dressers, sofas, or chairs.

Occlusion removal. For all the objects in the candidate list, we remove the occluding objects (from the associated camera viewpoint) by overlapping the object mask onto the instance segmentation mask. All overlapped pixels with different instance labels are detected to be associated with an occluding object. Since there may be "hidden" occlusions that are occluded in the first place, we repeat the same process after the initial detected occlusions are removed to reveal the remaining occlusions. This is repeated until no more occluding objects with respect to the candidate object is present.

In order to create a cleaner space for compositing objects, we also use a "thicker" object mask for the above removal procedure. To achieve this, we dilate the object mask with a 3×3 all-ones kernel for 10 times (*i.e.* "thicken" the object mask by 10 pixels).

Camera perturbation. For each of the provided camera viewpoints, we generate a camera perturbation by adding a random 3D-translation sampled from Unif$(-1, 1)$ in the forward-backward

direction, one sampled from Unif$(-1, 1)$ in the left-right direction (both scaled in meters as defined in the dataset), and a random azimuth rotation sampled from Unif$(-30, 30)$ (degrees).

After generating a camera perturbation, the same occlusion removal process described above is performed to ensure the wholeness of the object from the perturbed perspective. The candidate object rendered from the perturbed view serves as the foreground source for our experiment. However, if it becomes only partially or not visible, then the rendering is discarded.

Rendering. We use Mitsuba to render 120×160 realistic textures and the OpenGL toolbox to render object masks at 240×320 followed by $\times 2$ downscaling for anti-aliasing.

4.A.2 Warp Parameterization Details

We follow Mei *et al.* [122] to parameterize homography with the $\mathfrak{sl}(3)$ Lie algebra. Given a warp parameter vector $\mathbf{p} = [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8]^\top \in \mathfrak{sl}(3)$, the transformation matrix $\mathbf{H} \in \mathbb{SL}(3)$ can be written as

$$\mathbf{H}(\mathbf{p}) = \exp\left(\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & -p_1 - p_8 & p_5 \\ p_6 & p_7 & p_8 \end{bmatrix}\right) , \tag{4.7}$$

where $\exp$ is the exponential map (*i.e.* matrix exponential). $\mathbf{H}$ is the identity transformation when $\mathbf{p}$ is an all-zeros vector. Warp composition can thus be expressed as the addition of parameters, *i.e.* $\mathbf{p}_a \circ \mathbf{p}_b \equiv \mathbf{p}_a + \mathbf{p}_b \;\; \forall \mathbf{p}_a, \mathbf{p}_b \in \mathfrak{sl}(3)$; furthermore, $\det(\mathbf{H}) = 1 \;\; \forall \mathbf{H} \in \mathbb{SL}(3)$.

The exponential map is also Taylor-expandable as

$$\mathbf{H}(\mathbf{p}) = \exp(\mathbf{X}(\mathbf{p})) = \lim_{K \to \infty} \sum_{k=0}^{K} \frac{\mathbf{X}^k(\mathbf{p})}{k!} . \tag{4.8}$$

We implement the $\mathfrak{sl}(3)$ parameterization using the Taylor approximation expression with $K = 20$.

4.A.3 Training Details

For all experiments, we set the batch size for all experiments to be 20. Unless otherwise specified, we initialize all learnable weights in the networks from $\mathcal{N}(0, 0.01)$ and all biases to be 0. All deep learning approaches are trained with Adam optimization [85]. We set $\lambda_{\text{grad}} = 10$ following Gulrajani *et al.* [53].

We describe settings for specific experiments as follows.

3D cubes. We create 4000 samples of 3D cube/room pairs with random colors, as described in the paper. For the initial warp $\mathbf{p}_0$, we generate random homography perturbations $\mathbf{p}_0$ by sampling each element of $\mathbf{p}_0$ from $\mathcal{N}(0, 0.1)$, *i.e.* $\mathbf{p}_0 \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I})$. This is applied to a canonical frame with x and y coordinates normalized to $[-1, 1]$ and subsequently transformed back to the image frame. We train ST-GAN with 4 sequential warps, each for 50K iterations (with perturbations generated on the fly) with the learning rates for both $\mathcal{G}$ and $\mathcal{D}$ to be 10^{-4}. We set $\lambda_{\text{update}} = 0.1$ in this experiment.

Indoor objects. For the self-supervised baselines (HomographyNet [32] and SDM [199]), we generate random homography perturbations $\mathbf{p}_0$ using the same noise model as that from the 3D cubes experiment.

We train HomographyNet for 200K iterations (with perturbations generated on the fly) with a learning rate of 10^{-4}. For SDM, we vectorize the grayscale images to be the feature as was practiced for image alignment [100]; in our case, we concatenate those of the background and masked foreground as the final extracted feature. We generate 750K perturbed examples (more than 10 perturbed examples per training sample) to train each linear regressor. Also as was practiced [100, 199], we add an ℓ_2 regularization term to the SDM least-squares objective function and search for the penalty factor by evaluating on a separate validation set.

We initialize each of the ST-GAN generators $\mathcal{G}_i$ with the pretrained HomographyNet as we find it to be better-conditioned. During adversarial training, we train each $\mathcal{G}_i$ for 40K iterations with the learning rate for $\mathcal{G}_i$ to be 10^{-6} and that of $\mathcal{D}$ to be 10^{-4}. In the final end-to-end fine-tuning stage, we train all $\mathcal{G}_i$ for 40K iterations using the same learning rates (10^{-6} for all $\mathcal{G}_i$ and 10^{-4} for $\mathcal{D}$). The non-sequential ST-GAN baseline is trained for 160K iterations with the same learning rates. We set $\lambda_{\text{update}} = 0.3$ in this experiment.

Glasses. For data augmentation, we perturb the faces with random similarity transformations from $\mathcal{N}(0, 0.1)$ for rotation (radian) and $\mathcal{N}(0, 0.05)$ for translation (scaled by the image dimensions, in both x and y directions). The glasses are perturbed using the same random homography noise model as used in the 3D cubes experiment.

We train ST-GAN with 5 sequential warps, each for 50K iterations with the learning rates for both $\mathcal{G}$ and $\mathcal{D}$ to be 10^{-5}. As a preconditioning step, we also pretrain the discriminator $\mathcal{D}$ using only the initial fake samples and real samples for 50K iterations with the same learning rate. We set $\lambda_{\text{update}} = 1$ in this experiment.

4.A.4 Additional Indoor Object Results

We include additional qualitative results from the indoor object experiment in Fig. 4.12. Compared to the baselines, ST-GAN consistently predicts more realistic geometric corrections in most cases.

4.A.5 Additional Glasses Results

We also include additional qualitative results from the glasses experiment in Fig. 4.13. We re-emphasize that the training data here is unpaired and there is no information in the dataset about where the glasses are placed. Despite these, ST-GAN is able to consistently match the initial glasses foreground to the background faces.

(a)

(b)

Figure 4.11: **Glasses compositing results.** (a) The glasses progressively moves into a more realistic position. (b) ST-GAN learns to warp various kinds of glasses such that the resulting positions are usually realistic. The top rows indicates the initial composite, and the bottom rows indicates the ST-GAN output. The last 4 examples shows failure cases, where glasses fail to converge onto the faces.

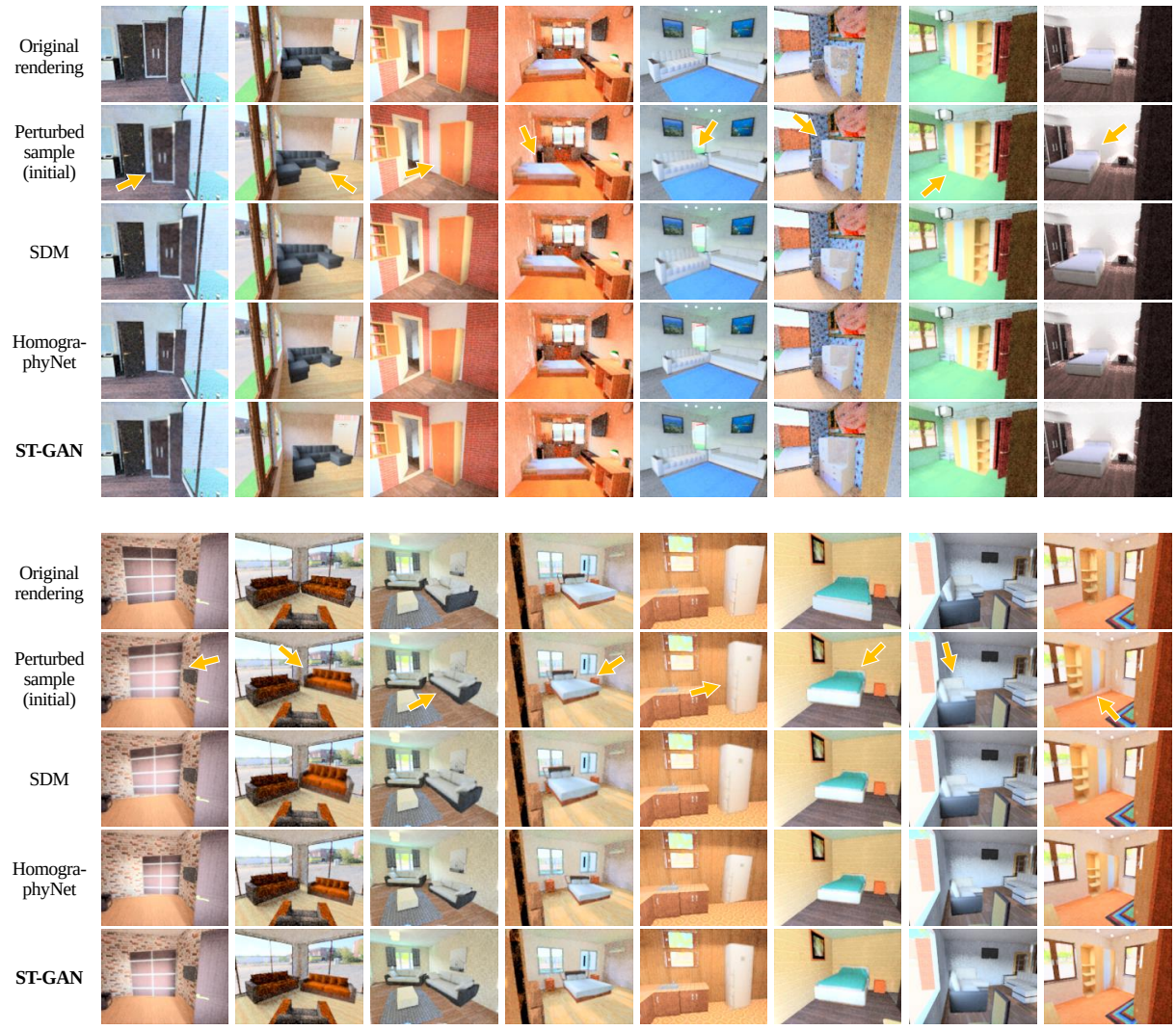


Figure 4.12: Additional qualitative results from the indoor object experiment (test set). The yellow arrows in the second row point to the composited foreground objects.

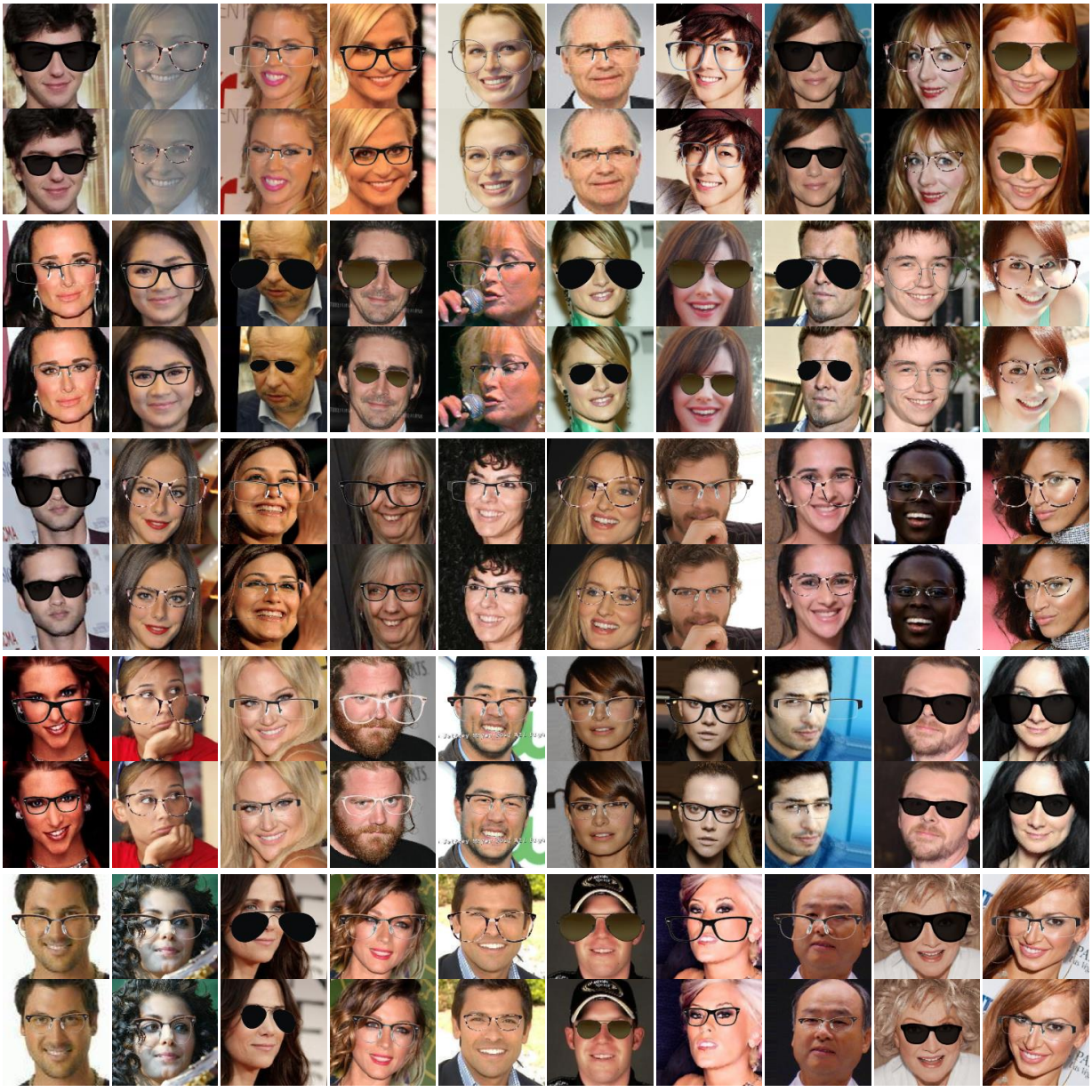


Figure 4.13: Additional qualitative results from the glasses experiment (test set). The top row indicates the initial composite, and the bottom row indicates the ST-GAN output.

Part II

Learning Dense 3D Reconstruction

Chapter 5

Multi-view Supervision from Depth Maps

5.1 Introduction

Generative models using convolutional neural networks (ConvNets) have achieved state of the art in image/object generation problems. Notable works of the class include variational autoencoders [86] and generative adversarial networks [48], both of which have drawn large success in various applications [70, 142, 188, 202, 220]. With the recent introduction of large publicly available 3D model repositories [16, 195], the study of generative modeling on 3D data using similar frameworks has also become of increasing interest.

In computer vision and graphics, 3D object models can take on various forms of representations. Of such, triangular meshes and point clouds are popular for their vectorized (and thus scalable) data representations as well as their compact encoding of shape information, optionally embedded with texture. However, this efficient representation comes with an inherent drawback as the dimensionality per 3D shape sample can vary, making the application of learning methods problematic. Furthermore, such data representations do not elegantly fit within conventional ConvNets as Euclidean convolutional operations cannot be directly applied. Hitherto, most existing works on 3D model generation resort to volumetric representations, allowing 3D Euclidean convolution to operate on regular discretized voxel grids. 3D ConvNets (as opposed to the classical 2D form) have been applied successfully to 3D volumetric representations for both discriminative [62, 121, 195] and generative [21, 47, 193, 203] problems.

Despite their recent success, 3D ConvNets suffer from an inherent drawback when modeling shapes with volumetric representations. Unlike 2D images, where every pixel contains meaningful spatial and texture information, volumetric representations are information-sparse. More specifically, a 3D object is expressed as a voxel-wise occupancy grid, where voxels "outside" the object (set to off) and "inside" the object (set to on) are unimportant and fundamentally not of particular interest. In other words, the richest information of shape representations lies on the surface of the 3D object, which makes up only a slight fraction of all voxels in an occupancy grid. Consequently, 3D ConvNets are extremely wasteful in both computation and memory in

trying to predict much unuseful data with high-complexity 3D convolutions, severely limiting the granularity of the 3D volumetric shapes that can be modeled even on high-end GPU-nodes commonly used in deep learning research.

In this chapter, we propose an efficient framework to represent and generate 3D object shapes with dense point clouds. We achieve this by learning to predict the 3D structures from multiple viewpoints, which is jointly optimized through 3D geometric reasoning. In contrast to prior art that adopts 3D ConvNets to operate on volumetric data, we leverage *2D convolutional operations* to predict points clouds that shape the *surface* of the 3D objects. Our experimental results show that we generate much denser and more accurate shapes than state-of-the-art 3D prediction methods.

Our contributions are summarized as follows:

- We advocate that 2D ConvNets are capable of generating dense point clouds that shapes the surface of 3D objects in an undiscretized 3D space.
- We introduce a pseudo-rendering pipeline to serve as a differentiable approximation of true rendering. We further utilize the pseudo-rendered depth images for 2D projection optimization for learning dense 3D shapes.
- We demonstrate the efficacy of our method on single-image 3D reconstruction problems, which significantly outperforms state-of-the-art methods.

5.2 Related Work

3D shape generation. As 2D ConvNets have demonstrated huge success on a myriad of image generation problems, most works on 3D shape generation follow the analogue using 3D ConvNets to generate volumetric shapes. Prior works include using 3D autoencoders [47] and recurrent networks [21] to learn a latent representation for volumetric data generation. Similar applications include the use of an additional encoded pose embedding to learn shape deformations [211] and using adversarial training to learn more realistic shape generation [44, 193]. Learning volumetric predictions from 2D projected observations has also been explored [44, 148, 203], which use 3D differentiable sampling on voxel grids for spatial transformations [71]. Constraining the ray consistency of 2D observations have also been suggested very recently [177].

Most of the above approaches utilize 3D convolutional operations, which is computationally expensive and allows only coarse 3D voxel resolution. The lack of granularity from such volumetric generation has been an open problem following these works. Riegler *et al.* [151] proposed to tackle the problem by using adaptive hierarchical octary trees on voxel grids to encourage encoding more informative parts of 3D shapes. Concurrent works follow to use similar concepts [56, 173] to predict 3D volumetric data with higher granularity.

Recently, Fan *et al.* [40] also sought to generate unordered point clouds by using variants of multi-layer perceptrons to predict multiple 3D coordinates. However, the required learnable parameters linearly proportional to the number of 3D point predictions and does not scale well;

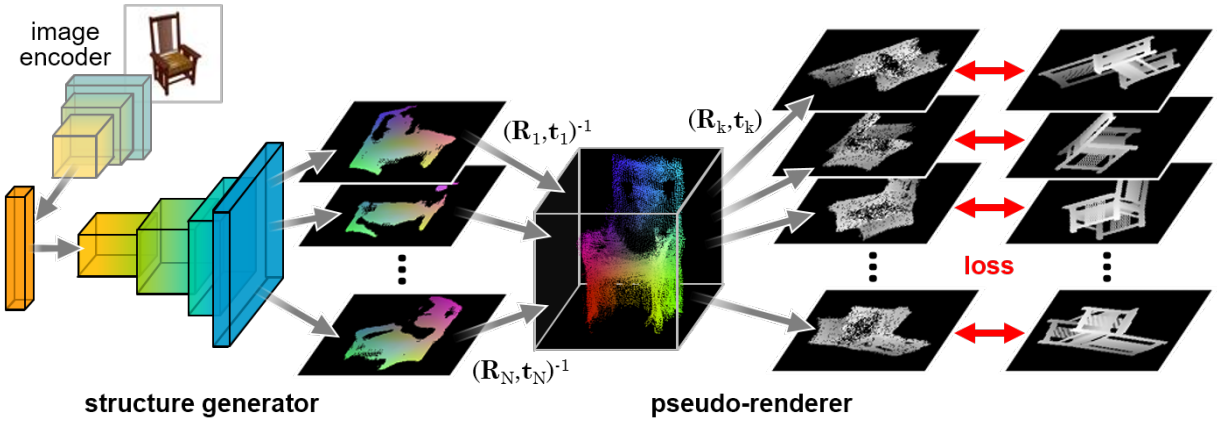


Figure 5.1: **Network architecture.** From an encoded latent representation, we propose to use a structure generator (Sec 5.3.1), which is based on *2D convolutional* operations, to predict the 3D structure at N viewpoints. The point clouds are fused by transforming the 3D structure at each viewpoint to the canonical coordinates. The pseudo-renderer (Sec. 5.3.2) synthesizes depth images from novel viewpoints, which are further used for joint 2D projection optimization. This contains no learnable parameters and reasons based purely on 3D geometry. s

in addition, using 3D distance metrics as optimization criteria is intractable for large number of points. In contrast, we leverage convolutional operations with a joint 2D project criterion to capture the correlation between generated point clouds and optimize in a more computationally tractable fashion.

3D view synthesis. Research has also been done in learning to synthesize novel 3D views of 2D objects in images. Most approaches using ConvNets follow the convention of an encoder-decoder framework. This has been explored by mixing 3D pose information into the latent embedding vector for the synthesis decoder [136, 172, 216]. A portion of these works also discussed the problem of disentangling the 3D pose representation from object identity information [91, 145, 206], allowing further control on the identity representation space.

The drawback of these approaches is their inefficiency in representing 3D geometry — as we later show in the experiments, one should explicitly factorize the underlying 3D geometry instead of implicitly encoding it into mixed representations. Resolving the geometry has been proven more efficient than tolerating in several works (*e.g.* Spatial Transformer Networks [71, 99]).

5.3 Approach

Our goal is to generate 3D predictions that compactly shape the surface geometry with dense point clouds. The overall pipeline is illustrated in Fig. 5.1. We start with an encoder that maps the input data to a latent representation space. The encoder may take on various forms of data depending

on the application; in our experiments, we focus on encoding RGB images for single-image 3D reconstruction tasks. From the latent representation, we propose to generate the dense point clouds using a structure generator based on 2D convolutions with a joint 2D projection criterion, described in detail as follows.

5.3.1 Structure Generator

The structure generator predicts the 3D structure of the object at N different viewpoints (along with their binary masks), *i.e.* the 3D coordinates $\hat{\mathbf{x}}_i = [\hat{x}_i \ \hat{y}_i \ \hat{z}_i]^\top$ at each pixel location. Pixel values in natural images can be synthesized through convolutional generative models mainly due to their exhibition of strong local spatial dependencies; similar phenomenons can be observed for point clouds when treating them as (x, y, z) multi-channel images on a 2D grid. Based on this insight, the structure generator is mainly based on 2D convolutional operations to predict the (x, y, z) images representing the 3D surface geometry. This approach circumvents the need of time-consuming and memory-expensive 3D convolutional operations for volumetric predictions. The evidence of such validity is verified in our experimental results.

Assuming the 3D rigid transforma matrices of the N viewpoints $(\mathbf{R}_1, \mathbf{t}_1)...(\mathbf{R}_N, \mathbf{t}_N)$ are given, each 3D point $\hat{\mathbf{x}}_i$ at viewpoint n can be transformed to the canonical 3D coordinates as $\hat{\mathbf{p}}_i$ via

$$\hat{\mathbf{p}}_i = \mathbf{R}_n^{-1} \left(\mathbf{K}^{-1} \hat{\mathbf{x}}_i - \mathbf{t}_n \right) \quad \forall i \ , \tag{5.1}$$

where $\mathbf{K}$ is the predefined camera intrinsic matrix. This defines the relationship between the predicted 3D points and the fused collection of point clouds in the canonical 3D coordinates, which is the outcome of our network.

5.3.2 Joint 2D Projection Optimization

To learn point cloud generation using the provided 3D CAD models as supervision, the standard approach would be to optimize over a 3D-based metric that defines the distance between the point cloud and the ground-truth CAD model (*e.g.* Chamfer distance [40]). Such metric usually involves computing surface projections for every generated point, which can be computationally expensive for very dense predictions, making it intractable.

We overcome this issue by alternatively optimizing over the *joint 2D projection error* of novel viewpoints. Instead of using only projected binary masks as supervision [44, 148, 203], a well-generated 3D shape should also have the ability to render reasonable depth images from any viewpoint. To realize this concept, we introduce the pseudo-renderer, a differentiable module to approximate true rendering, to synthesize novel depth images from dense point clouds.

Pseudo-rendering. Given the 3D rigid transformation matrix of a novel viewpoint $(\mathbf{R}_k, \mathbf{t}_k)$, each canonical 3D point $\hat{\mathbf{p}}_i$ can be further transformed to $\hat{\mathbf{x}}'_i$ back in the image coordinates via

$$\hat{\mathbf{x}}'_i = \mathbf{K} \left(\mathbf{R}_k \hat{\mathbf{p}}_i + \mathbf{t}_k \right) ; ; \forall i \ . \tag{5.2}$$

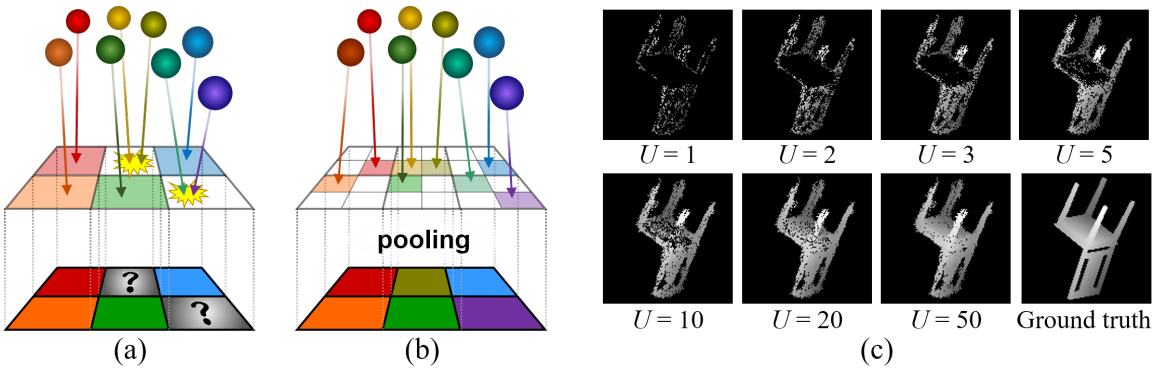


Figure 5.2: **Concept of pseudo-rendering.** Multiple transformed 3D points may correspond to projection on the same pixels in the image space. (a) Collision could easily occur if $(\hat{x}'_i, \hat{y}'_i)$ were directly discretized. (b) Upsampling the target image increases the precision of the projection locations and thus alleviates the collision effect. A max-pooling operation on the inverse depth values follows as to obtain the original resolution while maintaining the effective depth value at each pixel. (c) Examples of pseudo-rendered depth images with various upsampling factors U (only valid depth values without collision are shown). Pseudo-rendering achieves closer performance to true rendering with a higher value of U.

This is the inverse operation of Eq. (5.1) with different transformation matrices and can be combined with Eq. (5.1) together, composing a single effective transformation. By such, we obtain the $(\hat{x}'_i, \hat{y}'_i)$ location as well as the new depth value $\hat{z}'_i$ at viewpoint k.

To produce a pixelated depth image, one would also need to discretize all $(\hat{x}'_i, \hat{y}'_i)$ coordinates, resulting in possibly multiple transformed points projecting and "colliding" onto the same pixel (Fig. 5.2). We resolve this issue with the pseudo-renderer $f_{\text{PR}}(\cdot)$, which increases the projection resolution to alleviate such collision effect. Specifically, $\hat{\mathbf{x}}'_i$ is projected onto a target image upsampled by a factor of U, reducing the quantization error of $(\hat{x}'_i, \hat{y}'_i)$ as well as the probability of collision occurrence. A max-pooling operation on the inverse depth values with kernel size U follows to downsample back to the original resolution while maintaining the minimum depth value at each pixel location. We use such approximation of the rendering operation to maintain differentiability and parallelizability within the backpropagation framework.

Optimization. We use the pseudo-rendered depth images $\hat{\mathbf{Z}} = f_{\text{PR}}(\{\hat{\mathbf{x}}'_i\})$ and the resulting masks $\hat{\mathbf{M}}$ at novel viewpoints for optimization. The loss function consists of the mask loss $\mathcal{L}_{\text{mask}}$ and the depth loss $\mathcal{L}_{\text{depth}}$, respectively defined as

$$\mathcal{L}_{\text{mask}} = \sum_{k=1}^{K} -\mathbf{M}_k \log \hat{\mathbf{M}}_k - (1 - \mathbf{M}_k) \log \left(1 - \hat{\mathbf{M}}_k\right) \quad \text{and} \quad \mathcal{L}_{\text{depth}} = \sum_{k=1}^{K} \left\| \hat{\mathbf{Z}}_k - \mathbf{Z}_k \right\|_1 , \tag{5.3}$$

where we simultaneously optimize over K novel viewpoints at a time. $\mathbf{M}_k$ and $\mathbf{Z}_k$ are the ground-truth mask and depth images at the kth novel viewpoint. We use element-wise L_1 loss for the

Sec.	Input size	Latent vector	Number of filters	
			image encoder	structure generator
5.4.1	64×64	512-D	conv: 96, 128, 192, 256 linear: 2048, 1024, 512	linear: 1024, 2048, 4096 deconv: 192, 128, 96, 64, 48
5.4.2	128×128	1024-D	conv: 128, 192, 256, 384, 512 linear: 4096, 2048, 1024	linear: 2048, 4096, 12800 deconv: 384, 256, 192, 128, 96

Table 5.1: **Architectural details** of our proposed method for each experiment.

depth (posing it as a pixel-wise binary classification problem) and cross-entropy loss for the mask. The overall loss function is defined as $\mathcal{L} = \mathcal{L}_{\text{mask}} + \lambda \cdot \mathcal{L}_{\text{depth}}$, where λ is the weighting factor.

Optimizing the structure generator over novel projections enforces joint 3D geometric reasoning between the predicted point clouds from the N viewpoints. It also allows the optimization error to evenly distribute across novel viewpoints instead of focusing on the fixed N viewpoints.

5.4 Experiments

We evaluate our proposed method by analyzing its performance in the application of single-image 3D reconstruction and comparing against state-of-the-art methods.

Data preparation. We train and evaluate all networks using the ShapeNet database [16], which contains a large collection of categorized 3D CAD models. For each CAD model, we pre-render 100 depth/mask image pairs of size 128×128 at random novel viewpoints as the ground truth of the loss function. We consider the entire space of possible 3D rotations (including in-plane rotation) for the viewpoints and assume identity translation for simplicity. The input images are objects pre-rendered from a fixed elevation and 24 different azimuth angles.

Architectural details. The structure generator follows the structure of conventional deep generative models, consisting of linear layers followed by 2D convolution layers (with kernel size 3×3). The dimensions of all feature maps are halved after each encoder convolution and doubled after each generator convolution. Details of network dimensions are listed in Table 5.1. At the end of the decoder, we add an extra convolution layer with filters of size 1×1 to encourage individuality of the generated pixels. Batch normalization [69] and ReLU are added between all layers.

The generator predicts $N = 8$ images of size 128×128 with 4 channels (x, y, z and the binary mask), where the fixed viewpoints are chosen from the 8 corners of a centered cube. Orthographic projection is assumed in the transformation in (5.1) and (5.2). We use $U = 5$ for the upsampling factor of the pseudo-renderer in our experiments.

Training details. All networks are optimized using the Adam optimizer [85]. We take a two-stage training procedure: the structure generator is first pretrained to predict the depth images from the N viewpoints (with a constant learning rate of 1e-2), and then the entire network is fine-tuned with joint 2D projection optimization (with a constant learning rate of 1e-4). For the training parameters, we set $\lambda = 1.0$ and $K = 5$.

Quantitative metrics. We measure using the average point-wise 3D Euclidean distance between two 3D models: for each point $\hat{\mathbf{p}}_i$ in the source model, the distance to the target model $\mathcal{S}$ is defined as $\mathcal{E}_i = \min_{\mathbf{p}_j \in \mathcal{S}} \|\hat{\mathbf{p}}_i - \mathbf{p}_j\|_2$. This metric is defined bidirectionally as the distance from the predicted point cloud to the ground-truth CAD model and vice versa. It is necessary to report both metrics for they represent different aspects of quality — the former measures 3D shape similarity and the latter measures surface coverage [88]. We represent the ground-truth CAD models as collections of uniformly densified 3D points on the surfaces (100K densified points in our settings).

5.4.1 Single Object Category

We start by evaluating the efficacy of our dense point cloud representation on 3D reconstruction for a single object category. We use the chair category from ShapeNet, which consists of 6,778 CAD models. We compare against (a) Tatarchenko *et al.* [172], which learns implicit 3D representations through a mixed embedding, and (b) Perspective Transformer Networks (PTN) [203], which learns to predict volumetric data by minimizing the projection error. We include two variants of PTN as well as a baseline 3D ConvNet from Yan *et al.* [203]. We use the same 80%-20% training/test split provided by Yan *et al.* [203].

We pretrain our network for 200K iterations and fine-tune end-to-end for 100K iterations. For the method of Tatarchenko *et al.* [172], we evaluate by predicting depth images from our same N viewpoints and transform the resulting point clouds to the canonical coordinates. This shares the same network architecture to ours, but with 3D pose information additionally encoded using 3 linear layers (with 64 filters) and concatenated with the latent vector. We use the novel depth/mask pairs as direct supervision for the decoder output and train this network for 300K iterations with a constant learning rate of 1e-2. For PTN [203], we extract the surface voxels (by subtracting the prediction by its eroded version) and rescale them such that the tightest 3D bounding boxes of the prediction and the ground-truth CAD models have the same volume. We use the pretrained models readily provided by the authors.

The quantitative results on the test split are reported in Table 5.2. We achieve a lower average 3D distance than all baselines in both metrics, even though our approach is optimized with joint 2D projections instead of these 3D error metrics. This demonstrates that we are capable of predicting more accurate shapes with higher density and finer granularity. This highlights the efficiency of our approach using 2D ConvNets to generate 3D shapes compared to 3D ConvNet methods such as PTN [203] as they attempt to predict all voxel occupancies inside a 3D grid space. Compared to Tatarchenko *et al.* [172], an important takeaway is that 3D geometry should explicitly

Method	3D error metric	
	pred. $\rightarrow$ GT	GT $\rightarrow$ pred.
3D ConvNet (vol. only) [203]	1.827	2.660
PTN (proj. only) [203]	2.181	2.170
PTN (vol. & proj.) [203]	1.840	2.585
Tatarchenko *et al.* [172]	2.381	3.019
Proposed method	**1.768**	**1.763**

Table 5.2: **Average 3D test error** of the single-category experiment. Our method outperforms all baselines in both metrics, indicating the superiority in fine-grained shape similarity and point cloud coverage on the surface. (All numbers are scaled by 0.01)

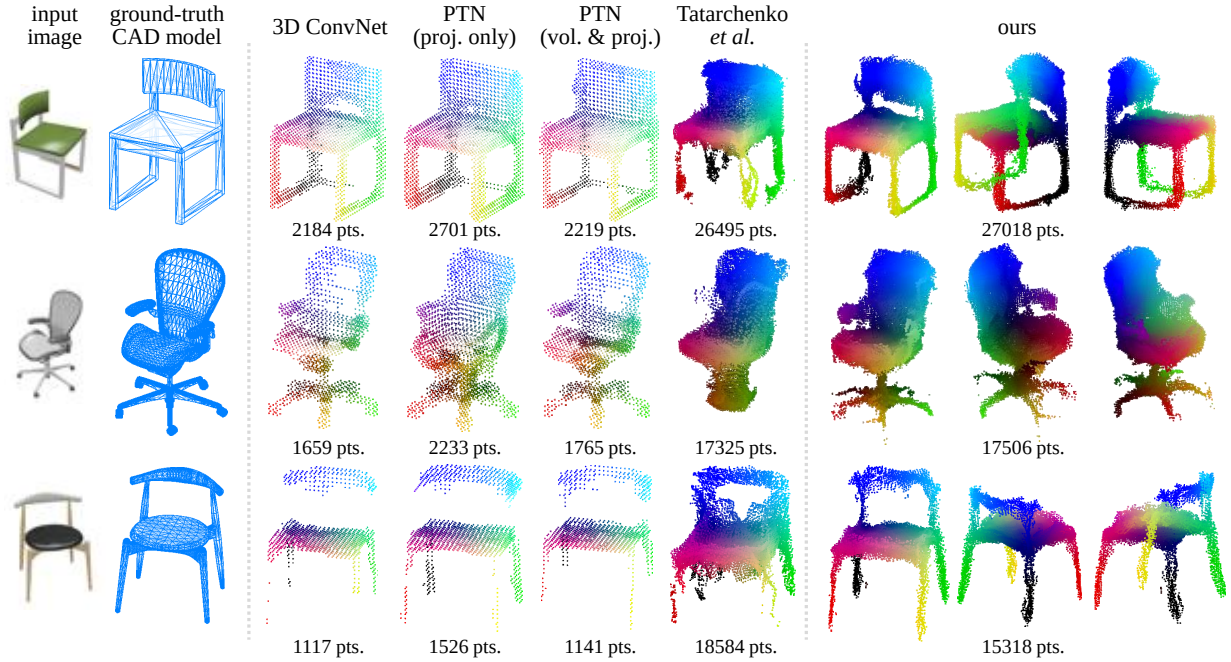


Figure 5.3: **Qualitative results** from the single-category experiment. Our method generates denser predictions compared to the volumetric baselines and more accurate shapes than Tatarchenko *et al.* [172], which learns 3D synthesis implicitly. The RGB values of the point cloud represents the 3D coordinate values. Best viewed in color.

factorized when possible instead of being implicitly learned by the network parameters. It is much more efficient to focus on predicting the geometry from a sufficient number of viewpoints and combining them with known geometric transformations.

We visualize the generated 3D shapes in Fig. 5.3. Compared to the baselines, we predict more accurate object structures with a much higher point cloud density (around $10\times$ higher than 32^3 volumetric methods). This further highlights the desirability of our approach — we are able to efficiently use 2D convolutional operations and utilize high-resolution supervision given similar memory budgets.

Category	3D-R2N2 [21]			Fan *et al.* [40]	Proposed
	1 view	3 views	5 views	(1 view)	(1 view)
airplane	3.207 / 2.879	2.521 / 2.468	2.399 / 2.391	1.301 / **1.488**	**1.294** / 1.541
bench	3.350 / 3.697	2.465 / 2.746	2.323 / 2.603	1.814 / 1.983	**1.757** / **1.487**
cabinet	1.636 / 2.817	1.445 / 2.626	**1.420** / 2.619	2.463 / 2.444	1.814 / **1.072**
car	1.808 / 3.238	1.685 / 3.151	1.664 / 3.146	1.800 / 2.053	**1.446** / **1.061**
chair	2.759 / 4.207	1.960 / 3.238	**1.854** / 3.080	1.887 / 2.355	1.886 / **2.041**
display	3.235 / 4.283	2.262 / 3.151	2.088 / 2.953	**1.919** / 2.334	2.142 / **1.440**
lamp	8.400 / 9.722	6.001 / 7.755	5.698 / 7.331	**2.347** / **2.212**	2.635 / 4.459
loudspeaker	2.652 / 4.335	2.577 / 4.302	2.487 / 4.203	3.215 / 2.788	**2.371** / **1.706**
rifle	4.798 / 2.996	4.307 / 2.546	4.193 / 2.447	1.316 / **1.358**	**1.289** / 1.510
sofa	2.725 / 3.628	2.371 / 3.252	2.306 / 3.196	2.592 / 2.784	**1.917** / **1.423**
table	3.118 / 4.208	2.268 / 3.277	2.128 / 3.134	1.874 / 2.229	**1.689** / **1.620**
telephone	2.202 / 3.314	1.969 / 2.834	1.874 / 2.734	**1.516** / 1.989	1.939 / **1.198**
watercraft	3.592 / 4.007	3.299 / 3.698	3.210 / 3.614	**1.715** / 1.877	1.813 / **1.550**
mean	3.345 / 4.102	2.702 / 3.465	2.588 / 3.342	1.982 / 2.146	**1.846** / **1.701**

Table 5.3: **Average 3D test error** of the multi-category experiment, where the numbers are shown as [prediction→GT / GT→prediction]. The mean is computed across categories. For the single-view case, we outperform all baselines in 8 and 10 out of 13 categories for the two 3D error metrics. (All numbers are scaled by 0.01)

5.4.2 General Object Categories

We also evaluate our network on the single-image 3D reconstruction task trained with multiple object categories. We compare against (a) 3D-R2N2 [21], which learns volumeric predictions through recurrent networks, and (b) Fan *et al.* [40], which predicts an unordered set of 1024 3D points. We use 13 categories of ShapeNet for evaluation (listed in Table 5.3), where the 80%-20% training/test split is provided by Choy *et al.* [21]. We evaluate 3D-R2N2 by its surface voxels using the same procedure as described in Sec. 5.4.1. We pretrain our network for 300K iterations and fine-tune end-to-end for 100K iterations; for the baselines, we use the pretrained models readily provided by the authors.

We list the quantitative results in Table 5.3, where the metrics are reported per-category. Our method achieves an overall lower error in both metrics. We outperform the volumetric baselines (3D-R2N2) by a large margin and has better prediction performance than Fan *et al.* in most cases. We also visualize the predictions in Fig. 5.4; again we see that our method predicts more accurate shapes with higher point density. Our method can be more problematic when objects contain very thin structures (*e.g.* lamps); adding hybrid linear layers [40] may help improve performance.

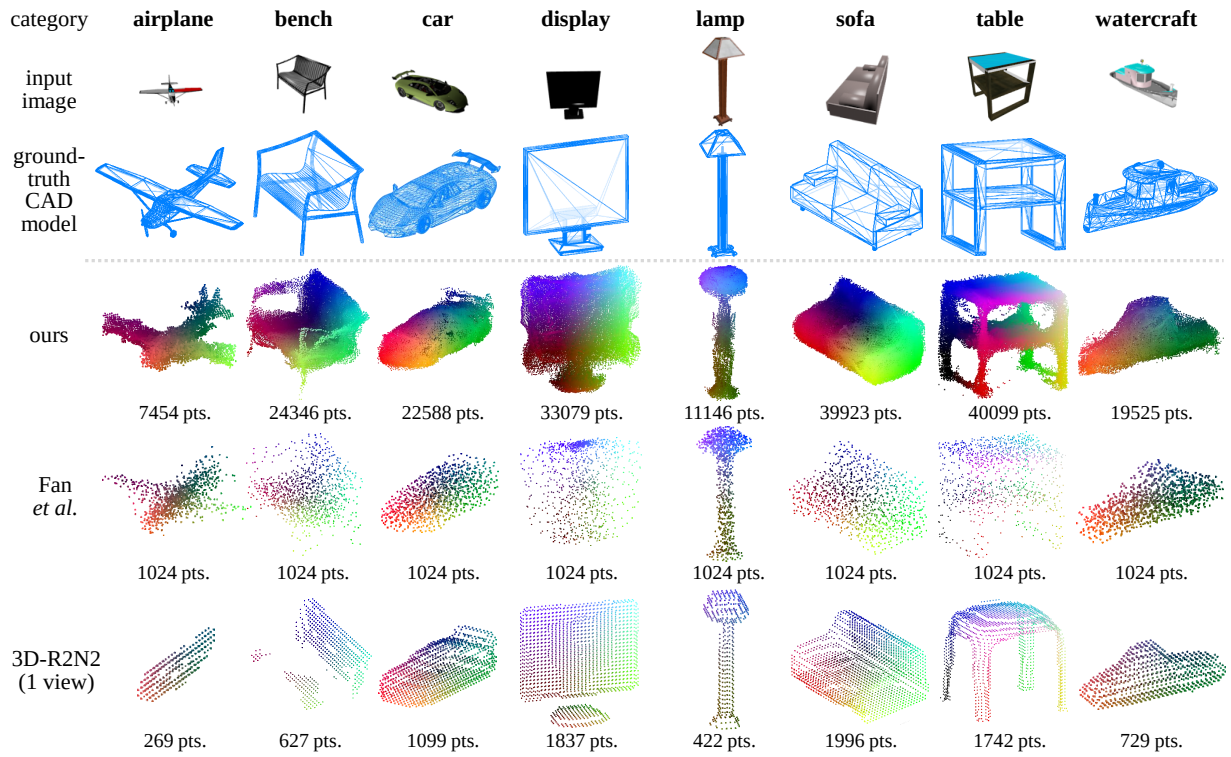


Figure 5.4: **Qualitative results** from the multi-category experiment. Our method generates denser and more certain predictions compared to the baselines.

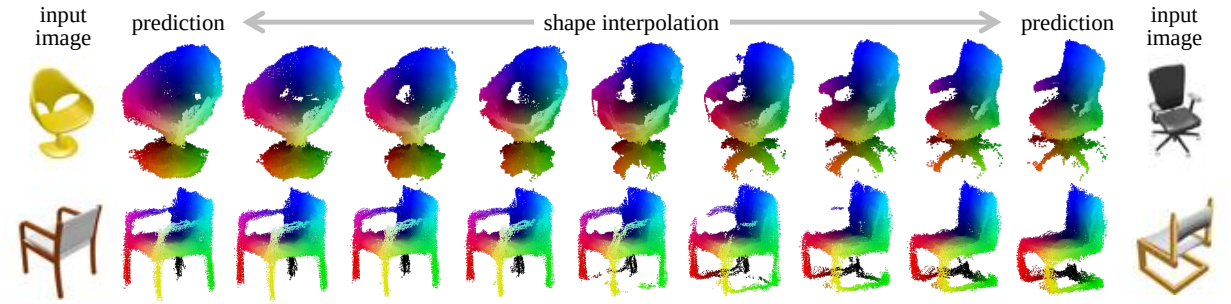


Figure 5.5: Dense shapes generated from interpolated latent embeddings of two input images (leftmost and rightmost). The interpolated shapes maintain reasonable structures of chairs.

5.4.3 Generative Representation Analysis

We analyze the learned generative representations by observing the 3D predictions from manipulation in the latent space. Previous works have demonstrated that deep generative networks can generate meaningful pixel/voxel predictions by performing linear operations in the latent space [34, 142, 193]; here, we explore the possibility of such manipulation for dense point clouds in an undiscretized space.

We show in Fig. 5.5 the resulting dense shapes generated from the embedding vector interpo-

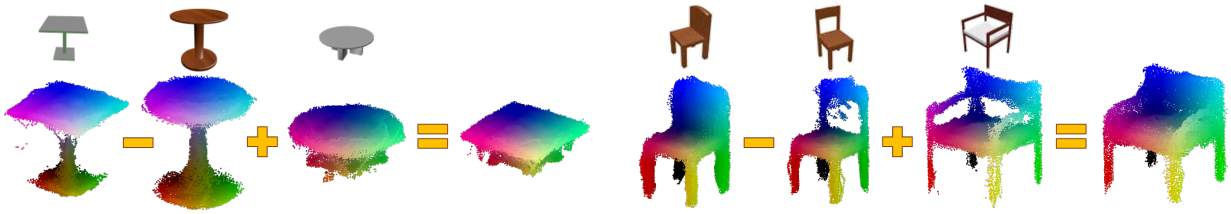


Figure 5.6: Dense shapes generated from arithmetic operations in the latent space (left: tables, right: chairs), where the input images are shown in the top row.

Figure 5.7: Comparison on the effects of the joint 2D optimization step (left figure: before, right figure: after). Optimizing only on the fixed viewpoints results in a denser point cloud but also with higher noise.

Joint 2D opt.	3D error metric	Predicted points
Before	1.933 / 1.307	31,972
After	1.768 / 1.763	25,401

Table 5.4: Optimizing on the novel viewpoints greatly reduces the noise while trading off partial surface coverage density. (The errors are shown as [prediction→GT / GT→prediction] and scaled by 100)

lated in the latent space. The morphing transition is smooth with plausible interpolated shapes, which suggests that our structure generator can generate meaningful 3D predictions from convex combinations of encoded latent vectors. The structure generator is also capable of generating reasonable novel shapes from arithmetic results in the latent space — from Fig. 5.6) we observe semantic feature replacement of table height/shape as well as chair arms/backs. These results suggest that the high-level semantic information encoded in the latent vectors are manipulable and interpretable of the resulting dense point clouds through the structure generator.

5.4.4 Ablative Analysis

We provide ablation studies on two of the key components of our proposed method: (1) the joint 2D optimization step and (2) the variability of the x, y coordinates of the structure generator output. We focus this analysis on the single-category experiment.

Joint 2D optimization. We validate the necessity of the second training stage of optimizing the network with supervision from novel viewpoints. We compare the performance of our network before and after the joint 2D projection optimization step in Table 3. We see that while optimizing

Output of structure generator	3D error metric	
	pred. $\rightarrow$ GT	GT $\rightarrow$ pred.
Depth image (z) only	1.764	2.086
xyz-channel images	1.768	1.763

Table 5.5: Comparison on the variability of the x, y coordinates of the multi-view output. Allowing the x, y coordinates to vary improves surface coverage. (All numbers are scaled by 100)

only on the fixed viewpoints results in more generated points closer to the ground-truth surface, it is also creates a considerable amount of noisy points in loss of shape accuracy. Fig. 5.7 visualizes the effect of joint optimization to eliminate most of the noisy points, demonstrating the necessity of such additional step.

Variability of x, y coordinates. Instead of having the structure generator to predict the x, y, z coordinates and the binary masks, one could alternatively design it to predict only the masked depth image (*i.e.* the z coordinates and the mask) and fix the x, y coordinates to the image regular grids. We show the difference in performance in Table 5.5. We see that enabling the x, y coordinates to vary not only leads to similar accuracy in shape prediction, but also allows higher surface coverage. There is also little increase in the number of learnable parameters from doubling the output channels in the final convolution layer.

5.5 Conclusion

In this chapter, we introduced a framework for generating 3D shapes in the form of dense point clouds. Compared to conventional volumetric prediction methods using 3D ConvNets, it is more efficient to utilize 2D convolutional operations to predict surface information of 3D shapes. We showed that by introducing a pseudo-renderer, we are able to synthesize approximate depth images from novel viewpoints to optimize the 2D projection error within a backpropagation framework. Experimental results for single-image 3D reconstruction tasks showed that we generate more accurate and much denser 3D shapes than state-of-the-art 3D reconstruction methods.

Chapter 6

Single-View Training from Static Image Collections

6.1 Introduction

Humans have strong capabilities to reason about 3D geometry in our visual world. When we see an object, not only can we infer its shape and appearance, but we can also speculate the underlying 3D structure. We learn to develop the concepts of 3D geometry and semantic priors, as well as the ability to mentally reconstruct the 3D world. Somehow through visual perception, *i.e.* just looking at a collection of 2D images, we have the ability to infer the 3D geometry of the objects in those images.

Researchers have sought to emulate such ability of 3D shape recovery from a single 2D image for AI systems, where success has been drawn specifically through neural networks. Although one could train such networks naively from images with associated ground-truth 3D shapes, such paired data are difficult to come by at scale. While most works have resorted to 3D object datasets, in which case synthetic image data can be created pain-free through rendering engines, the domain gap between synthetic and real images has prevented them from practical use. An abundant source of supervision that can be practically obtained for real-world image data is one problem that looms large in the field.

In the quest to eliminate the need for direct 3D supervision, recent research have attempted to tackle the problem of learning 3D shape recovery from 2D images with object silhouettes, which are easier to annotate in practice. This line of works seeks to maximize the reprojection consistency of 3D shape predictions to an ensemble of training images. While success has been shown on volumetric [177] and mesh-based [83, 109] reconstruction, such discretized 3D representations have drawbacks. Voxels are inefficient for representing shape surfaces as they are sparse by nature, while meshes are limited to deforming from fixed templates as learning adaptive mesh topologies is a nontrivial problem. Implicit shape representations become a more desirable choice for overcoming these limitations.

Figure 6.1: **Learning 3D SDF shape reconstruction from static images.** SDF-SRN learns implicit shape reconstruction from single-view images and 2D silhouettes at *training* time, allowing practical applications of real-world 3D object reconstruction trained from static image datasets.

Differentiable rendering methods for reconstructing 3D implicit representations have since sparked wide interest [108, 131]. Previous works, however, have required a multi-view setup, where objects are observed from multiple viewpoints *with* silhouette annotations. Since such data is difficult to obtain en masse, it has been unclear to the community how one can learn dense 3D reconstruction from single images at *training* time, where each individual object instance is assumed observed only once.

In this chapter, we make significant advances on learning dense 3D object reconstruction from single images and silhouettes, *without* the knowledge of the underlying shape structure or topology. To this end, we derive a formulation to learn signed distance functions (SDF) as the implicit 3D representation from images, where we take advantage of distance transform on silhouettes to provide rich geometric supervision from all pixels of an image. In addition, we build a differentiable rendering framework upon the recently proposed Scene Representation Network [161] for efficient optimization of shape surfaces. The proposed method, SDF-SRN, achieves state-of-the-art 3D object reconstruction results on challenging scenarios that requires only a single observation for each instance during *training* time. SDF-SRN also learns high-quality 3D reconstruction from real-world static images with *single-view* supervision (Fig. 6.1), which was not possible with previous implicit shape reconstruction methods.

In summary, we present the following contributions:

- We establish a novel mathematical formulation to optimize 3D SDF representations from 2D distance transform maps for learning dense 3D object reconstruction without 3D supervision.
- We propose an extended differentiable rendering algorithm that efficiently optimizes for the 3D shape surfaces from RGB images, which we show to be suitable only for SDF representations.
- Our method, SDF-SRN, significantly outperforms state-of-the-art 3D reconstruction methods on ShapeNet [16] trained with *single*-view supervision, as well as natural images from the PASCAL3D+ [197] dataset *without* 3D supervision nor externally pretrained 2.5D/3D priors.

6.2 Related Work

Learning 3D reconstruction without 3D supervision. The classical shape-from-silhouette problem [82, 93] can be posed as a learning problem by supervising the visual hull with a collection of silhouette images using a differentiable projection module. The reconstructed visual hull can take the forms of voxels [44, 203], point clouds [101], or 3D meshes [83, 109]. RGB images can also serve as additional supervisory signals if available [81, 103, 177]. While these methods learn from 2D supervision sources that are easier to obtain, they typically require multi-view observations of the same objects to be available. To this end, Kanazawa *et al.* [79] took a first step towards 3D reconstruction from static image collections by jointly optimizing for mesh textures and shape deformations.

Deep implicit 3D representations [19, 123, 137] has recently attracted wide interest to 3D reconstruction problems for their power to model complex shape topologies at arbitrary resolutions. Research efforts on learning implicit 3D shapes without 3D supervision have primarily resorted to binary occupancy [110, 131] as the representation, aiming to match reprojected 3D occupancy to the given binary masks. Current works adopting signed distance functions (SDF) either require a pretrained deep shape prior [108] or are limited to discretized representations [76] that do not scale up with resolution. Our method learns SDF representations *without* pretrained priors by establishing a more explicit geometric connection to 2D silhouettes via distance transform.

Neural image rendering. Rendering 2D images from 3D shapes is classically a non-differentiable operation in computer graphics, but recent research has advanced on making the operation differentiable and incorporable with neural networks. Differentiable (neural) rendering has been utilized for learning implicit 3D-aware representations [112, 129, 162], where earlier methods encode 3D voxelized features respecting the corresponding 2D pixel locations of images. Such 3D-aware representations, however, refrain one from interpreting *explicit* 3D geometric structures that underlies in the images. Recently, Scene Representation Networks (SRN) [161] offered up a solution of learning depth from images by keeping a close proximity of neural rendering to classical ray-tracing in computer graphics. An advantage of SRN lies in its efficiency by *learning* the ray-tracing steps, in contrast to methods that requires dense sampling along the rays [126, 131]. We take advantage of this property for 3D reconstruction, which we distinguish from "3D-aware representations", in the sense that globally consistent 3D geometric structures are recovered instead of view-dependent depth predictions.

6.3 Approach

Our 3D shape representation is a continuous implicit function $f : \mathbb{R}^3 \to \mathbb{R}$, where the 3D surface is defined by the zero level set $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = 0\}$. We define f as a multi-layer perceptron (MLP); since MLPs are composed of continuous functions (*i.e.* linear transformations and nonlinear activations), f is also continuous (almost everywhere) by construction. Therefore,

the surface of a 3D shape defined by $\mathcal{S}$ is dense and continuous, forming a 2-manifold embedded in the 3D space.

6.3.1 Learning 3D Signed Distance Functions from 2D Silhouettes

Shape silhouettes are important for learning 3D object representations: the projection of a 3D shape should match the silhouettes of a 2D observation under the given camera pose. One straightforward approach is to utilize silhouettes as *binary* masks that provide supervision on the projected occupancy, as adopted in most previous 3D-unsupervised reconstruction methods [83, 109, 110, 131]. This class of methods aims to optimize for the 3D shapes such that the projected occupancy maps are maximally matched across viewpoints. However, 2D binary occupancy maps offer little geometric supervision of the 3D shape surfaces except at pixel locations where the occupancy map changes value.

Our key insight is that the geometric interpretation of 2D silhouettes has a potentially richer explicit connection to the 3D shape surface $\mathcal{S}$, since 2D silhouettes result from the direct projection of the generating contours on $\mathcal{S}$. We can thus take advantage of the *2D distance transform* on the silhouettes, where each pixel encodes the minimum distance to the 2D silhouette(s) instead of binary occupancy. 2D distance transform is a deterministic operation on binary masks [26], and thus the output contains the same amount of information; however, such information about the geometry is dispersed to all pixels of an image. This allows us to treat 2D distance transform maps as "projections" of 3D signed distance functions (SDF), providing richer supervision on 3D shapes so that all pixels can contribute.

We discuss necessary conditions on the validity of 3D SDFs constrained by the given 2D distance transform maps, where we focus on the set of pixels exterior to the silhouette (denoted as $\mathbf{X}$). We denote $\mathbf{u} \in \mathbb{R}^2$ as the pixel coordinates and $z \in \mathbb{R}$ as projective depth. Furthermore, we denote $\mathcal{D} : \mathbb{R}^2 \to \mathbb{R}$ as the *(Euclidean) distance transform*, where $\mathcal{D}(\mathbf{u})$ encodes the distance of pixel coordinates $\mathbf{u}$ to the 2D silhouette. We assume the camera principal point is at $\mathbf{0} \in \mathbb{R}^2$.
Proposition. *If f is a valid 3D SDF, then under (calibrated) perspective cameras,*

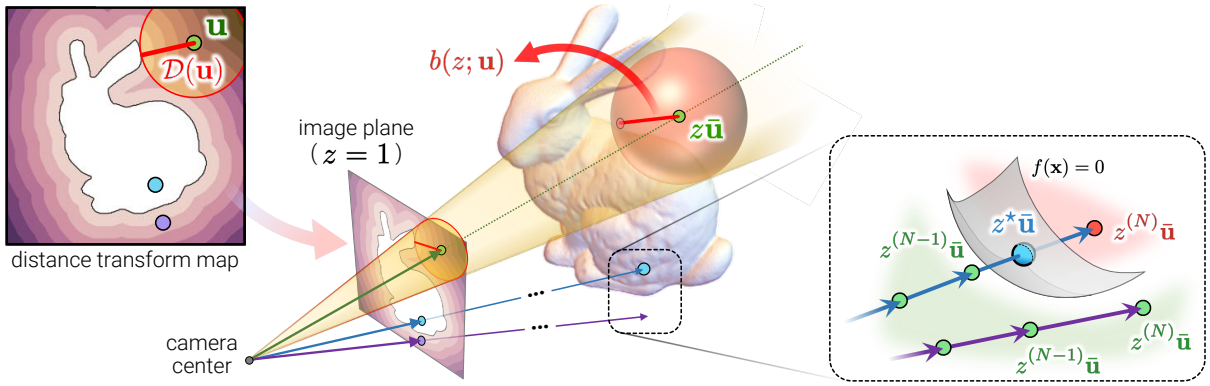
$$f(z\bar{\mathbf{u}}) \geq b(z; \mathbf{u}) = z \cdot \left\| \bar{\mathbf{u}} - \frac{\bar{\mathbf{v}}^\top \bar{\mathbf{u}}}{\bar{\mathbf{v}}^\top \bar{\mathbf{v}}} \bar{\mathbf{v}} \right\|_2 \quad \forall z \geq 1,\ \mathbf{u} \in \mathbf{X}\ , \tag{6.1}$$

where

$$\mathbf{v} = \left(1 + \frac{\mathcal{D}(\mathbf{u})}{\|\mathbf{u}\|_2}\right) \mathbf{u} \tag{6.2}$$

is the 2D point on the $\mathbf{u}$-centered circle of radius $\mathcal{D}(\mathbf{u})$ while being farthest away from the principle point, $\bar{\mathbf{u}} = [\mathbf{u}; 1] \in \mathbb{R}^3$ and $\bar{\mathbf{v}} = [\mathbf{v}; 1] \in \mathbb{R}^3$ are the respective homogeneous coordinates of $\mathbf{u}$ and $\mathbf{v}$, and $b(z; \mathbf{u})$ is a lower bound on the SDF value at 3D point location $z\bar{\mathbf{u}}$.

Proof outline. By the definition of distance transforms, any 2D point within the circle of radius $\mathcal{D}(\mathbf{u})$ centered at $\mathbf{u}$ must be free space. The back-projection of this circle from the camera



(a) SDF optimization from silhouettes (b) Zero-crossing from ray-marching

Figure 6.2: **Learning 3D SDFs from 2D images.** (a) For each pixel $\mathbf{u}$ exterior to the 2D silhouette and its distance transform value $\mathcal{D}(\mathbf{u})$, any 2D point inside the red circle (of radius $\mathcal{D}(\mathbf{u})$) must also be exterior to the silhouette. The back-projection of this circle, forming a cone in the 3D space, must correspondingly be free space. When $\mathbf{u}$ is back-projected into the 3D space to depth z, a lower bound $b(z;\mathbf{u})$ on the SDF value can be computed as the radius of the sphere centered at $z\bar{\mathbf{u}}$ and inscribed by the cone (see the supplementary material for detailed derivations). (b) For each pixel inside the 2D silhouette, the closest zero-crossing $z^{\star}\bar{\mathbf{u}}$ with the surface $\mathcal{S}=\{\mathbf{x}\in\mathbb{R}^3 \mid f(\mathbf{x})=0\}$ can be determined via the bisection method between the last two traced points $z^{(N-1)}\bar{\mathbf{u}}$ and $z^{(N)}\bar{\mathbf{u}}$, where opposite signs are encouraged via loss functions in (6.7) (indicated by the colors on the points).

center forms a cone (oblique if $\mathbf{u}\neq\mathbf{0}$), where any 3D point within must also be free space. Therefore, the radius of the sphere centered at $z\bar{\mathbf{u}}$ and inscribed by the cone serves as a lower bound $b(z;\mathbf{u})=\|z\bar{\mathbf{u}}-z'\bar{\mathbf{v}}\|_2$ for $f(z\bar{\mathbf{u}})$, where $z'\bar{\mathbf{v}}$ is the tangent point of the sphere and the conical surface (with $\mathbf{v}$ on the 2D circle back-projected to depth z'). One can thus find $b(z;\mathbf{u})$ by solving the minimization problem

$$\min_{\mathbf{v},z'\geq 0} \|z\bar{\mathbf{u}}-z'\bar{\mathbf{v}}\|_2^2 \quad \text{subject to } \|\mathbf{v}-\mathbf{u}\|_2=\mathcal{D}(\mathbf{u})\ . \tag{6.3}$$

The first-order optimality conditions on $\mathbf{v}$ and z' leads to the closed-form solution in (6.1). □

While we leave the full mathematical proof to the supplementary material, a visual sketch of the above proposition is shown in Fig. 6.2(a). This states that for any 3D point $z\bar{\mathbf{u}}\in\mathbb{R}^3$, we can always find a lower bound $b(z;\mathbf{u})$ on its possible SDF value from the given distance transform. It can also be shown that $b(z;\mathbf{u})=\mathcal{D}(\mathbf{u})$ for the special case of orthographic camera models.

We take advantage of (6.1) to optimize f, which we parametrize with $\boldsymbol{\theta}$. At each training iteration, we randomly sample M depth values $\widetilde{z}$ for each pixel $\mathbf{u}$ and formulate the loss as

$$\mathcal{L}_{\text{SDF}}(\boldsymbol{\theta})=\frac{\sum_{\mathbf{u}\in\mathbf{X}} w(\mathbf{u})\cdot\sum_{\widetilde{z}}\max\left(0,b(\widetilde{z};\mathbf{u})-f_{\boldsymbol{\theta}}(\widetilde{z}\bar{\mathbf{u}})\right)}{\sum_{\mathbf{u}\in\mathbf{X}} w(\mathbf{u})}\ . \tag{6.4}$$

We focus the loss more near the silhouettes with the weighting function $w(\mathbf{u}) = \frac{1}{\mathcal{D}(\mathbf{u})}$, decaying with the distance transform value. This is similar to the importance sampling strategy from recent works [87, 110] to improve fidelity of the 3D reconstruction or instance segmentation results.

6.3.2 Rendering Implicit 3D Surfaces

To optimize the 3D implicit surfaces from image data, a differentiable rendering function is required to interpret the continuous 3D feature space. Rendering shape surfaces involves solving for the pixel-wise depth z that corresponds to the 3D point stemming the appearance at pixel coordinates $\mathbf{u}$. Assuming perspective camera models, the problem can be defined as

$$z^\star = \arg\min_{z \geq 1} z \;\; \text{subject to} \;\; z\bar{\mathbf{u}} \in \mathcal{S} \; . \tag{6.5}$$

In other words, rendering $\mathcal{S}$ requires solving for the zero-crossing of each ray of sight closest to the camera center. Ray-casting approaches (*e.g.* sphere tracing [57] and volume rendering [96]) are classical rendering techniques for implicit surfaces in computer graphics, which have also inspired recent differentiable rendering methods [108, 126, 131, 161] for training neural networks with image data.

We build our rendering framework upon the differentiable ray-marching algorithm introduced in Scene Representation Networks (SRN) [161]. At the high level, SRN finds multi-view correspondences implicitly by searching for the terminating 3D point that would result in the most consistent RGB prediction across different viewpoints. SRN ray-marches each pixel from the camera center to predict the 3D geometry through a finite series of learnable steps. Starting at initial depth $z^{(0)}$ for pixel coordinates $\mathbf{u}$, the j-th ray-marching step and the update rule can be compactly written as

$$\begin{aligned} \Delta z^{(j)} &= \left| h_{\boldsymbol{\psi}}\left(z^{(j)}\bar{\mathbf{u}}; \boldsymbol{\eta}^{(j)}\right) \right| \, , \\ z^{(j+1)} &\leftarrow z^{(j)} + \Delta z^{(j)} \, , \qquad j \in \{0 \dots N-1\} \, , \end{aligned} \tag{6.6}$$

where $h_{\boldsymbol{\psi}} : \mathbb{R}^3 \rightarrow \mathbb{R}$ (parametrized by $\boldsymbol{\psi}$) consists of an MLP and an LSTM cell [64], and $\boldsymbol{\eta}^{(j)}$ summarizes the LSTM states for the j-th step, updated internally within in the LSTM cell. We take the absolute value on the output to ensure the ray marches away from the camera. The iterative update on the depth z is repeated N times until the final 3D point $z^{(N)}\bar{\mathbf{u}}$ is reached. Differentiable ray-marching can be viewed as a form of learned gradient descent [1] solving for the problem in (6.5), which has seen recent success in many applications involving bilevel optimization problems [41, 117, 171].

The ray-marched depth predictions from SRN, however, are view-dependent without guarantee that the resulting 3D geometry would be truly consistent across viewpoints. This can be problematic especially for untextured regions in the images, potentially leading to ambiguous multi-view correspondences being associated. By explicitly introducing the shape surface $\mathcal{S}$, we can resolve for such ambiguity and constrain the final depth prediction $z^\star$ to fall on $\mathcal{S}$. Therefore, we employ a second-stage bilevel optimization procedure following the original differentiable ray-marching

algorithm. We constrain $z^\star$ to fall within the last two ray-marching steps by encouraging negativity on the last (N-th) step (modeling a shape interior point) and positivity on the rest (modeling free space along the ray). If the conditions $f(z^{(N)}\bar{\mathbf{u}}) < 0$ and $f(z^{(N-1)}\bar{\mathbf{u}}) > 0$ are satisfied, there must exist $z^{(N-1)} < z^\star < z^{(N)}$ such that $f(z^\star\bar{\mathbf{u}}) = 0$ from the continuity of the implicit function f. Fig. 6.2(b) illustrates the above concept. We impose the penalty with margin ε as

$$\mathcal{L}_{\text{ray}}(\boldsymbol{\theta}, \boldsymbol{\psi}) = \sum_{\mathbf{u}} \sum_{j=0}^{N} \max\left(0, \alpha(\mathbf{u}) \cdot f_{\boldsymbol{\theta}}\left(z_{\boldsymbol{\psi}}^{(j)}(\mathbf{u})\bar{\mathbf{u}}\right) + \varepsilon\right), \ \alpha(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \in \mathbf{X}^{\text{c}} \text{ and } j = N \\ -1 & \text{otherwise} \end{cases} \tag{6.7}$$

by noting that the ray-marched depth values $z^{(j)}$ are dependent on h and thus parametrized by $\boldsymbol{\psi}$. We denote $\mathbf{X}^{\text{c}}$ as the complement set of $\mathbf{X}$, corresponding to the set of pixels inside the silhouettes. We also apply importance weighting using the same strategy described in Sec. 6.3.1. Finally, we solve for $z^\star_{\boldsymbol{\theta},\boldsymbol{\psi}}$ using the bisection method on $f_{\boldsymbol{\theta}}$, whose unrolled form is trivially differentiable.

The rendered pixel at the image coordinates $\mathbf{u}$ can subsequently be expressed as $\widehat{\mathcal{I}}(\mathbf{u}) = g_{\boldsymbol{\phi}}(z^\star_{\boldsymbol{\theta},\boldsymbol{\psi}}\bar{\mathbf{u}})$, where $g_{\boldsymbol{\phi}} : \mathbb{R}^3 \to \mathbb{R}^3$ (parametrized by $\boldsymbol{\phi}$) predicts the RGB values at the given 3D location. We optimize the rendered RGB appearance against the given image $\mathcal{I}$ with the loss

$$\mathcal{L}_{\text{RGB}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{\mathbf{u}} \left\| \widehat{\mathcal{I}}(\mathbf{u}) - \mathcal{I}(\mathbf{u}) \right\|_2^2 = \sum_{\mathbf{u}} \left\| g_{\boldsymbol{\phi}}\left(z^\star_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{u})\bar{\mathbf{u}}\right) - \mathcal{I}(\mathbf{u}) \right\|_2^2, \tag{6.8}$$

where we write the depth $z^\star_{\boldsymbol{\theta},\boldsymbol{\psi}}(\mathbf{u})$ as a function of the pixel coordinates $\mathbf{u}$. Ideally, $z^\star$ should be dependent only on $\boldsymbol{\theta}$, which parametrizes the surface $\mathcal{S}_{\boldsymbol{\theta}} = \{\mathbf{x} \in \mathbb{R}^3 \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = 0\}$; however, the extra dependency on $\boldsymbol{\psi}$ allows the ray-marching procedure to be much more computationally efficient as the step sizes are learned. This also poses an advantage over recent approaches based on classical sphere tracing [108] or volume rendering [126, 131], whose precision of differentiable rendering directly depends on the number of evaluations called on the implicit functions, making them inefficient.

Although SRN was originally designed to learn novel view synthesis from multi-view images, we found it to learn also from single images (in a category-specific setting). We believe this is because correspondences across individual objects still exist at the *semantic* level in single-view training, albeit unavailable at the pixel level. We take advantage of this observation for SDF-SRN. In our case, the hypernetwork learns implicit features that best explains the object semantics within the category; in turn, the ray-marching process discovers and associates implicit semantic correspondences in 3D, such that the ray-marched surfaces are semantically interpretable across all images. Therefore, shape/depth ambiguities can be resolved by learning to recover the appearance (with $\mathcal{L}_{\text{RGB}}$ here), a classical but important cue for disambiguating 3D geometry. This allows SDF-SRN to learn a strong category-specific object prior, making it trainable even from single-view images. Recent works have also shown initial success in this regard with 3D meshes [79] and 3D keypoints [183].

6.3.3 Implementation

We use an image encoder $\mathcal{E}$ as a hypernetwork [54] to predict $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and $\boldsymbol{\psi}$ (the respective function parameters of f, g, and h), written as $(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \mathcal{E}(\mathcal{I}; \boldsymbol{\Phi})$ where $\boldsymbol{\Phi}$ is the neural network weights. The implicit 3D shape $\mathcal{S}_{\boldsymbol{\theta}}$ thus corresponds to the reconstructed 3D shape from image $\mathcal{I}$, which is learned in an object-centric coordinate system. During optimization, $\mathcal{S}_{\boldsymbol{\theta}}$ is transformed to the camera frame at camera pose $(\mathbf{R}, \mathbf{t})$ corresponding to the given image $\mathcal{I}$, rewritten in a parametrized form as

$$\mathcal{S}'_{\boldsymbol{\theta}} = \mathcal{S}_{\boldsymbol{\theta}}(\mathbf{R}, \mathbf{t}) = \{\mathbf{x} \in \mathbb{R}^3 \mid f_{\boldsymbol{\theta}}(\mathbf{R}\mathbf{x} + \mathbf{t}) = 0\} \ . \tag{6.9}$$

A special property of SDFs is their differentiability with a gradient of unit norm, satisfying the eikonal equation $\|\nabla f\|_2 = 1$ (almost everywhere) [135]. Therefore, we also encourage our learned implicit 3D representation to satisfy the eikonal property by imposing the penalty

$$\mathcal{L}_{\text{eik}}(\boldsymbol{\theta}) = \sum_{\widetilde{\mathbf{x}}} \| \|\nabla f_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}})\|_2 - 1 \|_2^2 \ , \tag{6.10}$$

where $\widetilde{\mathbf{x}} \in \mathbb{R}^3$ is uniformly sampled from the 3D region of interest. Recent works on learning SDF representations have also sought to incorporate similar regularizations on implicit shapes [49, 76].

To summarize, given a dataset of D tuples $\{(\mathcal{I}, \mathbf{X}, \mathbf{R}, \mathbf{t})_d\}_{d=1}^D$ that consists of the RGB images, 2D silhouettes and camera poses, we train the network $\mathcal{E}$ end-to-end with the overall objective

$$\mathcal{L}_{\text{all}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = \lambda_{\text{SDF}}\mathcal{L}_{\text{SDF}}(\boldsymbol{\theta}) + \lambda_{\text{RGB}}\mathcal{L}_{\text{RGB}}(\boldsymbol{\theta}, \boldsymbol{\psi}) + \lambda_{\text{ray}}\mathcal{L}_{\text{ray}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) + \lambda_{\text{eik}}\mathcal{L}_{\text{eik}}(\boldsymbol{\theta}) \tag{6.11}$$

by noting that $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and $\boldsymbol{\psi}$ are predicted by the hypernetwork $\mathcal{E}(\mathcal{I}; \boldsymbol{\Phi})$.

6.4 Experiments

Architectural details. We use ResNet-18 [59] followed by fully-connected layers as the encoder. We implement the implicit functions $f_{\boldsymbol{\theta}}$, $g_{\boldsymbol{\phi}}$ and $h_{\boldsymbol{\psi}}$ as a shared MLP backbone (Fig. 6.3), with f and g connecting to shallow heads and h taking the backbone feature as the LSTM input (we do not predict the LSTM parameters with $\mathcal{E}$). The MLP backbone takes a 3D point with positional encoding [126] as input (also added as intermediate features to the hidden layers), which helps improve the reconstruction quality. We leave architectural details to the supplementary material.

Training settings. For a fair comparison, we train all networks with the Adam optimizer [85] with a learning rate of 10^{-4} and batch size 16. We choose $M = 5$ points for $\mathcal{L}_{\text{SDF}}$ and set the margin $\varepsilon = 0.01$ when training SDF-SRN. Unless otherwise specified, we choose the loss weights to be $\lambda_{\text{RGB}} = 1$, $\lambda_{\text{SDF}} = 3$, $\lambda_{\text{eik}} = 0.01$; we set λ_{ray} to be 1 for the last marched point and 0.1 otherwise. For each training iteration, we randomly sample 1024 pixels $\mathbf{u}$ from each image for faster training.

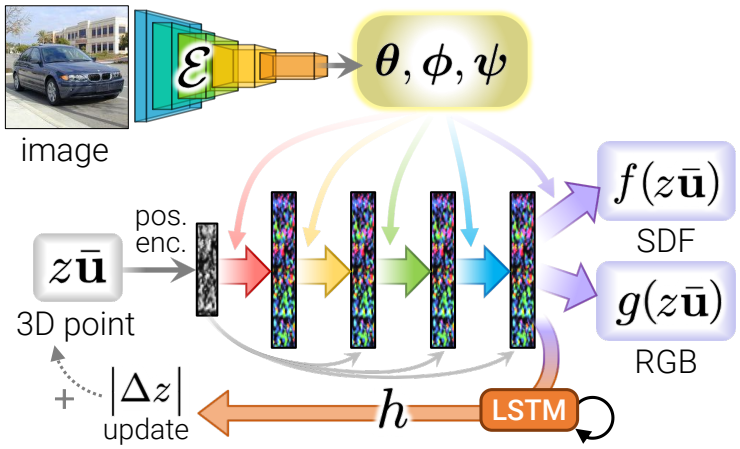


Figure 6.3: The implicit functions f, g, and h in SDF-SRN share an MLP backbone with the parameters encoded by $\mathcal{E}$.

Category	airplane		car		chair	
	accur.	cover.	accur.	cover.	accur.	cover.
SoftRas [109]	0.250	0.222	0.356	0.302	0.690	0.491
DVR [131]	1.795	0.258	1.538	0.432	1.274	0.699
SDF-SRN (ours)	**0.193**	**0.154**	**0.141**	**0.144**	**0.352**	**0.315**
DVR (w/ depth)	0.320	0.171	0.184	0.181	0.322	0.330

Table 6.1: **Quantitative results** on multi-view ShapeNet data *without* viewpoint association from CAD model correspondences. Note that SDF-SRN even outperforms DVR supervised with depth from visual hull [131]. All numbers are scaled by 10 (the lower the better).

Evaluation criteria. To convert implicit 3D surfaces $\mathcal{S}$ to explicit 3D representations, we sample SDF values at a voxel grid of resolution 128^3 and extract the 0-isosurface with Marching Cubes [114] to obtain watertight 3D meshes. We evaluate by comparing uniformly sampled 3D points from the mesh predictions to the ground-truth point clouds with the bidirectional metric of Chamfer distance, measuring different aspects of quality: shape accuracy and surface coverage [103, 123].

6.4.1 ShapeNet Objects

Datasets. We evaluate our method on the airplane, car, and chair categories from ShapeNet v2 [16], which consists of 4045, 3533, and 6778 CAD models respectively. We benchmark with renderings provided by Kato *et al.* [83], where the 3D CAD models were rendered at 24 uniform viewpoints to 64×64 images. We split the dataset into training/validation/test sets following Yan *et al.* [203].

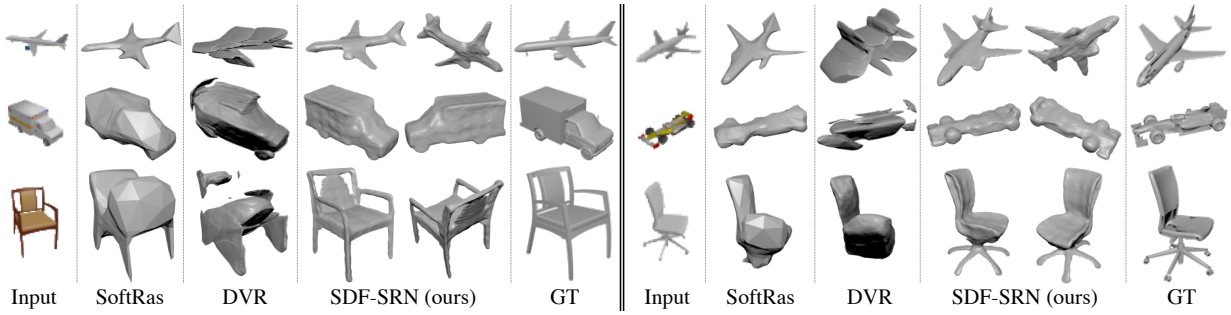


Figure 6.4: **Qualitative results** of ShapeNet 3D reconstruction from *single-view training* on multi-view data (*i.e.* no multi-view association is available). Both SoftRas and DVR learns to fit to the silhouettes but fail to learn reasonable 3D reconstruction from independent images, showing their reliance on multi-view constraints. SDF-SRN, in contrast, does not suffer from such limitation and reconstructs 3D objects with high fidelity and successfully recovers the target shape topologies.

Experimental settings. We consider a category-specific setting and compare against two baseline methods for learning-based 3D reconstruction without 3D supervision: (a) Differentiable Volumetric Rendering (DVR) [131], a state-of-the-art method for learning implicit occupancy functions, and (b) Soft Rasterizer (SoftRas) [109], a state-of-the-art method for learning 3D mesh reconstruction. We assume known camera poses as with previous works. We also set $\lambda_{\text{SDF}} = 1$ for the airplane category.

Previous works on learning 3D reconstruction from multi-view 2D supervision, including SoftRas and DVR, have assumed additional known *CAD model correspondences*, *i.e.* one knows a priori which images correspond to the same CAD model. This allows a training strategy of pairing images with viewpoints selected from the same 3D shape, making the problem more constrained and much easier to optimize. Practically, however, one rarely encounters the situation where such viewpoint associations are explicitly provided while also having silhouettes annotated. In pursuit of an even more practical scenario of 3D-unsupervised learning, we train all models where all rendered images are treated *independently*, *without* the knowledge of multi-view association. This is equivalent to *single-view training* on multi-view data, which is more challenging than "multi-view supervision", and can also be regarded as an autoencoder setting with additional camera pose supervision.

Results. We evaluate on the multi-view rendered images and train all airplane and car models for 100K iterations and all chair models for 200K iterations. The results are reported in Table 6.1 and visualized in Fig. 6.4. SDF-SRN outperforms both baseline methods by a wide margin and recovers accurate 3D object shapes, despite no explicit information of multi-view association being available. On the other hand, SoftRas and DVR trained on individual images without such multi-view constraint do not learn shape regularities within the category and cannot resolve for shape ambiguities, indicating their high reliance on viewpoint association of the same CAD model. Furthermore, SDF-SRN even outperforms DVR additionally supervised with depth information

# CADs	SoftRas [109] accur.	SoftRas [109] cover.	DVR [131] accur.	DVR [131] cover.	Ours accur.	Ours cover.
500	0.550	0.508	1.298	0.674	**0.475**	**0.422**
1K	0.547	0.551	1.284	0.701	**0.442**	**0.385**
2K	0.522	0.481	1.268	0.609	**0.423**	**0.349**
4.7K (all)	0.510	0.471	1.367	1.135	**0.401**	**0.329**

Table 6.2: **Performance analysis** of ShapeNet chairs on the amount of CAD models available as training data under *single*-view supervision. All numbers are scaled by 10 (the lower the better).

Figure 6.5: SDF-SRN recovers reasonable 3D shapes and topologies even from *single-view* supervision, where each instance appears only once in the training set.

extracted from visual hulls [131].

We further analyze the performances under the more practical *single-view* supervision setting on *single-view* data, where only one image (at a random viewpoint) per CAD model is included in the training set. We choose the chair category and train each model for 50K iterations; since there are much fewer available training images, we add data augmentation with random color jittering on the fly to reduce overfitting. Table 6.2 shows that SDF-SRN also outperforms current 3D-unsupervised methods in this challenging setting even trained with scarce data (500 images), which further improves as more training data becomes available. In addition, SDF-SRN is able to recover accurate shape topologies (Fig. 6.5) even under this scenario, giving hints to its applicability to real-world images.

Ablative analysis. We discuss the effects of different components essential to SDF-SRN (under the multi-view data setup) in Table 6.3. Training with binary cross-entropy (classifying 3D occupancy) results in a significant drop of performance, which we attribute to the nonlinear nature of MLPs using discontinuous binary functions as the objective. Table 6.3 also shows the necessity of differentiable rendering ($\mathcal{L}_{\text{RGB}}$) for resolving shape ambiguities (*e.g.* concavity cannot be inferred solely from silhouettes), importance weighting for learning finer shapes that aligns more accurately to silhouettes, and positional encoding of 3D point locations to extract more high-frequency details from images. We additionally show that if the camera pose were given, one could

	accur.	cover.
binary occupancy	0.425	0.790
w/o rendering loss $\mathcal{L}_{\text{RGB}}$	0.353	0.554
w/o importance weighting	0.483	1.168
w/o positional encoding	0.444	0.351
full model (SDF-SRN)	**0.352**	**0.315**
w/ test-time optimization	**0.332**	**0.303**

Table 6.3: Ablation studies of SDF-SRN (conducted on ShapeNet chairs). We additionally evaluate the performance with test-time optimization on the latent code over the fully trained networks if the camera poses were known a priori.

optimize the same losses in (6.11) for the latent code over the trained network, further reducing prediction uncertainties and improving test-time reconstruction performance [103, 137, 161].

6.4.2 Natural Images

Dataset. We demonstrate the efficacy of our method on PASCAL3D+ [197], a 3D reconstruction benchmarking dataset of real-world images with ground-truth CAD model annotations. We evaluate on the airplane, car, and chair categories from the ImageNet [31] subset, which consists of 1965, 5624, and 1053 images respectively. PASCAL3D+ is challenging in at least 3 aspects: (a) the images are much scarcer than ShapeNet renderings since human annotations of 3D CAD models is laborious and difficult to scale up; (b) object instances appear only once without available multi-view observations to associate with; (c) the large variations of textures and lighting in images makes it more difficult to learn from. We assume weak-perspective camera models and use the provided 2D bounding boxes to normalize the objects, square-crop the images, and resize them to 64×64.

Experimental settings. We compare against DVR [131] as well as Category-specific Mesh Reconstruction (CMR) [79], which learns 3D mesh reconstruction from static images. Only RGB images and object silhouettes are available during training; we do not assume any available pretrained 2.5D/3D shape priors from external 3D datasets [177, 194]. We initialize the meshes in CMR with a unit sphere; for a fair comparison, we consider CMR using ground-truth camera poses instead of predicting them from keypoints, which was also reported to yield better performance. We train each model for 30K iterations, augmenting the dataset with random color jittering and image rescaling (uniformly between $[0.8, 1.2]$). We set $\lambda_{\text{RGB}} = 10$ and $\lambda_{\text{eik}} = 1$ for SDF-SRN. We evaluate quantitatively with shape predictions registered to the ground truth using the Iterative Closest Point algorithm [11].

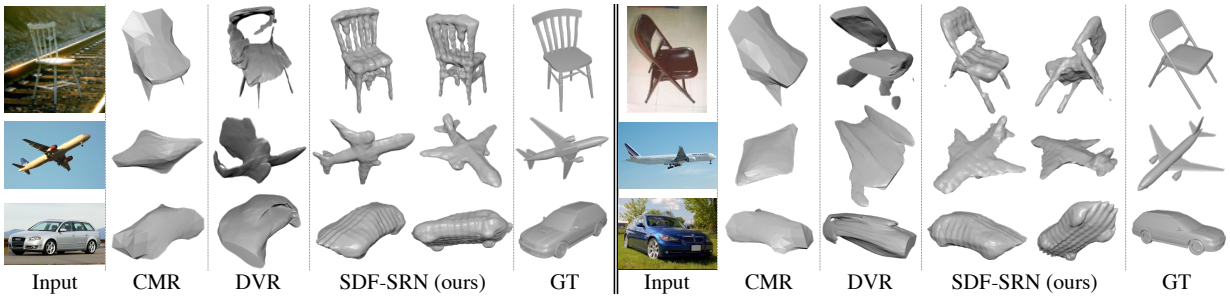


Figure 6.6: **Qualitative results** from PASCAL3D+ reconstruction. Compared to the two baseline methods, SDF-SRN recovers significantly more accurate 3D shapes and topologies from the images. Both CMR and DVR struggle to associate meaningful shape regularities within category, while CMR also suffers from topological limitations due to its mesh-based nature, as with SoftRas.

Category	airplane		car		chair	
	accur.	cover.	accur.	cover.	accur.	cover.
CMR [79]	0.625	0.803	0.474	0.623	1.396	1.168
DVR [131]	1.483	0.916	1.493	0.795	3.356	2.251
Ours	**0.582**	**0.543**	**0.391**	**0.402**	**0.478**	**0.398**

Table 6.4: **Quantitative comparison** on PASCAL3D+. All models were trained from scratch solely on the images and silhouettes, without utilizing 2.5D/3D shape priors pretrained from external 3D datasets. SDF-SRN consistently outperforms both baseline methods. All numbers are scaled by 10 (the lower the better).

Results. We present qualitative results in Fig. 6.6. Even though the texture and lighting variations in PASCAL3D+ makes it more difficult than ShapeNet to associate correspondences across images, SDF-SRN is able to recover more accurate shapes and topologies from the images compared to the baseline methods. CMR has difficulty learning non-convex shape parts (*e.g.* plane wings), while DVR has difficulty learning meaningful shape semantics within category. We note that SDF-SRN also suffers slightly from shape ambiguity (*e.g.* sides of cars tend to be concave); nonetheless, SDF-SRN still outperforms CMR and DVR quantitatively by a wide margin (Table 6.4). We also visualize colors and surface normals from the reconstructions rendered at novel viewpoints (Fig. 6.7), showing that SDF-SRN is able to capture meaningful semantics from singe images such as object symmetry.

Finally, we note that recent research has shown possible to obtain camera poses from more practical supervision sources (*e.g.* 2D keypoints [132, 184]), which could allow learning shape reconstruction from larger datasets that are much easier to annotate. We leave this to future work.

Figure 6.7: PASCAL3D+ reconstruction with color and surface normal predictions.

6.5 Conclusion

We have introduced SDF-SRN for learning dense 3D reconstruction from static image collections. Our framework learns SDF shape representations from distance transformed 2D silhouettes with improved differentiable rendering. SDF-SRN does not rely on associated multi-view supervision and learns from *single*-view images, demonstrating compelling 3D reconstruction results even on challenging natural images. We believe this is an exciting direction and opens up avenues for learning 3D reconstruction from larger real-world datasets with more practical supervision.

6.6 Broader Impact

Our proposed framework, SDF-SRN, allows for learning dense 3D geometry of object categories from real-world images using annotations (*i.e.* 2D silhouettes) that can be feasibly obtained at a large scale. Computer vision increasingly needs to perform 3D geometric reasoning from images, such as when an autonomous vehicle encounters a vehicle in the streets. To avoid catastrophe, the car must not only detect the existence of the vehicle but also exactly determine its spatial extent in the 3D world. Similarly, robots and drones are increasingly deployed in unconstrained environments where they must safely manipulate and avoid 3D objects. Health professionals are increasingly using computer vision to interpret 2D scans/imagery in 3D. Breakthroughs in dense geometric reasoning could allow researchers to extract unprecedented detail from visual data.

This work also offers up exciting new opportunities in the area of computer graphics for 3D content creation, where the laborious process of creating 3D models and animations could be significantly simplified. This could reduce the time and money costs required for many industrial applications (*e.g.* involving virtual reality). We should note that all new technologies have the potential for misuse, and our framework SDF-SRN is no different. However, we strongly believe

the myriad of possible societal and economic benefits of our work vastly outweigh such risks.

6.A Derivation of the Proposition

6.A.1 Formal Proof

We provide more detailed derivations of the lower bound $b(z; \mathbf{u})$. For clarity, we briefly restate the proof outline first before going into the main proof.

For pixel coordinates $\mathbf{u}$ on the image plane with distance transform value $\mathcal{D}(\mathbf{u})$, the set of 2D points $\{\mathbf{v} \mid \|\mathbf{v}-\mathbf{u}\|_2 \leq \mathcal{D}(\mathbf{u})\}$ within the circle of radius $\mathcal{D}(\mathbf{u})$ centered at $\mathbf{u}$ must be exterior to the 2D silhouette. It immediately follows that the set of 3D points $\{z'\bar{\mathbf{v}} \mid z' \geq 0, \|\mathbf{v}-\mathbf{u}\|_2 \leq \mathcal{D}(\mathbf{u})\}$ within the (oblique) cone formed by back-projecting the circle from the camera center must be free space. For a 3D point $z\bar{\mathbf{u}}$, the SDF value $f(z\bar{\mathbf{u}})$ is thus lower-bounded by the radius of the sphere centered at $z\bar{\mathbf{u}}$ while being inscribed by the cone. Let $b(z; \mathbf{u}) = \|z\bar{\mathbf{u}} - z'\bar{\mathbf{v}}\|_2$ be the lower bound for $f(z\bar{\mathbf{u}})$, where $z'\bar{\mathbf{v}}$ is the tangent point of the sphere and the conical surface (with $\mathbf{v}$ on the 2D circle back-projected to depth z'). We aim to find $b(z; \mathbf{u})$ by solving the problem

$$\min_{\mathbf{v}, z' \geq 0} \|z\bar{\mathbf{u}} - z'\bar{\mathbf{v}}\|_2^2 \quad \text{subject to } \|\mathbf{v}-\mathbf{u}\|_2 = \mathcal{D}(\mathbf{u}) \ . \tag{6.12}$$

First, by noting $\mathbf{u} = (u_x, u_y)$, we reparametrize $\mathbf{v}$ (the set of 2D points on the circle) with θ as

$$\mathbf{v} = \begin{bmatrix} u_x + \mathcal{D}(\mathbf{u}) \cos\theta \\ u_y + \mathcal{D}(\mathbf{u}) \sin\theta \end{bmatrix} \ . \tag{6.13}$$

The problem in (6.12) thus becomes

$$\min_{\theta, z' \geq 0} \left\| z \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} - z' \begin{bmatrix} u_x + \mathcal{D}(\mathbf{u}) \cos\theta \\ u_y + \mathcal{D}(\mathbf{u}) \sin\theta \\ 1 \end{bmatrix} \right\|_2^2 . \tag{6.14}$$

Without loss of generality, the first-order optimality condition on θ is

$$\begin{aligned} 0 &= 2 \cdot \left(z \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} - z' \begin{bmatrix} u_x + \mathcal{D}(\mathbf{u}) \cos\theta \\ u_y + \mathcal{D}(\mathbf{u}) \sin\theta \\ 1 \end{bmatrix} \right)^\top \left(-z' \begin{bmatrix} -\mathcal{D}(\mathbf{u}) \sin\theta \\ \mathcal{D}(\mathbf{u}) \cos\theta \\ 0 \end{bmatrix} \right) \\ &= \left(z \begin{bmatrix} u_x \\ u_y \end{bmatrix} - z' \begin{bmatrix} u_x + \mathcal{D}(\mathbf{u}) \cos\theta \\ u_y + \mathcal{D}(\mathbf{u}) \sin\theta \end{bmatrix} \right)^\top \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \\ &= \left(z \begin{bmatrix} u_x \\ u_y \end{bmatrix} - z' \begin{bmatrix} u_x \\ u_y \end{bmatrix} \right)^\top \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}^\top \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} , \end{aligned} \tag{6.15}$$

leading to $\theta = \tan^{-1} \frac{u_y}{u_x}$ and thus

$$\cos\theta = \frac{u_x}{\|\mathbf{u}\|_2} \quad \text{and} \quad \sin\theta = \frac{u_y}{\|\mathbf{u}\|_2} \,. \tag{6.16}$$

This indicates the optimality of θ is only dependent on the (normalized) image coordinates $\mathbf{u}$ while being *independent* of both the given depth z and the variable z'. Plugging (6.16) back into (6.13) leads to

$$\mathbf{v} = \begin{bmatrix} u_x + \mathcal{D}(\mathbf{u}) \dfrac{u_x}{\|\mathbf{u}\|_2} \\ u_y + \mathcal{D}(\mathbf{u}) \dfrac{u_y}{\|\mathbf{u}\|_2} \end{bmatrix} = \left(1 + \frac{\mathcal{D}(\mathbf{u})}{\|\mathbf{u}\|_2}\right) \mathbf{u} \,. \tag{6.17}$$

Having solved for $\mathbf{v}$, the problem in (6.12) simplifies into

$$\min_{z' \geq 0} \|z\bar{\mathbf{u}} - z'\bar{\mathbf{v}}\|_2^2 \,, \tag{6.18}$$

which is a linear problem with the solution

$$z' = z \cdot \frac{\bar{\mathbf{u}}^\top \bar{\mathbf{v}}}{\bar{\mathbf{v}}^\top \bar{\mathbf{v}}} \,. \tag{6.19}$$

Note that z' satisfies the non-negativity constraint by nature; one can verify by plugging (6.17) into (6.19).

Finally, the lower bound thus becomes

$$b(z; \mathbf{u}) = \|z\bar{\mathbf{u}} - z'\bar{\mathbf{v}}\|_2 = z \cdot \left\| \bar{\mathbf{u}} - \frac{\bar{\mathbf{v}}^\top \bar{\mathbf{u}}}{\bar{\mathbf{v}}^\top \bar{\mathbf{v}}} \bar{\mathbf{v}} \right\|_2 \tag{6.20}$$

by noting the expression of $\mathbf{v}$ is given in (6.17).

6.A.2 Orthographic cameras

We show that $b(z; \mathbf{u}) = \mathcal{D}(\mathbf{u})$ under orthographic cameras. Intuitively, one can imagine the camera center to be pulled away from the image to negative infinity, in which case the back-projected cone would approach an unbounded cylinder. Correspondingly, the radius of the inscribed sphere would always be $\mathcal{D}(\mathbf{u})$, irrespective of the queried depth z.

The back-projected 3D point (denoted as $\mathbf{x}$) from pixel coordinates $\mathbf{u}$ at depth z has a fixed distance $z - 1$ to the image plane. Denoting f_c as the camera focal length and writing the new depth as a function of f_c, we can rewrite the queried 3D point as

$$\mathbf{x} = \frac{z - 1 + f_\text{c}}{f_\text{c}} \begin{bmatrix} \mathbf{u} \\ f_\text{c} \end{bmatrix} \,, \tag{6.21}$$

which becomes $\mathbf{x} = z\bar{\mathbf{u}}$ when $f_c = 1$. Similarly, we can rewrite the tangent point of the cone and the inscribed sphere

$$\mathbf{x}' = \frac{z' - 1 + f_c}{f_c} \begin{bmatrix} \mathbf{v} \\ f_c \end{bmatrix} , \tag{6.22}$$

which becomes $\mathbf{x}' = z'\bar{\mathbf{v}}$ when $f_c = 1$. The lower bound $b(z; \mathbf{u}) = \|\mathbf{x} - \mathbf{x}'\|_2$ thus becomes

$$\begin{aligned} b(z; \mathbf{u}) &= \|\mathbf{x} - \mathbf{x}'\|_2 \\ &= \left\| \frac{z - 1 + f_c}{f_c} \begin{bmatrix} \mathbf{u} \\ f_c \end{bmatrix} - \frac{z' - 1 + f_c}{f_c} \begin{bmatrix} \mathbf{v} \\ f_c \end{bmatrix} \right\|_2 . \end{aligned} \tag{6.23}$$

For orthographic cameras (where f_c approaches infinity), taking $f_c \to \infty$ on (6.23) yields

$$\begin{aligned} \lim_{f_c \to \infty} b(z; \mathbf{u}) &= \lim_{f_c \to \infty} \left\| \frac{z - 1 + f_c}{f_c} \begin{bmatrix} \mathbf{u} \\ f_c \end{bmatrix} - \frac{z' - 1 + f_c}{f_c} \begin{bmatrix} \mathbf{v} \\ f_c \end{bmatrix} \right\|_2 \\ &= \|\mathbf{u} - \mathbf{v}\|_2 = \mathcal{D}(\mathbf{u}) . \end{aligned} \tag{6.24}$$

6.B Dataset

In this section, we provide more details on the datasets used in the experiments.

6.B.1 ShapeNet [16]

The dataset split from Yan *et al.* [203] were from ShapeNet v1. As nearly half of the CAD models in the car category were removed in ShapeNet v2, we take the intersection as the final splits for our experiments. The final statistics of ShapeNet CAD models are reported in Table 6.5. Following standard protocol, we used the validation set for hyperparameter tuning and the test set for evaluation.

The ground-truth point clouds provided by Kato *et al.* [83] were directly sampled from the ShapeNet CAD models. However, many interior details were also included in a significant portion of the CAD models, especially the car category (*e.g.* car seats and steering wheels). Such details cannot be recovered by 3D-unsupervised 3D reconstruction frameworks, including SoftRas [109], DVR [131], and the proposed SDF-SRN, as they are limited to reconstructing the outer (visible) surfaces of objects. Therefore, we use the ground truth provided by Groueix *et al.* [52], whose point clouds were created using a virtual mesh scanner from Wang *et al.* [186]. We re-normalize the point clouds to match that from Kato *et al.* [83], tightly fitting a zero-centered unit cube.

When performing on-the-fly data augmentation in the single-view supervision experiments, we randomly perturb the brightness by $[-20\%, 20\%]$, contrast by $[-20\%, 20\%]$, saturation by $[-20\%, 20\%]$, and hue uniformly in the entire range. We do not perturb the image scales for ShapeNet renderings.

Category	train	validation	test	total
airplane	2830	809	405	4044
car	2465	359	690	3514
chair	4744	678	1356	6778

Table 6.5: Dataset statistics of ShapeNet v2 (number of CAD models).

Category	train	validation	total
airplane	991	974	1965
car	2847	2777	5624
chair	539	514	1053

Table 6.6: Dataset statistics of PASCAL3D+ (number of images).

6.B.2 PASCAL3D+ [197]

The PASCAL3D+ dataset is comprised of two subsets from PASCAL VOC [39] and ImageNet [31], labeled with ground-truth CAD models and camera poses. We evaluate on the ImageNet subset as it exhibits much less object occlusions than the PASCAL VOC subset. We note that occlusion handling is still an open problem to all methods (including the proposed SDF-SRN) since appropriate normalization of object scales is required to learn meaningful semantics within category. The statistics of PASCAL3D+ used in the experiments are reported in Table 6.6.

Since the natural images from PASCAL3D+ come in different image and object sizes, we rescale by utilizing the (tightest) 2D bounding boxes. In particular, we center the object and rescale such that 1.2 times the longer side of the bounding box fits the canonical images (with a resolution of 64×64); for the car category, we rescale such that the height of the bounding box fits $1/3$ of the canonical image height. When performing on-the-fly data augmentation, we randomly perturb the brightness by $[-20\%, 20\%]$, contrast by $[-20\%, 20\%]$, saturation by $[-20\%, 20\%]$, and hue uniformly in the entire range. We additionally perturb the image scale by $[-20\%, 20\%]$.

We use the ground-truth CAD model and camera pose associated with each image to create the object silhouettes. For evaluation, we create ground-truth point clouds from the 3D CAD models using the virtual mesh scanner from Wang *et al.* [186] (described in Sec. 6.B.1) and rescale by

$$s = \frac{\text{camera focal length}}{\text{camera distance}} \cdot \frac{2}{64} \,, \tag{6.25}$$

which scales the point clouds to match the $[-1, 1]$ canonical image space where the silhouettes lie. Note that the camera parameters provided from the dataset are used only for evaluation. Since there may still exist misalignment mainly due to depth ambiguity, we run the rigid version of the Iterative Closest Point algorithm [11] for 50 iterations to register the prediction to the rescaled ground truth.

6.C Architectural and Training Details

We provide a more detailed description of the network architectures of SDF-SRN. As described, the implicit functions $f_{\boldsymbol{\theta}}$, $g_{\boldsymbol{\phi}}$ and $h_{\boldsymbol{\psi}}$ share an MLP backbone extracting point-wise deep features for each 3D point. The shared MLP backbone consists of 4 linear layers with 128 hidden units,

with layer normalization [8] and ReLU activations in between. The shallow heads of f and g are single linear layers, predicting the SDF and RGB values respectively. The encoder $\mathcal{E}$ is built with a ResNet-18 [59] followed by fully-connected layers, which consists of six 512-unit hidden layers for the ShapeNet [16] experiments and two 256-unit hidden layers for the PASCAL3D+ [197] experiments. We use separate branches of fully-connected layers to predict the weights and biases of each linear layer of the implicit function, *i.e.* the latent code (from the encoder) is passed to 4 sets of fully-connected layers for the 4 linear layers of the MLP backbone, and similarly for the shallow heads. We choose the hidden and output state dimension for the LSTM to be 32 and add a following linear layer to predict the update step for the depth δz.

Following prior practice [79, 131], we initialize ResNet-18 with weights pretrained with ImageNet [31]. To make the training of SDF-SRN better conditioned, we also pretrain the fully-connected part of the hypernetwork to initially predict an SDF space of a zero-centered sphere with radius r. In practice, we randomly sample $\widetilde{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ on the hidden latent space to predict the weights $\widetilde{\boldsymbol{\theta}}$ of an implicit function $f_{\widetilde{\boldsymbol{\theta}}}$. Subsequently, we uniformly sample $\widetilde{\mathbf{x}} \in \mathbb{R}^3$ in the 3D space and minimize the loss

$$\mathcal{L}_{\text{pretrain}} = \sum_{\widetilde{\mathbf{x}}} \left\| f_{\widetilde{\boldsymbol{\theta}}}(\widetilde{\mathbf{x}}) - \left(\|\widetilde{\mathbf{x}}\|_2 - r \right) \right\|_2^2 . \tag{6.26}$$

We choose radius $r = 0.5$ and pretrain the fully-connected layers randomly sampling 10K points for 2000 iterations. We find this pretraining step important for facilitating convergence at the early stage of training and avoiding degenerate solutions.

To encourage high-frequency components to be recovered in the 3D shapes, we take advantage of the positional encoding technique advocated by Mildenhall *et al.* [126]. For each input 3D point $\mathbf{x}$ of the implicit functions f, g, and h, we map $\mathbf{x}$ to higher dimensions with the function

$$\gamma(p) = \left[p, \cos(2^0 p), \sin(2^0 p), \ldots, \cos(2^{L-1} p), \sin(2^{L-1} p) \right] , \tag{6.27}$$

where p is applied to all 3D coordinates of $\mathbf{x} = (x, y, z)$ and subsequently concatenated. We choose $L = 6$ in our implementation.

6.D Additional Results

We visualize additional qualitative comparisons for the ShapeNet [16] multi-view supervision experiment in Fig. 6.8 for airplanes, Fig. 6.9 for cars, and Fig. 6.10 for chairs. We again emphasize that these results are from *unknown* multi-view associations, where each image is treated independently during training. SDF-SRN is able to consistently capture meaningful shape semantics within category, while SoftRas [109] and DVR [131] suffer from the lack of viewpoint correspondences. In addition, SDF-SRN can successfully capture various shape topologies that underlies in the images. We also provide additional results of SDF-SRN on natural images (PASCAL3D+ [197]) in Fig. 6.11 for airplanes, Fig. 6.12 for cars, and Fig. 6.13 for chairs.

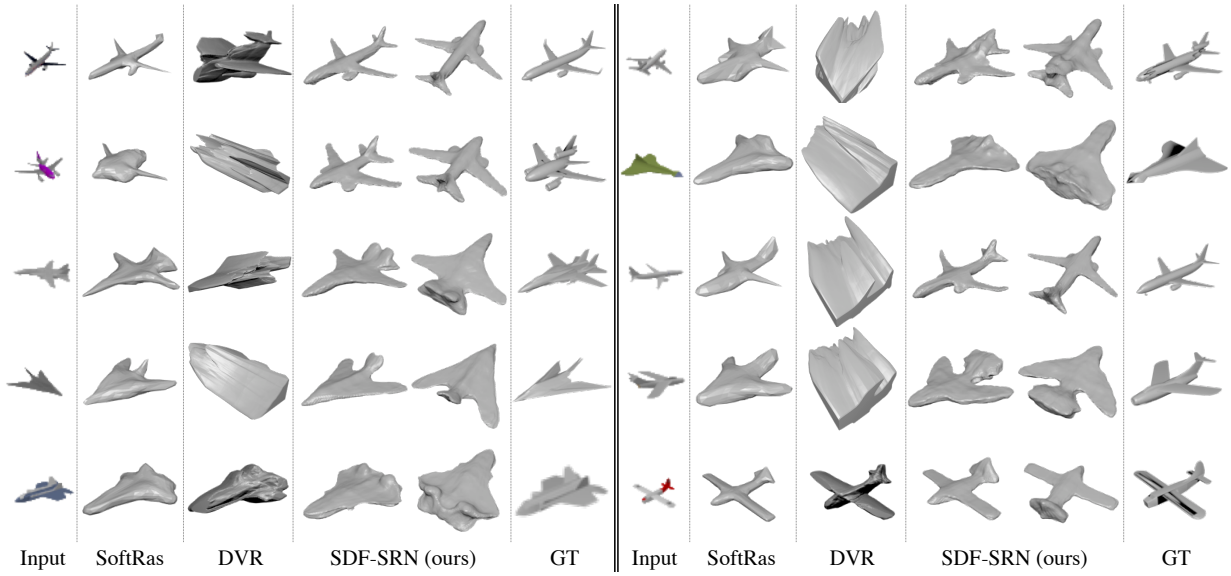


Figure 6.8: Additional results on ShapeNet airplanes.

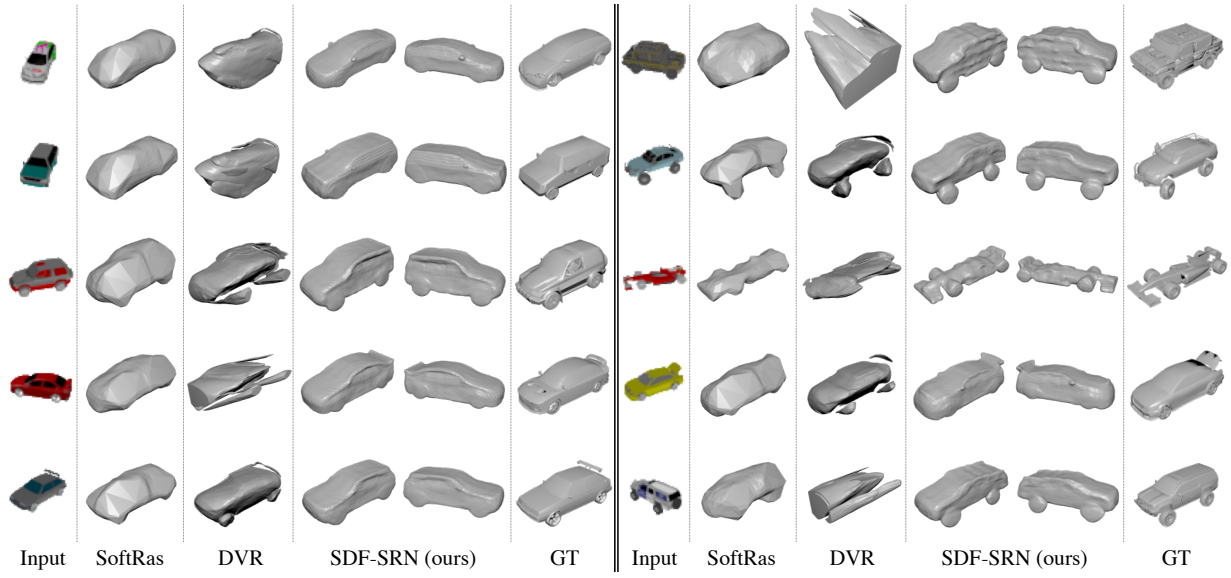


Figure 6.9: Additional results on ShapeNet cars.

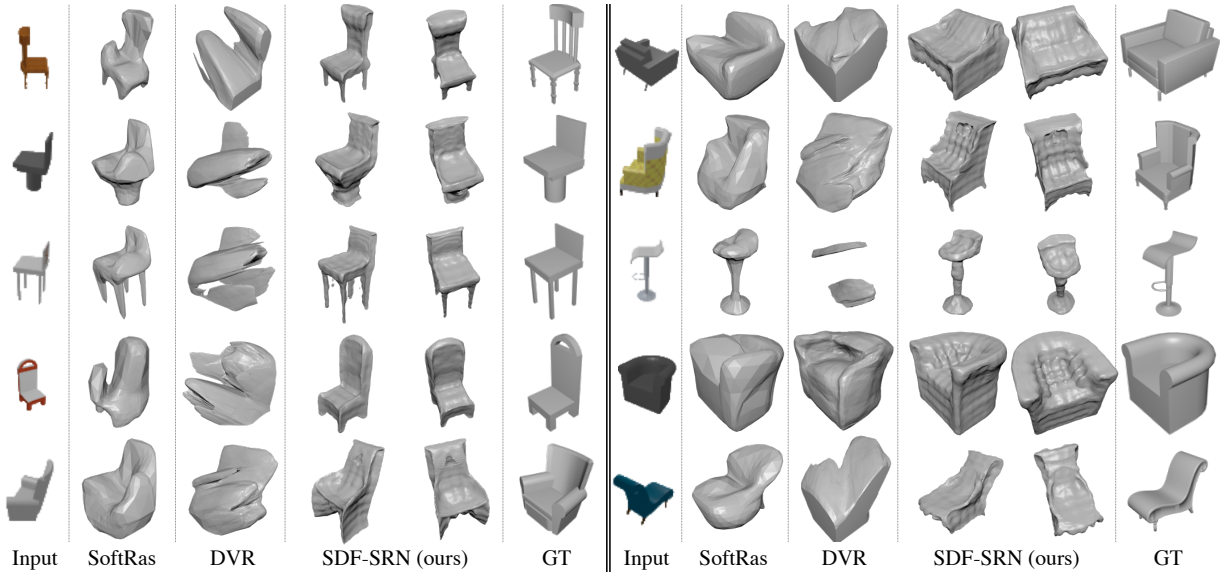


Figure 6.10: Additional results on ShapeNet chairs.

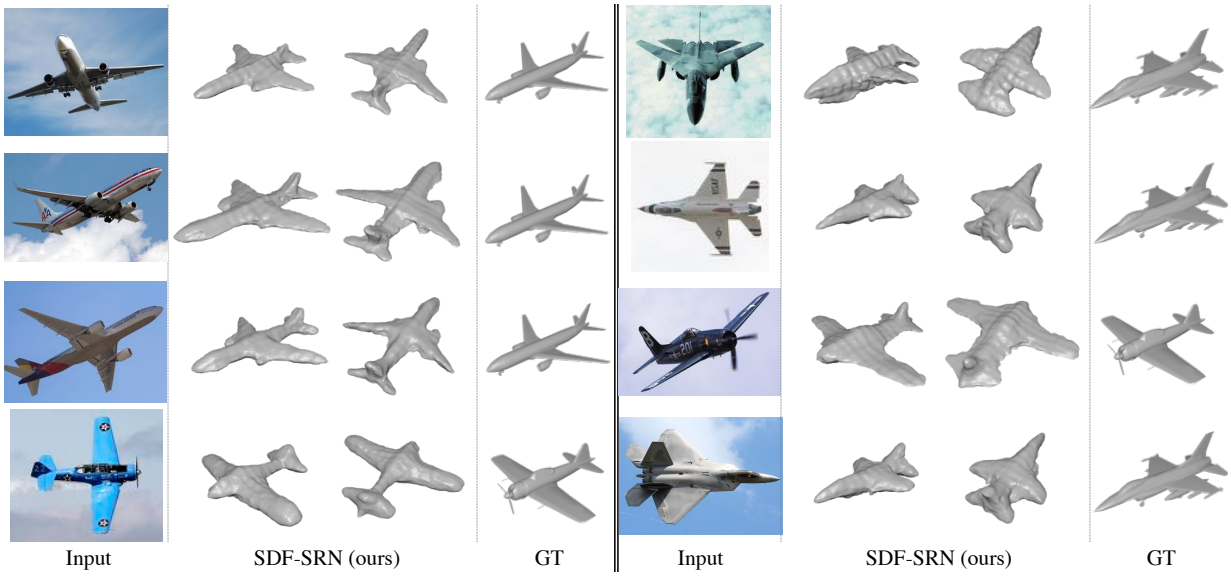


Figure 6.11: Additional results on PASCAL3D+ airplanes.

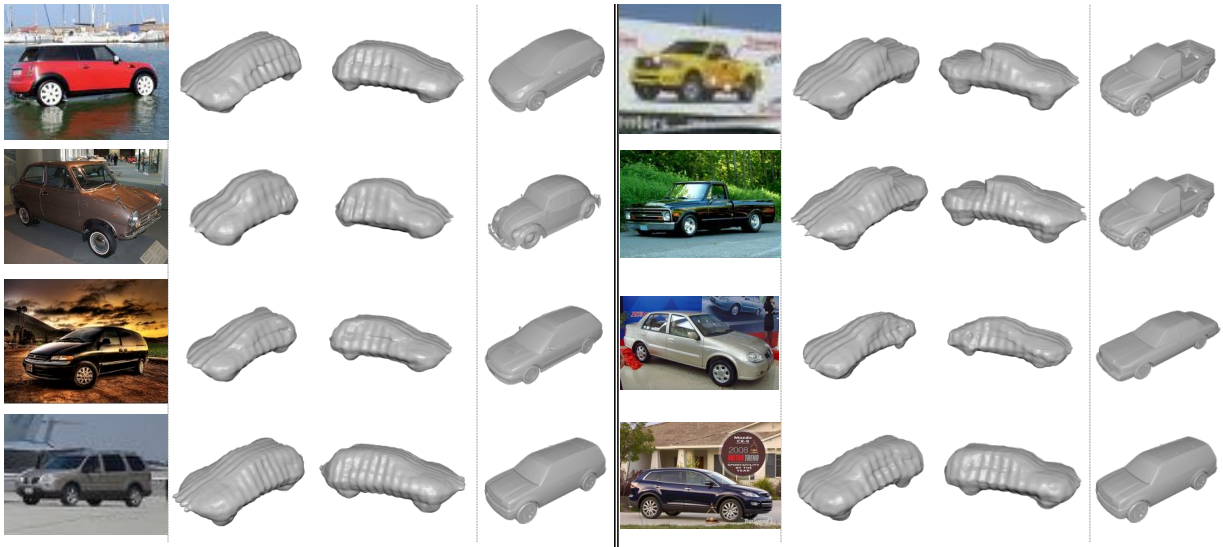


Figure 6.12: Additional results on PASCAL3D+ cars.

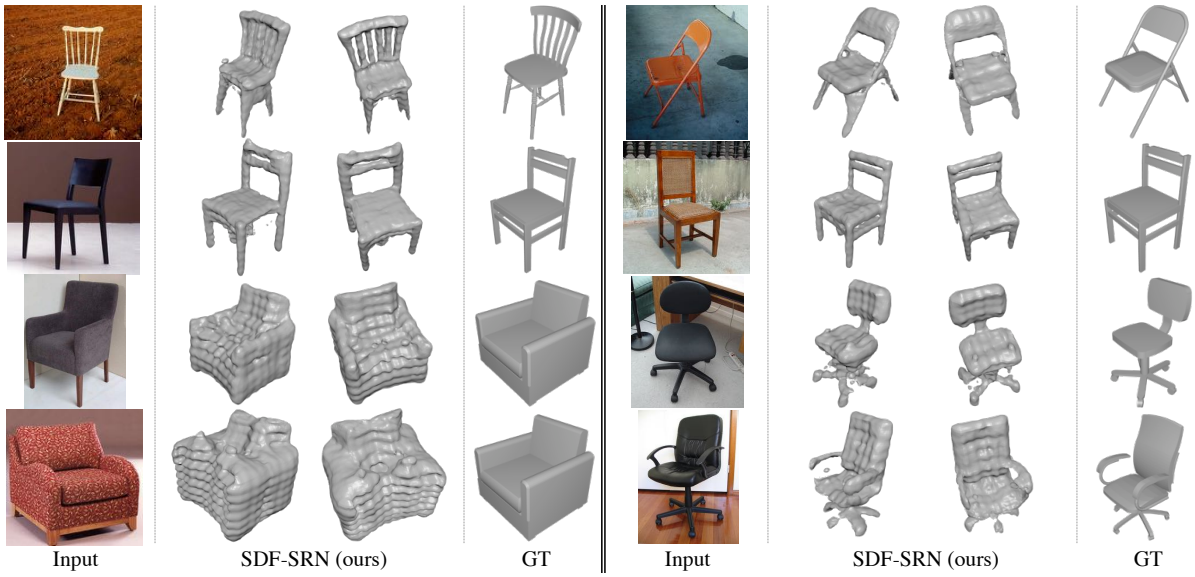


Figure 6.13: Additional results on PASCAL3D+ chairs.

Part III

3D Registration & Reconstruction

Chapter 7

Photometric Optimization for 3D Shape Alignment to Videos

7.1 Introduction

The choice of 3D representation plays a crucial role in 3D reconstruction problems from 2D images. Classical multi-view geometric methods, most notably structure from motion (S*f*M) and SLAM, recover point clouds as the underlying 3D structure of RGB sequences, often with very high accuracy [36, 128]. Point clouds, however, lack inherent 3D spatial structure that is essential for efficient reasoning. In many scenarios, *mesh representations* are more desirable – they are significantly more compact since they have inherent geometric structures defined by point connectivity, while they also represent continuous surfaces necessary for many applications such as robotics (*e.g.* accurate localization for autonomous driving), computer graphics (*e.g.* physical simulation, texture synthesis), and virtual/augmented reality.

Another drawback of classical multi-view geometric methods is reliance on hand-designed features and can be fragile when their assumptions are violated. This happens especially in textureless regions or when there are changes in illumination. Data-driven approaches [21, 52], on the other hand, learn priors to tackle ill-posed 3D reconstruction problems and have recently been widely applied to 3D prediction tasks from single images. However, they can only reliably reconstruct from the space of training examples it learns from, resulting in limited ability to generalize to unseen data.

In this chapter, we address the problem of 3D mesh reconstruction from image sequences by bringing together the best attributes of multi-view geometric methods and data-driven approaches (Fig. 7.1). Focusing on object instances, we use *shape priors* (specifically, neural networks) to reconstruct geometry with incomplete observations as well as *multi-view geometric constraints* to refine mesh predictions on the input sequences. Our approach allows dense reconstruction with object semantics from learned priors, which is not possible from the traditional pipelines of surface meshing [84] from multi-view stereo (MVS). Moreover, our approach generalizes to

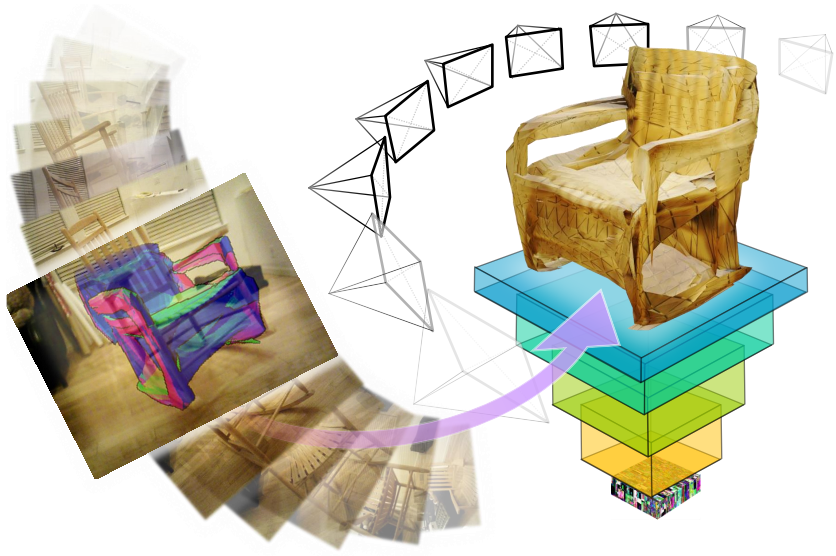


Figure 7.1: Our video-aligned object mesh reconstruction enforcing multi-view consistency while constraining shape deformations with shape priors, generating an output mesh with improved geometry with respect to the input views.

unseen objects by utilizing multi-view geometry to enforce consistency across viewpoints.

Given only RGB information, we achieve mesh reconstruction from image sequences by photometric optimization, which we pose as a piecewise image alignment problem of individual mesh faces. To avoid degeneracy, we introduce a novel virtual viewpoint rasterization to compute photometric gradients with respect to mesh vertices for 3D alignment, allowing the mesh to deform to the observed shape. A main advantage of our photometric mesh optimization is its non-reliance on any a-priori known depth or mask information [79, 177, 203] — a necessary condition to be able to reconstruct objects from real-world images. With this, we take a step toward practical usage of prior-based 3D mesh reconstruction aligned with RGB sequences.

In summary, we present the following contributions:

- We incorporate multi-view photometric consistency with data-driven shape priors for optimizing 3D meshes using 2D photometric cues.
- We propose a novel photometric optimization formulation for meshes and introduce a virtual viewpoint rasterization step to avoid gradient degeneracy.

Finally, we show 3D object mesh reconstruction results from both synthetic and real-world sequences, unachievable with either naïve mesh generators or traditional MVS pipelines without heavy manual post-processing.

7.2 Related Work

Our work on object mesh reconstruction touches several areas, including multi-view object reconstruction, mesh optimization, deep shape priors, and image alignment.

Multi-view object reconstruction. Multi-view calibration and reconstruction is a well-studied problem. Most approaches begin by estimating camera coordinates using 2D keypoint matching, a process known as SLAM [36, 127] or SfM [42, 153], followed by dense reconstruction methods such as MVS [43] and meshing [84]. More recent works using deep learning have explored 3D reconstruction from multiple-view consistency between various forms of 2D observations [101, 177, 178, 203, 222]. These methods all utilize forms of 2D supervision that are easier to acquire than 3D CAD models, which are relatively limited in quantity. Our approach uses both geometric and image-based constraints, which allows it to overcome common multi-view limitations such as missing observations and textureless regions.

Mesh optimization. Mesh optimization dates back to classical works of Active Shape Models [22] and Active Appearance Models [23, 120], which uses 2D meshes to fit facial landmarks. In this work, we optimize for 3D meshes using 2D photometric cues, a significantly more challenging problem due to the inherent ambiguities in the task. Similar approaches for mesh refinement have also been explored [29, 30]; however, a sufficiently good initialization is required with very small vertex perturbations allowed. As we show in our experiments, we are able to handle larger amount of noise perturbation by optimizing over a latent shape code instead of mesh vertices, making it more suitable for practical uses.

Several recent methods have addressed learning 3D reconstruction with mesh representations. AtlasNet [52] and Pixel2Mesh [185] are examples of learning mesh object reconstructions from 3D CAD models. Meanwhile, Neural Mesh Renderer [83] suggested a method of mesh reconstruction via approximate gradients for 2D mask optimization, and Kanazawa *et al.* [79] further advocated learning mesh reconstruction from 2D supervision of textures, masks, and 2D keypoints. Our approach, in contrast, does *not* assume any availability of masks or keypoints and operates purely via photometric cues across viewpoints.

Shape priors. The use of neural networks as object priors for reconstruction has recently been explored with point clouds [223]. However, it requires object masks as additional constraints during optimization. We eliminate the need for mask supervision by regularizing the latent code. Shape priors have also been explored for finding shape correspondences [51], where the network learns the deformation field from a template shape to match 3D observations. In our method, we directly optimize the latent shape code to match 2D cues from multiple viewpoints and do not require a known shape template for the object. A plane and primitive prior has been used for the challenging task of multi-view scene reconstruction [68]. Although the primitive prior does not need to be learned from an object dataset, the resulting reconstruction can differ significantly from the target geometry when it is not well represented by the chosen primitives.

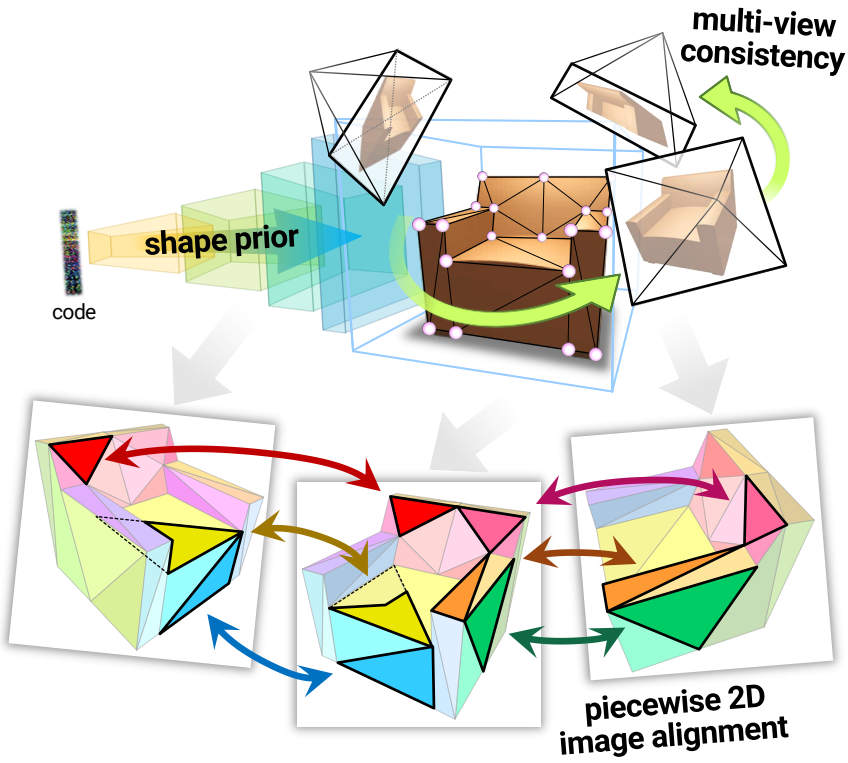


Figure 7.2: Overview. We perform 3D mesh reconstruction via *piecewise image alignment* of triangles to achieve per-triangle visibility-aware photometric consistency across multiple views, with mesh vertices optimized over the latent code of a shape prior learned by deep neural networks.

Image alignment. The most generic form of image alignment refers to prediction of inherent geometric misalignment between a pair of images. Image alignment using simple warping functions can be dated back to the seminal Lucas-Kanade algorithm [116] and its recent variants [9, 100]. Recent work has also explored learning a warp function to align images from neural networks for applications such as novel view synthesis [216, 217] and learning invariant representations [71, 99]. In this chapter, we pose our problem of mesh optimization as multiple image alignment problems of mesh faces, and solve it by optimizing over a latent code from a deep network rather than the vertices themselves.

7.3 Approach

We seek to reconstruct a 3D object mesh from an RGB sequence $\{(\mathcal{I}_f, \mathbf{\Omega}_f)\}$, where each frame $\mathcal{I}_f$ is associated with a camera matrix $\mathbf{\Omega}_f$. In this chapter, we assume that the camera matrices $\{\mathbf{\Omega}_f\}$ can be readily obtained from off-the-shelf S*f*M methods [153]. Fig. 7.2 provides an overview – we optimize for object meshes that maximize multi-view photometric consistency over a shape prior, where we use a pretrained mesh generator. We focus on triangular meshes here although our method is applicable to any mesh type.

7.3.1 Mesh Optimization over Shape Prior

Direct optimization on a 3D mesh $\mathcal{M}$ with N vertices involves solving for $3N$ degrees of freedom (DoFs) and typically becomes underconstrained when N is large. Therefore, reducing the allowed DoFs is crucial to ensure mesh deformations are well-behaved during optimization. We wish to represent the mesh $\mathcal{M} = \mathbf{G}(\mathbf{z})$ as a differentiable function $\mathbf{G}$ of a vector representation $\mathbf{z}$.

We propose to use an off-the-shelf generative neural network as the main part of $\mathbf{G}$ and reparameterize the mesh with an associated latent code $\mathbf{z} \in \mathbb{R}^K$, where $K \ll 3N$. The network serves as an object shape prior whose efficacy comes from pretraining on external shape datasets. Shape priors over point clouds have been previously explored [223]; here, we extend to mesh representations. We use AtlasNet [52] here although other mesh generators are also applicable. The shape prior allows the predicted 3D mesh to deform within a learned shape space, avoiding many local minima that exist with direct vertex optimization. To utilize RGB information from the given sequence for photometric optimization, we further add a 3D similarity transform to map the generated mesh to world cameras recovered by S*f*M (see Sec. 7.3.4).

We define our optimization problem as follows: given the RGB image sequence and cameras $\{(\mathcal{I}_f, \mathbf{\Omega}_f)\}$, we optimize a regularized cost consisting of a photometric loss $\mathcal{L}_{\text{photo}}$ for all pairs of frames over the representation $\mathbf{z}$, formulated as

$$\min_{\mathbf{z}} \sum_{a \neq b} \mathcal{L}_{\text{photo}}(\mathcal{I}_a, \mathcal{I}_b, \mathbf{\Omega}_a, \mathbf{\Omega}_b; \mathbf{G}(\mathbf{z})) + \mathcal{L}_{\text{reg}}(\mathbf{z}) \ , \tag{7.1}$$

where $\mathcal{L}_{\text{reg}}$ is a regularization term on $\mathbf{z}$. This objective allows the generated mesh to deform with respect to an effective shape prior. We describe each term in detail next.

7.3.2 Piecewise Image Alignment

Optimizing the mesh $\mathcal{M}$ with the photometric loss $\mathcal{L}_{\text{photo}}$ is based on the assumption that a dense 2D projection of the individual triangular faces of a 3D mesh should be globally consistent across multiple viewpoints. We cast the problem of 3D mesh alignment to the input views as a collection of *piecewise 2D image alignment* subproblems of each projected triangular face (Fig. 7.2).

To perform piecewise 2D image alignment between $\mathcal{I}_a$ and $\mathcal{I}_b$, we need to establish pixel correspondences. We first denote $\mathbf{V}_j(\mathbf{z}) \in \mathbb{R}^{3\times 3}$ as the 3D vertices of triangle j in mesh $\mathcal{M} = \mathbf{G}(\mathbf{z})$, defined as column vectors. From triangle j, we can sample a collection of 3D points $\mathbb{P}_j = \{\mathbf{p}_i(\mathbf{z})\}$ that lie within triangle j, related via $\mathbf{p}_i(\mathbf{z}) = \mathbf{V}_j(\mathbf{z})\boldsymbol{\alpha}_i$ through the barycentric coordinates $\boldsymbol{\alpha}_i$. For a camera $\mathbf{\Omega}$, let $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ be the projection function mapping a world 3D point $\mathbf{p}_i(\mathbf{z})$ to 2D image coordinates. The pixel intensity error between the two views $\mathbf{\Omega}_a$ and $\mathbf{\Omega}_b$ can be compared at the 2D image coordinates corresponding to the projected sampled 3D points. We formulate the photometric loss $\mathcal{L}_{\text{photo}}$ as the sum of ℓ_1 distances between pixel intensities at

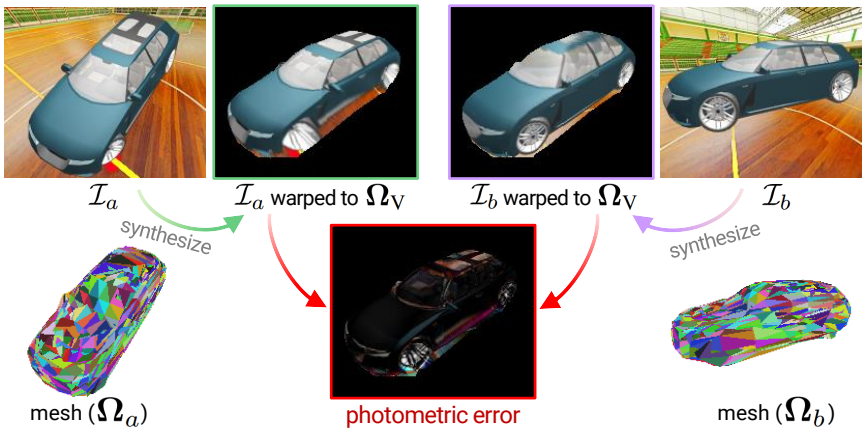


Figure 7.3: Visualization of the photometric loss $\mathcal{L}_\text{photo}$ between the synthesized appearances at virtual viewpoints $\mathbf{\Omega}_\mathrm{V}$ starting from input images $\mathcal{I}_a$ and $\mathcal{I}_b$. The photometric loss $\mathcal{L}_\text{photo}$ encourages consistent appearance syntheses from both input viewpoints $\mathbf{\Omega}_a$ and $\mathbf{\Omega}_b$.

these 2D image coordinates over all triangular faces,

$$\begin{aligned}&\mathcal{L}_\text{photo}(\mathcal{I}_a, \mathcal{I}_b, \mathbf{\Omega}_a, \mathbf{\Omega}_b; \mathbf{G}(\mathbf{z})) \\ =&\sum_j \sum_{i:\mathbf{p}_i \in \mathbb{P}_j} \left\| \mathcal{I}_a\left(\pi(\mathbf{p}_i(\mathbf{z}); \mathbf{\Omega}_a)\right) - \mathcal{I}_b\left(\pi(\mathbf{p}_i(\mathbf{z}); \mathbf{\Omega}_b)\right) \right\|_1 .\end{aligned} \tag{7.2}$$

As such, we can optimize the photometric loss $\mathcal{L}_\text{photo}$ with pixel correspondences established as a function of $\mathbf{z}$.

Visibility. As a 3D point $\mathbf{p}_i$ may not be visible in a given view due to possible object self-occlusion, we handle visibility by constraining $\mathbb{P}_j$ to be the set of samples in triangle j whose projection is visible in both views. We achieve this by returning a mesh index map using mesh rasterization, a standard operation in computer graphics, for each optimization step. The photometric gradients of each sampled point $\frac{\partial \mathcal{I}}{\partial \mathbf{V}_j} = \frac{\partial \mathcal{I}}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial \mathbf{V}_j}$ in turn backpropagate to the vertices $\mathbf{V}_j$. We obtain $\frac{\partial \mathcal{I}}{\partial \mathbf{x}_i}$ through differentiable image sampling [71], $\frac{\partial \mathbf{x}_i}{\partial \mathbf{p}_i}$ by taking the derivative of the projection π, and $\frac{\partial \mathbf{p}_i}{\partial \mathbf{V}_j}$ by associating with the barycentric coordinates $\boldsymbol{\alpha}_i$. We note that the entire process is differentiable and does not resort to approximate gradients [83].

7.3.3 Virtual Viewpoint Rasterization

We can efficiently sample a large number of 3D points $\mathbb{P}_j$ in triangle j by rendering the depth of $\mathcal{M}$ from a given view using mesh rasterization (Sec. 7.3.2). If the depth were rasterized from either input view $\mathbf{\Omega}_a$ or $\mathbf{\Omega}_b$, however, we would obtain zero photometric gradients. This degeneracy arises due to the fact that ray-casting from one view and projecting back to the same view results in $\frac{\partial \mathcal{I}}{\partial \mathbf{V}_j} = \mathbf{0}$.

To elaborate, we first note that depth rasterization of triangle j is equivalent to back-projecting regular grid coordinates $\bar{\mathbf{x}}_i$ to triangle j. We can express each depth point from camera $\boldsymbol{\Omega} \in \{\boldsymbol{\Omega}_a, \boldsymbol{\Omega}_b\}$ as $\mathbf{p}_i(\mathbf{z}) = \pi^{-1}(\bar{\mathbf{x}}_i; \mathbf{V}_j(\mathbf{z}), \boldsymbol{\Omega})$, where $\pi^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ is the inverse projection function realized by solving for ray-triangle intersection with $\mathbf{V}_j(\mathbf{z})$. Combining with the projection equation, we have

$$\mathbf{x}_i(\mathbf{z}, \boldsymbol{\Omega}) = \pi(\pi^{-1}(\bar{\mathbf{x}}_i; \mathbf{V}_j(\mathbf{z}), \boldsymbol{\Omega}); \boldsymbol{\Omega}) = \bar{\mathbf{x}}_i \quad \forall \bar{\mathbf{x}}_i \ , \tag{7.3}$$

becoming the identity mapping and losing the dependency of $\mathbf{x}_i$ on $\mathbf{V}_j(\mathbf{z})$, which in turn leads to $\frac{\partial \mathbf{x}_i}{\partial \mathbf{V}_j} = \mathbf{0}$. This insight is in line with the recent observation from Ham *et al.* [55].

To overcome this degeneracy, we rasterize the depth from a *third* virtual viewpoint $\boldsymbol{\Omega}_\text{V} \notin \{\boldsymbol{\Omega}_a, \boldsymbol{\Omega}_b\}$. This step allows correct gradients to be computed in both viewpoints $\boldsymbol{\Omega}_a$ and $\boldsymbol{\Omega}_b$, which is essential to maintain stability during optimization. We can form the photometric loss by synthesizing the image appearance at $\boldsymbol{\Omega}_\text{V}$ using the pixel intensities from both $\boldsymbol{\Omega}_a$ and $\boldsymbol{\Omega}_b$ (Fig. 7.3). We note that $\boldsymbol{\Omega}_\text{V}$ can be arbitrarily chosen. In practice, we choose $\boldsymbol{\Omega}_\text{V}$ to be the bisection between $\boldsymbol{\Omega}_a$ and $\boldsymbol{\Omega}_b$ by applying Slerp [156] on the rotation quaternions and averaging the two camera centers.

7.3.4 Implementation Details

Coordinate systems. Mesh predictions from a generative network typically lie in a canonical coordinate system [52, 185] independent of the world cameras recovered by S*f*M. Therefore, we need to account for an additional 3D similarity transform $\mathcal{T}(\cdot)$ applied to the mesh vertices. For each 3D vertex $\mathbf{v}'_k$ from the prediction, we define the similarity transform as

$$\mathbf{v}_k = \mathcal{T}(\mathbf{v}'_k; \boldsymbol{\theta}) = \exp(s) \cdot \mathbf{R}(\boldsymbol{\omega})\mathbf{v}'_k + \mathbf{t} \quad \forall k \ , \tag{7.4}$$

where $\boldsymbol{\theta} = [s; \boldsymbol{\omega}; \mathbf{t}] \in \mathbb{R}^7$ are the parameters and $\mathbf{R}$ is a 3D rotation matrix parameterized with the $\mathfrak{so}(3)$ Lie algebra. We optimize for $\mathbf{z} = [\mathbf{z}'; \boldsymbol{\theta}]$ together, where $\mathbf{z}'$ is the latent code associated with the generative network.

Since automated registration of noisy 3D data with unknown scales is still an open problem, we assume a coarse alignment of the coordinate systems can be computed from minimal annotation of rough correspondences (see Sec. 7.4.3 for details). We optimize for the similarity transform to more accurately align the meshes to the RGB sequences.

Regularization. Despite neural networks being effective priors, the latent space is only spanned by the training data. To avoid meshes from reaching a degenerate solution, we impose an extra penalty on the latent code $\mathbf{z}'$ to ensure it stays within a trust region of the initial code $\mathbf{z}_0$ (extracted from a pretrained image encoder), defined as $\mathcal{L}_\text{code} = \|\mathbf{z}' - \mathbf{z}_0\|_2^2$. We also add a scale penalty $\mathcal{L}_\text{scale} = -s$ that encourages the mesh to expand, since the mesh shrinking to infinitesimal is a trivial solution with zero photometric error. The regularization $\mathcal{L}_\text{reg}$ in cost (7.1) is written as

$$\mathcal{L}_\text{reg}(\mathbf{z}) = \lambda_\text{code} \cdot \mathcal{L}_\text{code}(\mathbf{z}') + \lambda_\text{scale} \cdot \mathcal{L}_\text{scale}(\boldsymbol{\theta}) \tag{7.5}$$

Figure 7.4: Sample sequences composited from ShapeNet renderings (top: car, bottom: airplane) and SUN360 scenes.

where λ_{code} and λ_{scale} are the penalty weights.

7.4 Experiments

We evaluate the performance of our method on a single (Sec. 7.4.1) and multiple (Sec. 7.4.2) object categories with synthetic data as well as real-world videos (Sec. 7.4.3).

Data preparation. We create datasets of 3D CAD model renderings for training a mesh generation network and evaluating our optimization framework. Our rendering pipeline aims to create realistic images with complex backgrounds so they could be applied to real-world video sequences. We use ShapeNet [16] for the object dataset and normalize all objects to fit an origin-centered unit sphere. We render RGB images of each object using perspective cameras at 24 equally spaced azimuth angles and 3 elevation angles.

To simulate realistic backgrounds, we randomly warp and crop spherical images from the SUN360 database [198] to create background images of the same scene taken at different camera viewpoints. By compositing the foreground and background images together at corresponding camera poses, we obtain RGB sequences of objects composited on realistic textured backgrounds (Fig. 7.4). Note that we do not keep any mask information that was accessible in the rendering and compositing process as such information is typically not available in real-world examples. All images are rendered/cropped at a resolution of 224×224.

Shape prior. We use AtlasNet [52] as the base network architecture for mesh generation, which we retrain on our new dataset. We use the same 80%-20% training/test split from Groueix *et al.* [52] and additionally split the SUN360 spherical images with the same ratio. During training, we augment background images at random azimuth angles.

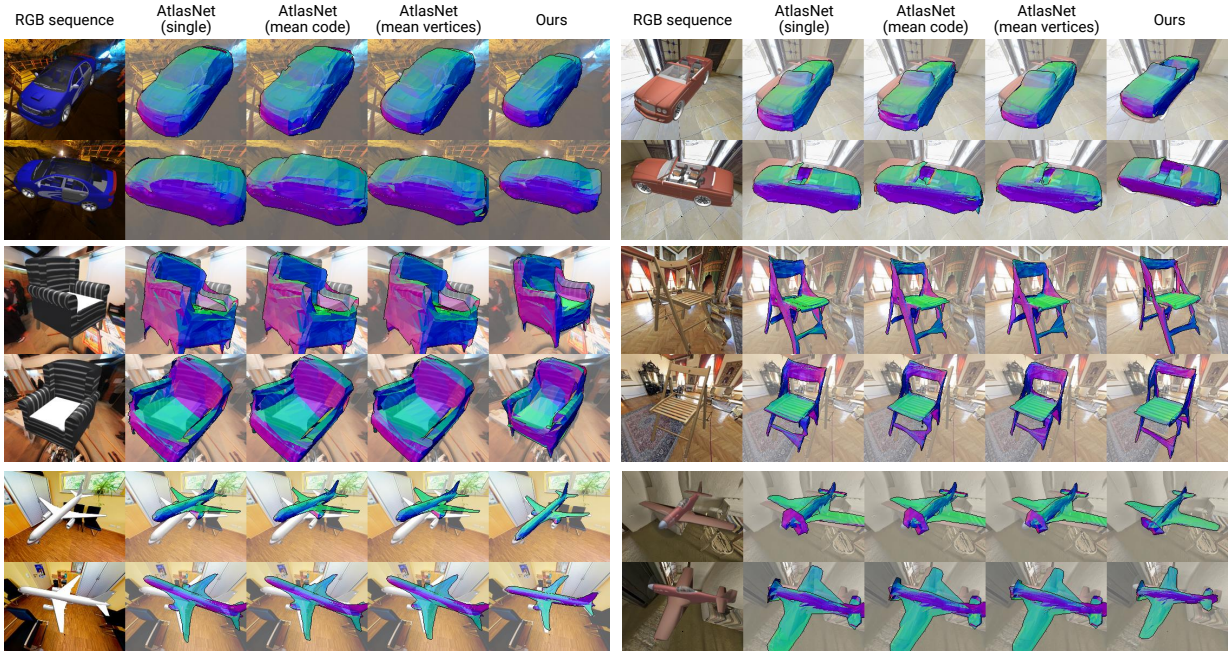


Figure 7.5: Qualitative results from category-specific models, where we visualize two sample frames from each test sequence. Our method better aligns initial meshes to the RGB sequences while optimizing for more subtle shape details (*e.g.* car spoilers and airplane wings) over baselines. The meshes are color-coded by surface normals with occlusion boundaries drawn.

Initialization. We initialize the code $\mathbf{z}_0$ by encoding an RGB frame with the AtlasNet encoder. For ShapeNet sequences, we choose frames with objects facing 45° sideways. For real-world sequences, we manually select frames where objects are center-aligned to the images as much as possible to match our rendering settings. We initialize the similarity transform parameters to $\boldsymbol{\theta} = \mathbf{0}$ (identity transform).

Evaluation criteria. We evaluate the result by measuring the 3D distances between the sampled 3D points from the predicted meshes and the ground-truth point clouds [52]. We follow Lin *et al.* [101] by reporting the 3D error between the predicted and ground-truth point clouds as $\eta(\mathcal{S}_1, \mathcal{S}_2) = \sum_{i:\mathbf{v}_i \in \mathcal{S}_1} \min_{\mathbf{v}_j \in \mathcal{S}_2} \|\mathbf{v}_i - \mathbf{v}_j\|_2$ for some source and target point sets $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. This metric measures the prediction shape accuracy when $\mathcal{S}_1$ is the prediction and $\mathcal{S}_2$ is the ground truth, while it indicates the prediction shape coverage when vice versa. We report quantitative results in both directions separately averaged across all instances.

7.4.1 Single Object Category

We start by evaluating our mesh alignment in a category-specific setting. We select the car, chair, and plane categories from ShapeNet, consisting of 703, 1356, and 809 objects in our test split, respectively. For each object, we create an RGB sequence by overlaying its rendering

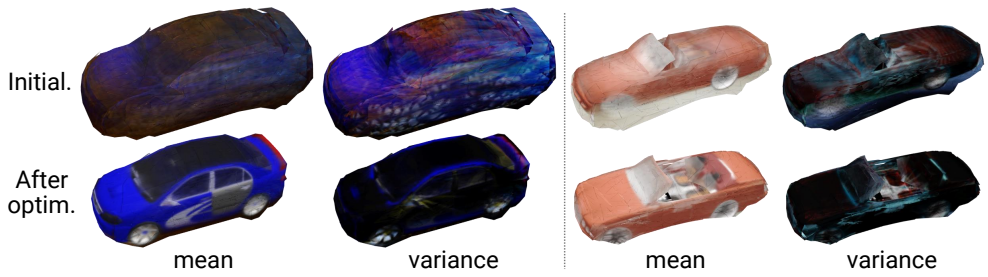


Figure 7.6: Mesh visualization with textures computed by averaging projections across all viewpoints. Our method successfully reduces variance and recovers dense textures that can be embedded on the surfaces.

onto a randomly paired SUN360 scene with the cameras in correspondence. We retrain each category-specific AtlasNet model on our new dataset using the default settings for 500 epochs. During optimization, we use the Adam optimizer [85] with a constant learning rate of 0.003 for 100 iterations. We manually set the penalty factors to be $\lambda_{\text{code}} = 0.05$ and $\lambda_{\text{scale}} = 0.02$.

One challenge is that the coordinate system for a mesh generated by AtlasNet is independent of the recovered world cameras $\{\mathbf{\Omega}_f\}$ for a real-world sequence. Determining such coordinate system mapping (defined by a 3D similarity transform) is required to relate the predicted mesh to the world. On the other hand, for the synthetic sequences, we know the exact mapping as we can render the views for AtlasNet and the input views $\{\mathcal{I}_f\}$ in the same coordinate system.

For our first experiment, we simulate the possibly incorrect mapping estimates by perturbing the ground-truth 3D similarity transform by adding Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma\mathbf{I})$ to its parameters, pre-generated per sequence for evaluation. We evaluate the 3D error metrics under such perturbations. Note that our method utilizes no additional information other than the RGB information from the given sequences.

We compare our mesh reconstruction approach against three baseline variants of AtlasNet: (a) mesh generations from a single-image feed-forward initialization, (b) generation from the mean latent code averaged over all frames in the sequence, and (c) the mean shape where vertices are averaged from the mesh generation across all frames.

We show qualitative results in Fig. 7.5 (compared under perturbation $\sigma = 0.12$). Our method is able to take advantage of multi-view geometry to resolve large misalignments and optimize for more accurate shapes. The high photometric error from the background between views discourages mesh vertices from staying in such regions. This error serves as a natural force to constrain the mesh within the desired 3D regions, eliminating the need of depth or mask constraints during optimization. We further visualize our mesh reconstruction with textures that are estimated from all images (Fig. 7.6). Note that the fidelity of mean textures increases while variance in textures decrease after optimization.

We evaluate quantitatively in Fig. 7.7, where we plot the average 3D error over mapping noise. This result demonstrates how our method handles inaccurate coordinate system mappings to successfully match the meshes against RGB sequences. We also ablate optimizing the latent

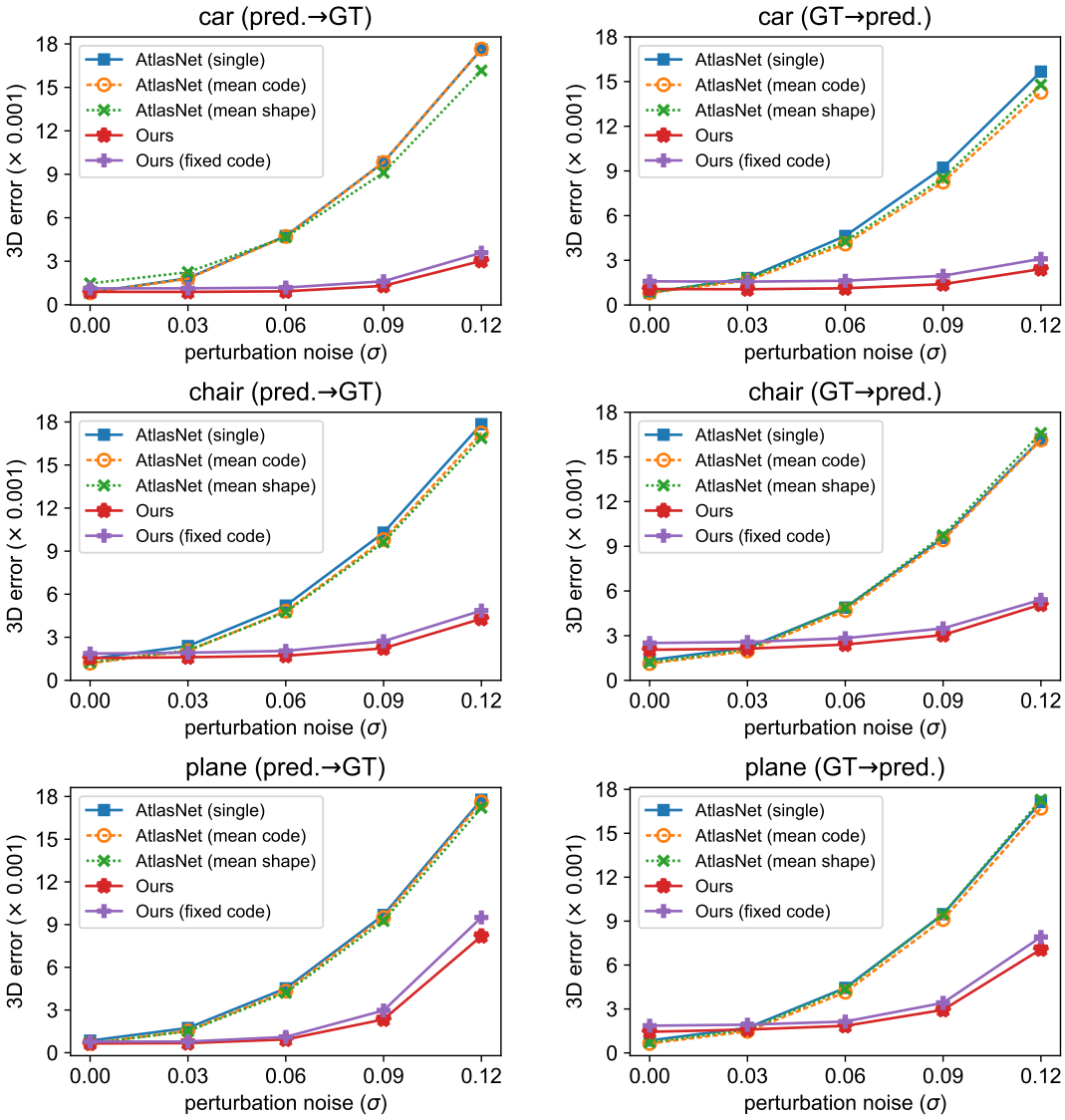


Figure 7.7: Category-specific performance to noise in coordinate system mapping. Our method is able to resolve for various extents of mesh misalignments from the sequence.

code $\mathbf{z}$, showing that allowing shape deformation improves reconstruction quality over a sole 3D similarity transform ("fixed code" in Fig. 7.7). Note that our method is slightly worse in shape coverage error (GT→pred.) when evaluated at the ground-truth mapping. This result is attributed to the limitation of photometric optimization that opts for degenerate solutions when objects are insufficiently textured.

Category	plane	bench	cabin.	car	chair	monit.	lamp	speak.	fire.	couch	table	cell.	water.	**mean**
AtlasNet (single)	3.872	4.931	5.708	4.269	4.869	4.687	8.684	7.245	3.864	5.017	4.964	4.571	4.290	5.152
AtlasNet (mean code)	3.746	4.496	5.600	4.286	4.571	4.634	7.366	6.976	3.632	4.798	4.903	4.286	3.860	4.858
AtlasNet (mean shape)	3.659	4.412	5.382	4.192	4.499	4.424	**7.200**	6.683	3.547	4.606	4.860	4.196	3.742	4.723
Ours	**0.704**	**1.821**	**2.850**	**0.597**	**1.441**	**1.115**	8.855	**3.430**	**1.255**	**0.983**	**1.725**	**1.599**	**1.743**	**2.163**

(a) 3D error: prediction → ground truth (shape accuracy).

Category	plane	bench	cabin.	car	chair	monit.	lamp	speak.	fire.	couch	table	cell.	water.	**mean**
AtlasNet (single)	4.430	4.895	5.024	4.461	4.896	4.640	8.906	6.994	4.407	4.613	5.350	4.254	4.263	5.164
AtlasNet (mean code)	4.177	4.507	4.962	4.384	4.635	4.143	**7.292**	6.990	4.307	4.463	**5.084**	4.036	3.718	4.823
AtlasNet (mean shape)	4.464	4.915	5.150	4.521	4.940	4.560	8.159	7.308	4.528	4.707	5.255	4.299	4.183	5.153
Ours	**2.237**	**3.215**	**1.927**	**0.734**	**2.377**	**2.119**	10.764	**4.152**	**2.583**	**1.735**	6.126	**1.851**	**2.926**	**3.288**

(b) 3D error: ground truth → prediction (shape coverage).

Table 7.1: Average 3D test error for general object categories (numbers scaled by 10^3). The mean is taken across categories. Our optimization method is effective on most object categories. Note that our method improves on accuracy of the table category despite worsening in shape coverage due to insufficient textures in object samples.

7.4.2 Multiple Object Categories

We extend beyond a model that reconstructs a single object category by training a single model to reconstruct multiple object categories. We take 13 commonly chosen CAD model categories from ShapeNet [21, 40, 52, 101]. We follow the same settings as in Sec. 7.4.1 except we retrain AtlasNet longer for 1000 epochs due to a larger training set.

We show visual results in Fig. 7.8 on the efficacy of our method for multiple object categories (under perturbation $\sigma = 0.12$). Our results show how we can reconstruct a shape that better matches our RGB observations (e.g., refining hollow regions, as in the bench backs and table legs). We also show category-wise quantitative results in Table 7.1, compared under perturbation noise $\sigma = 0.06$. We find photometric optimization to perform effectively across most categories except lamps, which consist of many examples where optimizing for thin structures is hard for photometric loss.

7.4.3 Real-world Videos

Finally, we demonstrate the efficacy of our method on challenging real-world video sequences orbiting an object. We use a dataset of RGB-D object scans [20], where we use the chair model to evaluate on the chair category. We select the subset of video sequences that are 3D-reconstructible using traditional pipelines [153] and where S*f*M extracts at least 20 reliable frames and 100 salient 3D points. We retain 82 sequences with sufficient quality for evaluation. We rescale the sequences to 240×320 and skip every 10 frames.

We compute the camera extrinsic and intrinsic matrices using off-the-shelf S*f*M with COLMAP [153]. For evaluation, we additionally compute a rough estimate of the coordinate system mapping by

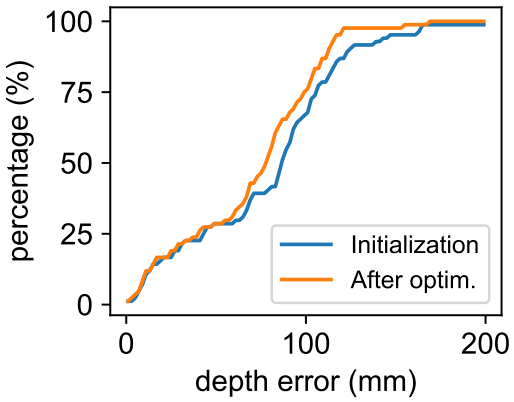


Figure 7.11: Metric-scale depth error before and after optimization (with S*f*M world cameras rescaled).

Dist.	Initial.	Optim.
1	6.504	4.990
2	9.064	6.979
3	10.984	8.528
4	12.479	9.788
6	14.718	11.665

Table 7.2: Average pixel reprojection error (scaled by 100) from real-world videos as a function of frame distances.

annotating 3 corresponding points between the predicted mesh and the sparse points extracted from S*f*M (Fig. 7.9), which allows us to fit a 3D similarity transform. We optimize using Adam with a learning rate of 2e-3 for 200 iterations, and we manually set the penalty factors to be $\lambda_{\text{code}} = 0.05$ and $\lambda_{\text{scale}} = 0.01$.

We demonstrate how our method is applicable to real-world datasets in Fig. 7.10. Our method is able to refine shapes such as armrests and office chair legs. Note that our method is sensitive to the quality of mesh initialization from real images, mainly due to the domain mismatch between synthetic and real data during the training/test phases of the shape prior. Despite this, it is still able to straighten and align to the desired 3D location. In addition, we report the average pixel reprojection error in Table 7.2 and metric depth error in Fig. 7.11 to quantify the effect of photometric optimization, which shows further improvement over coarse initializations.

Finally, we note that surface reconstruction is a challenging post-processing procedure for traditional pipelines. Fig. 7.10 shows sample results for S*f*M [153], PatchMatch Stereo [12], stereo fusion, and Poisson mesh reconstruction [84] from COLMAP [153]. In addition to the need of accurate object segmentation, the dense meshing problem with traditional pipelines typically yields noisy results without laborious manual post-processing.

7.5 Conclusion

We have demonstrated a method for reconstructing a 3D mesh from an RGB video by combining data-driven deep shape priors with multi-view photometric consistency optimization. We also show that mesh rasterization from a virtual viewpoint is critical for avoiding degenerate photometric gradients during optimization. We believe our photometric mesh optimization technique has merit for a number of practical applications. It enables the ability to generate more accurate models of real-world objects for computer graphics and potentially allows automated object segmentation from video data. It could also benefit 3D localization for robot navigation and autonomous driving,

where accurate object location, orientation, and shape from real-world cameras is crucial for more efficient understanding.

7.A Appendix

7.A.1 Architectural and Pretraining Details

We use AtlasNet [52] as the base network architecture for our experiments. Following Groueix *et al.* [52], the image encoder is the ResNet-18 [59] architecture where the last fully-connected layer is replaced with one with an output dimension of 1024, which is the size of the latent code. We use the 25-patch version of the AtlasNet mesh decoder, where each deformable patch is an open triangular mesh with $5^2 \times 2 = 50$ triangles on a 5×5 regular grid. We redirect the readers to Groueix *et al.* [52] for more details.

In the stage of pretraining AtlasNet on ShapeNet [16] with textured background from SUN360 [198], we train all networks using the Adam optimizer [85] with a constant learning rate of 10^{-4}. We set the batch size for all experiments to be 32. We initialize the AtlasNet encoder with the pretrained ResNet-18 on ImageNet [152] except for the last modified layer (before the latent code), and we initialize the decoder with that pretrained from a point cloud autoencoder from Groueix *et al.* [52].

7.A.2 Warp Parameterization Details

We parameterize the rotation component of 3D similarity transformations with the $\mathfrak{so}(3)$ Lie algebra. Given a warp parameter vector $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^\top \in \mathfrak{so}(3)$, the rotation matrix $\mathbf{R}(\boldsymbol{\omega}) \in \mathbb{SO}(3)$ can be written as

$$\mathbf{R}(\boldsymbol{\omega}) = \exp\left(\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}\right) , \tag{7.6}$$

where $\exp$ is the exponential map (*i.e.* matrix exponential). $\mathbf{R}$ is the identity transformation when $\boldsymbol{\omega}$ is an all-zeros vector. The exponential map is Taylor-expandable as

$$\mathbf{R}(\boldsymbol{\omega}) = \exp(\boldsymbol{\omega}_\times) = \lim_{K\to\infty} \sum_{k=0}^{K} \frac{\boldsymbol{\omega}_\times^k}{k!} . \tag{7.7}$$

We implement the $\mathfrak{so}(3)$ parameterization using the Taylor approximation expression with $K = 20$. We have also tried parametrizing the 3D similarity transformations with the self-contained Lie group $\mathrm{Sim}(3)$, where the scale is incorporated into the exponential map; we find it to yield almost identical results. We also take the exponential on the scale s to ensure positivityl; the resulting scale does not change when $s = 0$.

7.A.3 SUN360 Background Data Generation

The background images from SUN360 [198] are cropped from spherical images with a resolution of 1024×512, using a field of view of 90°. Fig. 7.12 illustrates an example of the original spherical image and its generated crops.

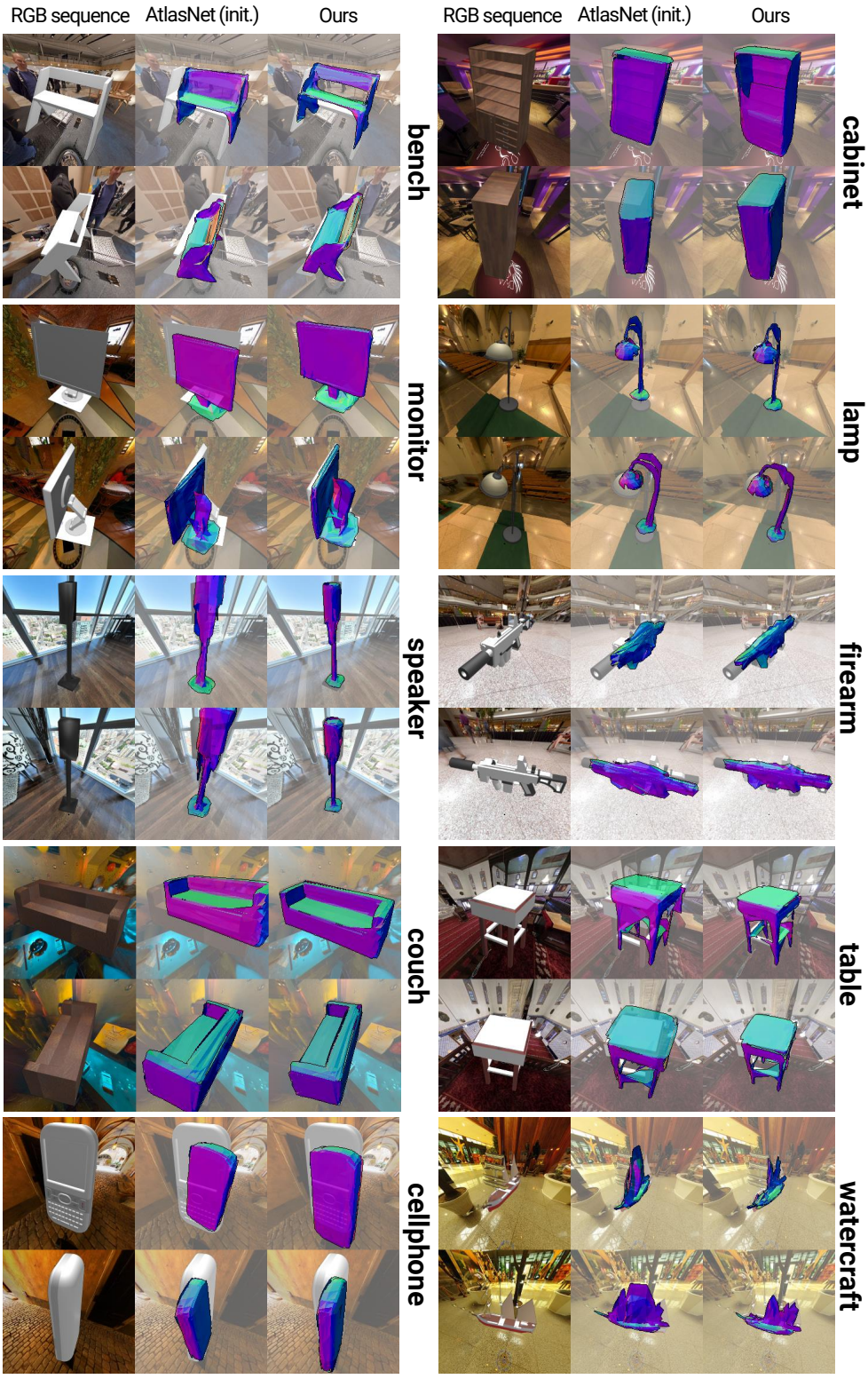


Figure 7.8: Qualitative results for general object categories. Our optimization method recovers subtle details such as back of benches, watercraft sails, and even starts to reveal cabinet open spaces which were initially occluded. Our method tends to fail more frequently with textureless objects (*e.g.* cellphone and firearm).

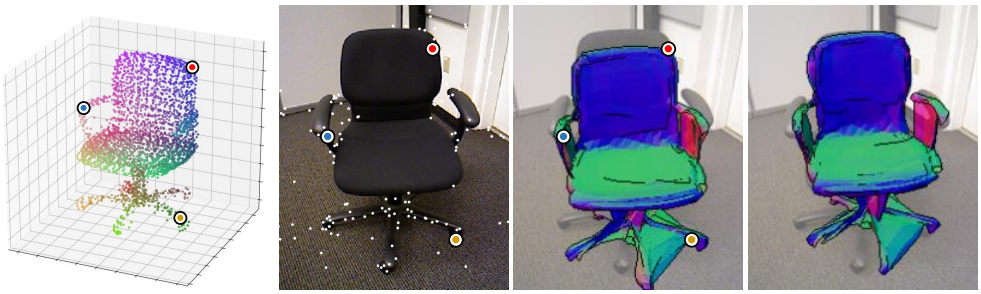


Figure 7.9: We select 3 correspondences between (a) the mesh vertices and (b) the S*f*M points to find (c) an estimated coordinate system mapping by fitting a 3D similarity transform. (d) Alignment result after our photometric optimization.

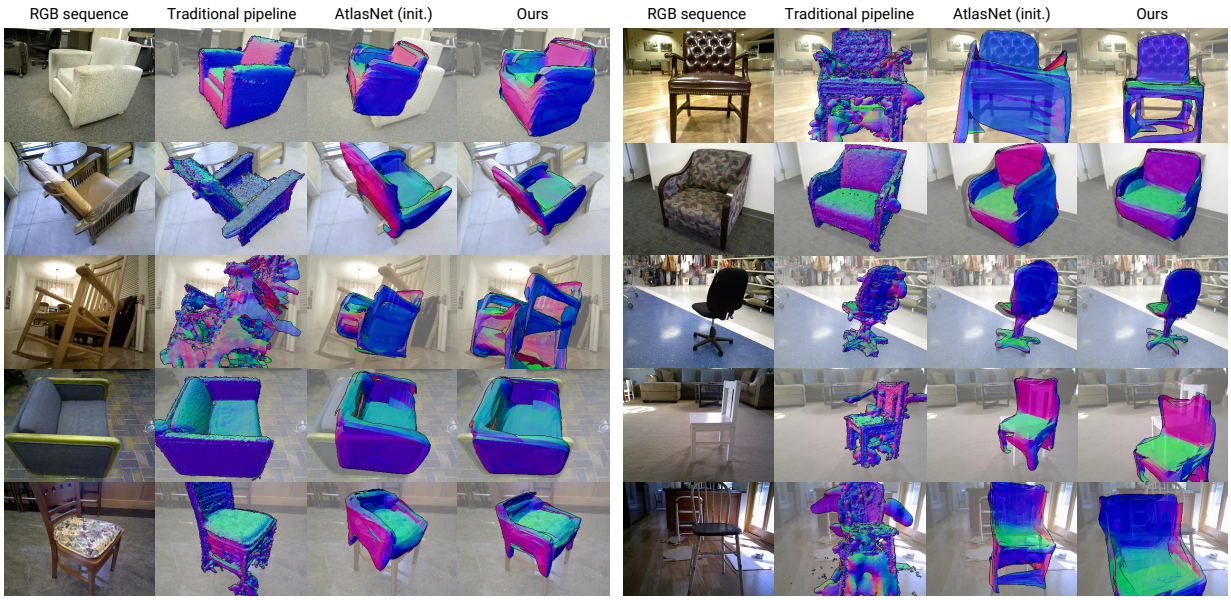


Figure 7.10: Qualitative results on real-world sequences. Given an initialization, our method accurately aligns a generated mesh to an RGB video. Even when the initial mesh is an inaccurate prediction of the real object, our method is still able to align the semantic parts (bottom left). We show failure cases in the last two examples in the bottom right, where there is insufficient background texture as photometric cues and where the initial mesh is insufficient to capture the thin structures. We also show the result of a traditional reconstruction pipeline [153] after manual cleanup. Due to the difficulty of the problem these meshes still often have undesirable artifacts.

(a)

(b)

Figure 7.12: (a) Example panoramic (spherical) image and (b) sample cropped images at different camera viewpoints.

Chapter 8

Beyond Shape Priors: Bundle-Adjusting Neural Radiance Fields

8.1 Introduction

Humans have strong capabilities of reasoning about 3D geometry through our vision from the slightest ego-motion. When watching movies, we can immediately infer the 3D spatial structures of objects and scenes inside the videos. This is because we have an inherent ability of associating spatial correspondences of the same scene across continuous observations, without having to make sense of the relative camera or ego-motion. Through pure visual perception, not only can we recover a mental 3D representation of *what* we are looking at, but meanwhile we can also recognize *where* we are looking at the scene from.

Simultaneously solving for the 3D scene representation from RGB images (*i.e.* **reconstruction**) and localizing the given camera frames (*i.e.* **registration**) is a long-standing chicken-and-egg problem in computer vision — recovering the 3D structure requires observations with known camera poses, while localizing the cameras requires reliable correspondences from the reconstruction. Classical methods such as structure from motion (S*f*M) [58, 153] or SLAM [36, 127] approach this problem through local registration followed by global geometric bundle adjustment (BA) on both the structure and cameras. S*f*M and SLAM systems, however, are sensitive to the quality of local registration and easily fall into suboptimal solutions. In addition, the sparse nature of output 3D point clouds (often noisy) limits downstream tasks that requires dense geometric reasoning.

Closely related to 3D reconstruction from imagery is the problem of view synthesis. Though not primarily purposed for recovering explicit 3D structures, recent advances on photorealistic view synthesis have opted to recover an intermediate dense 3D-aware representation (*e.g.* depth [41, 191], multi-plane images [166, 176, 218], or volume density [112, 126]), followed by neural rendering techniques [38, 124, 161, 174] to synthesize the target images. In particular, Neural Radiance Fields (NeRF) [126] have demonstrated its remarkable ability for high-fidelity view synthesis. NeRF encodes 3D scenes with a neural network mapping 3D point locations to color

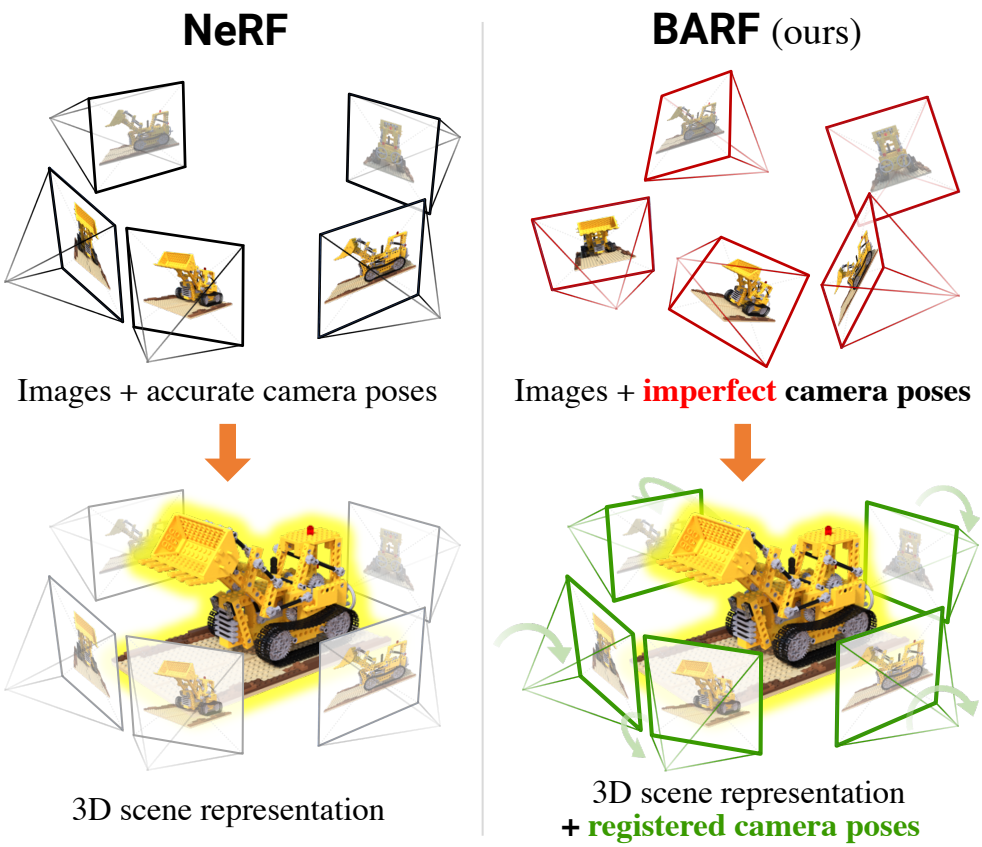


Figure 8.1: Training NeRF requires accurate camera poses for all images. We present **BARF** for learning 3D scene representations from *imperfect* (or even *unknown*) camera poses by jointly optimizing for registration and reconstruction.

and volume density. This allows the scenes to be represented with compact memory footprint without limiting the resolution of synthesized images. The optimization process of the network is constrained to obey the principles of classical volume rendering [96], making the learned representation interpretable as a continuous 3D volume density function.

Despite its notable ability for photorealistic view synthesis and 3D scene representation, a hard prerequisite of NeRF (as well as other view synthesis methods) is accurate camera poses of the given images, which is typically obtained through auxiliary off-the-shelf algorithms. One straightforward way to circumvent this limitation is to additionally optimize the pose parameters with the NeRF model via backpropagation. As later discussed, however, naïve pose optimization with NeRF is sensitive to initialization. It may lead to suboptimal solutions of the 3D scene representation, degrading the quality of view synthesis.

In this chapter, we address the problem of training NeRF representations from imperfect camera poses — the joint problem of *reconstructing* the 3D scene and *registering* the camera poses (Fig. 8.1). We draw inspiration from the success of classical image alignment methods and establish a theoretical connection, showing that coarse-to-fine registration is also critical to NeRF. Specifically, we show that positional encoding [180] of input 3D points plays a crucial

role — as much as it enables fitting to high-frequency functions [170], positional encoding is also more susceptible to suboptimal registration results. To this end, we present Bundle-Adjusting NeRF (BARF), a simple yet effective strategy for coarse-to-fine registration on coordinate-based scene representations. BARF can be regarded as a type of *photometric* BA [4, 103] using view synthesis as the proxy objective. Unlike traditional BA, however, BARF can learn the scene representation *from scratch* (*i.e.* from randomly initialized network weights), lifting the reliance of local registration subprocedures and allowing for more generic applications.

In summary, we present the following contributions:

- We establish a theoretical connection between classical image alignment to joint registration and reconstruction with Neural Radiance Fields (NeRF).
- We show that susceptibility to noise from positional encoding affects the basin of attraction for registration, and we present a simple strategy for coarse-to-fine registration on coordinate-based scene representations.
- Our proposed BARF can successfully recover scene representations from imperfect camera poses, allowing for applications such as view synthesis and localization of video sequences from unknown poses.

8.2 Related Work

Structure from motion (S*f*M) and SLAM. Given a set of input images, S*f*M [2, 163, 164, 192] and SLAM [36, 127, 128, 204] systems aim to recover the 3D structure and the sensor poses simultaneously. These can be classified into (a) *indirect* methods that rely on keypoint detection and matching [27, 127] and (b) *direct* methods that exploit photometric consistency [4, 37]. Modern pipelines following the indirect route have achieved tremendous success [153]; however, they often suffer at textureless regions and repetitive patterns, where distinctive keypoints cannot be reliably detected. Researchers have thus sought to use neural networks to learn discriminative features directly from data [33, 35, 134].

Direct methods, on the other hand, do not rely on such distinctive keypoints — every pixel can contribute to maximizing photometric consistency, leading to improved robustness in sparsely textured environments [187]. They can also be naturally integrated into deep learning frameworks through image reconstruction losses [182, 209, 217]. Our method BARF lies under the broad umbrella of direct methods, as BARF learns 3D scene representations from RGB images while also localizing the respective cameras. However, unlike classical S*f*M and SLAM that represent 3D structures with explicit geometry (*e.g.* point clouds), BARF encodes the scenes as coordinate-based representations with neural networks.

View synthesis. Given a set of posed images, view synthesis attempts to simulate how a scene would look like from novel viewpoints [18, 63, 97, 169]. The task has been closely tied to 3D reconstruction since its introduction [28, 61, 224]. Researchers have investigated blending

pixel colors based on depth maps [17] or leveraging proxy geometry to warp and composite the synthesized image [89]. However, since the problem is inherently ill-posed, there are still multiple restrictions and assumptions on the synthesized viewpoints.

State-of-the-art methods have capitalized on neural networks to learn both the scene geometry and statistical priors from data. Various representations have been explored in this direction, *e.g.* depth [41, 149, 150, 191], layered depth [155, 179], multi-plane images [166, 176, 218], volume density [112, 126], and mesh sheets [66]. Unfortunately, these view synthesis methods still require the camera poses to be known *a priori*, largely limiting their applications in practice. In contrast, our method BARF is able to effectively learn 3D representations that encodes the underlying scene geometry from imperfect or even unknown camera poses.

Neural Radiance Fields (NeRF). Recently, Mildenhall *et al.* [126] proposed NeRF to synthesize novel views of static, complex scenes from a set of posed input images. The key idea is to model the continuous radiance field of a scene with a multi-layer perceptron (MLP), followed by differentiable volume rendering to synthesize the images and backpropagate the photometric errors. NeRF has drawn wide attention across the vision community [130, 138, 144, 208, 212] due to its simplicity and extraordinary performance. It has also been extended on many fronts, *e.g.* reflectance modeling for photorealistic relighting [14, 167] and dynamic scene modeling that integrates the motion of the world [98, 141, 196]. Recent works have also sought to exploit a large corpus of data to pretrain the MLP, enabling the ability to infer the radiance field from a single image [46, 146, 154, 210].

While impressive results have been achieved by the above NeRF-based models, they have a common drawback — the requirement of *posed* images. Our proposed BARF allows us to circumvent such requirement. We show that with a simple coarse-to-fine bundle adjustment technique, we can recover from imperfect camera poses (including *unknown* poses of video sequences) and learn the NeRF representation simultaneously. Concurrent to our work, NeRF--[190] introduced an empirical, two-stage pipeline to estimate unknown camera poses. Our method BARF, in contrast, is motivated by mathematical insights and can recover the camera poses within a single course of optimization, allowing for direct utilities for various NeRF extensions.

8.3 Approach

We unfold this chapter by motivating with the simpler 2D case of classical image alignment as an example. Then we discuss how the same concept is also applicable to the 3D case, giving inspiration to our proposed BARF.

8.3.1 Planar Image Alignment (2D)

Let $\mathbf{x} \in \mathbb{R}^2$ be the 2D pixel coordinates and $\mathcal{I} : \mathbb{R}^2 \to \mathbb{R}^3$ be the imaging function. Image alignment aims to find the relative geometric transformation minimizing the photometric error between two images $\mathcal{I}_1$ and $\mathcal{I}_2$. We can formulate the problem with a synthesis-based objective:

$$\min_{\mathbf{p}} \sum_{\mathbf{x}} \|\mathcal{I}_1(\mathcal{W}(\mathbf{x};\mathbf{p})) - \mathcal{I}_2(\mathbf{x})\|_2^2 \ , \tag{8.1}$$

where $\mathcal{W} : \mathbb{R}^2 \to \mathbb{R}^2$ is the warp function parametrized by $\mathbf{p} \in \mathbb{R}^P$ (with P as the dimensionality). As this is a nonlinear problem, gradient-based optimization is the method of choice: given the current warp state $\mathbf{p}$, warp updates $\Delta\mathbf{p}$ are iteratively solved for and updated to the solution via $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$. Here, $\Delta\mathbf{p}$ can be written in a generic form of

$$\Delta\mathbf{p} = -\mathbf{A}(\mathbf{x};\mathbf{p}) \sum_{\mathbf{x}} \mathbf{J}(\mathbf{x};\mathbf{p})^\top \big(\mathcal{I}_1(\mathcal{W}(\mathbf{x};\mathbf{p})) - \mathcal{I}_2(\mathbf{x})\big) \ , \tag{8.2}$$

where $\mathbf{J} \in \mathbb{R}^{3\times P}$ is termed the steepest descent image, and $\mathbf{A}$ is a generic transformation which depends on the choice of the optimization algorithm. The seminal Lucas-Kanade algorithm [116] approaches the problem using Gauss-Newton optimization, *i.e.* $\mathbf{A}(\mathbf{x};\mathbf{p}) = (\sum_{\mathbf{x}} \mathbf{J}(\mathbf{x};\mathbf{p})^\top \mathbf{J}(\mathbf{x};\mathbf{p}))^{-1}$; alternatively, one could also choose first-order optimizers such as (stochastic) gradient descent which can be more naturally incorporated into modern deep learning frameworks, where $\mathbf{A}$ would correspond to a scalar learning rate.

The steepest descent image $\mathbf{J}$ can be expanded as

$$\mathbf{J}(\mathbf{x};\mathbf{p}) = \frac{\partial \mathcal{I}_1(\mathcal{W}(\mathbf{x};\mathbf{p}))}{\partial \mathcal{W}(\mathbf{x};\mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}} \ , \tag{8.3}$$

where $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}} \in \mathbb{R}^{2\times P}$ is the warp Jacobian constraining the pixel displacements with respect to the predefined warp. At the heart of gradient-based registration are the image gradients $\frac{\partial \mathcal{I}(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{3\times 2}$ modeling a local per-pixel linear relationship between appearance and spatial displacements, which is classically estimated via finite differencing. The overall warp update $\Delta\mathbf{p}$ can be more effectively estimated from pixel value differences if the per-pixel predictions are coherent (Fig. 8.2), *i.e.* the image signals are smooth. However, as natural images are typically complex signals, gradient-based registration on raw images is susceptible to suboptimal solutions if poorly initialized. Therefore, coarse-to-fine strategies have been practiced by blurring the images at earlier stages of registration, effectively widening the basin of attraction and smoothening the alignment landscape.

Images as neural networks. An alternative formulation of the problem is to learn a coordinate-based image representation with a neural network while also solving for the warp $\mathbf{p}$. Writing the network as $f : \mathbb{R}^2 \to \mathbb{R}^3$ and denoting $\mathbf{\Theta}$ as its parameters, one can instead choose to optimize the objective

$$\min_{\mathbf{p},\mathbf{\Theta}} \sum_{\mathbf{x}} \big(\|f(\mathbf{x};\mathbf{\Theta}) - \mathcal{I}_1(\mathbf{x})\|_2^2 + \|f(\mathcal{W}(\mathbf{x};\mathbf{p});\mathbf{\Theta}) - \mathcal{I}_2(\mathbf{x})\|_2^2 \big) \ , \tag{8.4}$$

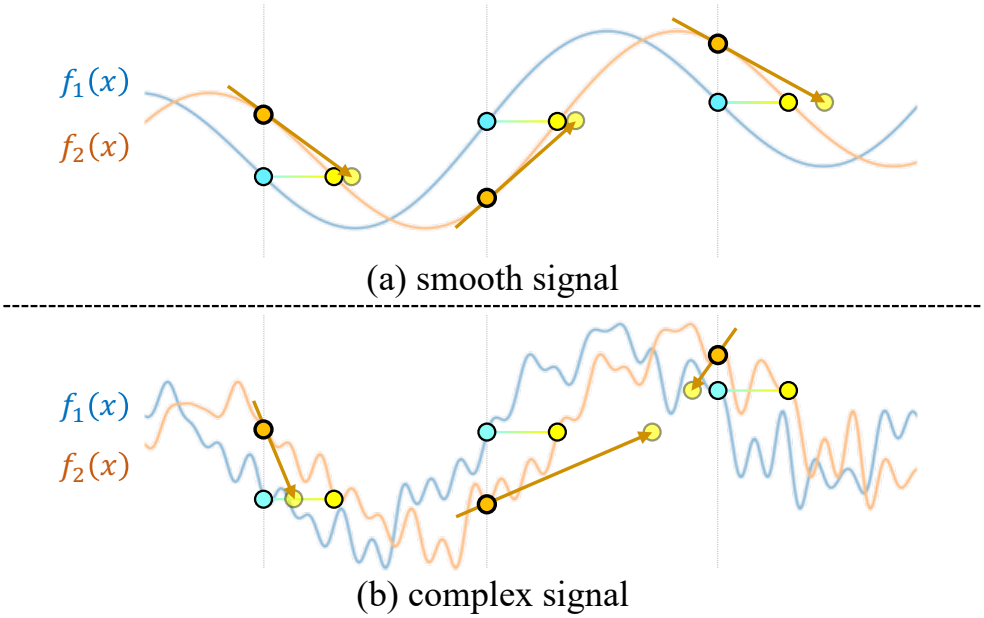


Figure 8.2: **Predicting alignment from signal differences.** Consider two 1D signals where $f_1(x) = f_2(x+c)$ differs by an offset c. When solving for alignment, smoother signals can predict more coherent displacements than complex signals, which easily results in suboptimal alignment.

or alternatively, one may choose to solve for warp parameters $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively for both images $\mathcal{I}_1$ and $\mathcal{I}_2$ through

$$\min_{\mathbf{p}_1,\mathbf{p}_2,\boldsymbol{\Theta}} \sum_{i=1}^{M} \sum_{\mathbf{x}} \| f(\mathcal{W}(\mathbf{x};\mathbf{p}_i);\boldsymbol{\Theta}) - \mathcal{I}_i(\mathbf{x}) \|_2^2 \ , \tag{8.5}$$

where $M = 2$ is the number of images. Albeit similar to (8.1), the image gradients become the analytical Jacobian of the network $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ instead of numerical estimation. By manipulating the network f, this also enables more principled control of the signal smoothness for alignment without having to rely on heuristic blurring on images, making these forms generalizable to 3D scene representations (Sec. 8.3.2).

8.3.2 Neural Radiance Fields (3D)

We discuss the 3D case of recovering the 3D scene representation from Neural Radiance Fields (NeRF) [126] *jointly* with the camera poses. To signify the analogy to Sec. 8.3.1, we deliberately overload the notations $\mathbf{x}$ as 3D points, $\mathcal{W}$ as camera pose transformations, and f as the network.

NeRF encodes a 3D scene as a continuous 3D representation using an MLP $f : \mathbb{R}^3 \to \mathbb{R}^4$ to predict the RGB color $\mathbf{c} \in \mathbb{R}^3$ and volume density $\sigma \in \mathbb{R}$ for each input 3D point $\mathbf{x} \in \mathbb{R}^3$. This can be summarized as $\mathbf{y} = [\mathbf{c}; \sigma]^\top = f(\mathbf{x};\boldsymbol{\Theta})$, where $\boldsymbol{\Theta}$ is the network parameters[1]. NeRF assumes an emission-only model, *i.e.* the rendered color of a pixel is dependent only on the emitted radiance of 3D points along the viewing ray, without considering external lighting factors.

We first formulate the rendering operation of NeRF in the camera view space. Given pixel coordinates $\mathbf{u} \in \mathbb{R}^2$ and denoting its homogeneous coordinates as $\bar{\mathbf{u}} = [\mathbf{u}; 1]^\top \in \mathbb{R}^3$, we can express a 3D point $\mathbf{x}_i$ along the viewing ray at depth z_i as $\mathbf{x}_i = z_i \bar{\mathbf{u}}$. The RGB color $\hat{\mathcal{I}}$ at pixel location $\mathbf{u}$ is extracted by volume rendering via

$$\hat{\mathcal{I}}(\mathbf{u}) = \int_{z_{\text{near}}}^{z_{\text{far}}} T(\mathbf{u}, z) \sigma(z\bar{\mathbf{u}}) \mathbf{c}(z\bar{\mathbf{u}}) \mathrm{d}z \ , \tag{8.6}$$

where $T(\mathbf{u}, z) = \exp\big(-\int_{z_{\text{near}}}^{z} \sigma(z'\bar{\mathbf{u}}) \mathrm{d}z'\big)$, and z_{near} and z_{far} are bounds on the depth range of interest. We refer our readers to Levoy [96] and Mildenhall *et al.* [126] for a more detailed treatment on volume rendering. In practice, the above integral formulations are approximated numerically via quadrature on discrete N points at depth $\{z_1, \ldots, z_N\}$ sampled along the ray. This involves N evaluations of the network f, whose output $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ are further composited through volume rendering. We can summarize the ray compositing function as $g : \mathbb{R}^{4N} \to \mathbb{R}^3$ and rewrite $\hat{\mathcal{I}}(\mathbf{u})$ as $\hat{\mathcal{I}}(\mathbf{u}) = g\left(\mathbf{y}_1, \ldots, \mathbf{y}_N\right)$. Note that g is differentiable but deterministic, *i.e.* there are no learnable parameters associated.

Under a 6-DoF camera pose parametrized by $\mathbf{p} \in \mathbb{R}^6$, a 3D point $\mathbf{x}$ in the camera view space can be transformed to the 3D world coordinates through a 3D rigid transformation $\mathcal{W} : \mathbb{R}^3 \to \mathbb{R}^3$. Therefore, the synthesized RGB value at pixel $\mathbf{u}$ becomes a function of the camera pose $\mathbf{p}$ as

$$\hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}) = g\Big(f(\mathcal{W}(z_1\bar{\mathbf{u}}; \mathbf{p}); \boldsymbol{\Theta}), \ldots, f(\mathcal{W}(z_N\bar{\mathbf{u}}; \mathbf{p}); \boldsymbol{\Theta})\Big) \ . \tag{8.7}$$

Given M images $\{\mathcal{I}_i\}_{i=1}^M$, our goal is to optimize NeRF *and* the camera poses $\{\mathbf{p}_i\}_{i=1}^M$ over the synthesis-based objective

$$\min_{\mathbf{p}_1, \ldots, \mathbf{p}_M, \boldsymbol{\Theta}} \sum_{i=1}^{M} \sum_{\mathbf{u}} \left\| \hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}_i, \boldsymbol{\Theta}) - \mathcal{I}_i(\mathbf{u}) \right\|_2^2 \ , \tag{8.8}$$

where $\hat{\mathcal{I}}$ also depends on the network parameters $\boldsymbol{\Theta}$.

One may notice the analogy between the synthesis-based objectives of 2D image alignment (8.5) and NeRF (8.8). Similarly, we can also derive the "steepest descent image" as

$$\mathbf{J}(\mathbf{u}; \mathbf{p}) = \sum_{i=1}^{N} \frac{\partial g(\mathbf{y}_1, \ldots, \mathbf{y}_N)}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i(\mathbf{p})}{\partial \mathbf{x}_i(\mathbf{p})} \frac{\partial \mathcal{W}(z_i\bar{\mathbf{u}}; \mathbf{p})}{\partial \mathbf{p}} \ , \tag{8.9}$$

which is formed via backpropagation in practice. The linearization (8.9) is also analogous to the 2D case of (8.3), where the Jacobian of the network $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ linearly relates the change of color $\mathbf{c}$ and volume density σ with 3D spatial displacements. To solve for effective camera pose updates $\Delta\mathbf{p}$ through backpropagation, it is also desirable to control the smoothness of f for predicting coherent geometric displacements from the sampled 3D points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.

[1] In practice, f is also conditioned on the viewing direction [126] for modeling view-dependent effects, which we omit here for simplicity.

8.3.3 On Positional Encoding and Registration

The key of enabling NeRF to synthesize views with high fidelity is positional encoding [180], a deterministic mapping of input 3D coordinates $\mathbf{x}$ to higher dimensions of different sinusoidal frequency bases[2]. We denote $\gamma:\mathbb{R}^3\rightarrow\mathbb{R}^{3+6L}$ as the positional encoding with L frequency bases, defined as

$$\gamma(\mathbf{x})=\big[\mathbf{x},\gamma_0(\mathbf{x}),\gamma_1(\mathbf{x}),\dots,\gamma_{L-1}(\mathbf{x})\big]\in\mathbb{R}^{3+6L}\ , \tag{8.10}$$

where the k-th frequency encoding $\gamma_k(\mathbf{x})$ is

$$\gamma_k(\mathbf{x})=\big[\cos(2^k\pi\mathbf{x}),\sin(2^k\pi\mathbf{x})\big]\in\mathbb{R}^6\ , \tag{8.11}$$

with the sinusoidal functions operating coordinate-wise. The special case of $L=0$ makes γ an identity mapping function. The network f is thus a composition of $f(\mathbf{x})=f'\circ\gamma(\mathbf{x})$, where f' is the subsequent learnable MLP. Positional encoding allows coordinate-based neural networks, which are typically bandwidth limited, to represent signals of higher frequency with faster convergence behaviors [170].

The Jacobian of the k-th positional encoding γ_k is

$$\frac{\partial\gamma_k(\mathbf{x})}{\partial\mathbf{x}}=2^k\pi\cdot\big[-\sin(2^k\pi\mathbf{x}),\cos(2^k\pi\mathbf{x})\big]\ , \tag{8.12}$$

which amplifies the gradient signals from the MLP f' by $2^k\pi$ with its direction changing at the same frequency. This makes it difficult to predict effective updates $\Delta\mathbf{p}$, since gradient signals from the sampled 3D points are incoherent (in terms of both direction and magnitude) and can easily cancel out each other. Therefore, naïvely applying positional encoding can become a double-edged sword to NeRF for the task of joint registration and reconstruction.

8.3.4 Bundle-Adjusting Neural Radiance Fields

We describe our proposed BARF, a simple yet effective strategy for coarse-to-fine registration for NeRF. The key idea is to apply a smooth mask on the encoding at different frequency bands (from low to high) over the course of optimization, which acts like a dynamic low-pass filter. Inspired by recent work of learning coarse-to-fine deformation flow fields [138], we weigh the k-th frequency component of γ as

$$\gamma_k(\mathbf{x};\alpha)=w_k(\alpha)\cdot\big[\cos(2^k\pi\mathbf{x}),\sin(2^k\pi\mathbf{x})\big]\ , \tag{8.13}$$

where the weight w_k is defined as

$$w_k(\alpha)=\begin{cases}0 & \text{if } \alpha<k\\ \dfrac{1-\cos((\alpha-k)\pi)}{2} & \text{if } 0\leq\alpha-k<1\\ 1 & \text{if } \alpha-k\geq 1\end{cases} \tag{8.14}$$

[2]Although we focus on 3D input coordinates here, positional encoding is also directly applicable to 2D image coordinates in Sec. 8.3.1 as well.

and $\alpha \in [0, L]$ is a controllable parameter proportional to the optimization progress. The Jacobian of γ_k thus becomes

$$\frac{\partial \gamma_k(\mathbf{x};\alpha)}{\partial \mathbf{x}} = w_k(\alpha) \cdot 2^k\pi \cdot \big[-\sin(2^k\pi\mathbf{x}), \cos(2^k\pi\mathbf{x})\big] \,. \tag{8.15}$$

When $w_k(\alpha) = 0$, the contribution to the gradient from the k-th (and higher) frequency component is nullified.

Starting from the raw 3D input $\mathbf{x}$ ($\alpha = 0$), we gradually activate the encodings of higher frequency bands until full positional encoding is enabled ($\alpha = L$), equivalent to the original NeRF model. This allows BARF to discover the correct registration with an initially smooth signal and later shift focus to learning a high-fidelity scene representation.

8.4 Experiments

We validate the effectiveness of our proposed BARF with a simple experiment of 2D planar image alignment, and show how the same coarse-to-fine registration strategy can be generalized to NeRF [126] for learning 3D scene representations.

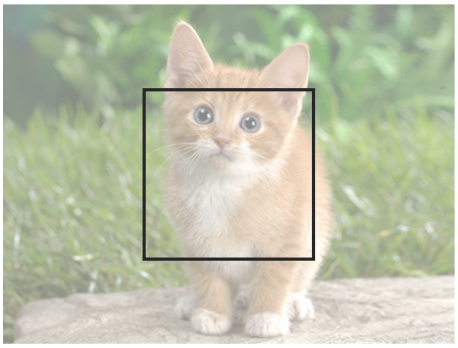
8.4.1 Planar Image Alignment (2D)

We choose a representative image from ImageNet [31], shown in Fig. 8.3. Given $M = 5$ patches from the image generated with homography perturbations (Fig. 8.3(a)), we aim to find the homography warp parameters $\mathbf{p} \in \mathbb{R}^8$ for each patch (Fig. 8.3(b)) while *also* learning the neural representation of the entire image with a network f by optimizing (8.5). We initialize all M patches with a center crop (Fig. 8.3(c)), and we anchor the warp of the first patch as identity so the recovered image would be implicitly aligned to the raw image. We parametrize homography warps with the $\mathfrak{sl}(3)$ Lie algebra.

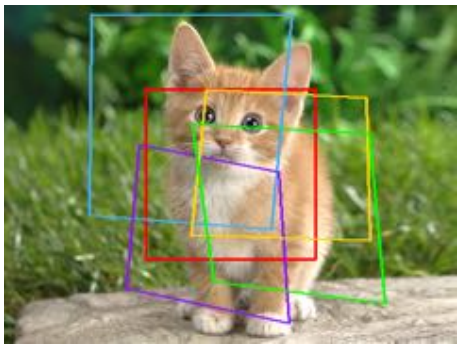
Experimental settings. We investigate how positional encoding impacts this problem by comparing networks with naïve (full) positional encoding and without any encoding. We use a simple ReLU MLP for f with four 256-dimensional hidden units, and we use the Adam optimizer [85] to optimize both the network weights and the warp parameters for 5000 iterations with a learning rate of 0.001. For BARF, we linearly adjust α for the first 2000 iterations and activate all frequency bands ($L = 8$) for the remaining iterations.

Results. We visualize the registration results in Fig. 8.4. Alignment with full positional encoding results in suboptimal registration with ghostly artifacts in the recovered image representation. On the other hand, alignment without positional encoding achieves decent registration results, but cannot recover the image with sufficient fidelity. BARF discovers the precise geometric warps with

(a) image patches given for optimization



(b) initialization



(c) ground-truth warps

Figure 8.3: Given image patches color-coded in (a), we aim to recover the alignment *and* the neural representation of the entire image, with the patches initialized to center crops shown in (b) and the corresponding ground-truth warps shown in (c).

positional encoding	$\mathfrak{sl}(3)$ error	patch PSNR
naïve (full)	0.2949	23.41
without	0.0641	24.72
BARF (coarse-to-fine)	0.0096	35.30

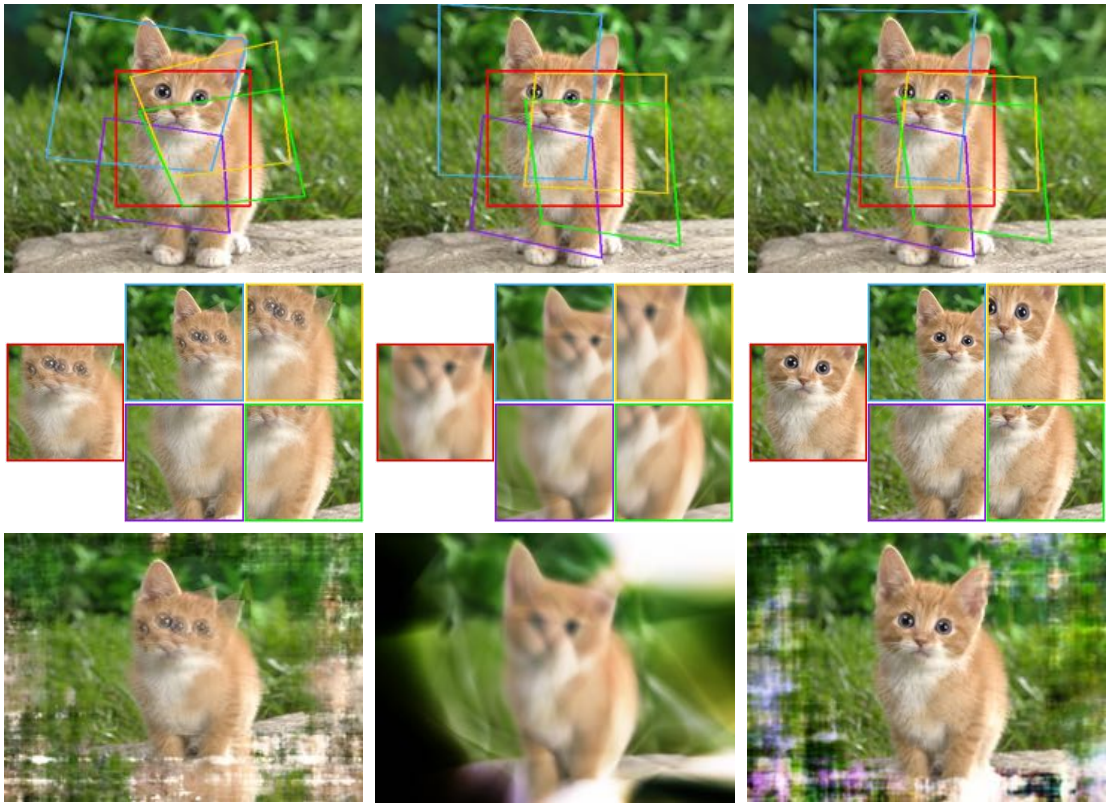
Table 8.1: **Quantitative results** of planar image alignment. BARF optimizes for more accurate alignment and patch reconstruction compared to the baselines.

the image representation optimized with high fidelity, quantitatively reflected in Table 8.1. This experiment demonstrates the general advantage of BARF for coordinate-based representations.

8.4.2 NeRF (3D): Synthetic Objects

We investigate the problem of learning 3D scene representations with Neural Radiance Fields (NeRF) [126] from imperfect camera poses. We experiment with the 8 synthetic object-centric scenes provided by Mildenhall *et al.* [126], which consists of $M = 100$ rendered images with ground-truth camera poses for each scene for training.

Experimental settings. We parametrize the camera poses $\mathbf{p}$ with the $\mathfrak{se}(3)$ Lie algebra and assume known intrinsics. For each scene, we synthetically perturb the camera poses with additive noise $\delta\mathbf{p} \sim \mathcal{N}(\mathbf{0}, 0.15\mathbf{I})$, which corresponds to a standard deviation of $14.9°$ in rotation and 0.26



(a) naïve pos. enc. (b) w/o pos. enc. (c) BARF

Figure 8.4: **Qualitative results** of the planar image alignment experiment. We visualize the optimized warps (top row), the patch reconstructions in corresponding colors (middle row), and recovered image representation from f (bottom row). BARF is able to recover accurate alignment and high-fidelity image reconstruction, while baselines result in suboptimal alignment with naïve positional encoding and blurry reconstruction without any encoding. Best viewed in color.

in translational magnitude (Fig. 8.5(a)). We optimize the objective in (8.8) jointly for the scene representation and the camera poses. We evaluate BARF mainly against the original NeRF model with naïve (full) positional encoding; for completeness, we also compare with the same model without positional encoding.

Implementation details. We follow the architectural settings from the original NeRF [126] with some modifications. We train a single MLP with 128 hidden units in each layer and without additional hierarchical sampling for simplicity. We resize the images to 400×400 pixels and randomly sample 1024 pixel rays at each optimization step. We choose $N = 128$ sample for numerical integration along each ray, and we use the softplus activation on the volume density output σ for improved stability. We use the Adam optimizer and train all models for 2000 epochs, with a learning rate of 5×10^{-4} exponentially decaying to 1×10^{-4} for the network f and 1×10^{-3} decaying to 1×10^{-5} for the camera poses $\mathbf{p}$. For BARF, we linearly adjust α from epoch 400 to 800 and activate all frequency bands (up to $L = 10$) subsequently.

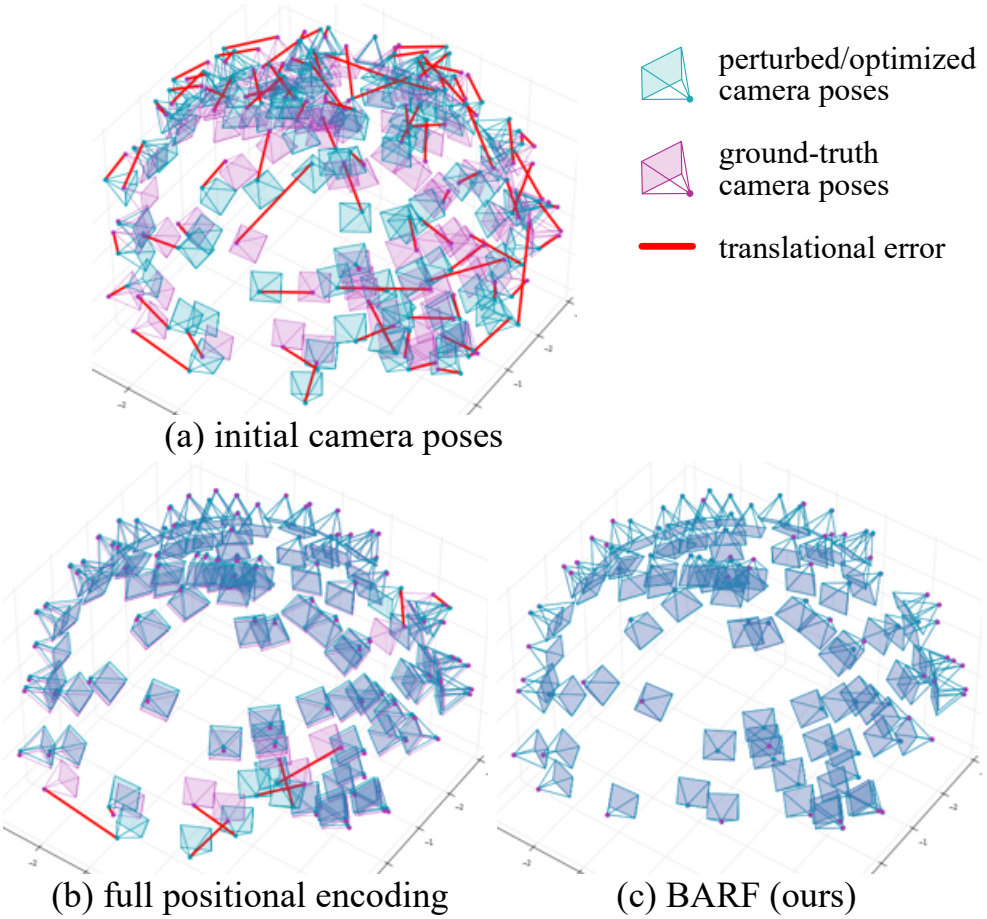


Figure 8.5: Visual comparison of the initial and optimized camera poses (Procrustes aligned) for the *chair* scene. BARF successfully realigns all the camera frames while NeRF naïve positional encoding gets stuck at suboptimal solutions.

Evaluation criteria. We measure the performance in two aspects: pose error for registration and view synthesis quality for the scene representation. Since both the scene and camera poses are variable up to a 3D similarity transformation, we evaluate the quality of registration by pre-aligning the optimized poses to the ground truth with Procrustes analysis on the camera locations. For evaluating view synthesis, we run an additional step of test-time photometric optimization on the trained models [103, 208] to factor out the pose error that may contaminate the view synthesis quality. We report the average rotation and translation errors for pose and PSNR, SSIM and LPIPS [213] for view synthesis.

Results. We visualize the results in Fig. 8.6 and report the quantitative results in Table 8.2. BARF takes the best of both worlds of recovering the neural scene representation with the camera pose successfully registered, while naïve NeRF with full positional encoding finds suboptimal solutions. Fig. 8.5 shows that BARF can achieve near-perfect registration for the synthetic scenes. Although the NeRF model without positional encoding can also successfully recover alignment,

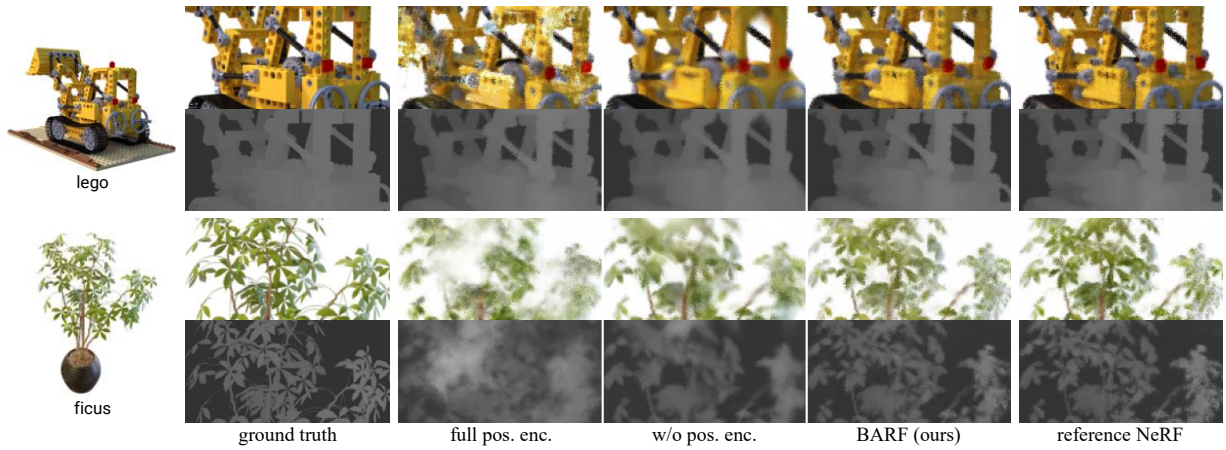


Figure 8.6: **Qualitative results** of NeRF on synthetic scenes. We visualize the image synthesis (top) and the expected depth through ray compositing (bottom). BARF achieves comparable synthesis quality to the reference NeRF (trained under perfect camera poses), while full positional encoding results in suboptimal registration, leading to synthesis artifacts.

Scene	Camera pose registration						View synthesis quality											
	Rotation (°) ↓			Translation ↓			PSNR ↑				SSIM ↑				LPIPS ↓			
	full pos.enc.	w/o pos.enc.	BARF	full pos.enc.	w/o pos.enc.	BARF	full pos.enc.	w/o pos.enc.	BARF	ref. NeRF	full pos.enc.	w/o pos.enc.	BARF	ref. NeRF	full pos.enc.	w/o pos.enc.	BARF	ref. NeRF
Chair	1.382	1.034	0.115	5.921	0.554	0.679	28.68	28.93	30.25	30.42	0.931	0.929	0.943	0.948	0.073	0.083	0.058	0.060
Drums	2.944	0.071	0.062	8.556	0.335	0.434	21.58	22.65	23.12	23.42	0.855	0.873	0.884	0.889	0.149	0.155	0.128	0.120
Ficus	4.861	0.139	0.095	8.870	0.632	0.583	21.34	23.64	25.33	25.61	0.866	0.898	0.922	0.926	0.161	0.104	0.078	0.116
Hotdog	0.845	0.235	0.273	1.702	1.184	1.600	32.15	32.43	32.97	33.37	0.956	0.957	0.961	0.963	0.058	0.054	0.053	0.047
Lego	0.648	0.112	0.069	2.432	0.408	0.335	26.85	25.32	27.20	27.70	0.907	0.859	0.906	0.920	0.079	0.136	0.072	0.064
Materials	2.712	0.049	0.541	7.991	0.313	1.433	21.34	25.52	26.07	27.43	0.850	0.899	0.910	0.930	0.153	0.099	0.087	0.070
Mic	5.283	0.078	0.065	12.227	0.437	0.314	20.88	30.14	30.62	31.21	0.884	0.964	0.965	0.968	0.201	0.060	0.056	0.052
Ship	4.274	0.156	0.642	3.827	0.540	1.519	25.10	26.02	26.61	27.14	0.806	0.818	0.829	0.839	0.196	0.200	0.164	0.154
Mean	2.869	0.234	0.233	6.441	0.550	0.862	24.74	26.83	27.77	28.29	0.882	0.900	0.915	0.923	0.134	0.111	0.087	0.085

Table 8.2: **Quantitative results** of NeRF on synthetic scenes. BARF successfully optimizes for camera registration (with less than 1° rotation error) while still consistently achieving high-quality view synthesis that is comparable to the reference NeRF models (trained under perfect camera poses). Translation errors are scaled by 100.

the learned scene representations (and thus the synthesized images) lack the reconstruction fidelity. As a reference, we also compare the view synthesis quality against standard NeRF models trained under ground-truth poses, showing that BARF can achieve comparable view synthesis quality in all metrics, albeit initialized from imperfect camera poses.

8.4.3 NeRF (3D): Real-World Scenes

We investigate the challenging problem of learning neural 3D representations with NeRF on real-world scenes, where the camera poses are *unknown*. We consider the LLFF dataset [125], which consists of 8 forward-facing scenes with RGB images captured by hand-held cameras.

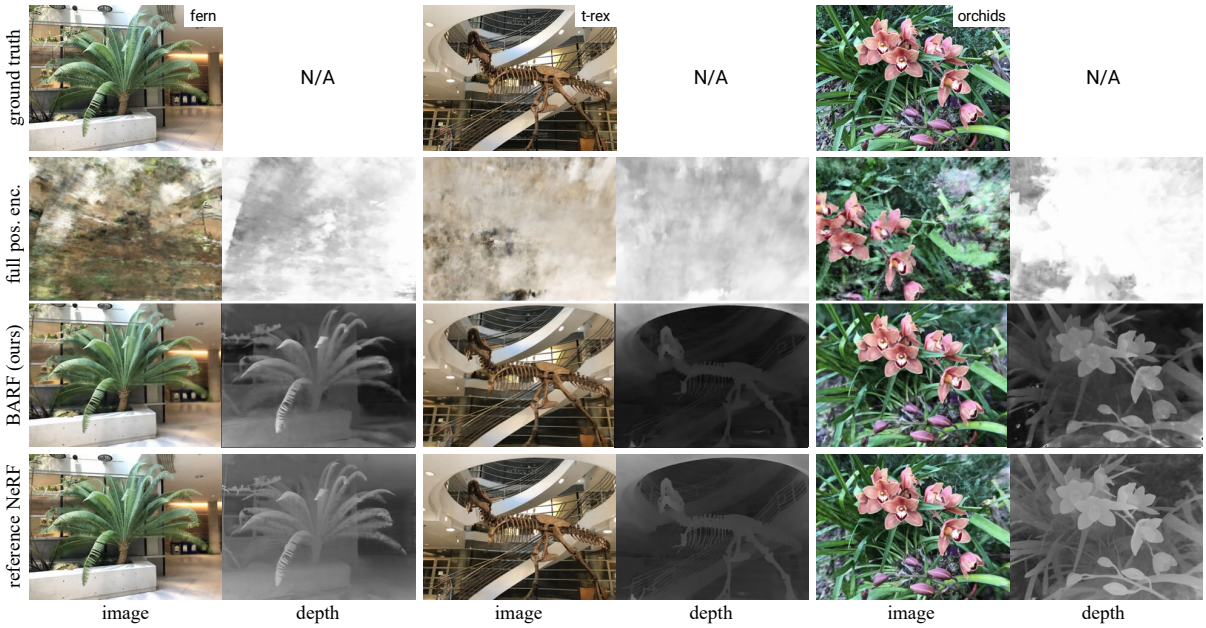
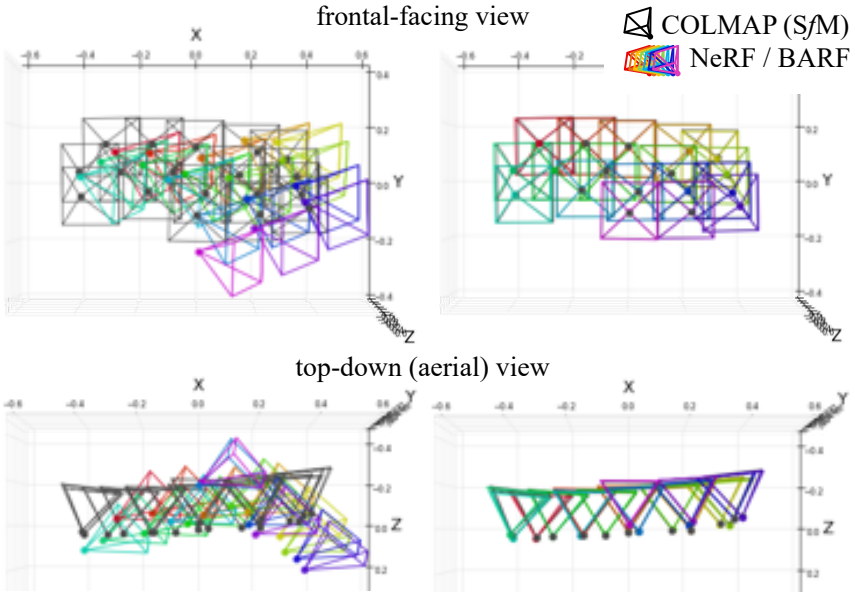


Figure 8.7: **Qualitative results** of NeRF on real-world scenes from *unknown* camera poses. Compared to a reference NeRF model trained with camera poses provided from S*f*M [153], BARF can effectively optimize for the poses jointly with the scene representation. NeRF models with full positional encoding diverge to incorrect localization and hence poor synthesis quality.

Experimental settings. We parametrize the camera poses $\mathbf{p}$ with $\mathfrak{se}(3)$ following Sec. 8.4.2 but initialize all cameras with the *identity* transformation, *i.e.* $\mathbf{p}_i = \mathbf{0}\ \ \forall i$. We assume known camera intrinsics (provided by the dataset). We compare against the original NeRF model with naïve positional encoding, and we use the same evaluation criteria described in Sec. 8.4.2. However, we note that the camera poses provided in LLFF are also estimations from S*f*M packages [153]; therefore, the pose evaluation is at most an indication of how well BARF agrees with classical geometric pose estimation.

Implementation details. We follow the same the architectural settings from the original NeRF *et al.* [126] and resize the images to 480×640 pixels. We train all models for 10000 epochs, with a learning rate of 1×10^{-3} for the network f and 3×10^{-3} for the pose $\mathbf{p}$, both decaying to 1×10^{-5}. We linearly adjust α for BARF from epoch 2000 to 4000 and activate all frequency bands (up to $L=10$) subsequently.

Results. The quantitative results (Table 8.3) show that the recovered camera poses from BARF highly agrees with those estimated from off-the-shelf S*f*M methods (visualized in Fig. 8.8), demonstrating the ability of BARF to localize from scratch. Furthermore, BARF can successfully recover the 3D scene representation with high fidelity (Fig. 8.7). In contrast, NeRF with naïve positional encoding diverge to incorrect camera poses, which in turn results in poor view synthesis. This highlights the effectiveness of BARF for coarse-to-fine joint registration and reconstruction.



(a) full pos. enc. (b) BARF (ours)

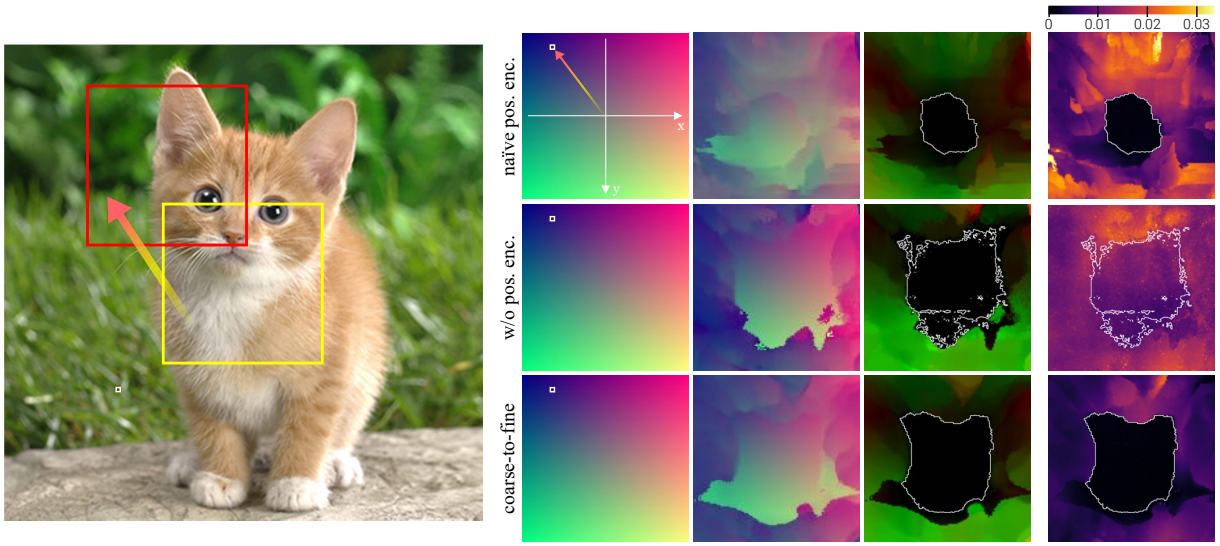
Figure 8.8: Visualization of optimized camera poses from the *fern* scene (Procrustes aligned). Results from BARF highly agrees with S*f*M, whereas the baseline poses are suboptimal.

Scene	Camera pose registration				View synthesis quality								
	Rotation (°) ↓		Translation ↓		PSNR ↑			SSIM ↑			LPIPS ↓		
	full pos.enc.	BARF	full pos.enc.	BARF	full pos.enc.	BARF	ref. NeRF	full pos.enc.	BARF	ref. NeRF	full pos.enc.	BARF	ref. NeRF
Fern	59.081	0.982	20.081	0.479	10.68	22.82	24.03	0.233	0.658	0.734	0.857	0.385	0.273
Flower	28.076	3.205	5.434	0.735	11.57	20.30	22.94	0.203	0.555	0.690	0.851	0.325	0.207
Fortress	149.785	1.993	46.019	0.434	14.08	27.01	27.10	0.363	0.752	0.806	0.761	0.234	0.133
Horns	177.604	2.424	64.518	0.264	9.87	20.07	22.15	0.187	0.669	0.743	0.804	0.374	0.272
Leaves	41.954	1.832	7.808	0.419	7.93	17.06	15.90	0.079	0.420	0.349	0.750	0.432	0.487
Orchids	31.386	2.095	29.613	0.666	9.08	18.03	19.13	0.090	0.478	0.587	0.759	0.397	0.269
Room	49.661	1.275	29.342	0.195	10.84	28.50	31.06	0.421	0.893	0.936	0.856	0.176	0.099
T-rex	165.731	1.913	43.226	0.956	7.37	21.90	23.68	0.288	0.741	0.816	0.948	0.248	0.146
Mean	87.910	1.965	30.755	0.519	10.18	21.96	23.25	0.233	0.646	0.708	0.823	0.321	0.236

Table 8.3: **Quantitative results** of NeRF on the LLFF forward-facing scenes from *unknown* camera poses. BARF can optimize for accurate camera poses (with an average rotation error of $2°$) and high-fidelity scene representations, enabling novel view synthesis with comparable quality to reference NeRF model trained under S*f*M poses. Translation errors are scaled by 100.

8.5 Conclusion

We present Bundle-Adjusting Neural Radiance Fields (BARF), a simple yet effective strategy for training NeRF from imperfect camera poses. By establishing a theoretical connection to classical image alignment, we demonstrate that coarse-to-fine registration is necessary for joint registration



(a) source and target patch (b) target warp (c) optimized warp (d) warp error (e) image recon. error

Figure 8.9: Visualization of the **basin of attraction**. (a) We aim to align a center box (yellow) to a target patch (red) at *every* possible location within the raw image. For each target patch, we jointly optimize f and the translational warp $\mathbf{p}$ to analyze the final warp error and the image reconstruction loss. (b) The target offsets forms a color-coded map, where green indicates horizontal offsets and red indicates vertical offsets. The above example corresponds to the highlighted pixel. (c) The optimized warp parameters and (d) the warp error for every target patch location, where the white contours highlight the offset error threshold of 0.5 pixels. BARF effectively widens the basin of attraction (range of successful alignment) with a smoother landscape compared to naïve positional encoding. (e) Without positional encoding, f has limited capacity of representing the image details, resulting in nonzero image errors despite the registration being successful as well.

and reconstruction with coordinate-based scene representations. Our experiments show that BARF can effectively learn the 3D scene representations from scratch and resolve large camera pose misalignment at the same time.

Despite the intriguing results at the current stage, BARF has similar limitations to the original NeRF formulation [126] (*e.g.* slow optimization and rendering, rigidity assumption, sensitivity to dense 3D sampling), as well as reliance on heuristic coarse-to-fine scheduling strategies. Nevertheless, since BARF keeps a close formulation to NeRF, many of the latest advances on improving NeRF are potentially transferable to BARF as well. We believe BARF opens up exciting avenues for rethinking visual localization for SfM/SLAM systems and self-supervised dense 3D reconstruction frameworks using view synthesis as a proxy objective.

8.A Visualizing the Basin of Attraction

The planar image alignment setting allows us to analyze how positional encoding affects the basin of attraction. We use the same image in Fig. 8.3 and consider the simpler case of aligning two

image patches differing by an offset. We use a translational warp $\mathbf{p} \in \mathbb{R}^2$ on a square box whose size is $1/3$ of the raw image height and initialized to the raw center. We aim to register the center box to a single target patch of the same size shifted by some offset, shown in Fig. 8.9(a). We optimize the image neural network f with the objective in (8.4), where $\mathcal{I}_1$ is the center patch and $\mathcal{I}_2$ is the target patch, and investigate the convergence behavior of translational alignment as a function of target offsets. We search over the entire pixel grid to as far as where the target patch has no overlapping region with the initial center box.

We visualize the results in Fig. 8.9. Naïve positional encoding results in a more nonlinear alignment landscape and a smaller basin of attraction, while not using positional encoding sacrifices the reconstruction quality due to the limited representability of the network f. In contrast, BARF can widen the basin of attraction while reconstructing the image representation with high fidelity. This also justifies the importance of coarse-to-fine registration for NeRF.

8.B Additional NeRF Details & Results

We provide more details and results from our NeRF experiments in this section (for real-world scenes in particular).

8.B.1 Evaluation Details

As mentioned in the main chapter, the optimized solutions of the 3D scenes and camera poses are up to a 3D similarity transformation. Therefore, we evaluate the quality of registration by pre-aligning the optimized poses to the reference poses, which are the ground truth poses for the synthetic objects (Sec. 8.4.2) and pose estimation computed from S*f*M packages [153] for the real-world scenes (Sec. 8.4.3).

We use Procrustes analysis on the camera locations for aligning the coordinate systems. The algorithm details are described in Alg. 1. We write the reference poses $\{[\mathbf{R}_i,\mathbf{t}_i]\}_{i=1}^M$ and the optimized poses $\{[\widehat{\mathbf{R}}_i,\hat{\mathbf{t}}_i]\}_{i=1}^M$ in the form of camera extrinsic matrices, and the aligned poses can be written as $\{[\widehat{\mathbf{R}}'_i,\hat{\mathbf{t}}'_i]\}_{i=1}^M = \textsc{PreAlign}(\{[\mathbf{R}_i,\mathbf{t}_i]\}_{i=1}^M,\{[\widehat{\mathbf{R}}_i,\hat{\mathbf{t}}_i]\}_{i=1}^M)$. After the cameras are Procrustes-aligned, we apply the relative rotation (solved for via the Procrustes analysis process) to account for rotational differences. We measure the rotation error between the S*f*M poses and the aligned poses from NeRF/BARF by the angular distance as

$$\Delta\theta_i = \cos^{-1}\left(2\langle \mathbf{q}_i,\hat{\mathbf{q}}'_i\rangle^2-1\right)\ ,\quad i=\{1,\dots,M\}\ , \tag{8.16}$$

where $\mathbf{q}_i$ and $\hat{\mathbf{q}}'_i$ are the quaternion representation of $\mathbf{R}_i$ and $\widehat{\mathbf{R}}'_i$ (the rotation of the Procrusted-aligned poses) respectively and $\langle\cdot,\cdot\rangle$ is the quaternion inner product. For additional clarity, we provide a more detailed visualization of the optimized camera poses in Fig. 8.10 (for LLFF).

To evaluate the quality of novel view synthesis while being minimally affected by camera misalignment, we transform the test views (provided by Mildenhall *et al.* [125]) to the coordinate

Algorithm 1: Pre-align camera poses for evaluation

Function PREALIGN($\{[\mathbf{R}_i, \mathbf{t}_i]\}_{i=1}^M, \{[\widehat{\mathbf{R}}_i, \hat{\mathbf{t}}_i]\}_{i=1}^M$)**:**

Input : reference poses $\{[\mathbf{R}_i, \mathbf{t}_i]\}_{i=1}^M$, optimized poses $\{[\widehat{\mathbf{R}}_i, \hat{\mathbf{t}}_i]\}_{i=1}^M$

Output : optimized poses $\{[\widehat{\mathbf{R}}'_i, \hat{\mathbf{t}}'_i]\}_{i=1}^M$ aligned to the reference poses

for $i = \{1, \dots, M\}$ **do**

$\mathbf{o}_i = -\mathbf{R}_i^\top \mathbf{t}_i$

$\hat{\mathbf{o}}_i = -\widehat{\mathbf{R}}_i^\top \hat{\mathbf{t}}_i$

end

$s, \hat{s}, \mathbf{t}, \hat{\mathbf{t}}, \mathbf{R} = \text{PROCRUSTES}(\{\mathbf{o}_i\}_{i=1}^M, \{\hat{\mathbf{o}}_i\}_{i=1}^M)$

for $i = \{1, \dots, M\}$ **do**

$\hat{\mathbf{o}}'_i = s\mathbf{R}\left(\frac{1}{\hat{s}}(\hat{\mathbf{o}}_i - \hat{\mathbf{t}})\right) + \mathbf{t}$

$\widehat{\mathbf{R}}'_i = \widehat{\mathbf{R}}_i \mathbf{R}^\top$

$\hat{\mathbf{t}}'_i = -\widehat{\mathbf{R}}_i'^\top \hat{\mathbf{o}}'_i$

end

return $\{[\widehat{\mathbf{R}}'_i, \hat{\mathbf{t}}'_i]\}_{i=1}^M$

end

Function PROCRUSTES($\{\mathbf{o}_i\}_{i=1}^M, \{\hat{\mathbf{o}}_i\}_{i=1}^M$)**:**

Input : reference camera centers $\{\mathbf{o}_i\}_{i=1}^M$, optimized camera centers $\{\hat{\mathbf{o}}_i\}_{i=1}^M$

Output : scale s, $\hat{s}$, translation $\mathbf{t}$, $\hat{\mathbf{t}}$, rotation $\mathbf{R}$

$\mathbf{t} = \frac{1}{M}\sum_{i=1}^M \mathbf{o}_i \ \in \mathbb{R}^3$

$\hat{\mathbf{t}} = \frac{1}{M}\sum_{i=1}^M \hat{\mathbf{o}}_i \ \in \mathbb{R}^3$

$s = \sqrt{\frac{1}{M}\sum_{i=1}^M \|\mathbf{o}_i - \mathbf{t}\|_2^2} \ \in \mathbb{R}$

$\hat{s} = \sqrt{\frac{1}{M}\sum_{i=1}^M \left\|\hat{\mathbf{o}}_i - \hat{\mathbf{t}}\right\|_2^2} \ \in \mathbb{R}$

$\mathbf{X} = \frac{1}{s}\left([\mathbf{o}_1, \dots, \mathbf{o}_M] - \mathbf{t}\mathbf{1}_M^\top\right) \ \in \mathbb{R}^{3\times M}$

$\widehat{\mathbf{X}} = \frac{1}{\hat{s}}\left([\hat{\mathbf{o}}_1, \dots, \hat{\mathbf{o}}_M] - \hat{\mathbf{t}}\mathbf{1}_M^\top\right) \ \in \mathbb{R}^{3\times M}$

$\mathbf{U}, \mathbf{S}, \mathbf{V}^\top = \text{SVD}(\mathbf{X}\widehat{\mathbf{X}}^\top)$

$\mathbf{R} = \mathbf{U}\mathbf{V}^\top \ \in \mathbb{R}^{3\times 3}$

if $\det(\mathbf{R}) = -1$ **then**

multiply last row of $\mathbf{R}$ by -1

end

return s, $\hat{s}$, $\mathbf{t}$, $\hat{\mathbf{t}}$, $\mathbf{R}$

end

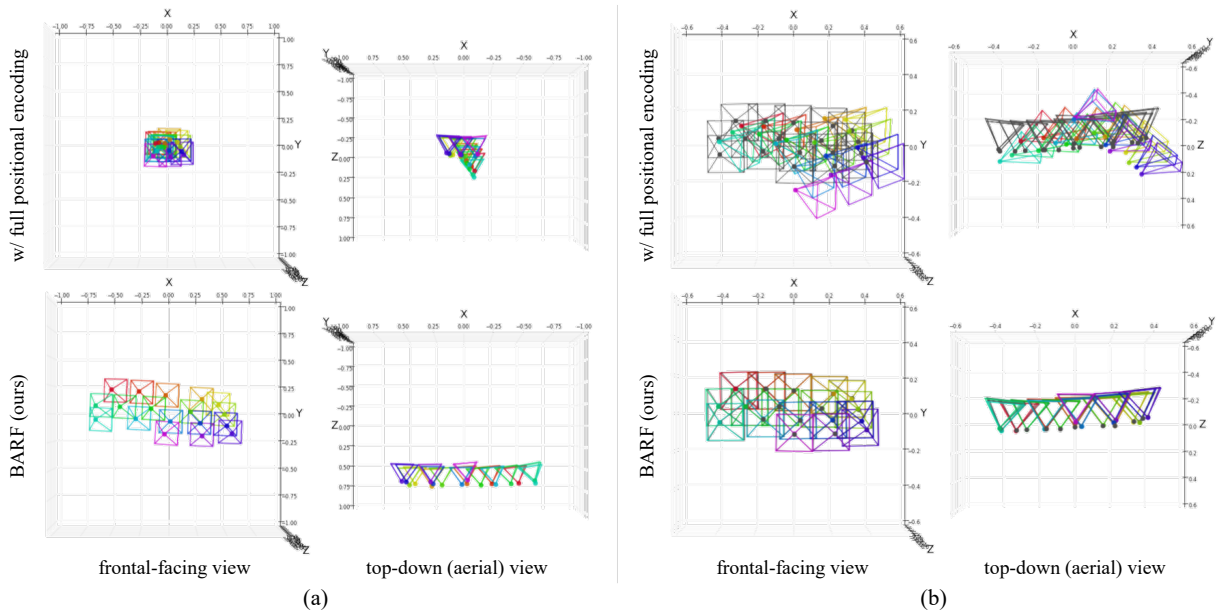


Figure 8.10: Visualization of the **optimized camera poses** for the *fern* scene. The poses for both the baseline NeRF (with full positional encoding) and BARF are initialized to the identity transform for all frames. (a) The camera poses of the baseline NeRF get stuck in a suboptimal solution that does not accurately reflect the actual viewpoints, whereas BARF can effectively optimize for the underlying poses. (b) We compare the optimized poses to those computed from S*f*M [153] (colored in black), where we align the pose trajectories using Procrustes analysis. The camera poses optimized by BARF highly agree with those from S*f*M, whereas those from the baseline NeRF cannot be well-aligned with Procrustes analysis. Therefore, there is no systematic way of finding a reasonable set of corresponding held-out views with respect to the optimized coordinate system.

system of the optimized poses by applying the scale/rotation/translation from the Procrustes analysis, as in Alg. 1. The camera trajectories from the baseline NeRF with naïve full positional encoding exhibits large rotational and translational differences compared to S*f*M poses in general. For this reason, the view synthesis results from the baseline NeRF, whose corresponding test views are also determined using Procrustes analysis, are far from plausible. Unfortunately, there is no other systematic way of determining what the corresponding views held out from the S*f*M poses would be in the learned coordinate system. Nevertheless, we provide additional qualitative results in Fig. 8.11, where the novel views are selected from a *training* view closest to the average pose and sampling translational perturbations.

8.B.2 Real-World Scenes (LLFF Dataset)

Dataset. The LLFF dataset [125] consists of 8 forward-facing scenes with RGB images sequentially captured by hand-held cameras. In the original NeRF paper [125], the test views were selected by holding out every 8th frame from the video sequence and training with the remaining

	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-rex
split	18 / 2	31 / 3	38 / 4	56 / 6	24 / 2	23 / 2	37 / 4	50 / 5
total	20	34	42	62	26	25	41	55

Table 8.4: Dataset statistics of the train/test splits for the real-world scene (LLFF) experiments, where we hold out the last 10% frames from each sequences.

Scene	Camera pose registration				View synthesis quality								
	Rotation (°) ↓		Translation ↓		PSNR ↑			SSIM ↑			LPIPS ↓		
	full pos.enc.	BARF	full pos.enc.	BARF	full pos.enc.	BARF	ref. NeRF	full pos.enc.	BARF	ref. NeRF	full pos.enc.	BARF	ref. NeRF
Fern	64.822	1.042	24.035	0.607	10.56	22.40	22.35	0.132	0.645	0.638	1.106	0.404	0.450
Flower	3.265	3.496	0.662	0.776	22.14	21.81	22.83	0.600	0.609	0.650	0.340	0.260	0.287
Fortress	2.882	2.055	1.200	0.463	23.89	25.44	25.14	0.531	0.663	0.626	0.647	0.379	0.489
Horns	6.831	2.328	8.248	0.255	18.73	18.72	21.40	0.556	0.676	0.703	0.494	0.313	0.321
Leaves	11.123	2.546	1.821	0.455	10.28	15.89	17.71	0.093	0.405	0.484	0.864	0.395	0.345
Orchids	32.764	2.482	12.851	0.855	10.31	17.57	17.70	0.074	0.447	0.434	1.057	0.426	0.550
Room	31.344	1.235	20.592	0.249	11.08	28.84	29.26	0.263	0.902	0.894	1.076	0.168	0.192
T-rex	105.177	2.131	42.785	1.107	9.25	22.63	23.43	0.100	0.787	0.812	1.224	0.173	0.154
Mean	32.276	2.164	14.024	0.596	14.53	21.66	22.48	0.294	0.642	0.655	0.851	0.314	0.349

Table 8.5: **Quantitative results** of NeRF on the LLFF forward-facing scenes from *unknown* camera poses, sampling the 3D points in the regular depth space. BARF consistently optimizes for accurate camera poses (also with an average rotation error of $2°$) and enables high-quality view synthesis, demonstrating the justification of coarse-to-fine registration and that it is agnostic to the choice of dense 3D point sampling strategies in NeRF. Translation errors are scaled by 100.

frames. Unlike Mildenhall *et al.* [125], however, we hold out the last 10% of the frames for evaluation and train with the first 90% frames. This train/test split lifts the assumption that the held-out views are interpolations of the training views, which allows a more practical simulation of predicting future viewpoints from previous observations. The statistics of the train/test split for each scene is provided in Table 8.4.

Depth parametrization. In the main LLFF experiments, we sample $N = 128$ points along each ray linearly in the inverse depth (disparity) space, where the lower and upper bounds are $1/z_{\text{near}} = 1$ and $1/z_{\text{far}} = 0.05$ respectively. To analyze the effect of depth parametrization on the performance of real-world scenes, we run an additional set of the same experiments by sampling the 3D points in the regular (metric) depth space with the same bounds of $z_{\text{near}} = 1$ and $z_{\text{far}} = 20$.

We report the quantitative results in Table 8.5. Despite the baseline NeRF (with full positional encoding) showing considerable improvement in pose registration on certain scenes when sampling in the regular space, BARF still outperforms the baseline by a large margin in terms of both the registration and view synthesis quality. This shows that irrespective of the choice of 3D point sampling strategies, BARF can find a much more desirable solution of joint registration and reconstruction by utilizing a coarse-to-fine strategy.

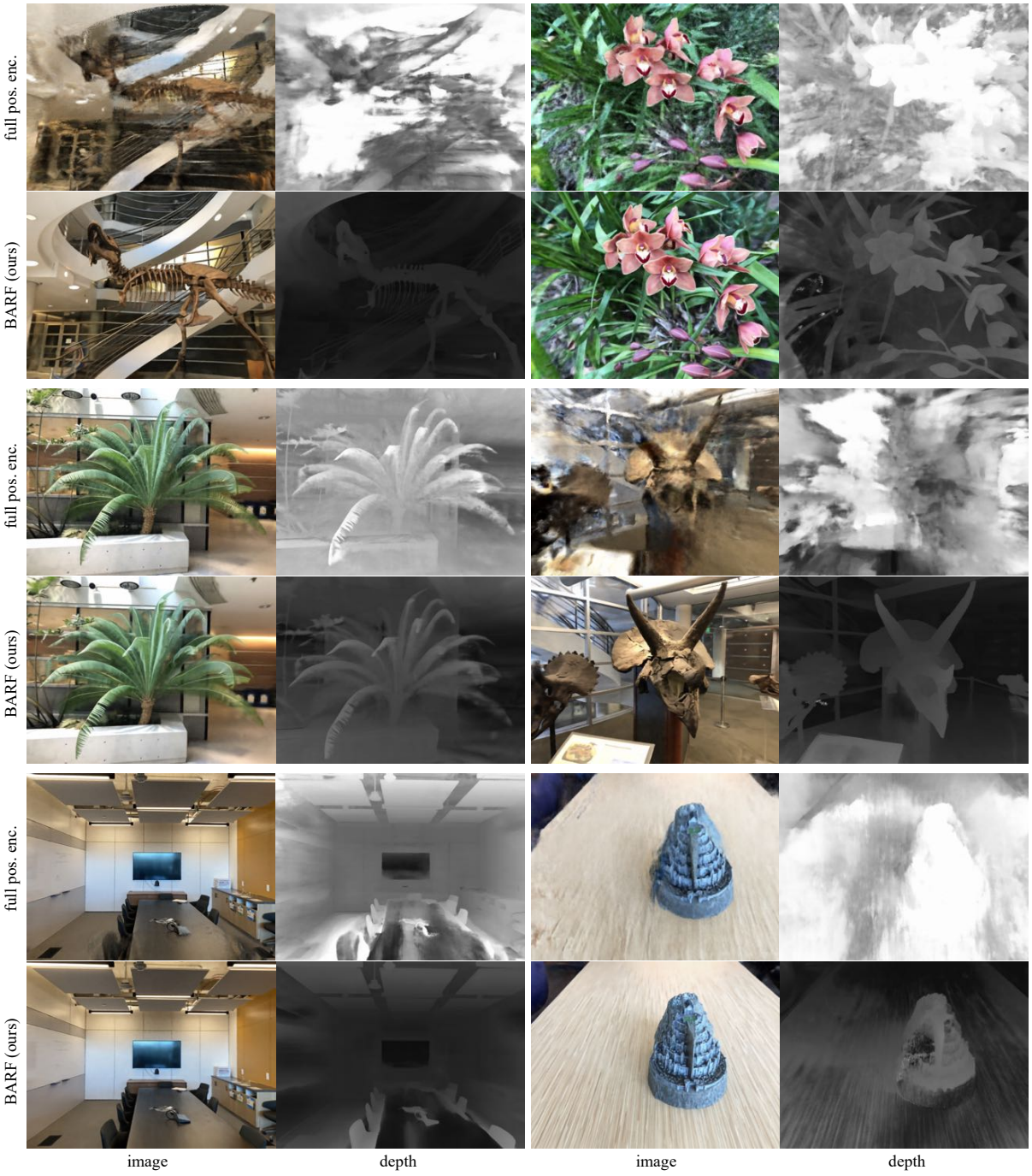


Figure 8.11: Additional novel view synthesis results from the real-world scene experiment (LLFF dataset). Instead of visualizing the held-out views computed by Procrustes analysis, we show qualitative results at new viewpoints by sampling camera pose perturbations around the viewpoint from the training set (closest to the average pose). Note that for this set of qualitative results, we do not have ground-truth RGB images to compare against. BARF can optimize for scene representations of much higher quality.

Chapter 9

Conclusion & Discussions

In this dissertation, we have discussed several learning-based methods for the problems of dense image registration and dense 3D reconstruction. We briefly summarize this thesis as follows.

In Part I, we have explored learning-based direct methods for image registration. We showed that structured geometric priors can allow learning registration models more efficiently from data (Chapter 2), geometric misalignment within an image dataset can be resolved indirectly through a discriminative objective (Chapter 3), and that spatial alignment of objects can be discovered via an adversarial objective against an unpaired image dataset (Chapter 4). Direct image registration establishes dense pixel correspondences between images, and these methods give hints to application to more sophisticated warp functions, such as 3D geometric shapes for registering the images together in 3D. Factorizing geometric information from data-driven models also them to reduce the learnable parameters for improved learning efficiency.

In Part II, we have explored learning-based 3D shape reconstruction methods from image datasets, without the use of explicit 3D supervision. We showed that generalizable 3D shape reconstruction can be naively trained from a dataset of multi-view depth images (Chapter 5), and that 3D shape reconstruction can be trained from single-view static images with neural rendering to discover semantic correspondences and lifting the ill-posedness of the problem (Chapter 6). With suitable designs of differentiable/neural rendering, we demonstrated that it is possible to utilize self-supervised learning techniques to extract 3D geometric information from large-scale image datasets, serving as a strong learned prior for dense 3D prediction and understanding tasks.

Finally in Part III, we have explored the joint problem of registration and 3D reconstruction, using neural networks for photometric bundle adjustment on objects and scenes. We showed that given a video sequence, one can use a neural network pretrained on 3D shapes as a learned shape prior to reconstruct 3D shapes pixel-aligned to the sequence (Chapter 7), and that one can use a generic 3D volume rendering prior to solve for a neural 3D scene representation of an arbitrary scene while also optimizing for the camera poses (Chapter 8). These works echo back to the lessons from registration that designing the suitable geometric prior is a very powerful tool to factorize the 3D object and scene geometry out of images and videos in a self-supervised fashion.

Future work. Despite the remarkable progress we have seen for self-supervised registration and dense 3D reconstruction, there are still much work that remains to be done. We provide discussions and potential future research directions below.

- **Learning dense 3D reconstruction of dynamic scenes.** The works that consist of this thesis has largely focused on the simpler case of learning registration and reconstruction from rigid scenes. However, our world is rarely static. In fact, the interesting actions and events captured by the majority of video sequences are *dynamic*, mostly centered around humans in motion and moving objects such as cars in street scenes. Previous works have strove to reconstruct and understand specifically dynamic humans [78, 80] as well as small-motion dynamic scenes [98, 138], but would it be possible to recover the dense 3D geometry for extreme motions as well? Relaxing the rigidity assumption and searching for a suitable prior to model such dynamic motion would be a interesting future direction.
- **Efficient neural 3D representations.** Over the past year, coordinate-based scene representations have emerged as a powerful way of modeling 3D scenes, with NeRF [126] being the most representative. The volume rendering nature of NeRF, however, is computationally prohibitive compared to other 3D renderers (*e.g.* mesh rasterization) due to the requirement of evaluating dense ray samples. Such coordinate-based representations that *implicitly* encode 3D scenes are still inferior in many aspects compared to *explicit* representations (*e.g.* object compositionality and scene expansion), limiting their practicality. Investigating efficient neural 3D scene representations would also be an interesting line of future work.
- **Scaling up self-supervision on visual data.** Fully self-supervised learning of 3D reconstruction is still far from ideal today. One of the most limiting factors is the requirement of accurate camera poses for multi-view supervision, which are often times difficult to come by as S*f*M methods are not 100% perfect. We believe BARF (Chapter 8) is one step towards alleviating this constraint. In addition, handling dynamic scenes has been a major challenge as previously discussed. An alternative future direction would be to consider scaling-up omni-supervised learning [143]. One idea would be to leverage pretrained models on existing annotated datasets such as COCO [106] and enforcing multi-view consistency on video sequences with neural rendering, such that instance segmentation predictions can be refined. Objects could also potentially be utilized as a cue in turn to refine for more accurate camera poses. Thinking about scaling up self-supervision on image and video data would be a critical ingredient towards learning dense 3D reconstruction in the wild.

This dissertation wraps up a series of work for learning 3D registration and reconstruction from the visual world, but it is by no means the end. In contrast, this opens up many more new exciting avenues for future investigation. Finally, this also marks the start of my long-term research goal: to enable the ability of AI systems to be able to truly learn from large-scale visual data to factorize the 3D geometry from all images and videos, in order to revolutionize downstream visual recognition and 3D understanding towards true spatial 3D artificial intelligence.

Bibliography

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017. 86

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *ACM Communications*, 2011. 125

[3] Hatem Alismail, Brett Browning, and Simon Lucey. Bit-planes: Dense subpixel alignment of binary descriptors. *arXiv preprint arXiv:1602.00307*, 2016. 5, 11, 21, 23

[4] Hatem Alismail, Brett Browning, and Simon Lucey. Photometric bundle adjustment for vision-based slam. In *ACCV*, 2016. 4, 5, 125

[5] Epameinondas Antonakos, Joan Alabort-i Medina, Georgios Tzimiropoulos, and Stefanos P Zafeiriou. Feature-based lucas–kanade and active appearance models. *IEEE Transactions on Image Processing*, 24(9):2617–2632, 2015. 11

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 50

[7] Akshay Asthana, Stefanos Zafeiriou, Shiyang Cheng, and Maja Pantic. Incremental face alignment in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1859–1866, 2014. 32

[8] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 99

[9] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 11, 13, 30, 31, 32, 36, 37, 49, 108

[10] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 50

[11] 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. 92, 98

[12] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *British Machine Vision Conference*, 2011. 117

[13] 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. 4

[14] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. *arXiv*, 2020. 126

[15] Hilton Bristow and Simon Lucey. In defense of gradient-based alignment on densely sampled sparse features. In *Dense Image Correspondences for Computer Vision*, pages 135–152. Springer, 2016. 11, 13, 16

[16] 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. 4, 69, 74, 82, 89, 97, 99, 112, 118

[17] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *TOG*, 2013. 126

[18] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 1993. 125

[19] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 83

[20] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016. 116

[21] 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. 69, 70, 77, 105, 116

[22] Timothy F Cootes and Christopher J Taylor. Active shape models—'smart snakes'. In *BMVC92*, pages 266–275. Springer, 1992. 107

[23] Timothy F Cootes, Gareth J Edwards, Christopher J Taylor, et al. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001. 107

[24] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017. 47

[25] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2005. 11, 32

[26] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980. 84

[27] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *TPAMI*, 2007. 4, 125

[28] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the*

23rd annual conference on Computer graphics and interactive techniques, 1996. 125

[29] Amaël Delaunoy and Marc Pollefeys. Photometric bundle adjustment for dense multi-view 3d modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1486–1493, 2014. 107

[30] Amaël Delaunoy and Emmanuel Prados. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3d reconstruction problems dealing with visibility. *International journal of computer vision*, 95(2):100–123, 2011. 107

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 4, 92, 98, 99, 131

[32] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016. 54, 55, 61

[33] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR*, 2018. 125

[34] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015. 78

[35] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *arXiv*, 2019. 125

[36] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014. 4, 105, 107, 123, 125

[37] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *TPAMI*, 2017. 125

[38] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. 123

[39] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 98

[40] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 70, 72, 77, 116

[41] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 86, 123, 126

[42] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1): 55–81, 2015. 107

[43] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010. 107

[44] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, pages 402–411. IEEE, 2017. 70, 72, 83

[45] Yaroslav Ganin, Daniil Kononenko, Diana Sungatullina, and Victor Lempitsky. Deepwarp: Photorealistic image resynthesis for gaze manipulation. In *European Conference on Computer Vision*, pages 311–326. Springer, 2016. 47

[46] Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. Portrait neural radiance fields from a single image. *arXiv*, 2020. 126

[47] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016. 69, 70

[48] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 6, 45, 47, 69

[49] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 88

[50] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010. 19

[51] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246, 2018. 107

[52] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 97, 105, 107, 109, 111, 112, 113, 116, 118

[53] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 50, 60

[54] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 88

[55] Christopher Ham, Simon Lucey, and Surya Singh. Proxy templates for inverse compositional photometric bundle adjustment. *arXiv preprint arXiv:1704.06967*, 2017. 5, 111

[56] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv preprint arXiv:1704.00710*, 2017. 70

[57] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996. 86

[58] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 3, 5, 123

[59] 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. 29, 36, 88, 99, 118

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016. 36

[61] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3d photography. *TOG*, 2017. 125

[62] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016. 69

[63] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999*, pages 94–101. Springer, 1999. 125

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 86

[65] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 49

[66] Ronghang Hu and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. *arXiv*, 2020. 126

[67] Gary Huang, Marwan Mattar, Honglak Lee, and Erik G Learned-Miller. Learning to align from scratch. In *Advances in neural information processing systems*, pages 764–772, 2012. 29

[68] Jingwei Huang, Angela Dai, Leonidas Guibas, and Matthias Nießner. 3dlite: Towards commodity 3d scanning for content creation. *ACM Transactions on Graphics 2017 (TOG)*, 2017. 107

[69] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 74

[70] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 47, 51, 69

[71] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 6, 29, 33, 45, 47, 70, 71, 108, 110

[72] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 53, 59

[73] Tony Jebara and Alex P Pentland. *Discriminative, generative and imitative learning*. PhD thesis, PhD thesis, Media laboratory, MIT, 2001. 11

[74] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks.

In *Advances in Neural Information Processing Systems*, pages 667–675, 2016. 47

[75] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 38

[76] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 83, 88

[77] Angjoo Kanazawa, David W Jacobs, and Manmohan Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2016. 47

[78] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018. 146

[79] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 83, 87, 92, 93, 99, 106, 107

[80] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019. 146

[81] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017. 83

[82] Kevin Karsch, Zicheng Liao, Jason Rock, Jonathan T Barron, and Derek Hoiem. Boundary cues for 3d object shape recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2163–2170, 2013. 83

[83] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4, 81, 83, 84, 89, 97, 107, 110

[84] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. 105, 107, 117

[85] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 60, 75, 88, 114, 118, 131

[86] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013. 69

[87] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 86

[88] Chen Kong, Chen-Hsuan Lin, and Simon Lucey. Using locally corresponding cad models for dense 3d reconstructions from a single image. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition, 2017. 75

[89] Johannes Kopf, Michael F Cohen, and Richard Szeliski. First-person hyper-lapse videos. *TOG*, 2014. 126

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 29

[91] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015. 71

[92] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 47

[93] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence*, 16(2):150–162, 1994. 83

[94] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. 39

[95] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *IJCV*, 2009. 5

[96] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9(3):245–261, 1990. 86, 124, 129

[97] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 125

[98] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv*, 2020. 126, 146

[99] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7, 47, 49, 54, 71, 108

[100] Chen-Hsuan Lin, Rui Zhu, and Simon Lucey. The conditional lucas & kanade algorithm. In *European Conference on Computer Vision (ECCV)*, pages 793–808. Springer International Publishing, 2016. 5, 6, 32, 49, 61, 108

[101] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 7, 83, 107, 113, 116

[102] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 7

[103] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,

2019. 7, 8, 83, 89, 92, 125, 134

[104] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 7

[105] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. *arXiv preprint arXiv:2104.06405*, 2021. 8

[106] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 146

[107] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *arXiv preprint arXiv:1703.00848*, 2017. 47

[108] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. *arXiv preprint arXiv:1911.13225*, 2019. 82, 83, 86, 87

[109] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019. 4, 81, 83, 84, 89, 90, 91, 97, 99

[110] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 8293–8304, 2019. 83, 84, 86

[111] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 57

[112] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 83, 123, 126

[113] Jonathan L Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2014. 29

[114] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 89

[115] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 5, 11, 32

[116] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, 1981. 5, 11, 29, 30, 49, 108, 127

[117] Wei-Chiu D. Ma, Shenlong Wang, Jiayuan Gu, Sivabalan Manivasagam, Antonio Torralba, and Raquel Urtasun. Deep feedback inverse problem solver. In *European conference on computer vision*. Springer, 2020. 86

[118] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016. 50

[119] David Marr. Vision: A computational investigation into the human representation and processing of visual information, henry holt and co. *Inc., New York, NY*, 2(4.2), 1982. 17

[120] Iain Matthews and Simon Baker. Active appearance models revisited. *International journal of computer vision*, 60(2):135–164, 2004. 24, 107

[121] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 69

[122] Christopher Mei, Selim Benhimane, Ezio Malis, and Patrick Rives. Homography-based tracking for central catadioptric cameras. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 669–674. IEEE, 2006. 51, 60

[123] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 83, 89

[124] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6878–6887, 2019. 123

[125] 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. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 135, 139, 141, 142

[126] 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*, 2020. 8, 83, 86, 87, 88, 99, 123, 126, 128, 129, 131, 132, 133, 136, 138, 146

[127] 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. 107, 123, 125

[128] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011. 105, 125

[129] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Computer Vision (ICCV), 2019 IEEE International Conference on*. IEEE, 2019. 83

[130] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. *arXiv*, 2020. 126

[131] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable

volumetric rendering: Learning implicit 3d representations without 3d supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 82, 83, 84, 86, 87, 89, 90, 91, 92, 93, 97, 99

[132] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7688–7697, 2019. 93

[133] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2002. 11

[134] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. *arXiv*, 2018. 125

[135] Stanley Osher, Ronald Fedkiw, and K Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004. 88

[136] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 47, 71

[137] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 83, 92

[138] 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. 126, 130, 146

[139] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 47

[140] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM Transactions on graphics (TOG)*, volume 22, pages 313–318. ACM, 2003. 47

[141] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv*, 2020. 126

[142] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 47, 69, 78

[143] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4119–4128, 2018. 146

[144] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. *arXiv*, 2020. 126

[145] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems*, pages 1252–1260, 2015. 71

[146] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned radiance fields from a single view. *arXiv*, 2021. 126

[147] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Face alignment at 3000 fps via regressing local binary features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1685–1692, 2014. 32

[148] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems*, pages 4996–5004, 2016. 70, 72

[149] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 126

[150] Gernot Riegler and Vladlen Koltun. Stable view synthesis. *arXiv*, 2020. 126

[151] Gernot Riegler, Ali Osman Ulusoys, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 70

[152] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015. 118

[153] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 4, 107, 108, 116, 117, 121, 123, 125, 136, 139, 141

[154] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv*, 2020. 126

[155] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 126

[156] Ken Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*. ACM, 1985. 111

[157] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *Computer Vision–ECCV 2012*, pages 746–760, 2012. 53, 59

[158] Patrice Y Simard, David Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003. 29

[159] Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001. 15

[160] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 29

[161] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2016. 7, 82, 83, 86, 92, 123

[162] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 83

[163] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. 125

[164] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *IJCV*, 2008. 125

[165] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 53, 59

[166] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019. 123, 126

[167] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. *arXiv*, 2020. 126

[168] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1453–1460. IEEE, 2011. 42

[169] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *ICCV*, 1998. 125

[170] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, 2020. 125, 130

[171] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018. 86

[172] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016. 71, 75, 76

[173] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *arXiv preprint arXiv:1703.09438*, 2017. 70

[174] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wiley Online Library, 2020. 123

[175] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan

Yang. Deep image harmonization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 47

[176] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020. 123, 126

[177] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 4, 70, 81, 83, 92, 106, 107

[178] Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 107

[179] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 126

[180] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017. 124, 130

[181] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016. 47

[182] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *CVPR*, 2018. 125

[183] Chaoyang Wang, Chen Kong, and Simon Lucey. Distill knowledge from nrsfm for weakly supervised 3d pose learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 743–752, 2019. 87

[184] Chaoyang Wang, Chen-Hsuan Lin, and Simon Lucey. Deep nrsfm++: Towards 3d reconstruction in the wild. *arXiv preprint arXiv:2001.10090*, 2020. 93

[185] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. *arXiv preprint arXiv:1804.01654*, 2018. 107, 111

[186] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017. 97, 98

[187] Rui Wang, Martin Schworer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *ICCV*, 2017. 5, 125

[188] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pages 318–335. Springer, 2016. 69

[189] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. *arXiv preprint arXiv:1704.03414*, 2017. 47

[190] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf −−: Neural radiance fields without known camera parameters. *arXiv*, 2021. 126

[191] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 123, 126

[192] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011. 125

[193] 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 *Advances in Neural Information Processing Systems*, pages 82–90, 2016. 47, 69, 70, 78

[194] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*, pages 540–550, 2017. 92

[195] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 69

[196] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv*, 2020. 126

[197] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE, 2014. 82, 92, 98, 99

[198] Jianxiong Xiao, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2695–2702. IEEE, 2012. 112, 118, 119

[199] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 532–539, 2013. 5, 11, 14, 15, 32, 49, 54, 61

[200] Xuehan Xiong and Fernando De la Torre. Supervised descent method for solving nonlinear least squares problems in computer vision. *arXiv preprint arXiv:1405.0601*, 2014. 11

[201] Xuehan Xiong and Fernando De la Torre. Global supervised descent method. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2664–2673, 2015. 11, 14

[202] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016. 69

[203] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016. 47, 69, 70, 72, 75, 76, 83, 89, 97, 106, 107

[204] Anqi Joyce Yang, Can Cui, Ioan Andrei Bârsan, Raquel Urtasun, and Shenlong Wang. Asynchronous multi-view SLAM. In *ICRA*, 2021. 4, 125

[205] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 47

[206] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems*, pages 1099–1107, 2015. 71

[207] Raymond Yeh, Ziwei Liu, Dan B Goldman, and Aseem Agarwala. Semantic facial expression editing using autoencoded flow. *arXiv preprint arXiv:1611.09961*, 2016. 47

[208] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. *arXiv preprint arXiv:2012.05877*, 2020. 126, 134

[209] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018. 125

[210] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 126

[211] M Ersin Yumer and Niloy J Mitra. Learning semantic deformation flows with 3d convolutional networks. In *European Conference on Computer Vision*, pages 294–311. Springer, 2016. 70

[212] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv*, 2020. 126

[213] 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. 134

[214] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 53, 59

[215] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. 50

[216] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer, 2016. 47, 71, 108

[217] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 47, 108, 125

[218] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 123, 126

[219] Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A Efros. Learning a discriminative model for the perception of realism in composite images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3943–3951, 2015. 47

[220] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. 47, 69

[221] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017. 47

[222] Rui Zhu, Hamed Kiani Galoogahi, Chaoyang Wang, and Simon Lucey. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 57–65. IEEE, 2017. 107

[223] Rui Zhu, Chaoyang Wang, Chen-Hsuan Lin, Ziyan Wang, and Simon Lucey. Object-centric photometric bundle adjustment with deep shape prior. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 894–902. IEEE, 2018. 107, 109

[224] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *TOG*, 2004. 125