# REPRESENTING AND RECONSTRUCTING GENERAL NON-RIGID OBJECTS WITH NEURAL MODELS

EDITH TRETSCHK

Dissertation zur Erlangung des Grades der

*Doktorin der Ingenieurwissenschaften (Dr.-Ing.)*

der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken, 2023

| | |
|---|---|
| **Date of Colloquium** | January 29, 2024 |
| **Dean of the Faculty** | Jürgen Steimle |
| **Chair of the Committee** | Prof. Dr. Philipp Slusallek |
| **Reviewers** | Prof. Dr. Christian Theobalt |
| | Prof. Dr. Hans-Peter Seidel |
| | Prof. Dr. Lourdes Agapito |
| **Academic Assistant** | Dr. Rishabh Dabral |

*To Alma and Ella.*

# ABSTRACT

Digitizing the real world is a wide problem area at the intersection of computer vision and computer graphics and, lately, machine learning. Despite a lot of effort, creating virtual clones of real-world objects remains an unsolved scientific challenge. Still, it is of great interest as it enables interactions with the environment in augmented reality, digital clones for virtual reality, and consistent visual effects. While human-centered approaches are already advanced, the handling of general deformable objects is far less explored and the topic of this thesis.

To digitize an object, it first needs to be reconstructed from sensor observations and then represented in a suitable manner for downstream tasks. Many classical methods have explored these closely related areas. However, these reconstruction methods still fall short of practical applicability, and representing general deformable objects is unduly limited by hand-crafted priors. Over the past decade, neural techniques have led to great advancement in both areas. Meshes have become accessible to deep learning thanks to graph convolutions, graphics representations have expanded to include coordinate-based neural networks, and the entire reconstruction field has been revolutionized by neural radiance fields.

This thesis contributes to both areas. In the first part, it focuses on representing deformations and geometry. In particular, it introduces a low-dimensional deformation model. Unlike prior work that handcrafts these for specific categories, it can be trained for any general non-rigid object category via mesh auto-encoding using graph convolutions. Furthermore, by integrating insights from classical deformation modeling, it avoids artifacts common to prior work, which is purely learning-based.

Next, coordinate-based networks model geometry at infinite resolution but they do not generalize due to their global representation. This thesis makes them generalizable, thereby making these new models much easier to apply to general objects where training data is lacking.

In the second part, this thesis advances the reconstruction side. It extends neural radiance fields, which were previously restricted to static scenes, to deformable objects. This approach seeds a new category of methods for general non-rigid reconstruction from monocular input.

Finally, this thesis extends the previous method to handle large motions, a non-trivial endeavor due to backwards deformation modeling. Unlike prior work on general non-rigid reconstruction, it achieves time consistency even for studio-scale motion.

# ZUSAMMENFASSUNG

Die echte Welt digital zu klonen, ist ein weites Forschungsfeld an der Grenze von Computer Vision und Computergrafik und, seit kurzem, maschinellem Lernen. Trotz vieler Bemühungen ist das Erstellen digitaler Kopien von echten Objekten nach wie vor ein ungelöstes Problem. Nichtsdestotrotz ist es ein wichtiges Unterfangen, das viele Anwendungen hat: Interaktionen mit der Umwelt in Augmented Reality, das Erstellen digitaler Assets für Videospiele und konsistente visuelle Effekte. Methoden, die sich auf Menschen konzentrieren, sind bereits weit fortgeschritten. Allgemeine verformbare Objekte hingegen sind bisher nur wenig untersucht worden und Gegenstand dieser Arbeit.

Um ein echtes Objekt zu digitalisieren, muss es zunächst aus Sensormessungen rekonstruiert werden und dann passend für die eigentlichen Ziele repräsentiert werden. Viele klassische Methoden haben sich diese eng verwandten Gebiete angeschaut. Allerdings sind diese Rekonstruktionsmethoden nicht hinreichend praxistauglich und die Repräsentationen allgemeiner verformbarer Objekte sind übermäßig durch manuelle Annahmen eingeschränkt. Im letzten Jahrzehnt haben neuronale Techniken in beiden Gebieten zu großem Fortschritt geführt. Meshes sind dank Graph-Faltungen für Deep Learning zugänglich, koordinaten-basierte neuronale Netze haben Grafikrepräsentationen erweitert, und das gesamte Rekonstruktionsgebiet hat durch neuronale Radiance Fields eine Revolution durchlaufen.

Diese Arbeit bringt beide Gebiete voran. Der erste Teil dreht sich um das Modellieren von Verformungen und von Geometrie. Konkret wird ein niedrig-dimensionales Modell für Verformungen vorgestellt. Im Gegensatz zu existierenden Arbeiten, die diese Modelle speziell für bestimmte Objektkategorien entwerfen, kann das vorgestellte Modell für jede beliebige allgemeine verformbare Objektkategorie via Auto-Encoding für Meshes mit Graph-Faltungen trainiert werden. Des Weiteren vermeidet es ungewünschte Artefakte, die existierende Arbeiten aufgrund ihres reinen Maschinelles-Lernen-Ansatzes aufweisen, indem es Erkenntnisse aus der klassischen Computergrafik über das Modellieren von Verformungen einbindet.

Koordinaten-basierte Netze stellen Geometrie mit unendlicher Auflösung dar, lassen sich aber aufgrund ihrer globalen Repräsentation nicht auf beliebige Objekte anwenden. Diese Arbeit entfernt diese Beschränkung und vereinfacht damit die Anwendung dieser neuen Modelle auf allgemeine Objekte, die außerhalb der Trainingsdaten liegen.

Der zweite Teil dieser Arbeit dreht sich um Rekonstruktion. Zunächst werden neuronale Radiance Fields, die bisher auf unbewegliche Szenen beschränkt waren, auf verformbare Objekte erweitert. Diese Methoden begründet eine neue Richtung von Methoden zur Rekonstruktion von allgemeinen verformbaren Objekten aus monokularen Messungen.

Schließlich erweitert diese Arbeit die vorangegangene Methode, sodass diese auch große Bewegungen rekonstruieren kann. Das ist ein kompliziertes Unterfangen aufgrund der rückwärts gerichteten Verformungsrepräsentation. Im Gegensatz zu existierenden Arbeiten zur allgemeinen verformbaren Rekonstruktion bleibt diese Methode selbst für Bewegungen durch ein ganzes Studio zeitlich konsistent.

# ACKNOWLEDGMENTS

I am very grateful for having been able to spend so many years as a seminar student, a Hiwi, as part of a research immersion lab, as a Master student, and finally as a PhD candidate in the Graphics, Vision, and Video group, which eventually became the Visual Computing and Artificial Intelligence department during this long journey. It allowed me to pursue the research of my interest and to satisfy my curiosity in a great environment. It is a unique place and I hope it remains so for a long time.

In particular, I want to thank Christian Theobalt for having built this group (and department) into what it is today and for taking me on as a PhD candidate. The seminar sparked my interest in pursuing a PhD at all and I am very grateful for that. The guidance always struck a great balance between staying on topic while following wherever curiosity led.

I am also thankful for everyone I got to work and interact with. The fruitful discussions we had were greatly enjoyable. I especially want to thank everyone who helped me out with the main projects in the thesis over all the years and who lent an ear whenever I needed a second opinion: Vladislav Golyanik, Ayush Tewari, Michael Zollhöfer, Christoph Lassner, Carsten Stoll, and Aljaž Božič. A further thank you to everyone whose great projects I, in turn, got to be a part of: Soshi Shimada, Vikramjit Sidhu, Hsiao-yu Chen, Navami Kairanda, Harshil Bhatia, Maximilian Krahn, Marcel Seelbach Benkner, Lakshika Rathi, and Viktor Rudnev, as well as Michael Möller and Eddy Ilg. I also want to thank the competent administrative staff who keep everything running smoothly: Sabine Budde, Ellen Fries, everyone at IST, and our main administrator, Gereon Fox. And of course I want to thank my former and current office mates, Abhimitra Meka and Mohit Mendiratta. Finally, I want to thank the thesis committee members: Lourdes Agapito, Hans-Peter Seidel, and Christian Theobalt, and all the proofreaders of this thesis: Mallikarjun B R, Heming Zhu, Linjie Lyu, Soshi Shimada, Jian Wang, and Mohit Mendiratta, and Rishabh Dabral for being the "akademischer Beisitzer". And, last but not least, my family and friends for their support.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# 1 INTRODUCTION

## 1.1 MOTIVATION

The world around us consists of countless different entities, many of which are deformable. The most obvious example are humans, but we are also surrounded by animals and vegetation, or, more artificially, pieces of cloth, blankets, plush toys, and many others. When interacting with all of these, people are greatly aided by a 3D perception of them and their deformations (Yildirim et al., 2023), which yields a 4D perspective. As of now, however, computers still lack behind humans in their 4D understanding of the real world (Tretschk et al., 2023). To get closer to people's 4D understanding, computational methods are required that obtain a 4D analysis of the real world. In other words, it is necessary to *digitize* the real world. For the purposes of this thesis, digitization can be split into two steps: (1) creating (or *reconstructing*) a digital clone of the real-world entity that resembles its real counterpart in terms of geometry, appearance, and deformations, and then (2) modeling (or *representing*) these factors in an easy-to-use manner for certain desired goals. 4D digitization constitutes a fundamental problem at the intersection of computer vision (using camera measurements as input) and computer graphics (representing the entity) and is thus an important subject of study on its own.

In addition, digitization makes the real world accessible as an input to numerous downstream applications in various domains: Virtual reality, and telepresence in particular, benefits from high-quality novel-view synthesis of real-world entities (Lawrence et al., 2021; Orts-Escolano et al., 2016). Augmented reality and robotics require a 3D/4D model of the scene geometry for interactions with the real-world environment. Video game development can become easier if real objects can be cloned into virtual assets (Chen et al., 2022). VFX and social media might see creative new ways for visual content creation and editing. Finally, scientific fields such as physics and biology may analyze an object's motion given its reconstruction (Kairanda et al., 2022; Wang et al., 2021).

Many of these applications are centered around humans and thus a lot of prior work specializes on human bodies (Habermann et al., 2020; Liu et al., 2021; Zhao et al., 2022), faces (Egger et al., 2020b), or hands (Corona et al., 2022; Iwase et al., 2023; Qian et al., 2020). These methods already achieve almost photo-realistic results in certain settings. Similarly, arbitrary static objects can be reconstructed and modeled impressively well (Mildenhall et al., 2020; Park et al., 2019). However, the reconstruction of general deformable (or *non-rigid*) objects have seen significantly less work over the years and this thesis focuses on them instead. Thus, the ability to generalize to arbitrary object categories is central to the design of all methods presented in this thesis.

Two common themes appear again and again in this thesis: neural techniques and temporal correspondences. Traditionally, reconstructing and representing general non-rigid objects has been tackled by tracking or deforming a template mesh across time or by reconstructing a point cloud for each timestep. However, recent years have witnessed a revolution in the reconstruction field with the introduction of neural techniques from machine learning. In particular, geometric deep learning (Bronstein et al., 2021) enables deep learning on meshes via graph convolutions; coordinate-based neural networks (Xie et al., 2022) enable an entire new way of representing geometry and other attributes of an object; and differentiable volumetric rendering (Tewari et al., 2022) enables straightforward integration of rendering into gradient-based optimization schemes, thereby naturally connecting 2D images of the real world with a virtual 3D reconstruction. The methods presented in this thesis are each based on at least one of these neural techniques.

Finally, both when reconstructing and representing non-rigid objects, an ideal handling of them involves factoring out the deformations from the geometry and appearance. Such a factorization yields correspondences across time and deformations, and thereby a deeper analysis of the object. Sophisticated downstream applications benefit immensely from the resulting temporal consistency. For example, editing the appearance of an objects becomes tremendously easier if only a single canonical model needs to be adapted and the changes are then propagated across time via the object's deformations. This thesis encounters temporal correspondences in many different forms: when learning a motion model, when adding global semantics to local patches across different shapes, when reconstructing from monocular input, and when reconstructing large motions.

## 1.2 OVERVIEW

To successfully digitize an object, it first needs to be reconstructed from sensor observations and then represented in an easy-to-use manner, which in turn enables downstream tasks like editing or novel interactions. The main organization of this thesis is along this split: The first part focuses on representing general non-rigid objects and the second part focuses on reconstructing general non-rigid objects. In the first part, the two presented methods take 3D data as input and convert it into novel representations that better achieve a certain goal than the input data representations. In the second part, the two presented methods take 2D data as input and reconstruct 4D models from it. Although downstream applications are not the main focus, editing and novel deformations will be shortly touched upon by these methods. Next, the four methods contained in this thesis are motivated shortly.

One core aspect of modeling general non-rigid objects is their deformations. Category-specific parametric models, such as the human-body model SMPL (Loper et al., 2015), have become an influential paradigm for reconstructing and modeling deformable entities over the past two decades (Blanz and Vetter, 1999; Egger et al., 2020b). They represent the deformations of a certain category via a low-dimensional parameter space. However, their design is *hand-tailored* to the specific category they seek to represent (Loper et al., 2015; Romero et al., 2017; Zuffi et al., 2017). Is it instead possible to automatically build a low-dimensional motion model purely from registered 3D meshes, *regardless of the object category*? Unlike hand-tailored methods, a category-agnostic method cannot make strong assumptions about the deformations, for example that they are merely articulated rather than fully non-rigid. The first method presented in this thesis (Chapter 4) thus tackles this question by combining a hierarchical, graph-convolutional auto-encoder for meshes with expert knowledge about modeling general non-rigid deformations in computer graphics. This leads to a method that learns a low-dimensional motion representation while avoiding surface artifacts that previous methods, which do not incorporate graphics knowledge, suffer from. The method's general applicability is demonstrated on the categories of human bodies, faces, and hands, and on cloths.

Another core aspect of an object representation apart from deformations is geometry. While traditional representations like meshes or point clouds can easily be applied to arbitrary objects, coordinate-based neural networks (Park et al., 2019) are restricted to object categories within their training distribution. This is because they are, in their original form, global representations that represent an object via a single latent vector without spatial factorization. However, in other respects, such neural representations are *more* flexible than traditional representations. For example, they provide infinite resolution in principle and they empirically appear to be easier to optimize. Their inherently neural nature also promises the potential for novel problem settings that have previously been intractable. Addressing their shortcomings is thus of high interest. The second method presented in this thesis (Chapter 5) focuses on making coordinate-based networks generalize. Naïvely training on more diverse data is difficult in practice since such data is hard to obtain at scale. The method design thus needs to also allow for object categories not seen during training and that are far away from the training distribution. This includes generalizing to non-rigid object categories when training only on rigid objects. To achieve this, it represents geometry in a spatially factorized manner via local coordinate-based patch networks. This enables generalization since objects are similar at the local patch level, even across object categories. In addition, it also enables classical modeling of articulated deformations (for example, of human bodies) via modifying the positions and rotations of the patches, which even preserves correspondences.

The second half of the thesis concerns reconstruction. Early on, coordinate-based neural networks were only trained from full 3D supervision, like the previous method. However, they were later found to be trainable from 2D supervision (Mildenhall et al., 2020) as well. These neural radiance fields (NeRFs) reconstruct general static scenes so well that they enable photo-realistic novel-view synthesis. However, they originally could not handle deformations. The third method of this thesis (Chapter 6) is one of the first works to remedy this shortcoming. Using a single monocular video as input, a separate coordinate-based deformation network is trained jointly with a static neural radiance field that represents the scene's geometry and appearance. It assumes local rigidity, a widely applicable prior for general deformable objects, and thus generalizes to many real-world object categories. Furthermore, factorizing out the deformations leads to temporal consistency. This method thus establishes a novel category of methods to reconstruct general non-rigid objects from monocular input, which is important as the existing fields of Shape from Template (SfT) and Non-Rigid Structure from Motion (NRSfM) have plateaued in performance over the past years (Tretschk et al., 2023). That is because the task of reconstructing a general non-rigid object from monocular input is highly ambiguous and thus challenging. For example, depth ambiguity, occlusions, and a lack of correspondence annotations in the input necessitate a careful introduction of generic priors on the reconstruction. The presented method handles these difficulties via a simple design that, unlike concurrent work, does not use additional annotations like estimated depth maps, segmentation masks, optical flow, or 3D static points to work on real data, and, unlike SfT and NRSfM, reconstructs the background as well. It is thus easy to apply in practice.

Finally, a major shortcoming of the previous method is addressed: its inability to handle large motion. This inability stems from the strong prior assumptions required by the severely underconstrained problem setting. This also explains why estimating temporal correspondences for objects that undergo large motions has only been successfully tackled by category-specific methods that exploit, for example, strong human-specific priors on the deformations. In contrast, the fourth method presented in this thesis (Chapter 7) aims for long-term, long-range correspondences for *general* non-rigid objects. To this end, it optimizes its deformation and static-NeRF networks in an online manner. To concentrate on the main goal of handling large-scale motion, calibrated multi-view RGB videos captured in a recording studio are used as input to lessen the impact of occlusions. However, repetitive appearance still leads to ambiguity and requires a deformation prior to handle. Doing so naïvely as in prior work is not feasible in a reasonable amount of time and thus several design choices are made to reduce the optimization time per timestep. In addition, backwards deformation modeling leads to unintuitive difficulties, including issues specific to large motion, which need to be adressed for the method to work. The effectiveness of the method is demonstrated on studio-scale motion, which previous methods cannot handle in a time-consistent manner, and fine non-rigid deformations of different dresses.

## 1.3 STRUCTURE

This thesis is structured as follows.

- Chapter 1 motivates the thesis' topic of representing and reconstructing general non-rigid objects and provides an overview of the methods presented in this thesis. Furthermore, it describes the structure and contributions of this thesis.
- Chapter 2 discusses the wider literature on representing and reconstructing general objects, especially non-rigid objects.
- Chapter 3 introduces the narrower technical background on deformation types and neural models used in the thesis.
- Chapter 4 introduces a low-dimensional motion representation that combines geometric deep learning and classical graphics knowledge, and evaluates it against the state of the art.
- Chapter 5 presents a geometry representation that makes coordinate-based neural networks generalizable, and demonstrates this generalizability in extensive experiments.

- Chapter 6 describes a dynamic reconstruction method for monocular inputs that makes neural radiance fields applicable to non-rigid objects, and shows its effectiveness in various experiments.
- Chapter 7 introduces a dynamic reconstruction method that can handle large motion in a time-consistent manner in a studio setting, and evaluates both its novel-view synthesis quality and time consistency extensively.
- Chapter 8 summarizes the insights of this thesis and discusses future steps.

## 1.4 CONTRIBUTIONS

This thesis makes the following main contributions:

The contributions of Chapter 4 (published as Tretschk et al. (2020b)) are:

- A general-purpose mesh auto-encoder that learns a low-dimensional motion model purely from mesh datasets of moderate sizes for any deformable object category, while avoiding artifacts.
- A novel differentiable embedded deformation layer that models the deformable meshes using lower-dimensional deformation graphs with physically interpretable deformation parameters, thereby decoupling the parameterization of object motion from the mesh resolution and introducing local spatial coherence via vertex skinning.

The contributions of Chapter 5 (published as Tretschk et al. (2020a)) are:

- A novel neural, coordinate-based surface representation that can generalize to different object categories.
- An optimization scheme for a mid-level, spatially factorized representation of surfaces, at the level of patches, that is based on auto-decoding and can be trained with only few shapes.

The contributions of Chapter 6 (published as Tretschk et al. (2021)) are:

- A free viewpoint rendering method that only requires a monocular video of the dynamic scene. The spatio-temporal camera trajectory for test-time novel view synthesis can differ significantly from the trajectory of the input video. Moreover, dense correspondences relating arbitrary (input or novel) frames can be extracted.

- A rigidity network which can segment the scene into non-rigid foreground and rigid background without direct supervision.
- Regularizers on the estimated deformations which constrain the problem by encouraging small, volume-preserving deformations.
- Several extensions for handling of view dependence and multi-view data, and applications for simple scene editing.

The contributions of Chapter 7 (published as Tretschk et al. (2024)):

- An end-to-end differentiable, time-consistent 4D reconstruction method for general dynamic scenes undergoing large motions captured with static, multi-view RGB cameras.
- A general approach to make backward deformation models work for larger motion via frame-by-frame tracking and "extending the deformation field".

## 1.5 PUBLICATIONS

The methods presented in this thesis are also publicly available in the following self-contained works:

- Edith Tretschk et al. (2020b). "DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects." In: *European Conference on Computer Vision (ECCV)*
- Edith Tretschk et al. (2020a). "PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations." In: *European Conference on Computer Vision (ECCV)*
- Edith Tretschk et al. (2021). "Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video." In: *International Conference on Computer Vision (ICCV)*
- Edith Tretschk et al. (2024). "SceNeRFlow: Time-Consistent Reconstruction of General Dynamic Scenes." In: *International Conference on 3D Vision (3DV)*

Further significant contributions were made to the following works, which are not part of this thesis:

- Soshi Shimada et al. (2019). "DispVoxNets: Non-Rigid Point Set Alignment with Supervised Learning Proxies." In: *International Conference on 3D Vision (3DV)*

- Vikramjit Sidhu et al. (2020). "Neural Dense Non-Rigid Structure from Motion with Latent Space Constraints." In: *European Conference on Computer Vision (ECCV)*
- Ayush Tewari et al. (2022). "Advances in Neural Rendering." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*
- Hsiao yu Chen et al. (2022). "Virtual Elastic Objects." In: *Computer Vision and Pattern Recognition (CVPR)*
- Navami Kairanda et al. (2022). "$\phi$-SfT: Shape-from-Template with a Physics-Based Deformation Model." In: *Computer Vision and Pattern Recognition (CVPR)*
- Marcel Seelbach Benkner et al. (2023). "QuAnt: Quantum Annealing with Learnt Couplings." In: *International Conference on Learning Representations (ICLR)*
- Edith Tretschk et al. (2023). "State of the Art in Dense Monocular Non-Rigid 3D Reconstruction." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*
- Harshil Bhatia et al. (2023). "CCuantuMM: Cycle-Consistent Quantum-Hybrid Matching of Multiple Shapes." In: *Computer Vision and Pattern Recognition (CVPR)*

# 2 RELATED WORK

This chapter discusses the wider literature on representing and reconstructing general objects and scenes, with a focus on non-rigidity. Next, Chapter 3 focuses on the specific technical background needed for this thesis.

## 2.1 REPRESENTING GENERAL OBJECTS

The first part of the thesis is about representing general objects and scenes: Chapter 4 introduces a deformation model for general non-rigid objects, and Chapter 5 introduces a patch-based geometry representation for general objects and scenes. This section thus discusses prior work on geometry and deformation modeling of general objects.

### 2.1.1 *Global Geometry Representations*

Widely-used classical data structures can be adapted for geometric deep learning such as voxel grids (Choy et al., 2016), point clouds (Qi et al., 2017), and meshes (Wang et al., 2018). To alleviate the memory limitations and speed-up training, improved versions of voxel grids with hierarchical space partitioning (Riegler et al., 2017) and tri-linear interpolation (Shimada et al., 2019) were recently proposed. All these data structures enable a limited level of detail given a constant memory size. In contrast, other representations such as sign distance functions (SDF) (Curless and Levoy, 1996) represent surfaces implicitly as the zero-crossing of a volumetric level set function.

Recently, neural counterparts of implicit representations and approaches operating on them were proposed in the literature (Chen and Zhang, 2019; Mescheder et al., 2019; Michalkiewicz et al., 2019; Park et al., 2019). Similarly to SDFs, these methods extract surfaces as zero level sets or decision boundaries, while differing in the type of the learned function. Thus, DeepSDF is a learnable variant of SDFs (Park et al., 2019), whereas Mescheder et al. (2019) train a spatial classifier (indicator function) for regions inside and outside of the scene. In theory, both methods allow for surface extraction at unlimited resolution. Neural implicit functions have already demonstrated their effectiveness and robustness in many follow-up works and applications such as single-view 3D reconstruction

(Liu et al., 2019a; Saito et al., 2019) as well as static (Sitzmann et al., 2019b) and dynamic (Niemeyer et al., 2019) object representation. SAL (Atzmon and Lipman, 2020) performs shape completion from noisy full raw scans.

Unlike the aforementioned global approaches, the method presented in Chapter 5 is based on local patches and hence generalizes much better, for example to new categories.

### 2.1.2 *Patch-Based Geometry Representations*

Ohtake et al. (2003) use a combination of implicit functions for versatile shape representation and editing. Several neural techniques use mixtures of geometric primitives as well (Deng et al., 2019; Deprelle et al., 2019; Genova et al., 2019; Lombardi et al., 2021; Tulsiani et al., 2017; Williams et al., 2020). The latter have been shown to be helpful abstractions in such tasks as shape segmentation, interpolation, classification and recognition, as well as 3D reconstruction. Tulsiani et al. (2017) learn to assemble shapes of various categories from explicit 3D geometric primitives (*e.g.*, cubes and cuboids). Their method discovers a consistent structure and allows to establish semantic correspondences between the samples. AtlasNet (Groueix et al., 2018) stitches local patches, which can lead to artifacts. Genova et al. (2019) further develop the idea and learn a general template from data which is composed of implicit functions with local support. Due to the function choice, *i.e.*, scaled axis-aligned anisotropic 3D Gaussians, shapes with sharp edges and thin structures are challenging for their method. In *CVXNets* (Deng et al., 2019), solid objects are assembled in a piecewise manner from convex elements. This results in a differentiable form which is directly usable in physics and graphics engines. Deprelle et al. (2019) decompose shapes into learnable combinations of deformable elementary 3D structures. VoronoiNet (Williams et al., 2020) is a deep generative network which operates on a differentiable version of Voronoi diagrams. NASA (Deng et al., 2020) focuses on articulated deformations.

In contrast to other patch-based approaches, Chapter 5 presents a method whose learned patches are not limited to hand-crafted priors but instead are more flexible and expressive.

### 2.1.3 *Classical Mesh Deformations.*

Chapter 4 introduces an embedded deformation layer. It is inspired by as-rigid-as-possible modeling (Sorkine and Alexa, 2007) and the method of Sumner et al. (2007) for mesh editing and manipulation. Deformation cages (Nieto and Susín, 2012) are another widespread paradigm for deforming general non-rigid objects. While these methods have been shown to be very useful for mesh manipulation in computer graphics, Chapter 4 is the first time a model-based regularizer is used in a mesh auto-encoder.

### 2.1.4 *Auto-Encoding-Based Mesh Deformations.*

Chapter 4 proposes a mesh auto-encoder. Previously, several mesh auto-encoders with various applications were introduced. A new hierarchical variational mesh auto-encoder with fully connected layers for facial geometry parameterization learns an accurate face model from small databases and accomplishes depth-to-mesh fitting tasks (Bagautdinov et al., 2018). Tan et al. (2018a) introduce a mesh auto-encoder with a rotation-invariant mesh representation as a generative model. Their network can generate new meshes by sampling in the latent space and can perform mesh interpolation. To cope with meshes of arbitrary connectivity, they use fully-connected layers and do not explicitly encode neighbor relations.

#### 2.1.4.1 *Graph Convolutions*

Graph convolutions generalize image convolutions from the image grid to arbitrary graphs. They allow for intuitive learned computation on meshes. This enables learning of correspondences between shapes, shape retrieval (Boscaini et al., 2016; Masci et al., 2015; Monti et al., 2017), and segmentation (Yi et al., 2017). Masci et al. (2015) proposed geodesic CNNs operating on Riemannian manifolds for shape description, retrieval, and correspondence estimation. Boscaini et al. (2016) introduce spatial weighting functions based on simulated heat propagation and projected anisotropic convolutions. Monti et al. (2017) extend graph convolutions to variable patches through Gaussian mixture model CNNs. In FeaSTNet (Verma et al., 2018), the correspondences between filter weights and graph neighborhoods with arbitrary connectivities are established dynamically from the learned features. The localized spectral interpretation of Defferrard et al. (2016) is based on recursive feature learning with Chebyshev polynomials and has linear evaluation complexity.

In particular, graph convolutions can be used for mesh auto-encoding, thereby learning a deformation model. Tan et al. (2018b) train a network with graph convolutions to extract sparse localized deformation components from meshes. Their method is suitable for large-scale deformations and meshes with irregular connectivity. Gao et al. (2018) transfer mesh deformations by training a generative adversarial network with a cycle consistency loss to map shapes in the latent space, while a variational mesh auto-encoder encodes deformations. The Convolutional facial Mesh Auto-encoder (CoMA) of Ranjan et al. (2018) allows to model and sample stronger deformations compared to previous methods and supports asymmetric facial expressions. The Neural 3DMM of Bouritsas et al. (2019) exploits the fixed ordering of neighboring vertices in its graph convolution. It improves quantitatively over CoMA due to better training parameters and task-specific graph convolutions.

Unlike prior mesh-auto-encoding approaches, Chapter 4 adds a model-based regularizer inspired by classical graphics knowledge, thereby reducing artifacts.

## 2.2 RECONSTRUCTING GENERAL OBJECTS

The second part of the thesis deals with reconstruction and its related work is described next. Before discussing the reconstruction of non-rigid objects, adjacent areas are first touched upon. In particular, an overview of the reconstruction of rigid objects (which cannot handle deformations), novel-view synthesis (which neglects 3D consistency), and temporal correspondence estimation (which neglects the geometry and appearance) is provided. Then, approaches for non-rigid reconstruction without and with neural radiance fields (NeRF (Mildenhall et al., 2020)) are discussed. Chapter 6 and Chapter 7 introduce NeRF-based methods for non-rigid reconstruction. A recent survey (Tretschk et al., 2023) offers an introduction to the fundamentals of reconstruction.

### 2.2.1 *Rigid Objects*

Several algorithms use variants of shape-from-silhouette to approximate real scene geometry, such as visual hull reconstruction or visual hulls improved via multi-view photo-consistency in Kutulakos and Seitz (2000) and Starck et al. (2006). While reconstruction is fast and feasible with fewer cameras, the coarse approximate geometry introduces rendering artifacts, and the reconstruction is usually limited to the separable foreground.

An emerging algorithm class uses neural networks to augment or replace established graphics and vision concepts for reconstruction and novel-view rendering. Given a depth image, the network of Sinha et al. (2017) reconstructs rigid objects from single images. Similarly, Groueix et al. (2018) reconstruct object surfaces from a point cloud or single monocular image with an atlas parameterization. Meshes can be reconstructed via differentiable mesh rendering (Kato et al., 2018). The approaches of Kurenkov et al. (2018) and Jack et al. (2018) modify the identity of a predefined object-class template to match the observed object appearance in an image. The Pixel2Mesh approach of Wang et al. (2018) reconstructs an accurate mesh of an object in a segmented image. Initializing the 3D reconstruction with an ellipsoid, their method gradually deforms it until the appearance matches the observation. The template-based approaches (Jack et al., 2018; Kurenkov et al., 2018), as well as Pixel2Mesh (Wang et al., 2018), produce complete 3D meshes. These have spawned a wide field of follow-up works on reconstructing static scenes (Eslami et al., 2018; Flynn et al., 2019; Hedman et al., 2018; Meshry et al., 2019; Mildenhall et al., 2020; Nguyen-Phuoc et al., 2018; Riegler and Koltun, 2020; Sitzmann et al., 2019a,b).

### 2.2.2 *Static and Dynamic Novel-View Synthesis*

Novel-view synthesis is closely related to 3D reconstruction. It differs in its focus on 2D view synthesis, thereby foregoing 3D/multi-view consistency.

Early methods for image-based novel and free-viewpoint rendering combined traditional concepts of multi-view camera geometry, explicit vision-based 3D shape and appearance reconstruction, and classical computer graphics or image-based rendering. These methods are based on light fields (Buehler et al., 2001; Gortler et al., 1996; Levoy and Hanrahan, 1996), multi-view stereo to capture dense depth maps (Zhang et al., 2003), layered depth images (Shade et al., 1998), or representations using 3D point clouds (Agarwal et al., 2011; Liu et al., 2010; Schönberger and Frahm, 2016), meshes (Matsuyama et al., 2004; Tung et al., 2009) or surfels (Carceroni and Kutulakos, 2002; Pfister et al., 2000; Waschbüsch et al., 2005) for dynamic scenes. Some approaches use 3D templates and combine vision-based reconstruction with appearance modeling to enable free-viewpoint video relighting, *e.g.*, by estimating reflectance models under general lighting or under controlled light stage illumination (Guo et al., 2019; Li et al., 2013; Nagano et al., 2015; Theobalt et al., 2007).

#### 2.2.2.1 *Neural Approaches*

Thies et al. (2019) combine neural textures with the classical graphics pipeline for novel view synthesis of static objects and monocular video re-rendering. Other methods combine explicit dynamic scene reconstruction and traditional graphics rendering with neural re-rendering (Kim et al., 2019, 2018; Martin Brualla et al., 2018; Yoon et al., 2020). Several approaches address generating images of humans in new poses (Balakrishnan et al., 2018; Ma et al., 2018; Neverova et al., 2018; Sarkar et al.,

2020) or body reenactment from monocular videos (Chan et al., 2019). Shysheya et al. (2019) propose a neural rendering approach for human avatars with texture warping. Zhu et al. (2018) leverage geometric constraints and optical flow for synthesizing novel views of humans from a single image. Although some of these methods use an underlying 3D feature parametrization, all of them ultimately forego spatial consistency, in contrast to full 3D reconstruction.

### 2.2.3 *3D Correspondence Estimation*

For temporal data, the seminal work by Vedula et al. (1999) introduced scene flow as the 3D equivalent of optical flow, which led to many approaches tackling the problem (Basha et al., 2013; Huguet and Devernay, 2007; Quiroga et al., 2014; Teed and Deng, 2021; Vogel et al., 2015; Yoon et al., 2018), as a recent survey by Zhai et al. (2021) summarizes. Lately, learning-based methods (Gu et al., 2019; Li et al., 2021a; Liu et al., 2019b; Mittal et al., 2020) focus on 3D point clouds as input. More generally, non-rigid 3D point-cloud registration (Deng et al., 2022) estimates correspondences of provided 3D point clouds. Unlike full reconstruction, correspondence estimation neglects the geometry and appearance aspect and solely focuses on the deformations.

### 2.2.4 *Non-Rigid Objects*

Multiple lines of research target 4D reconstruction without relying on neural radiance fields (Mildenhall et al., 2020). As the recent survey Tretschk et al. (2023) discusses, monocular RGB input has traditionally been tackled by Non-Rigid Structure-from-Motion (Garg et al., 2013; Graßhof and Brandt, 2022; Kumar et al., 2018; Parashar et al., 2020; Russell et al., 2012) and Shape-from-Template methods (Bartoli et al., 2015; Casillas-Perez et al., 2021; Kairanda et al., 2022; Ngo et al., 2015; Salzmann et al., 2007). Dense non-rigid structure from motion requires dense point tracks over input images, which are then factorized into camera poses and non-rigid 3D states per view (Garg et al., 2013; Kumar et al., 2018; Sidhu et al., 2020). The correspondences are usually obtained with dense optical flow methods, which makes them prone to occlusions and inaccuracies, and which can have a detrimental effect on the reconstructions. Monocular template-based methods do not assume dense matches and rely on a known 3D state of a deformable object (a 3D template), which is then tracked across time (Ngo et al., 2015; Perriollat et al., 2011; Xu et al., 2018; Yu et al., 2015), or a training dataset with multiple object states (Golyanik et al., 2018). Obtaining templates for complex objects and scenes is often non-trivial and requires specialized setups. Due to the restrictive input setting, these methods either only handle small motions or lack time consistency.

Slightly larger motions are possible with a single RGB-D camera (Zollhöfer et al., 2018), although these methods (Bozic et al., 2020a,b; Cai et al., 2022; Guo et al., 2015, 2017; Innmann et al., 2016; Lin et al., 2022; Newcombe et al., 2015; Slavcheva et al., 2017, 2018; Zollhöfer et al., 2014) all update (Curless and Levoy, 1996) both previously unseen *and previously seen* parts of the canonical model over time during their online optimization, thereby losing time consistency (except for the template-based Zollhöfer et al. (2014)).

Multi-view input has seen comparatively less work over the years (Goldluecke and Magnor, 2004; Larsen et al., 2007; Tung et al., 2009; Zhang et al., 2003). Fusion4D (Dou et al., 2016; Orts-Escolano et al., 2016) operates in real time but also modifies the canonical model over time. Mustafa et al. (2016) track a mesh through a temporal sequence according to optical flow and refine its topology for each timestamp, but do not reconstruct the appearance. Unlike such multi-view mesh-tracking approaches (Cagniart et al., 2010; Zhao et al., 2022), the method presented in Chapter 7 is end-to-end differentiable and can easily integrate recent advances in neural scene representations (Tewari et al., 2022). Bansal et al. (2020) obtain impressive but time-inconsistent novel-view results.

#### 2.2.4.1 *Neural Approaches*

Only a few supervised neural learning approaches for 3D reconstruction from monocular images tackle the deformable nature of non-rigid objects. Several methods (Golyanik et al., 2018; Pumarola et al., 2018; Shimada et al., 2019) train networks for deformation models with synthetic thin plates datasets. These approaches can infer non-rigid states of the observed surfaces such as paper sheets or membranes. Still, their accuracy and robustness on real images are limited. Bednařík et al. (2018) propose an encoder-decoder network for texture-less surfaces relying on shading cues. They train on a real dataset and show an enhanced reconstruction accuracy on real images, but support only trained object classes. Fuentes-Jimenez et al. (2018) train a network to deform an object template for depth map recovery. They achieve impressive results on real image sequences but require an accurate 3D model of every object in the scene, which restricts the method's practicality.

Beyond these early works, weakly supervised learning-based approaches (Kanazawa et al., 2018; Kokkinos and Kokkinos, 2021; Li et al., 2020; Wu et al., 2021; Yang et al., 2022; Yoon et al., 2020) have grown in popularity over the last years.

### 2.2.5 *Dynamic NeRFs*

2D neural rendering (Tewari et al., 2020) and 3D neural scene representations (Tewari et al., 2022) based on NeRF (Mildenhall et al., 2020) have seen great success in recent years. Neural Volumes (Lombardi et al., 2019) is an early neural scene representation that learns object models which can be animated and rendered from novel views, given multi-view video data. Apart from it, current methods (Du et al., 2021; Gao et al., 2021; Li et al., 2021b; Menapace et al., 2022; Park et al., 2021a,b; Pumarola et al., 2021; Xian et al., 2021) for dynamic scene reconstruction with neural representations build on NeRF. The method in Chapter 6 is among the first to extend NeRF to deformable reconstruction. Some approaches focus on surface (Johnson et al., 2022), fast (Fang et al., 2022; Guo et al., 2022), generalizable (Wang et al., 2022), or depth-supported (Attal et al., 2021) reconstruction. The methods in Chapter 6 and Chapter 7 are most related to this field of work but differ in their goal: they do not primarily aim for novel-view synthesis but rather for a *time-consistent* reconstruction. A few prior methods (Liu et al., 2022; Pumarola et al., 2021) do obtain time consistency but either only in synthetic (Pumarola et al., 2021) or small-motion (Liu et al., 2022) settings, including the method presented in Chapter 6. Furthermore, most works use monocular input (Gao et al., 2022) and only a few (Li et al., 2022; Liu et al., 2022; Song et al., 2022; Wang et al., 2022) explore multi-view input, where the latter forego long-term time consistency or only handle small motion (Liu et al., 2022). Unlike these approaches, Chapter 7 achieves time consistency even for large motion.

# 3 BACKGROUND

After discussing the broader literature in the previous chapter, this chapter focuses on the specific technical background of the thesis. In particular, this chapter first discusses deformation types and how to model them (Sec. 3.1), and then introduces the neural techniques used by the methods presented in this thesis, as mentioned in Chapter 1: graph convolutions (Sec. 3.2), coordinate-based neural networks (Sec. 3.3), and volumetric neural rendering (Sec. 3.4).

## 3.1 DEFORMATION TYPES AND PARAMETRIZATIONS

As this thesis focuses on non-rigid objects and scenes, handling their deformations is crucial throughout. This section discusses mathematical deformation types and then presents practical parametrizations for them. It is based on the Fundamentals section in Tretschk et al. (2023).

### 3.1.1 *Deformation Types*

There are different types of deformations that can be applied to an object. This section focuses on the deformation itself and thus abstracts away the specific geometry. Therefore, deformations may be described by a space warping $d$: $\mathbf{x}' = d(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{x}' \in \mathbb{R}^3$ are canonical and deformed points in space, respectively. In the following, each deformation type generalizes the previous type.

**Static** objects do not move locally or globally, with the deformation trivially defined as $\mathbf{x}' = \mathbf{x}$.

**Rigid** objects may move around globally, without changing their shape and size. Specifically, the object can change its orientation and position, but cannot scale or shear. Mathematically, the deformation allows for a rotation and translation: $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$, for a 3D rotation matrix $\mathbf{R} \in SO(3)$ and a 3D translation $\mathbf{t} \in \mathbb{R}^3$. If $\det R = -1$ is also allowed, it is a *reflection*. If $\det R = \pm 1$ and a scaling $s \in \mathbb{R}$ is allowed, it is a *similarity* transform: $\mathbf{x}' = s\mathbf{R}\mathbf{x} + \mathbf{t}$.

**Affine** deformations are also global like rigid deformations, except that now shearing and scaling are allowed: $\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{t}$, where $\mathbf{A} \in \mathbb{R}^{3\times 3}$ is an invertible linear mapping.

**Articulated** deformations generalize the previous classes to (piece-wise) ensembles of $N$ local rigid/affine deformations $\{d_i\}_{i=1}^N$, where $d_i(\mathbf{x}) = \mathbf{R}_i\mathbf{x} + \mathbf{t}_i$ in the rigid case:

$$\mathbf{x}' = d_i(\mathbf{x}) \quad \text{if } \mathbf{x} \in U_i, \tag{3.1}$$

where $\{U_i \subset \mathbb{R}^3\}_{i=1}^N$ is a partition of $\mathbb{R}^3$. $U_i$ is a *local part* and deforms according to its own associated deformation $d_i$. Thus, the deformed object is obtained by deforming each local part $U_i$ of the canonical object according to its own rigid/affine transform $d_i$. Humans or animals are sometimes modeled with articulated deformations. There, each large bone (e. g.an upper arm) would be a local part. Chapter 5 shows an application where a deep implicit surface representation undergoes articulated deformations.

**Non-Rigid** objects undergo arbitrary deformations as they respond naturally to applied forces, constraints and contacts with themselves or obstacles. This most generic formulation describes the behavior of most real, physical objects and deformation $d$ is any (in general non-linear) mapping that displaces an undeformed point to a deformed one:

$$\mathbf{x}' = d(\mathbf{x}), \text{ where } d : \mathbb{R}^3 \to \mathbb{R}^3. \tag{3.2}$$

For instance, cloths deform freely and often form fine local surface deformations such as folds and wrinkles. Chapter 4, Chapter 6, and Chapter 7 model non-rigid deformations of general objects.

### 3.1.2 *Parametrizing Deformations*

There are several commonly used possibilities for parametrizing the deformations described in the previous section.

**Physics Simulation.** The most accurate way to model deformations is by imposing the true physical laws that govern an object's behavior. Despite being ideal in principle, physics simulation is difficult to model completely and to implement, and is computationally expensive. Thus, the vast majority of 3D reconstruction works may take inspiration from physics but ultimately apply approximations, as discussed next.

**Template Offsets.** A simple deformation model consists of per-vertex offsets of a template, which is particularly popular due to recent methods trained on general image-collections (Kanazawa et al., 2018). As per-vertex offsets are severely underconstrained, they allow for fully non-rigid deformations. However, for stable optimization, they are often combined with additional priors to constrain them.

**Skinning.** As deformations tend to be spatially smooth, it is common to skin a detailed template mesh to a coarser graph embedded in 3D (whose parameters are thus the deformation parameters) by specifying skinning weights. When deforming the coarse graph, its deformations are transferred to the detailed template by interpolating according to these weights. Embedded graphs (Sumner et al., 2007) are common for general objects and skeleton skinning (itself based on a kinematic chain) is common for category-specific models (Loper et al., 2015). Skeleton skinning induces an articulated deformation on the mesh, while high-resolution embedded graphs approximate non-rigid behavior well. Chapter 4 uses embedded graphs to model non-rigid deformations.

**Linear Subspace Models.** Instead of deforming a single template, linear subspace models linearly combine a limited number of basis deformations to obtain the deformed surface. The coefficients of this combination are often globally constant across space. This kind of parametrization is common in non-rigid structure from motion (Chapter 2). Linear subspaces are sometimes used for the underlying skeletons in skinned models (Loper et al., 2015). Models that parametrize a low-dimensional space, especially by linear combinations of some (usually fixed) basis, are called parametric models. Chapter 4 learns a low-dimensional space of the deformation parameters of an embedded graph.

**Neural Volumetric Deformations.** Implicit parametrizations tend to use volumetric backward deformation models to avoid the need for directly accessing the surface. Rather than taking, for example, a vertex from a mesh and deforming it (forward) in space, backwards models take any point in space and deform it backwards into the canonical space, where the geometry and appearance lie (*e.g.* in the form of a mesh or a neural radiance field, as discussed below). The earlier work Neural Volumes (Lombardi et al., 2019) uses an explicit mixture of affine warps, while recent neural-rendering methods use an MLP parametrization. In the fully non-rigid case, such an MLP can output an offset per point in 3D space, while more articulated deformations benefit from an $SE(3)$ output (Park et al., 2021a). Chapter 6 uses volumetric backwards models to parametrize deformations, and Chapter 7 makes them work for large deformations.

## 3.2 GRAPH CONVOLUTIONS

Deep convolutional neural networks (CNNs) allow to effectively capture contextual information of input data modalities and can be trained for various tasks. Lately, image convolutions operating on regular grids have been generalized to more general topologically connected structures such as meshes and two-dimensional manifolds (Bruna et al., 2013; Niepert et al., 2016). Their design space is much larger and thus less straightforward than convolutions on regular image or voxel grids. In general, graph convolutions are designed to preserve image convolutions' main properties: locality and an inductive bias of translation invariance. They thus aggregate information over a local graph neighborhood with the filter weights being constant across the graph. Next, two specific realizations are presented that are employed in Chapter 4.

### 3.2.1 *Spiral Graph Convolutions*

Given an $F_{in}$-channel feature tensor $\mathbf{x} \in \mathbb{R}^{N \times F_{in}}$, where the features are defined at the $N$ graph nodes, let $\mathbf{x}_{*,i} \in \mathbb{R}^N$ denote the $i$-th input graph feature map. The complete output feature tensor, that stacks all $F_{out}$ feature maps, is denoted as $\mathbf{y} \in \mathbb{R}^{N \times F_{out}}$. The graph convolutions act without stride, *i.e.,* the input graph resolution equals the output resolution.

Chapter 4 mainly works with spiral graph convolutions (Bouritsas et al., 2019). Let $\mathbf{x}^T_{n,*} \in \mathbb{R}^{F_{in}}$ denote the feature vector of graph node $n$. Assuming a fixed topology of the graph, some ordering of the neighboring nodes of graph node $n$ can be determined: $n_0, \dots, n_s, \dots, n_{S-1}$. Bouritsas et al. (2019) pick a spiral ordering that starts with $n_0 = n$ and proceeds along the 1-ring $(n^1_0, n^1_1, \dots)$, then the 2-ring $(n^2_0, n^2_1, \dots)$, and so on. The spiral ordering is thus given by: $n, n^1_0, n^1_1, \dots, n^2_0, n^2_1, \dots$ All the spirals of the graph are ultimately cut to a fixed length $S$ and zero-padded if necessary. The output of the spiral convolution is then defined as:

$$\mathbf{y}^T_{j,*} = \sum_{s=0}^{S-1} G_s \cdot \mathbf{x}^T_{n_s,*} \quad , \tag{3.3}$$

where $G_s \in F_{out} \times F_{in}$ is a trainable matrix.

### 3.2.2 *Spectral Graph Convolutions*

The second type of graph convolutions is based on fast localized spectral filtering (Defferrard et al., 2016), used in CoMA (Ranjan et al., 2018). The $j$-th output graph feature map $\mathbf{y}_{*,j} \in \mathbb{R}^N$ is computed as follows:

$$\mathbf{y}_{*,j} = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(\mathbf{L}) \cdot \mathbf{x}_{*,i} \quad . \tag{3.4}$$

Here, $\mathbf{L}$ is the normalized Laplacian matrix of the graph and the filters $g_{\theta_{i,j}}(\mathbf{L})$ are parameterized using Chebyshev polynomials of order $K$. More specifically,

$$g_{\theta_{i,j}}(\mathbf{L}) = \sum_{k=0}^{K-1} \theta_{i,j,k} \cdot T_k(\widetilde{\mathbf{L}}), \tag{3.5}$$

where $\theta_{i,j,k} \in \mathbb{R}$ and $\widetilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}$, with $\lambda_{max} = 2$ being an upper bound on the spectrum of $\mathbf{L}$. The Chebychev polynomial $T_k$ is defined as $T_k(x) = 2x \cdot T_{k-1}(x) - T_{k-2}(x)$, $T_1(x) = x$, and $T_0(x) = 1$.

This leads to $K$-localized filters that operate on the $K$-neighborhoods of the nodes. Each filter $g_{\theta_{i,j}}(\mathbf{L})$ is parameterized by $K$ coefficients, which in total leads to $F_{in} \times F_{out} \times K$ trainable parameters for each graph convolution layer.

## 3.3 COORDINATE-BASED NETWORKS

Several works (Mescheder et al., 2019; Park et al., 2019) concurrently introduced coordinate-based neural networks. A coordinate-based network $f$ represents the volume enclosing a scene implicitly rather than, for example, by regressing a mesh. Volumetric representations are particularly suitable because they allow for flexibility in the optimization whereas surface representations like meshes are in practice difficult to optimize. Furthermore, coordinate-based networks avoid discretization and, in principle, represent the scene at infinite resolution.

Specifically, an object can be represented by encoding its signed distance field in a neural network: $f(\mathbf{x}) = s(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^3$ is any 3D point in space and $s(\mathbf{x})$ is its signed distance to the object surface. Therefore, the networks need to be queried for each point that is to be evaluated, which makes them (in their original form) much more computationally expensive than classical representations.

The first works required full 3D supervision for training, with several attempts (Niemeyer et al., 2020; Sitzmann et al., 2019a,b) at 2D supervision showing insufficient performance for wide adaption. Coordinate-based networks have grown tremendously in popularity over the last years and are now also known as *neural fields*, as a recent survey (Xie et al., 2022) discusses in detail.

Chapter 5 replaces the global network with local patch networks to improve the generalization capabilities of coordinate-based networks.

## 3.4 NEURAL RADIANCE FIELDS

NeRF (Mildenhall et al., 2020) enables coordinate-based neural networks to be trained from 2D image supervision via volumetric rendering. In contrast to previous attempts at 2D supervision, it makes two crucial design choices: (1) the use of a density rather than an SDF representation, and (2) backpropagating gradients into all points sampled along a ray rather than only into points directly on the surface. Both of these relax the optimization problem to the right degree. A more technical design choice that enables sufficient spatial resolution beyond the natural smoothness induced by an MLP is positional encoding. Positional encoding makes the fine-grained high-frequency components of the coordinate input easily accessible to the network. To this end, it applies sinusoidal functions with different frequencies to the input and feeds the concatenation of all of them into the MLP. NeRFs have seen quick and widespread adaptation to numerous related fields, as a recent survey (Tewari et al., 2022) discusses. Chapter 6 and Chapter 7 extend NeRF to deformable scenes.

NeRF renders a 3D volume into an image by accumulating color, weighted by accumulated transmittance and density, along camera rays. The 3D volume is parametrized by an MLP $\mathbf{v}(\mathbf{x}, \mathbf{d}) = (\mathbf{c}, o)$ that regresses an RGB color $\mathbf{c} = \mathbf{c}(\mathbf{x}, \mathbf{d}) \in [0,1]^3$ and an opacity $o = o(\mathbf{x}) \in [0,1]$ for a point $\mathbf{x} \in \mathbb{R}^3$ on a ray with direction $\mathbf{d} \in \mathbb{R}^3$.

Consider a pixel $(u,v)$ of an image $\hat{\mathbf{c}}_i$. For a pinhole camera, the associated ray $\mathbf{r}_{u,v}(j) = \mathbf{o} + j\mathbf{d}(u,v)$ can be calculated using $\mathbf{R}_i, \mathbf{t}_i$, and $\mathbf{K}_i$, which yield the ray origin $\mathbf{o} \in \mathbb{R}^3$ and ray direction $\mathbf{d}(u,v) \in \mathbb{R}^3$. Integrating along the ray from the near plane $j_n$ to the far plane $j_f$ of the camera frustum yields the final color $\mathbf{c}$ at $(u,v)$:

$$\mathbf{c}(\mathbf{r}_{u,v}) = \int_{j_n}^{j_f} V(j) \cdot o(\mathbf{r}_{u,v}(j)) \cdot \mathbf{c}(\mathbf{r}_{u,v}(j), \mathbf{d}(u,v))\, dj \ , \tag{3.6}$$

with $V(j) = \exp(-\int_{j_n}^{j} o(\mathbf{r}_{u,v}(s))\, ds)$ being the accumulated transmittance along the ray from $j_n$ up to $j$. In practice, the integrals are approximated by discrete samples $\mathbf{x}$ along the ray. NeRF employs a coarse volume $\mathbf{v}_c$ with network weights $\theta_c$ and a fine volume $\mathbf{v}_f$ with network weights $\theta_f$. Both volumes have the same architecture, but do not share weights: $\theta = \theta_c \,\dot{\cup}\, \theta_f$. When rendering a ray, $\mathbf{v}_c$ is accessed first at uniformly distributed samples along the ray. These coarse samples are used to estimate the transmittance distribution, from which fine samples are sampled. $\mathbf{v}_f$ is then evaluated at the combined set of coarse and fine sample points.

# Part I

# Representing General Non-Rigid Objects

# 4 DEMEA: DEEP MESH AUTOENCODERS FOR NON-RIGIDLY DEFORMING OBJECTS

The first part of this thesis is about representing general non-rigid objects. Three aspects need to be accounted for when representing an object in computer graphics: its geometry, its appearance, and its deformations. This chapter looks at the third aspect. How can neural techniques be used to model deformations for general non-rigid objects, in particular in a low-dimensional manner?

Mesh autoencoders are commonly used for dimensionality reduction, sampling and mesh modeling. This chapter proposes a general-purpose DEep MEsh Autoencoder (DEMEA) which adds a novel embedded deformation layer to a graph-convolutional mesh autoencoder (also published as Tretschk et al. (2020b)). The embedded deformation layer (EDL) is a differentiable deformable geometric proxy which explicitly models point displacements of non-rigid deformations in a lower dimensional space and serves as a local rigidity regularizer. DEMEA decouples the parameterization of the deformation from the final mesh resolution since the deformation is defined over a lower dimensional embedded deformation graph. A large-scale study on four different datasets of deformable objects confirms the effectiveness of the proposed method. Reasoning about the local rigidity of meshes using EDL allows DEMEA to achieve higher-quality results for highly deformable objects, compared to directly regressing vertex positions. Finally, this chapter demonstrates multiple applications of DEMEA, including non-rigid 3D reconstruction from depth and shading cues, non-rigid surface tracking, as well as the transfer of deformations over different meshes.

## 4.1 INTRODUCTION

With the increasing volume of datasets of deforming objects enabled by modern 3D acquisition technology, the demand for compact data representations and compression grows. Dimensionality reduction of mesh data has multiple applications in computer graphics and vision, including shape retrieval, generation, interpolation, and completion. Crucially, when focusing on registered meshes (as this chapter does), dimensionality reduction becomes low-dimensional deformation modeling. Apart from earlier classical methods discussed in Chapter 2, recent deep

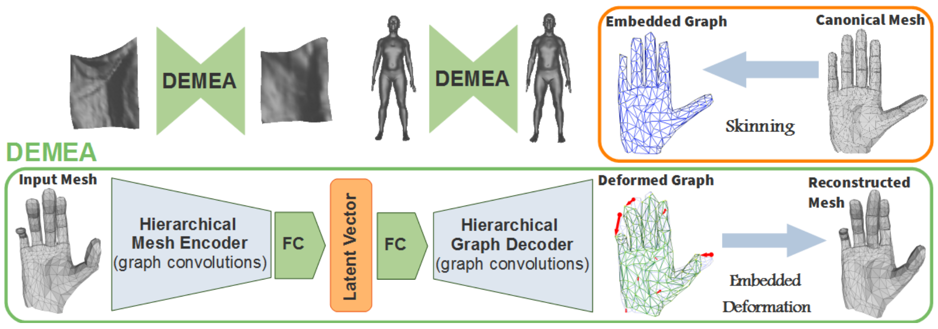


Figure 4.1: Pipeline: DEMEA encodes a mesh using graph convolutions on a mesh hierarchy. The graph decoder first maps the latent vector to node features of the coarsest graph level. A number of upsampling and graph convolution modules infer the node translations and rotations of the embedded graph. An embedded deformation layer applies the node translations to a template graph, against which a template mesh is skinned. With the node rotations and the skinning, this deformed graph allows reconstructing a deformed mesh.

convolutional autoencoder networks can *learn* highly compact mesh representations (Bagautdinov et al., 2018; Bouritsas et al., 2019; Ranjan et al., 2018; Tan et al., 2018b).

Dynamic real-world objects do not deform arbitrarily. While deforming, they preserve topology, and nearby points are more likely to deform similarly compared to more distant points. Current convolutional mesh autoencoders exploit this coherence by learning the deformation properties of objects directly from data and are already suitable for mesh compression and representation learning. On the other hand, they do not explicitly reason about the deformation field in terms of local rotations and translations. This chapter shows that explicitly reasoning about the local rigidity of meshes enables higher-quality results for highly deformable objects, compared to directly regressing vertex positions.

At the other end of the spectrum, mesh manipulation techniques such as As-Rigid-As-Possible Deformation (Sorkine and Alexa, 2007) and Embedded Deformation (Sumner et al., 2007) only require a single mesh and enforce deformation properties, such as smoothness and local rigidity, based on a set of hand-crafted priors. These hand-crafted priors are effective and work surprisingly well, but since they do not model the real-world deformation behavior of the physical object, they often lead to unrealistic deformations and artifacts in the reconstructions.

This chapter proposes a general-purpose mesh autoencoder with a model-based deformation layer, combining the best of both worlds, *i.e.* supervised learning with deformable meshes and a novel *differentiable embedded deformation* layer that models the deformable meshes using lower-dimensional deformation graphs with physically interpretable deformation parameters. While the core of DEep MEsh Autoencoder (DEMEA) learns the deformation model of objects from data using the state-of-the-art convolutional mesh autoencoder (CoMA) (Ranjan et al., 2018), the novel embedded deformation layer decouples the parameterization of object motion from the mesh resolution and introduces local spatial coherence via vertex skinning. DEMEA is trained on mesh datasets of moderate sizes that have recently become available (Bednařík et al., 2018; Bogo et al., 2017; Loper et al., 2014; Malik et al., 2018). DEMEA is a general mesh autoencoding approach that can be trained for any deformable object class. DEMEA is evaluated on datasets of three objects with large deformations like articulated deformations (body, hand) and large non-linear deformations (cloth), and one object with small localized deformations (face). Quantitatively, DEMEA outperforms standard convolutional mesh autoencoder architectures in terms of vertex-to-vertex distance error. Qualitatively, DEMEA produces visually higher fidelity results due to the physically based embedded deformation layer. DEMEA enables several applications in computer vision and graphics. Once trained, the decoder of the autoencoder can be used for shape compression, high-quality depth-to-mesh reconstruction of human bodies and hands, and even poorly textured RGB-image-to-mesh reconstruction for deforming cloth.

The low-dimensional latent space learned by DEMEA is meaningful and well-behaved, as different applications of latent space arithmetic demonstrate. Thus, DEMEA provides a well-behaved general-purpose category-specific generative model of highly deformable objects.

## 4.2 METHOD

This section describes the architecture of the proposed DEMEA, which learns a low-dimensional deformation model from large datasets of registered meshes.

To start, its auto-encoder architecture is based on spiral graph convolutions (Bouritsas et al., 2019) that are defined on a multi-resolution mesh hierarchy (Sec. 4.2.1). Importantly, at the end of the decoder, it uses an embedded deformation layer to decouple the complexity of the learned deformation field from the actual mesh resolution (Sec. 4.2.2). The deformation is represented relative to a canonical mesh $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ with $N_v$ vertices $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N_v}$, and edges $\mathbf{E}$. To this end, the encoder-decoder uses a coarse deformation graph and the embedded deformation layer to drive the deformation of the final high-resolution mesh, see Fig. 4.1. This architecture is trained with a simple reconstruction loss (Sec. 4.2.4), and,

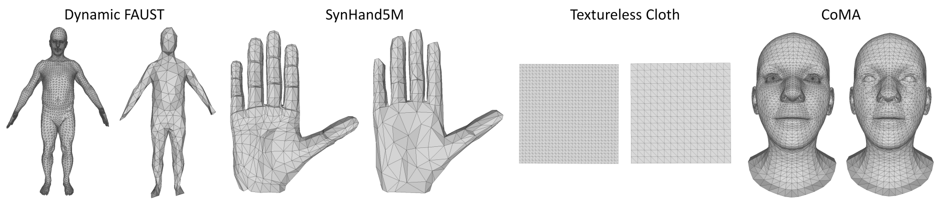


Figure 4.2: Template mesh and the corresponding embedded deformation graph pairs automatically generated using Cignoni et al. (2008).

given paired data, can be extended to image input (Sec. 4.2.5). Finally, this section concludes with low-level architecture details (Sec. 4.2.6).

### 4.2.1 *Mesh Hierarchy*

DEMEA is a convolutional mesh autoencoder. Similar to image autoencoders up- and downsampling the feature grids they act on, DEMEA's up- and downsampling is defined over a multi-resolution mesh hierarchy, inspired by the CoMA (Ranjan et al., 2018) architecture. DEMEA employs a hierarchy with five resolution levels, where the finest level is the mesh. The mesh hierarchy is generated using the CoMA code (Ranjan et al., 2018).

Given the multi-resolution mesh hierarchy, up- and downsampling operations (Ranjan et al., 2018) for feature maps defined on the graph are applied. To enable this, during downsampling, the nodes of the coarser level need to be a subset of the nodes of the next finer level. A feature map can be transferred to the next coarser level by a similar naïve subsampling operation. The inverse operation, *i.e.*, feature map upsampling, is implemented based on a barycentric interpolation of close features. During edge collapse, each collapsed node gets projected onto the closest triangle of the coarser level. The barycentric coordinates of this closest point with respect to the triangle's vertices then allow to define the interpolation weights.

**Embedded Graphs in the Hierarchy.** The design so far follows Ranjan et al. (2018). However, DEMEA requires additional steps. Given a canonical mesh, a corresponding coarse embedded deformation graph needs to be picked. In practice, the first or the second level of the automatically generated mesh hierarchy can be used as the embedded graph. Instead, this chapter proposes to use more uniform embedded graphs, see Fig. 4.2.

To this end, the embedded graph can be computed fully automatically based on quadric edge collapses (Garland and Heckbert, 1997) of the canonical mesh. Using MeshLab's (Cignoni et al., 2008) implementation of quadric edge collapse decimation with default settings works well. (Note that the embedded graph uses the same number of graph nodes as used by Ranjan et al. (2018).) Experiments with different ways of obtaining better embedded graphs, including hand-crafting them, yielded no major differences, except that graphs generated by Ranjan et al. (2018) were too non-uniform. Among the tested methods, MeshLab constitutes the least involved method of obtaining a uniform graph.

The deformation graph is used as one of the two levels immediately below the mesh in the mesh hierarchy (depending on the resolution of the graph) of the autoencoder. When generating the mesh hierarchy, the subset relationship between levels needs to be preserved. However, the quadric edge collapse algorithm of Ranjan et al. (2018) might delete nodes of the embedded graph when computing intermediate levels between the mesh and the embedded graph. To ensure that those nodes are not removed, the cost of removing them from levels that are at least as fine as the embedded graph is set to infinity. Furthermore, since the embedded graph needs to be a subset of the mesh, graph nodes obtained this way need to be projected to their closest vertices. This may lead to multiple nodes projecting to the same vertex. A greedy assignment from nodes to vertices resolves this issue: looping over all nodes, the current node is assigned to its closest vertex that is not yet taken by another node. This enables a fully automatic generation of the mesh hierarchy for this embedded graph.

### 4.2.2 *Embedded Deformation Layer (EDL)*

The embedded deformation layer models a space deformation that maps the vertices of the canonical template mesh $\mathbf{V}$ to a deformed version $\hat{\mathbf{V}}$. Suppose $\mathcal{G} = (\mathbf{N}, \mathbf{E})$ is the embedded deformation graph (Sumner et al., 2007) with $L$ canonical nodes $\mathbf{N} = \{\boldsymbol{g}_l\}_{i=1}^{L}$ and $K$ edges $\mathbf{E}$, with $\boldsymbol{g}_l \in \mathbb{R}^3$. The global space deformation is defined by a set of local, rigid, per-graph node transformations. Each local rigid space transformation is defined by a tuple $\boldsymbol{T}_l = (\boldsymbol{R}_l, \boldsymbol{t}_l)$, with $\boldsymbol{R}_l \in \mathbf{SO}(3)$ being a rotation matrix and $\boldsymbol{t}_l \in \mathbb{R}^3$ being a translation vector. Parameterizing the rotation matrices based on three Euler angles enforces that $\boldsymbol{R}_l^{\mathsf{T}} = \boldsymbol{R}_l^{-1}$ and $\det(\boldsymbol{R}_l) = 1$. Each $\boldsymbol{T}_l$ is anchored at the canonical node position $\boldsymbol{g}_l$ and maps every point $\boldsymbol{p} \in \mathbb{R}^3$ to a new position in the following manner (Sumner et al., 2007):

$$\boldsymbol{T}_l(\boldsymbol{p}) = \boldsymbol{R}_l[\boldsymbol{p} - \boldsymbol{g}_l] + \boldsymbol{g}_l + \boldsymbol{t}_l. \tag{4.1}$$

To obtain the final global space deformation $\mathbf{G}$, the local per-node transformations are linearly combined:

$$\mathbf{G}(\boldsymbol{p}) = \sum_{l \in \mathcal{N}_{\boldsymbol{p}}} w_l(\boldsymbol{p}) \cdot \boldsymbol{T}_l(\boldsymbol{p}) \ . \tag{4.2}$$

Here, $\mathcal{N}_{\boldsymbol{p}}$ is the set of approximate closest deformation nodes. The linear blending weights $w_l(\boldsymbol{p})$ for each position are based on the distance to the respective deformation node (Sumner et al., 2007):

$$w_l(\boldsymbol{p}) = \exp\left(\frac{-\|\boldsymbol{g}_l - \boldsymbol{p}\|^2}{2 \cdot \sigma^2}\right), \tag{4.3}$$

where $\sigma \in \mathbb{R}$ is determined heuristically as follows:

$$\sigma = \sigma_0 \cdot d_{max} \cdot \frac{1}{\sqrt{L}}, \tag{4.4}$$

where $\sigma_0 = \frac{2}{3}$ and $d_{max}$ is the maximum Euclidean distance between any two mesh vertices, a proxy for the absolute scale of the mesh. Note that $\mathcal{N}_{\boldsymbol{p}}$, $w_l$ and $\sigma$ are pre-computed on the template mesh and graph and are kept fixed within each dataset. DEMEA uses $|\mathcal{N}_{\boldsymbol{p}}| = 6$ for embedded graphs on the first level and $|\mathcal{N}_{\boldsymbol{p}}| = 12$ for embedded graphs on the second level.

Finally, the deformed mesh $\hat{\mathbf{V}} = \mathbf{G}(\mathbf{V})$ is obtained by applying the global space deformation to the canonical template mesh $\mathbf{V}$. The free parameters are the local per-node rotations $\boldsymbol{R}_l$ and translations $\boldsymbol{t}_l$, *i.e.*, $6L$ parameters with $L$ being the number of nodes in the graph. These parameters are input to the embedded deformation layer and are regressed by the graph convolutional decoder.

### 4.2.3 *Differentiable Space Deformation*

The novel EDL is fully differentiable and can be used during network training to decouple the parameterization of the space deformation from the resolution of the final high-resolution output mesh. The reconstruction loss can thus be defined on the final high-resolution output mesh, while still enabling backpropagating the errors via the skinning transform to the coarse parameterization of the space deformation. Thus, DEMEA enables finding the best space deformation by only supervising the final output mesh.

### 4.2.4 *Training*

DEMEA is trained end-to-end in Tensorflow (Abadi et al., 2015) using Adam (Kingma and Ba, 2015). Its loss is a dense geometric per-vertex $\ell_1$-loss with respect to the ground-truth mesh:

$$\mathcal{L}_{vertex} = \frac{1}{N_v} \sum_{i=1}^{N_v} \|\hat{\mathbf{v}}_i - \mathbf{v}_i^*\|_1, \tag{4.5}$$

where $\hat{\mathbf{v}}_i$ is the $i$-th deformed vertex and $\mathbf{v}_i^*$ is the $i$-th ground-truth vertex. The loss is averaged across the batch. The deformed vertex can be either directly regressed (CA, MCA, FCA) or it can be computed via EDL (DEMEA, FCED). For the latter case, Eq. 4.2 is used. DEMEA is trained with Tensorflow 1.5.0 (Abadi et al., 2015) on Debian with an Nvidia V100 GPU.

All experiments use a learning rate of $10^{-4}$ and default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ for Adam. The training takes 50 epochs for Dynamic Faust, 30 epochs for SynHand5M, 50 epochs for the CoMA dataset and 300 epochs for the Cloth dataset. The batch size is 8.

### 4.2.5 *Reconstructing Meshes from Images/Depth*

DEMEA can be adapted to take images as input. This image/depth-to-mesh network consists of an image encoder and a mesh decoder, see Fig. 4.7a. The mesh decoder is initialized from the corresponding mesh auto-encoder, the image/depth encoder is based on a ResNet-50 (He et al., 2016) architecture, and the latent code is shared between the encoder and decoder. The ResNet-50 component is initialized using pre-trained weights from ImageNet (Deng et al., 2009). To obtain training data, the meshes are rendered into synthetic depth maps. For RGB input, paired data is necessary. The training uses the same settings as for mesh auto-encoding.

### 4.2.6 *Network Architecture Details*

In the following, more details of the encoder-decoder architectures are provided. Fig. 4.3 contains the low-level architecture.
**Encoding Meshes.** Input to the first layer of the mesh encoder is an $N_v \times 3$ tensor that stacks the coordinates of all $N_v$ vertices. DEMEA applies four *downsampling modules*. Each module applies a graph convolution and is followed by a downsampling to the next coarser level of the mesh hierarchy. DEMEA uses spiral graph convolutions (Bouritsas et al., 2019)

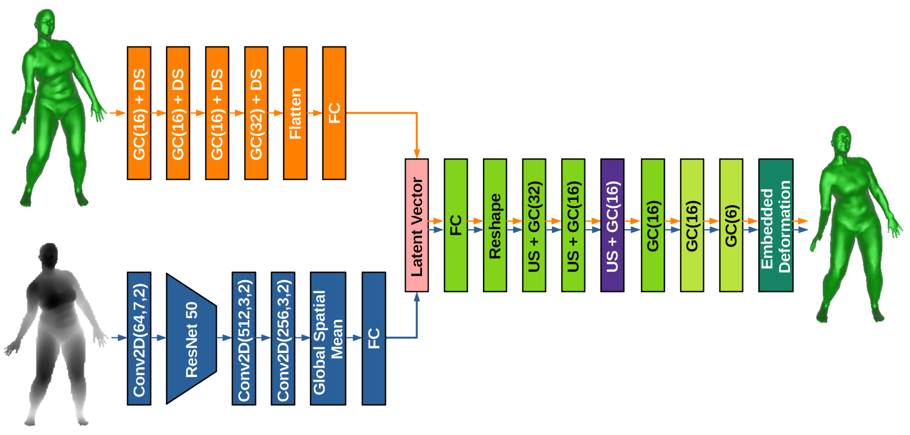


Figure 4.3: The low-level architecture of DEMEA (orange path) and the depth-to-mesh network (blue path). Note that the two paths are not trained simultaneously. *GC(f)* is a graph-convolutional layer with $f$ output features. *DS* is a downsampling layer and *US* is an upsampling layer. *Conv2D(f,k,s)* is a 2D convolution with $f$ output features, kernel size $k \times k$, and stride $s$.

and similarly applies an ELU non-linearity after each convolution. Finally, the output of the final module is fed into a fully connected layer followed by an ELU non-linearity to obtain a latent space embedding.

**Encoding Images/Depth.** To encode images/depth, a 2D convolutional network maps color/depth input to a latent space embedding. Input to this encoder are images of resolution 256 × 256 pixels. Removing the first convolutional layer of the ResNet-50 (He et al., 2016) architecture enables it to take a single or three-channel input image. (The first 2D convolutional layer in the depth encoder has no activation function.) Furthermore, its final non-convolutional layers are removed and two additional convolution layers added at the end instead, which are followed by global average pooling. Finally, a fully connected layer with a subsequent ELU non-linearity maps the activations to the latent space.

**Decoding Graphs.** The task of the graph decoder is to map from the latent space back to the embedded deformation graph. First, a fully connected layer in combination with reshaping yields the input to the graph convolutional *upsampling modules*. DEMEA uses a sequence of three or four upsampling modules until the resolution level of the embedded graph is reached. The fourth upsampling module (*i.e.* upsampling layer followed by a graph convolution) is only used for higher-resolution embedded graphs. Each upsampling module first up-samples the features to the next finer graph resolution and then performs a graph convolution, which is then followed by an ELU non-linearity. Then, two graph

| | Mesh | 1st | 2nd | 3rd | 4th |
|---|---|---|---|---|---|
| DFaust (Bogo et al., 2017) | 6890 | 1723 | **431** | 108 | 27 |
| CoMA (Ranjan et al., 2018) | 5023 | **1256** | 314 | 79 | 20 |
| SynHand5M (Malik et al., 2018) | 1193 | **299** | 75 | 19 | 5 |
| Cloth (Bednařík et al., 2018) | 961 | **256** | 100 | 36 | 16 |

Table 4.1: Number of vertices on each level of the mesh hierarchy. Bold levels denote the embedded graph. Note that except for Cloth these values were computed automatically based on Ranjan et al. (2018).

convolutions with ELUs refine the features, followed by a final convolution without an activation function. The resulting tensor is passed to the embedded deformation layer. The spiral graph convolutions use the default settings of Bouritsas et al. (2019) to determine the length of the spirals. The spectral graph convolutions always use $K = 6$, except for the last two layers, which use $K = 2$ for local refinement.

## 4.3 RESULTS

In this section, DEMEA is evaluated quantitatively and qualitatively on several challenging datasets. First, the baselines and datasets are introduced in Sec. 4.3.1. Then, Sec. 4.3.2 conducts an ablation study of the embedded deformation layer. Finally, Sec. 4.3.3 shows that the proposed method achieves state-of-the-art results for mesh auto-encoding. The next section (Sec. 4.4) then shows applications, namely reconstruction from RGB images and depth maps and that the learned latent space enables well-behaved latent arithmetic.

### 4.3.1 *Evaluation Settings*

**Metric.** Since registered meshes are available, the reconstructions are quantitatively evaluated via average per-vertex Euclidean errors.
**Datasets.** Experiments with body (Dynamic Faust, DFaust (Bogo et al., 2017)), hand (SynHand5M (Malik et al., 2018)), textureless cloth (Cloth (Bednařík et al., 2018)), and face (CoMA (Ranjan et al., 2018)) datasets demonstrate DEMEA's generality. Tab. 4.1 gives the number of graph nodes used on each level of the hierarchical architecture. All meshes live in metric space.
*DFaust (Bogo et al., 2017).* The training set consists of 28,294 meshes. The test set splits off two identities (female 50004, male 50002) and two dynamic performances, *i.e., one-leg jump* and *chicken wings*. Overall, this

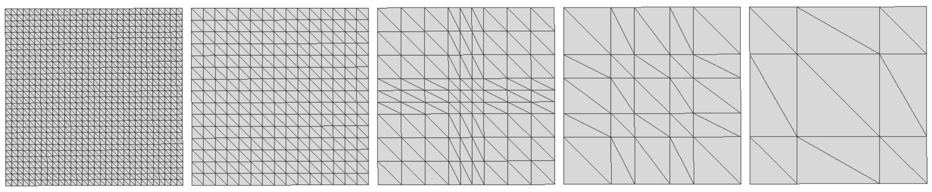


Figure 4.4: The Cloth hierarchy.

results in a test set with 12,926 elements. For the depth-to-mesh results, the synthetic depth maps from the DFaust training set are insufficient for generalization, *i.e.,* the test error is high. Thus, more pose variety is necessary for DFaust for the depth-to-mesh experiments. To this end, 28$k$ randomly sampled poses from the CMU Mocap[1] dataset are added to the training data, where the identities are randomly sampled from the SMPL (Loper et al., 2015) model (14$k$ female, 14$k$ male). Similarly, 12$k$ such samples are added to the test set (6$k$ female, 6$k$ male).

*Textureless Cloth (Bednařík et al., 2018).* The *textureless cloth* data set of Bednařík et al. (2018) allows to evaluate DEMEA on general non-rigidly deforming surfaces. It contains real depth maps and images of a white deformable sheet — observed in different states and differently shaded — as well as ground-truth meshes. In total, pre-filtering yields 3,861 meshes with consistent edge lengths. 3,167 meshes are used for training and 700 meshes are reserved for evaluation. Since the canonical mesh is a perfectly flat sheet, it lacks geometric features, which causes downsampling methods like Garland and Heckbert (1997), Ranjan et al. (2018), and Cignoni et al. (2008) to introduce severe artifacts. Hence, the entire mesh hierarchy for this dataset is generated procedurally, see Fig. 4.4. This hierarchy is also used to train the other methods in this section.

*SynHand5M (Malik et al., 2018).* For the experiments with hands, 100$k$ random meshes from the synthetic *SynHand5M* dataset of Malik et al. (2018) are used and additionally rendered to obtain the corresponding depth maps. The training set is comprised of 90$k$ meshes, and the remaining 10$k$ meshes are used for evaluation.

*CoMA (Ranjan et al., 2018).* The training set contains 17,794 meshes of the human face in various expressions (Ranjan et al., 2018). The test set contains two challenging expressions, *i.e., high smile* and *mouth extreme*. This gives 2,671 test meshes in total.

**Baseline Architectures.** DEMEA is compared to a number of strong baselines.

*Convolutional Baseline.* The *convolutional ablation (CA)* is a version of DEMEA where the ED layer is replaced by learned upsampling modules

1 mocap.cs.cmu.edu

| | DFaust | | SynHand5M | | Cloth | | CoMA | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 32 | 8 | 32 | 8 | 32 | 8 | 32 |
| w/ GL | 8.92 | 2.75 | 9.02 | 2.95 | **11.26** | 6.45 | 1.38 | 0.99 |
| w/ LP | 7.71 | **2.22** | **8.00** | 2.52 | 11.46 | 7.96 | 1.25 | **0.79** |
| DEMEA | **6.69** | 2.23 | 8.12 | **2.51** | 11.28 | **6.40** | **1.23** | 0.81 |

Table 4.2: EDL integration ablation. Average per-vertex errors on the test sets of DFaust (*cm*), SynHand5M (*mm*), textureless cloth (*mm*) and CoMA (*mm*) for 8 and 32 latent dimensions are reported.

that upsample to the mesh resolution. In this case, the extra refinement convolutions occur on the level of the embedded graph. *Modified CA (MCA)* is an architecture where the refinement convolutions are moved to the end of the network, such that they operate on mesh resolution.
*Fully-Connected Baseline. FC ablation (FCA)* is an an almost-linear baseline. The input is given to a fully-connected layer, after which an ELU is applied. The resulting latent vector is decoded using another FC layer that maps to the output space. Finally, the *FCED* is variant where the fully-connected decoder maps to the deformation graph, which the embedded deformation layer (EDL) in turn maps to the full-resolution mesh.

## 4.3.2 *Ablation Study*

The first question is the exact manner of integrating the EDL into the training. DEMEA regresses node positions and rotations and then uses the EDL to obtain the deformed mesh, on which the reconstruction loss is applied.

As an alternative, the *graph loss (GL)* encourages the regressed graph nodes positions to be close to the ground-truth vertex positions. It is an $\ell_1$ reconstruction loss directly on the graph node positions where the vertex positions of the input mesh that correspond to the graph nodes are used as ground truth. The graph nodes $\mathbf{N}$ are a subset of the mesh vertices $\mathbf{V}$, and the vertex index corresponding to a graph node $l$ is denoted as $i_l$. Then, the graph loss is:

$$\mathcal{L}_{graph} = \frac{1}{L} \sum_{l=1}^{L} \|\mathbf{t}_l - \mathbf{v}^*_{i_l}\|_1. \tag{4.6}$$

The GL setting uses the EDL only at test time to map to the full mesh, but not for training. Although the trained network predicts graph node positions $\boldsymbol{t}_l$ at test time, it does not regress graph node rotations $\boldsymbol{R}_l$ which are necessary for the EDL. It computes the missing rotation for

| | DFaust | | SynHand5M | | Cloth | | CoMA | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 32 | 8 | 32 | 8 | 32 | 8 | 32 |
| Spiral | 6.69 | **2.23** | **8.12** | **2.51** | **11.28** | **6.40** | **1.23** | **0.81** |
| Spectral | **6.56** | 2.40 | 8.74 | 3.83 | 11.76 | 6.52 | 1.40 | 0.98 |

Table 4.3: Comparison of DEMEA with spiral and spectral graph convolutions. Average per-vertex errors on the test sets of DFaust (in *cm*), SynHand5M (in *mm*), textureless cloth (in *cm*) and CoMA (in *mm*) are reported.

each graph node $l$ as follows: Assuming that each node's neighborhood transforms roughly rigidly, it solves a small Procrustes problem that computes the rigid rotation between the 1-ring neighborhoods of $l$ in the template graph and in the regressed network output. It then directly uses this rotation as $\boldsymbol{R}_l$.

An alternative is to estimate the local Procrustes rotation *inside the network during training (LP)*. The same loss (Eq. 4.5) makes DEMEA learn to regress the translation parameters, $\boldsymbol{t}_l$. However, instead of also learning to regress the rotation parameters of EDL, $\boldsymbol{R}_l$, as in DEMEA, they can alternatively be obtained via local Procrustes as described in the previous paragraph. Here, backpropagation through the rotation computation is turned off to avoid training instabilities.

Tab. 4.2 shows the quantitative results using the average per-vertex Euclidean error. Using the EDL during training leads to better quantitative results, as the network is aware of the skinning function and can move the graph nodes accordingly. In addition to being an order of magnitude faster than LP, regressing rotations either gives the best results or is close to the best. DEMEA therefore uses the EDL with regressed rotation parameters during training in all further experiments.

DEMEA uses spiral graph convolutions (Bouritsas et al., 2019), but spectral graph convolutions (Defferrard et al., 2016) also give similar results, as Tab. 4.3 shows. Except for DFaust on latent dimension 8, spiral graph convolutions always perform at least slightly better than spectral graph convolutions. These results show that EDL obtains similar accuracy with both graph convolutions, which further validates its robustness.

### 4.3.3 *Evaluations of the Autoencoder*

**Qualitative Evaluations.** DEMEA significantly outperforms the baselines qualitatively on the DFaust and SynHand5M datasets, as seen in Figs. 4.6 and 4.5. Convolutional architectures without an embedded graph produce strong artifacts in the hand, feet, and face regions in the presence of large

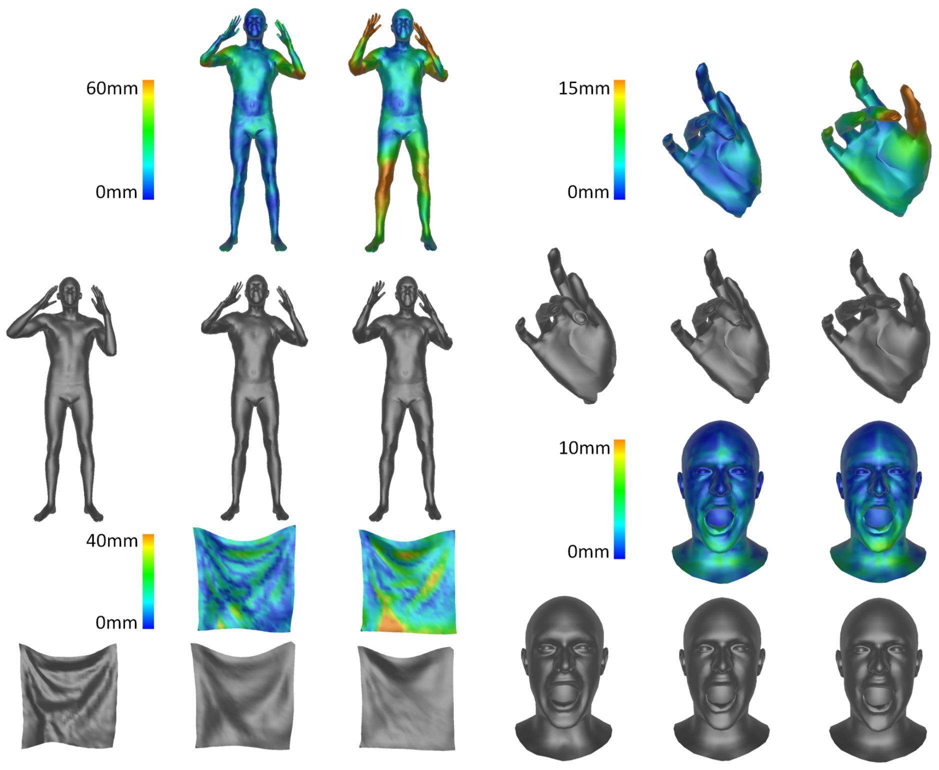


Figure 4.5: Auto-encoding results on all four datasets. From left to right: ground-truth, DEMEA with latent dimension 32, DEMEA with latent dimension 8.

| | DFaust | | SynHand5M | | Cloth | | CoMA | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 32 | 8 | 32 | 8 | 32 | 8 | 32 |
| FCA | 6.51 | 2.17 | 15.10 | 2.95 | 15.63 | 5.99 | 1.77 | **0.67** |
| FCED | 6.26 | 2.14 | 14.61 | 2.75 | 15.87 | **5.94** | 1.81 | 0.73 |
| CA | 6.35 | **2.07** | 8.12 | 2.60 | **11.21** | 6.50 | **1.17** | 0.72 |
| MCA | **6.21** | 2.13 | **8.11** | 2.67 | 11.64 | 6.59 | 1.20 | 0.71 |
| DEMEA | 6.69 | 2.23 | 8.12 | **2.51** | 11.28 | 6.40 | 1.23 | 0.81 |

Table 4.4: Architecture baselines. Average per-vertex errors on the test sets of DFaust (*cm*), SynHand5M (*mm*), textureless cloth (*mm*) and CoMA (*mm*) for 8 and 32 latent dimensions are reported.

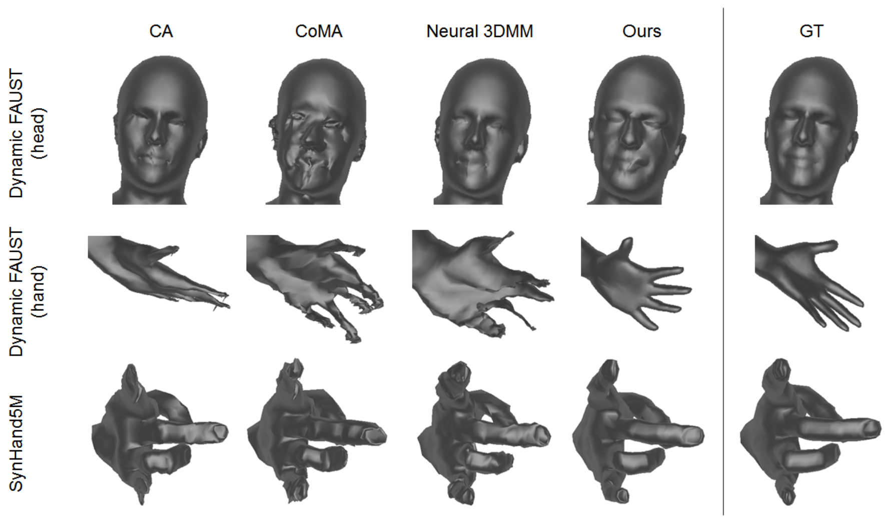


Figure 4.6: In contrast to graph-convolutional networks that directly regress vertex positions, the embedded graph layer does not show artifacts. These results use a latent dimension of 32.

deformations. Since EDL explicitly models deformations, it preserves fine details of the template under strong non-linear deformations and articulations of extremities.

**Quantitative Evaluations.** Tab. 4.4 compares the proposed DEMEA to the baselines on the autoencoding task.

While the fully-connected baselines are competitive for larger dimensions of the latent space, their memory demand increases drastically. On the other hand, they perform significantly worse for low dimensions on all datasets, except for DFaust. This chapter aims for low latent dimensions, *e.g.* less than 32, as the goal is to learn mesh representations that are as compact as possible. Adding EDL to the fully-connected baselines maintains their performance. Furthermore, the lower test errors of FCED on Cloth indicate that network capacity (and not EDL) limits the quantitative results.

On SynHand5M, Cloth, and CoMA, the convolutional baselines perform on par with DEMEA. On DFaust, DEMEA is slightly worse, perhaps because other architectures can also fit to the high-frequency details and noise. EDL regularizes deformations to avoid artifacts, which also prevents fitting to high-frequency or small details. Thus, explicitly modelling deformations via the EDL and thereby avoiding artifacts has no negative impact on the quantitative performance. Since CoMA mainly contains small and local deformations, DEMEA does not lead to any quantitative improvement. This is more evident in the case of latent dimension 32,

| | DFaust | | SynHand5M | | Cloth | | CoMA | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 32 | 8 | 32 | 8 | 32 | 8 | 32 |
| N. 3DMM | 7.09 | **1.99** | 8.50 | 2.58 | 12.64 | 6.49 | 1.34 | **0.71** |
| DEMEA | **6.69** | 2.23 | **8.12** | **2.51** | **11.28** | **6.40** | **1.23** | 0.81 |

Table 4.5: Comparison with Neural 3DMM (Bouritsas et al., 2019). Average per-vertex errors on the test sets of DFaust (in *cm*), SynHand5M (in *mm*), textureless cloth (in *mm*) and CoMA (in *mm*) for 8 and 32 latent dimensions are reported.

as the baselines can better reproduce noise and other high-frequency deformations.

**Comparisons.** Extensive comparisons with several competitive baselines demonstrate the usefulness of DEMEA for autoencoding strong non-linear deformations and articulated motion. Next, DEMEA is compared to the existing state-of-the-art CoMA approach (Ranjan et al., 2018). Their architecture is applied to all mentioned datasets with a latent dimension of 8, which is also used in Ranjan et al. (2018). DEMEA outperforms their method quantitatively on DFaust (6.7*cm vs.* 8.4*cm*), on SynHand5M (8.12*mm vs.* 8.93*mm*), on Cloth (1.13*cm vs.* 1.44*cm*), and even on CoMA (1.23*mm vs.* 1.42*mm*), where the deformations are not large. Finally, Tab. 4.5 compares DEMEA to Neural 3DMM (Bouritsas et al., 2019) on latent dimensions 8 and 32, similarly to Ranjan et al. (2018) on their proposed hierarchy. DEMEA performs better than Neural 3DMM in almost all cases. Fig. 4.6 shows that DEMEA avoids many of the artifacts present in the case of Ranjan et al. (2018), Bouritsas et al. (2019), and other baselines.

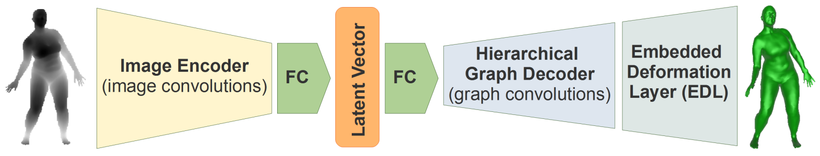
## 4.4 APPLICATIONS

This section shows that DEMEA can be used for mesh reconstruction from RGB or depth input, and that its learned latent space is surprisingly structured.

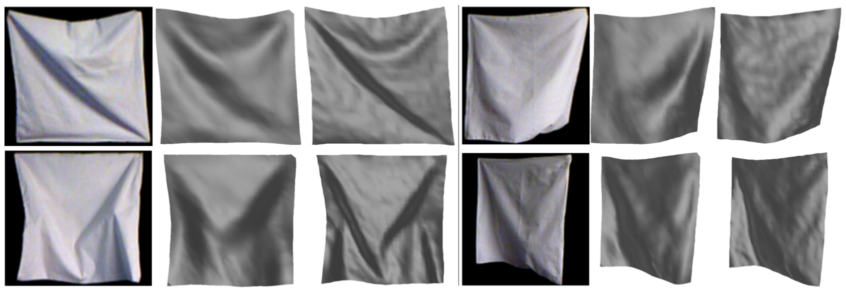
### 4.4.1 *RGB to Mesh*

On the Cloth (Bednařík et al., 2018) dataset, DEMEA can reconstruct meshes from RGB images. See Fig. 4.7b for qualitative examples with a latent dimension of 32.

On the test set, DEMEA achieves RGB-to-mesh reconstruction errors of 16.1*mm* and 14.5*mm* for latent dimensions 8 and 32, respectively. Bednařík et al. (2018), who use a different split, report an error of 21.48*mm*. The



(a) To train an image/depth-to-mesh reconstruction network, a convolutional image encoder is combined with a decoder that is initialized to a pre-trained graph decoder.



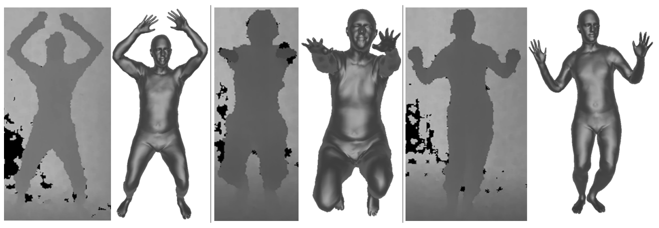
(b) From left to right: real RGB image, DEMEA's reconstruction, ground truth.

Figure 4.7: (Top) Image/depth-to-mesh pipeline. (Bottom) RGB-to-mesh results on the test set.

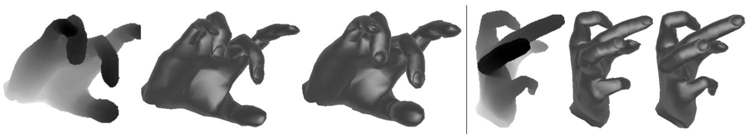
authors of IsMo-GAN (Shimada et al., 2019) report results on their own split for IsMo-GAN and the Hybrid Deformation Model Network (HDM-net) (Golyanik et al., 2018). On their split, HDM-Net achieves an error of $17.65mm$ after training for 100 epochs using a batch size of 4. IsMo-GAN obtains an error of $15.79mm$. Re-training DEMEA under the same settings as HDM-Net, without pre-training the mesh decoder, gives test errors of $16.6mm$ and $13.8mm$ using latent dimensions of 8 and 32, respectively.

### 4.4.2 *Depth to Mesh*

**Bodies.** DEMEA uses a latent space dimension of 32. Quantitatively, this yields an error of $2.3cm$ on un-augmented synthetic data. Fig. 4.8a shows results on real data. To this end, it is necessary to augment the depth images with artificial noise to lessen the domain gap.
**Hands.** DEMEA can reconstruct hands from depth as well, see Fig. 4.8b. It achieves a reconstruction error of $6.73mm$ for a latent dimension of 32. Malik et al. (2018) report an error of $11.8\ mm$. The test set in this chapter is composed of a random sample of fully randomly generated hands from the dataset, which is very challenging. Furthermore, this chapter uses $256 \times 256$, whereas Malik et al. (2018) use images of size $96 \times 96$.



(a) DEMEA (right) on real Kinect depth images (left) with latent dimension 32.



(b) Reconstruction results from synthetic depth images of hands using a latent dimension of 32. From left to right: depth, DEMEA's reconstruction, ground truth.

Figure 4.8: Reconstruction from a single depth image.

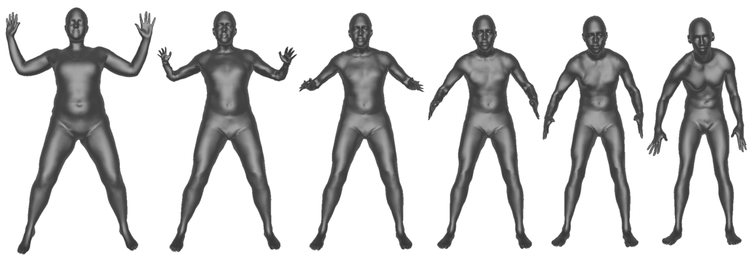
### 4.4.3 *Latent Space Arithmetic*

Although DEMEA does not employ any regularization on the latent space, it empirically learns a well-behaved latent space.

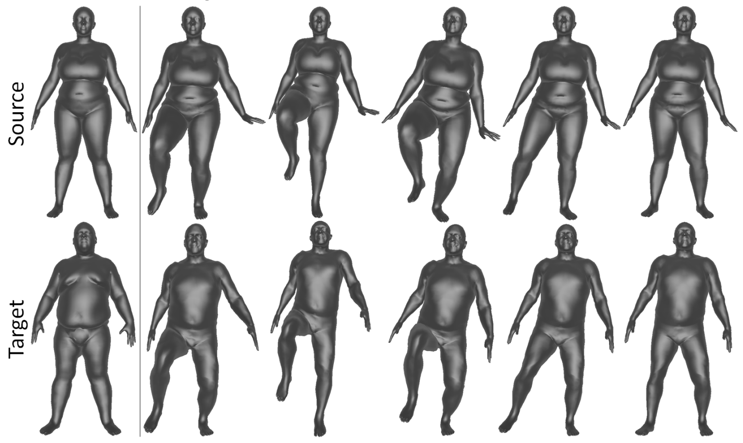
**Latent Interpolation.** DEMEA allows to linearly interpolate the latent vectors $\mathcal{S}$ and $\mathcal{T}$ of a source and a target mesh: $\mathcal{I}(\alpha) = (1-\alpha)\mathcal{S} + \alpha\mathcal{T}$. Even for highly different poses and identities, these $\mathcal{I}(\alpha)$ yield plausible in-between meshes, see Fig. 4.9a.

**Smooth Tracking.** DEMEA can temporally smooth tracked meshes from a depth stream. Specifically, temporal smoothing of the reconstruction of a sequence of real depth images $\{\mathbf{D}_i\}_i$ is achieved by decoding a running (causal) exponential average of the latent vectors of this sequence. First, the sequence is encoded into latent vectors $\{\mathcal{D}_i\}_i$. The smoothed sequence of latent vectors $\{\mathcal{D}'_i\}_i$ is defined as follows: Let $\mathcal{D}'_0 = \mathcal{D}_0$ and set $\mathcal{D}'_i = \alpha \cdot \mathcal{D}_i + (1-\alpha) \cdot \mathcal{D}'_{i-1}$ for $i > 0$ for some $\alpha \in [0,1]$. The smoothed sequence of meshes $\{\mathbf{M}_i\}_i$ is obtained by decoding $\{\mathcal{D}'_i\}_i$.

**Deformation Transfer.** The learned latent space allows to transfer poses between different identities on DFaust. Let a sequence of source meshes $\mathbf{S} = \{\mathbf{M}_i\}_i$ of person $A$ and a target mesh $\mathbf{M}'_0$ of person $B$ be given, where w.l.o.g. $\mathbf{M}_0$ and $\mathbf{M}'_0$ correspond to the same pose. The goal is to obtain sequence of target meshes $\mathbf{S}' = \{\mathbf{M}'_i\}_i$ of person $B$ performing the same poses as person $A$ in $\mathbf{S}$. To this end, $\mathbf{S}$ and $M'_0$ are encoded into the latent space of the mesh auto-encoder, yielding the corresponding latent vectors $\{\mathcal{M}_i\}_i$ and $\mathcal{M}'_0$. The identity difference is given by $d = \mathcal{M}'_0 - \mathcal{M}_0$, which



(a) Interpolation results, from left to right: source mesh, $\alpha = 0.2$, $\alpha = 0.4$, $\alpha = 0.6$, $\alpha = 0.8$, target mesh.



(b) Deformation transfer from a source sequence to a target identity. The first column shows $\mathbf{M}_0$ and $\mathbf{M}'_0$.

Figure 4.9: Latent space arithmetic.

allows to set $\mathcal{M}'_i = \mathcal{M}_i + d$ for $i > 0$. Decoding $\{\mathcal{M}'_i\}_i$ using the mesh decoder than yields $\mathbf{S}'$. See Fig. 4.9b for qualitative results.

## 4.5 LIMITATIONS

While the embedded deformation graph excels on highly articulated, non-rigid motions, it has difficulties accounting for very subtle actions. Since the faces in the CoMA (Ranjan et al., 2018) dataset do not undergo large deformations, the EDL-based architecture does not offer a significant advantage. Similar to all other 3D deep learning techniques, DEMEA also requires reasonably sized mesh datasets for supervised training, which might be difficult to capture or model. DEMEA is trained in an object-specific manner. Generalizing DEMEA across different object categories is an interesting direction for future work.

## 4.6 CONCLUSION

This chapter proposes DEMEA — the first deep mesh autoencoder for highly deformable and articulated scenes, such as human bodies, hands, and deformable surfaces, that builds on a new differentiable embedded deformation layer. The deformation layer reasons about local rigidity of the mesh and achieves higher quality autoencoding results compared to several baselines and existing approaches. It enables multiple applications, including non-rigid reconstruction from real depth maps and 3D reconstruction of textureless surfaces from images.

Next, this thesis turns from modeling deformations for general non-rigid objects to modeling geometry for general non-rigid objects, again with neural techniques.

# 5 PATCHNETS: PATCH-BASED GENERALIZABLE DEEP IMPLICIT 3D SHAPE REPRESENTATIONS

This chapter concerns itself with the geometry aspect of general non-rigid objects. Implicit surface representations, such as signed-distance functions, combined with deep learning have led to impressive models which can represent detailed shapes of objects with arbitrary topology. Since a continuous function is learned, the reconstructions can also be extracted at any arbitrary resolution. However, even when training on large datasets such as ShapeNet (Chang et al., 2015), these models only generalize to similar test sets.

This chapter presents a new mid-level patch-based surface representation (also published as Tretschk et al. (2020a)). At the level of patches, objects across different categories share similarities, which leads to more generalizable models. This chapter introduces a novel method to learn this patch-based representation in a canonical space, such that it is as object-agnostic as possible. Experiments show that this representation, when trained on one category of objects from ShapeNet, can also well represent detailed shapes from any other category. In particular, it allows to represent shapes of general non-rigid object categories (*e.g.*, humans) despite only having been trained on shapes of static object categories (*e.g.*, chairs). In addition, it can be trained using much fewer shapes, compared to existing approaches. This chapter shows several applications of this new representation, including shape interpolation and partial point cloud completion. Due to explicit control over positions, orientations and scales of patches, this representation is also more controllable compared to object-level representations, which allows to deform encoded shapes non-rigidly.

## 5.1 INTRODUCTION

Several 3D shape representations exist in the computer vision and computer graphics communities, such as point clouds, meshes, voxel grids, and implicit functions. Learning-based approaches have mostly focused on voxel grids due to their regular structure, suited for convolutions. However, such methods (Choy et al., 2016) ultimately have limited output resolution due to the finite resolution of the voxel grid. Point cloud based approaches have also been explored (Qi et al., 2017). While most approaches assume a fixed number of points, recent methods also allow for variable resolution outputs (Mescheder et al., 2019; Sitzmann et al., 2019b). Point clouds only offer a sparse representation of the surface. Meshes with fixed topology are commonly used in constrained settings with known object categories (Wang et al., 2018). However, they are not suitable for representing objects with varying topology. Very recently, implicit function-based representations were introduced (Chen and Zhang, 2019; Mescheder et al., 2019; Park et al., 2019). DeepSDF (Park et al., 2019) learns a network which represents the continuous signed distance functions for a class of objects. The surface is represented as the 0-isosurface. Similar approaches (Chen and Zhang, 2019; Mescheder et al., 2019) use occupancy networks, where only the occupancy values are learned (similar to voxel grid-based approaches), but in a continuous representation. Implicit functions allow for representing (closed) shapes of arbitrary topology. The reconstructed surface can be extracted at any resolution, since a continuous function is learned.

All existing implicit function-based methods rely on large datasets of 3D shapes for training. This chapter aims to build a generalizable surface representation which can be trained with much fewer shapes, and can also generalize to different object categories. Instead of learning an object-level representation, *PatchNet* learns a mid-level representation of surfaces, at the level of patches. At the level of patches, objects across different categories share similarities. PatchNet learns these patches in a canonical space to further abstract from object-specific details. Patch extrinsics (position, scale, and orientation of a patch) allow each patch to be translated, rotated, and scaled. Multiple patches can be combined in order to represent the full surface of an object. Experiments show that the patches can be learned using very few shapes, and can generalize across different object categories, see Fig. 5.1. This representation also allows to build object-level models, *ObjectNets*, which is useful for applications which require an object-level prior.

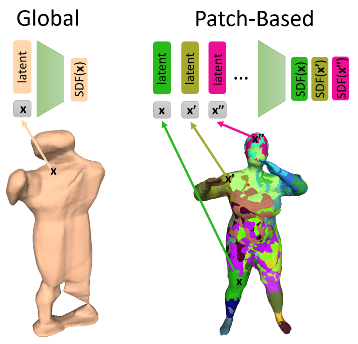


Figure 5.1: In contrast to a global approach, the proposed patch-based method generalizes to non-rigid human shapes after being trained on rigid ShapeNet objects.

The trained models enable several applications, including partial point cloud completion from depth maps, shape interpolation, and a generative model for objects. While implicit function-based approaches can reconstruct high-quality and detailed shapes, they lack controllability.

The proposed patch-based implicit representation *natively* allows for controllability due to the explicit control over patch extrinsics. User-guided rigging of the patches to the surface allows for articulated deformation of humans without re-encoding the deformed shapes. In addition to the generalization and editing capabilities, the PatchNet representation includes all advantages of implicit surface modeling. The patches can represent shapes of any arbitrary topology, and the reconstructions can be extracted at any arbitrary resolution using *Marching Cubes* (Lorensen and Cline, 1987). Similar to DeepSDF (Park et al., 2019), the network in PatchNet uses an auto-decoder architecture, combining classical optimization with learning, resulting in high-quality geometry.

## 5.2 method

The proposed method represents the surface of any object as a combination of several surface patches. The patches form a mid-level representation, where each patch represents the surface within a specified radius from its center. This representation is generalizable across object categories, as most objects share similar geometry at the patch level. This section explains how the patches are represented using artificial neural networks, the losses required to train such networks, as well as the algorithm to combine multiple patches for smooth surface reconstruction.

### 5.2.1 *Implicit Patch Representation*

PatchNets represents a full object $i$ as a collection of $N_P = 30$ patches. A patch $p$ represents a surface within a sphere of radius $r_{i,p} \in \mathbb{R}$, centered at $\mathbf{c}_{i,p} \in \mathbb{R}^3$. Each patch can be oriented by a rotation about a canonical frame, parametrized by Euler angles $\phi_{i,p} \in \mathbb{R}^3$. Let $\mathbf{e}_{i,p} = (r_{i,p}, \mathbf{c}_{i,p}, \phi_{i,p}) \in \mathbb{R}^7$ denote all extrinsic patch parameters. Representing the patch surface in a canonical frame of reference allows to normalize the query 3D point, leading to more object-agnostic and generalizable patches.

The patch surface is represented as an implicit signed-distance function (SDF), which maps 3D points to their signed distance from the closest surface. This offers several advantages, as these functions are a continuous representation of the surface, unlike point clouds or meshes. In addition, the surface can be extracted at any resolution without a large memory requirement, unlike for voxel grids. In contrast to prior work (Genova et al., 2019; Williams et al., 2020), which uses simple patch primitives, the proposed approach parametrizes the patch surface as a neural network (PatchNet). Its network architecture is based on the auto-decoder of DeepSDF (Park et al., 2019). The input to the network is a *patch latent* *code* $\mathbf{z} \in \mathbb{R}^{N_z}$ of length $N_z = 128$, which describes the patch surface, and a 3D query point $\mathbf{x} \in \mathbb{R}^3$. The output is the scalar SDF value of the surface at $\mathbf{x}$. Similar to DeepSDF, its architecture consists of eight weight-normalized (Salimans and Kingma, 2016) fully-connected layers with 128 output dimensions and ReLU activations, with $\mathbf{z}$ and $\mathbf{x}$ concatenated to the input of the fifth layer. The last fully-connected layer outputs a single scalar to which *tanh* is applied to obtain the SDF value.

### 5.2.2 *Preliminaries*

PREPROCESSING: A given watertight mesh is first preprocessed to obtain SDF values for 3D point samples. First, each mesh is centered and fit tightly into the unit sphere. This is followed by sampling points, mostly close to the surface, and computing their truncated signed distance to the object surface, with truncation at 0.1. For more details on the sampling strategy, please refer to Park et al. (2019).

AUTO-DECODING: Unlike the usual setting, PatchNets do not use an encoder that regresses patch latent codes and extrinsics. Instead, following DeepSDF (Park et al., 2019), shapes are *auto-decoded*: the patch latent codes and extrinsics of each object are treated as free variables to be optimized for during training. *I.e.*, instead of back-propagating into an encoder, the proposed method employs the gradients to learn these parameters directly during training.

INITIALIZATION: Due to auto-decoding, the patch latent codes and extrinsics are treated as free variables, similar to classical optimization. Therefore, they can be directly initialized. All patch latent codes are initially set to zero, and the patch positions are initialized by greedy farthest point sampling of point samples of the object surface. Each patch radius is set to the minimum such that each surface point sample is covered by its closest patch. The patch orientation aligns the $z$-axis of the patch coordinate system with the surface normal.

### 5.2.3 *Loss Functions*

PatchNet is trained by auto-decoding $N$ full objects. The patch latent codes of an object $i$ are $\mathbf{z}_i = [\mathbf{z}_{i,0}, \mathbf{z}_{i,1}, \dots, \mathbf{z}_{i,N_P-1}]$, with each patch latent code of length $N_z$. Patch extrinsics are represented as

$\mathbf{e}_i = [\mathbf{e}_{i,0}, \mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,N_P-1}]$. Let $\theta$ denote the trainable weights of PatchNet. PatchNets employ the following loss function:

$$\mathcal{L}(\mathbf{z}_i, \mathbf{e}_i, \theta) = \mathcal{L}_{\text{recon}}(\mathbf{z}_i, \mathbf{e}_i, \theta) + \mathcal{L}_{\text{ext}}(\mathbf{e}_i) + \mathcal{L}_{\text{reg}}(\mathbf{z}_i). \tag{5.1}$$

Here, $\mathcal{L}_{recon}$ is the surface reconstruction loss, $\mathcal{L}_{ext}$ is the extrinsic loss guiding the extrinsics for each patch, and $\mathcal{L}_{reg}$ is a regularizer on the patch latent codes.

RECONSTRUCTION LOSS: The reconstruction loss minimizes the SDF values between the predictions and the ground truth for each patch:

$$\mathcal{L}_{\text{recon}}(\mathbf{z}_i, \mathbf{e}_i, \theta) = \frac{1}{N_P} \sum_{p=0}^{N_P-1} \frac{1}{|S(\mathbf{e}_{i,p})|} \sum_{\mathbf{x} \in S(\mathbf{e}_{i,p})} \left\| f(\mathbf{x}, \mathbf{z}_{i,p}, \theta) - s(\mathbf{x}) \right\|_1, \tag{5.2}$$

where $f(\cdot)$ and $s(\mathbf{x})$ denote a forward pass of the network and the ground truth truncated SDF values at point $\mathbf{x}$, respectively; $S(\mathbf{e}_{i,p})$ is the set of all (normalized) point samples that lie within the bounds of patch $p$ with extrinsics $\mathbf{e}_{i,p}$.

EXTRINSIC LOSS: The composite extrinsic loss ensures all patches contribute to the surface and are placed such that the surfaces are learned in a canonical space:

$$\mathcal{L}_{\text{ext}}(\mathbf{e}_i) = \mathcal{L}_{\text{sur}}(\mathbf{e}_i) + \mathcal{L}_{\text{cov}}(\mathbf{e}_i) + \mathcal{L}_{\text{rot}}(\mathbf{e}_i) + \mathcal{L}_{\text{scl}}(\mathbf{e}_i) + \mathcal{L}_{\text{var}}(\mathbf{e}_i). \tag{5.3}$$

$\mathcal{L}_{\text{sur}}$ ensures that every patch stays close to the surface:

$$\mathcal{L}_{\text{sur}}(\mathbf{e}_i) = \omega_{\text{sur}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} \max(\min_{\mathbf{x} \in \mathbf{O}_i} \left\| \mathbf{c}_{i,p} - \mathbf{x} \right\|_2^2, t). \tag{5.4}$$

Here, $\mathbf{O}_i$ is the set of surface points of object $i$. This term is only applied when the distance between a patch and the surface is greater than a threshold $t = 0.06$.

A symmetric coverage loss $\mathcal{L}_{\text{cov}}$ encourages each point on the surface to be covered by at least one patch:

$$\mathcal{L}_{\text{cov}}(\mathbf{e}_i) = \omega_{\text{cov}} \cdot \frac{1}{|\mathbf{U}_i|} \sum_{\mathbf{x} \in \mathbf{U}_i} \frac{w_{i,p,\mathbf{x}}}{\sum_p w_{i,p,\mathbf{x}}} (\left\| \mathbf{c}_{i,p} - \mathbf{x} \right\|_2 - r_{i,p}), \tag{5.5}$$

where $\mathbf{U}_i \subseteq \mathbf{O}_i$ are all surface points that are not covered by any patch, *i.e.*, outside the bounds of all patches. $w_{i,p,\mathbf{x}}$ weighs the patches based on their distance from $\mathbf{x}$, with $w_{i,p,\mathbf{x}} = \exp\left(-0.5 \cdot ((\left\| \mathbf{c}_{i,p} - \mathbf{x} \right\|_2 - r_{i,p})/\sigma)^2\right)$ where $\sigma = 0.05$.

An additional loss aligns the patches with the surface normals. This encourages the patch surface to be learned in a canonical frame of reference:

$$\mathcal{L}_{\text{rot}}(\mathbf{e}_i) = \omega_{\text{rot}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} (1 - \langle \phi_{i,p} \cdot [0,0,1]^T, \mathbf{n}_{i,p} \rangle)^2. \tag{5.6}$$

Here, $\mathbf{n}_{i,p}$ is the surface normal at the point $\mathbf{o}_{i,p}$ closest to the patch center, *i.e.*, $\mathbf{o}_{i,p} = \operatorname{argmin}_{\mathbf{x} \in \mathbf{O}_i} \|\mathbf{x} - \mathbf{c}_{i,p}\|_2$.

Finally, two losses regularize the extent of the patches. The first loss encourages the patches to be reasonably small. This prevents significant overlap between different patches:

$$\mathcal{L}_{\text{scl}}(\mathbf{e}_i) = \omega_{\text{scl}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} r_{i,p}^2. \tag{5.7}$$

The second loss encourages all patches to be of similar sizes. This prevents the surface to be reconstructed only using very few large patches:

$$\mathcal{L}_{\text{var}}(\mathbf{e}_i) = \omega_{\text{var}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} (r_{i,p} - m_i)^2, \tag{5.8}$$

where $m_i$ is the mean patch radius of object $i$.

REGULARIZER: Similar to DeepSDF, an $\ell_2$-regularizer on the latent codes encourages a Gaussian prior distribution:

$$\mathcal{L}_{\text{reg}}(\mathbf{z}_i) = \omega_{\text{reg}} \cdot \frac{1}{N_P} \sum_{p=0}^{N_P-1} \|\mathbf{z}_{i,p}\|_2^2. \tag{5.9}$$

OPTIMIZATION: During training, the following problem is optimized:

$$\operatorname*{argmin}_{\theta, \{\mathbf{z}_i\}_i, \{\mathbf{e}_i\}_i} \sum_{i=0}^{N-1} \mathcal{L}(\mathbf{z}_i, \mathbf{e}_i, \theta). \tag{5.10}$$

At test time, any surface can be reconstructed using the learned patch-based representation. Using the same initialization of extrinsics and patch latent codes, and given point samples with their SDF values, the patch latent codes and the patches extrinsics are optimized for with fixed network weights.

### 5.2.4 *Blended Surface Reconstruction*

For a smooth surface reconstruction of object $i$, *e.g.* for Marching Cubes, the blended SDF prediction $g_i(\mathbf{x})$ is obtained by blending between different patches in the overlapping regions. Specifically, $g_i(\mathbf{x})$ is computed as a weighted linear combination of the SDF values $f(\mathbf{x}, \mathbf{z}_{i,p}, \theta)$ of the overlapping patches:

$$g_i(\mathbf{x}) = \sum_{p \in P_{i,\mathbf{x}}} \frac{w_{i,p,\mathbf{x}}}{\sum_{p \in P_{i,\mathbf{x}}} w_{i,p,\mathbf{x}}} f(\mathbf{x}, \mathbf{z}_{i,p}, \theta), \tag{5.11}$$

with $P_{i,\mathbf{x}}$ denoting the patches which overlap at point $\mathbf{x}$. For empty $P_{i,\mathbf{x}}$, $g_i(\mathbf{x}) = 1$. The blending weights are defined as:

$$w_{i,p,\mathbf{x}} = \exp\left(-\frac{1}{2}\left(\frac{\left\|\mathbf{c}_{i,p} - \mathbf{x}\right\|_2}{\sigma}\right)^2\right) - \exp\left(-\frac{1}{2}\left(\frac{r_{i,p}}{\sigma}\right)^2\right), \tag{5.12}$$

with $\sigma = r_{i,p}/3$. The offset ensures that the weight is zero at the patch boundary, to avoid stitching artifacts.

## 5.3 RESULTS

In the following, experiments show the effectiveness of the proposed patch-based representation on several different problems.

### 5.3.1 *Settings*

**Datasets.** Most experiments employ *ShapeNet* (Chang et al., 2015). Preprocessing is performed using the code of Stutz and Geiger (2018), similar to Genova et al. (2020) and Mescheder et al. (2019), to make the meshes watertight and normalize them within a unit cube. The training and test splits follow Choy et al. (2016). The results in Tab. 5.1 and 5.2 use the full test set. Other results refer to a reduced test set, where 50 objects from each of the 13 categories are randomly picked. Sec. B.1 shows that results on the reduced test set are representative of the full test set. In addition, Dynamic FAUST (Bogo et al., 2017) is used for testing. There, the test set from Chapter 4 is subsampled by concatenating all test sequences and taking every 20th mesh. Preprocessing generates 200$k$ SDF point samples per shape.

**Metrics.** Three error metrics are used to quantitatively assess the results: IoU, Chamfer distance, and F-score, similar to Genova et al. (2020). The mean values across different test sets are reported.

*IoU*: For a given watertight ground-truth mesh, marching cubes extracts the reconstructed mesh at $128^3$ resolution. Then, 100$k$ points are uniformly sampled in the bounding box of the ground truth. Then, each point lies inside or outside of the generated mesh and of the ground truth. The final value is the fraction of intersection over union, multiplied by a factor of 100. Higher is better.

*Chamfer Distance*: Here, $100k$ points are sampled on the surface of both the ground truth and the reconstructed mesh. A kD-tree is used to compute the closest points from the reconstructed to the ground-truth mesh and vice-versa. The square of these distances ($L2$ Chamfer) is averaged for each of the two directions and the two resulting numbers are summed. For better readability, the numbers are finally multiplied by 100. Lower is better.

*F-score*: For each shape, the point-wise distances computed before are thresholded at 0.01 (all meshes are normalized to a unit cube). Then, separately for each direction, the fraction of distances below the threshold is determined. Finally, the harmonic mean of both these values is multiplied by 100 to yield the final result. Higher is better.

In cases where a network does not produce any surface, the value of IoU is set to 0, the Chamfer distance to 100, and the F-score to 0.

**Training Details.** PatchNets is trained using *PyTorch* (Paszke et al., 2019). The number of epochs is 1000, the learning rate for the network is initially $5 \cdot 10^{-4}$, and for the patch latent codes and extrinsics $10^{-3}$. Both learning rates are halved every 200 epochs. The optimization uses Adam (Kingma and Ba, 2015) and a batch size of 64. For each object in the batch, $3k$ SDF point samples are randomly sampled. The weights for the losses are: $\omega_{\text{scl}} = 0.01$, $\omega_{\text{var}} = 0.01$, $\omega_{\text{sur}} = 5$, $\omega_{\text{rot}} = 1$, $\omega_{\text{sur}} = 200$. $\omega_{\text{reg}}$ is linearly increased from 0 to $10^{-4}$ for 400 epochs and then kept constant.

**Baseline.** For fairer comparisons, the experiments compare against a "global-patch" baseline similar to DeepSDF, which only uses a single patch without extrinsics. The patch latent size is 4050, matching PatchNets'. The learning rate scheme is the same as for the proposed method.

### 5.3.2 *Surface Reconstruction*

First, the surface reconstruction of the test set, in challenging generalization settings, and under ablations is evaluated.

#### 5.3.2.1 *Results*

The proposed approach is trained on a subset of the training data, with 100 randomly picked shapes from each category. In addition to a comparison with the global baseline, PatchNets is compared with DeepSDF (Park

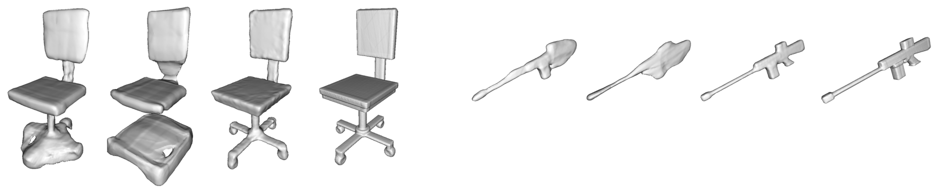


Figure 5.2: Surface Reconstruction. From left to right: DeepSDF, baseline, PatchNets, ground truth.

| | IoU | | | Chamfer | | | F-score | | |
|---|---|---|---|---|---|---|---|---|---|
| Category | DeepSDF | Baseline | PatchNets | DeepSDF | Baseline | PatchNets | DeepSDF | Baseline | PatchNets |
| airplane | 84.9 | 65.3 | **91.1** | 0.012 | 0.077 | **0.004** | 83.0 | 72.9 | **97.8** |
| bench | 78.3 | 68.0 | **85.4** | 0.021 | 0.065 | **0.006** | 91.2 | 80.6 | **95.7** |
| cabinet | 92.2 | 88.8 | **92.9** | **0.033** | 0.055 | 0.110 | **91.6** | 86.4 | 91.2 |
| car | 87.9 | 83.6 | **91.7** | **0.049** | 0.070 | **0.049** | 82.2 | 74.5 | **87.7** |
| chair | 81.8 | 72.9 | **90.0** | 0.042 | 0.110 | **0.018** | 86.6 | 75.5 | **94.3** |
| display | 91.6 | 86.5 | **95.2** | **0.030** | 0.061 | 0.039 | 93.7 | 87.0 | **97.0** |
| lamp | 74.9 | 63.0 | **89.6** | 0.566 | 0.438 | **0.055** | 82.5 | 69.4 | **94.9** |
| rifle | 79.0 | 68.5 | **93.3** | 0.013 | 0.039 | **0.002** | 90.9 | 82.3 | **99.3** |
| sofa | 92.5 | 85.4 | **95.0** | 0.054 | 0.226 | **0.014** | 92.1 | 84.2 | **95.3** |
| speaker | 91.9 | 86.7 | **92.7** | **0.050** | 0.094 | 0.243 | 87.6 | 79.4 | **88.5** |
| table | 84.2 | 71.9 | **89.4** | 0.074 | 0.156 | **0.018** | 91.1 | 79.2 | **95.0** |
| telephone | 96.2 | 95.0 | **98.1** | 0.008 | 0.016 | **0.003** | 97.7 | 96.2 | **99.4** |
| watercraft | 85.2 | 79.1 | **93.2** | 0.026 | 0.041 | **0.009** | 87.8 | 80.2 | **96.4** |
| mean | 77.4 | 76.5 | **92.1** | 0.075 | 0.111 | **0.044** | 89.9 | 80.6 | **94.8** |

Table 5.1: Surface Reconstruction. PatchNets significantly outperform DeepSDF (Park et al., 2019) and the global baseline on all categories of ShapeNet almost everywhere.

et al., 2019) as setup in their paper. Both DeepSDF and the global baseline use the subset. Qualitative results are shown in Fig. 5.2 and 5.3.

Tab. 5.1 shows the quantitative results for surface reconstruction. PatchNets significantly outperform DeepSDF and the global baseline almost everywhere, demonstrating the higher-quality afforded by its patch-based representation.

Next, experiments compare PatchNets with several state-of-the-art approaches on implicit surface reconstruction, OccupancyNetworks (Mescheder et al., 2019), Structured Implicit Functions (Genova et al., 2019) and Deep Structured Implicit Functions (Genova et al., 2020)[1]. While they are trained on the full ShapeNet shapes, the proposed method is trained only on a small subset. Even in this disadvantageous and challenging setting, PatchNets outperform these approaches on most categories, see Tab. 5.2. Note that the metrics are computed consistently with Genova et al. (2020) and thus are directly comparable to numbers reported in their paper.

1 DSIF is also known as *Local Deep Implicit Functions for 3D Shape*.

| | IoU | | | | Chamfer | | | | F-score | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | OccNet | SIF | DSIF | PatchNets | OccNet | SIF | DSIF | PatchNets | OccNet | SIF | DSIF | PatchNets |
| airplane | 77.0 | 66.2 | **91.2** | 91.1 | 0.016 | 0.044 | 0.010 | **0.004** | 87.8 | 71.4 | 96.9 | **97.8** |
| bench | 71.3 | 53.3 | **85.6** | 85.4 | 0.024 | 0.082 | 0.017 | **0.006** | 87.5 | 58.4 | 94.8 | **95.7** |
| cabinet | 86.2 | 78.3 | **93.2** | 92.9 | 0.041 | 0.110 | **0.033** | 0.110 | 86.0 | 59.3 | **92.0** | 91.2 |
| car | 83.9 | 77.2 | 90.2 | **91.7** | 0.061 | 0.108 | **0.028** | 0.049 | 77.5 | 56.6 | 87.2 | **87.7** |
| chair | 73.9 | 57.2 | 87.5 | **90.0** | 0.044 | 0.154 | 0.034 | **0.018** | 77.2 | 42.4 | 90.9 | **94.3** |
| display | 81.8 | 69.3 | 94.2 | **95.2** | 0.034 | 0.097 | **0.028** | 0.039 | 82.1 | 56.3 | 94.8 | **97.0** |
| lamp | 56.5 | 41.7 | 77.9 | **89.6** | 0.167 | 0.342 | 0.180 | **0.055** | 62.7 | 35.0 | 83.5 | **94.9** |
| rifle | 69.5 | 60.4 | 89.9 | **93.3** | 0.019 | 0.042 | 0.009 | **0.002** | 86.2 | 70.0 | 97.3 | **99.3** |
| sofa | 87.2 | 76.0 | 94.1 | **95.0** | 0.030 | 0.080 | 0.035 | **0.014** | 85.9 | 55.2 | 92.8 | **95.3** |
| speaker | 82.4 | 74.2 | 90.3 | **92.7** | 0.101 | 0.199 | **0.068** | 0.243 | 74.7 | 47.4 | 84.3 | **88.5** |
| table | 75.6 | 57.2 | 88.2 | **89.4** | 0.044 | 0.157 | 0.056 | **0.018** | 84.9 | 55.7 | 92.4 | **95.0** |
| telephone | 90.9 | 83.1 | 97.6 | **98.1** | 0.013 | 0.039 | 0.008 | **0.003** | 94.8 | 81.8 | 98.1 | **99.4** |
| watercraft | 74.7 | 64.3 | 90.1 | **93.2** | 0.041 | 0.078 | 0.020 | **0.009** | 77.3 | 54.2 | 93.2 | **96.4** |
| mean | 77.8 | 66.0 | 90.0 | **92.1** | 0.049 | 0.118 | **0.040** | 0.044 | 81.9 | 59.0 | 92.2 | **94.8** |

Table 5.2: Surface Reconstruction. PatchNets outperform OccupancyNetworks (OccNet, Mescheder et al. (2019)), Structured Implicit Functions (SIF, Genova et al. (2019)), and Deep Structured Implicit Functions (DSIF, Genova et al. (2020)) almost everywhere.

### 5.3.2.2 *Generalization*

The proposed patch-based representation is more generalizable compared to existing representations. Several experiments with different training data demonstrate this in the following. In each case, the learning rate schemes are modified to equalize the number of network weight updates. For each experiment, PatchNets is compared with the baseline approaches described above. The experiments use a reduced ShapeNet test set, which consists of 50 shapes from each category. Fig. 5.3 shows qualitative results and comparisons. Evaluations on 647 meshes from the Dynamic FAUST (Bogo et al., 2017) test set show cross-dataset generalization. In the first experiment, PatchNets is trained on shapes from the *Cabinet* category and attempts to reconstruct shapes from every other category. It significantly outperforms the baselines almost everywhere, see Tab. 5.3. The improvement is even more noticeable for cross dataset generalization with around 70% improvement in the F-score compared to the global-patch baseline.

The second experiment evaluates the amount of training data required to train PatchNets. PatchNets as well as the baselines are trained on 30, 10, 3, and 1 shape(s) per category of ShapeNet. An additional experiment trains the networks on a single randomly picked shape from ShapeNet, which is visualized in Fig. 5.4. Fig. 5.5 shows the errors for ShapeNet (mean across categories) and Dynamic FAUST. The performance of the proposed approach degrades only slightly with a decreasing number of training shapes. However, the baseline approach of DeepSDF degrades much more severely. This is even more evident for cross dataset generalization on Dynamic FAUST, where the baseline cannot perform well

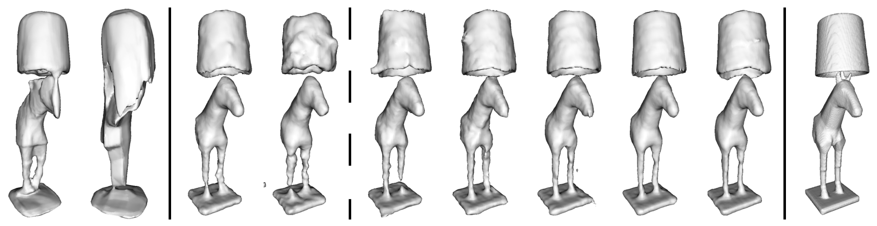


Figure 5.3: Generalization. From left to right: DeepSDF, baseline, PatchNets on one category, PatchNets on one shape, PatchNets on 1 shape per category, PatchNets on 3 per category, PatchNets on 10 per category, PatchNets on 30 per category, PatchNets on 100 per category, and ground truth.

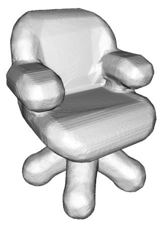


Figure 5.4: Single Shape. In one of the generalization experiments, the networks are trained on this randomly chosen ground-truth shape.

even with a larger number of training shapes, while PatchNets perform similarly across datasets.

#### 5.3.2.3 *Ablation Experiments*

Several ablative analysis experiments in this section evaluate PatchNets deeper. The mean error metrics on the reduced ShapeNet test set are reported when training on the reduced ShapeNet training set, which consists of 100 shapes per ShapeNet category.

**Number of Patches.** First, the number of patches required to reconstruct surfaces is evaluated. Tab. 5.4 contains summary results, while Fig. 5.7

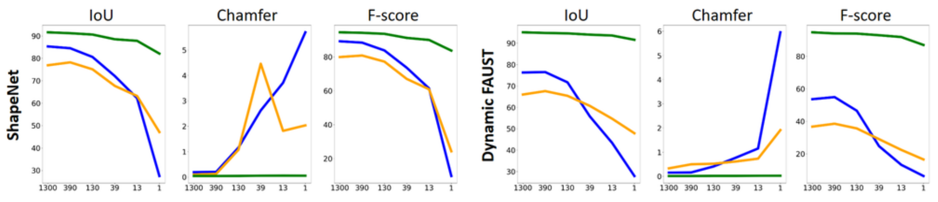


Figure 5.5: Generalization. PatchNet (green), the global-patch baseline (orange), and DeepSDF (blue) are trained on different numbers of shapes (x-axis). Results on different metrics on the reduced test sets are shown on the y-axis. For IoU and F-score, higher is better. For Chamfer distance, lower is better.

| | IoU | | | Chamfer | | | F-score | | |
|---|---|---|---|---|---|---|---|---|---|
| Category | BL | DSDF | PatchNets | BL | DSDF | PatchNets | BL | DSDF | PatchNets |
| airplane | 33.5 | 56.9 | **88.2** | 0.668 | 0.583 | **0.005** | 33.5 | 61.7 | **96.3** |
| bench | 49.1 | 58.8 | **80.4** | 0.169 | 0.093 | **0.006** | 63.6 | 76.3 | **93.3** |
| *cabinet* | 86.0 | 91.1 | **91.4** | 0.045 | **0.025** | 0.121 | 86.4 | **92.6** | 91.7 |
| car | 78.4 | 83.7 | **92.0** | 0.101 | 0.074 | **0.050** | 62.7 | 73.9 | **87.2** |
| chair | 50.7 | 61.8 | **86.9** | 0.473 | 0.287 | **0.012** | 49.1 | 65.2 | **92.5** |
| display | 83.2 | 87.6 | **94.4** | 0.111 | 0.065 | **0.052** | 83.9 | 89.6 | **96.9** |
| lamp | 49.7 | 59.3 | **86.6** | 0.689 | 2.645 | **0.082** | 50.4 | 64.5 | **93.4** |
| rifle | 56.4 | 56.1 | **91.8** | 0.114 | 2.669 | **0.002** | 71.0 | 54.7 | **99.1** |
| sofa | 81.1 | 87.3 | **94.8** | 0.245 | 0.193 | **0.010** | 74.2 | 84.6 | **95.2** |
| speaker | 83.2 | 88.3 | **90.5** | 0.163 | **0.080** | 0.232 | 71.8 | 80.1 | **84.9** |
| table | 55.0 | 73.6 | **88.4** | 0.469 | 0.222 | **0.020** | 61.8 | 82.8 | **95.0** |
| telephone | 90.4 | 94.7 | **97.3** | 0.051 | 0.015 | **0.004** | 90.8 | 96.1 | **99.2** |
| watercraft | 66.5 | 73.5 | **91.8** | 0.115 | 0.157 | **0.006** | 63.0 | 74.2 | **96.2** |
| mean | 66.4 | 74.8 | **90.3** | 0.263 | 0.547 | **0.046** | 66.3 | 76.6 | **93.9** |
| DFAUST | 57.8 | 71.2 | **94.4** | 0.751 | 0.389 | **0.012** | 25.0 | 45.4 | **94.0** |

Table 5.3: Generalization. Networks trained on the *Cabinet* category, but evaluated on every category of ShapeNet, as well as on Dynamic FAUST. PatchNets significantly outperform the baseline (BL) and DeepSDF (DSDF) almost everywhere.

| | IoU | Chamfer | F-score | Time |
|---|---|---|---|---|
| $N_P = 3$ | 73.8 | 0.15 | 72.9 | 1h |
| $N_P = 10$ | 85.2 | 0.049 | 88.0 | 1.5h |
| size 32 | 82.8 | 0.066 | 84.7 | 1.5h |
| size 512 | 95.3 | 0.048 | 97.2 | 8h |
| full dataset | 92.2 | 0.050 | 94.8 | 156h |
| PatchNets | 91.6 | 0.045 | 94.5 | 2h |

Table 5.4: Ablation evaluating the performance using different numbers of patches, as well as using variable sizes of the patch latent code/hidden dimensions, and the training data. The training time is measured on an Nvidia V100 GPU.

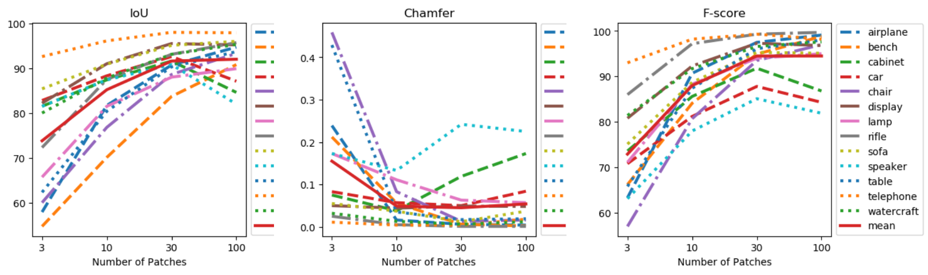


Figure 5.6: Per-category error metrics on the reduced ShapeNet test set for different numbers of patches.

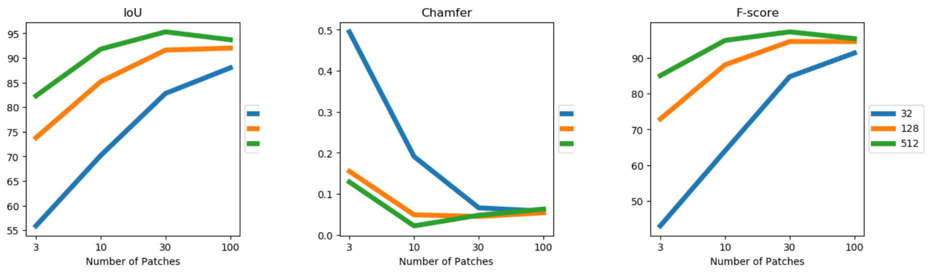


Figure 5.7: Mean error metrics on the reduced ShapeNet test set for different numbers of patches and network/latent code sizes.

contains detailed results. As expected, the performance becomes better with a larger number of patches, since this would lead to smaller patches which can capture more details and generalize better. Fig. 5.6 shows the per-category error metrics. Apart from the outlier categories *cabinet*, *car*, and *speaker*, the error metrics behave very similar across categories. They improve strongly when going from 3 to 10 and from 10 to 30 patches, and they improve at most slightly when going from 30 to 100 patches.

**Neural Sizes.** Next, the impact of different sizes of the latent codes and hidden dimensions used for the patch network is evaluated. Size refers to both the dimensions of the patch latent vector and the hidden dimensions of PatchNet. Again, Tab. 5.4 contains summary results, while Fig. 5.7 contains detailed results. The gap between size 128 and 512 is much smaller than between 32 and 128. Larger latent codes and hidden dimensions lead to higher quality results. Similarly, training on the full training dataset, consisting of 33*k* shapes leads to higher quality. However, all design choices with better performance come at the cost of longer training times, see Tab. 5.4.

**Losses.** Finally, each of the extrinsics losses is ablated as well as the need for guiding the rotation via initialization and a loss function. Tab. 5.5 contains the results. Due to the initialization, as described in Sec. 5.2.2, the extrinsics losses are not necessary in this setting. However, as shown

| | IoU | Chamfer | F-score |
|---|---|---|---|
| no $\mathcal{L}_{\text{sur}}$ | 92.0 | 0.049 | 94.8 |
| no $\mathcal{L}_{\text{cov}}$ | 90.7 | 0.051 | 93.6 |
| no $\mathcal{L}_{\text{rot}}$ | 92.5 | 0.043 | 95.4 |
| no $\mathcal{L}_{\text{scl}}$ | 91.2 | 0.031 | 94.3 |
| no $\mathcal{L}_{\text{var}}$ | 91.6 | 0.045 | 94.4 |
| random rotation initialization and no $\mathcal{L}_{\text{rot}}$ | 89.0 | 0.048 | 93.1 |
| PatchNets | 91.6 | 0.045 | 94.5 |
| PatchNets with $\mathcal{L}_{\text{recon}}$ on mixture | 94.0 | 0.026 | 96.8 |

Table 5.5: Ablation Study of PatchNet. Each of the extrinsics losses is removed. Also, the reconstruction loss is imposed on the mixture (using $g_i(\mathbf{x})$ from Eq. 11 instead of $f(\mathbf{x}, \mathbf{z}_{i,p}, \theta)$).

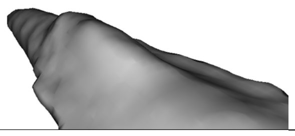
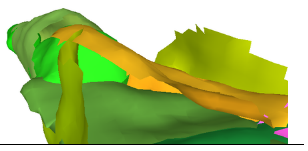
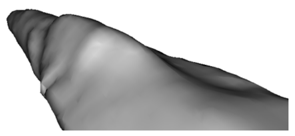
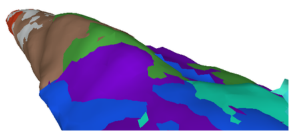
| | mixture | patches |
|---|---|---|
| $\mathcal{L}_{\text{recon}}$ on mixture |  |  |
| PatchNets |  |  |

Figure 5.8: Mixture Reconstruction Loss. Imposing the reconstruction loss on the mixture instead of directly on the patches leads to individual patches not matching the surface.

in Sec. B.2, they are necessary when the extrinsics are regressed instead of free. Initializing and encouraging the rotation towards normal alignment helps. Using $\mathcal{L}_{\text{recon}}$ on the mixture does not sufficiently constrain the patches to individually reconstruct the surface, as Fig. 5.8 shows.

### 5.3.3 *Object-Level Priors*

While PatchNet benefits from its local representation, certain tasks need a global prior. Thus, the following experiments concern *category-specific* object priors. To this end, ObjectNet (four FC layers with hidden dimension 1024 and ReLU activations) is added in front of PatchNet and the baselines. From object latent codes of size 256, ObjectNet regresses patch latent codes and extrinsics as an intermediate representation usable with

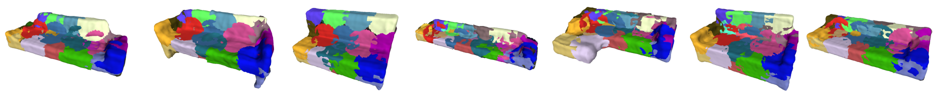


Figure 5.9: Coarse Correspondences. Note the consistent coloring of the patches.

PatchNet. ObjectNet effectively increases the network capacity of the baselines.

**Training.** All object latents are initialized with zeros and the weights of ObjectNet's last layer with very small numbers. The bias of ObjectNet's last layer is initialized with zeros for patch latent codes and with the extrinsics of an arbitrary object from the category as computed by the initialization in Sec. 5.2.2. PatchNet is pretrained on ShapeNet. For the proposed method, the PatchNet is kept fixed from this point on. The full training split of the ShapeNet category considered in the respective experiment is used for training. Removing $\mathcal{L}_{\text{rot}}$ completely significantly improves quality. The $L2$ regularization is only applied to the object latent codes. The training uses $\omega_{\text{var}} = 5$. ObjectNet is trained in three phases, each lasting 1000 epochs. The same initial learning rates as when training PatchNet are used, except in the last phase, where they are reduced by a factor of 5. The batch size is 128.

*Phase I*: Pretraining ObjectNet ensures good patch extrinsics. For this, the extrinsic loss, $\mathcal{L}_{\text{ext}}$ in Eq. 5.3, and the regularizer are used, with $\omega_{\text{scl}} = 2$.

*Phase II*: Next, ObjectNet learns to regress patch latent codes. First, before training, a layer that multiplies the regressed scales by 1.3 is added. Then, these extrinsics are stored. Afterwards, ObjectNet is trained using $\mathcal{L}_{\text{recon}}$ and two $L2$ losses that keep the regressed position and scale close to the stored extrinsics, with respective weights 1, 3, and 30.

*Phase III*: The complete loss $\mathcal{L}$ in Eq. 5.1, with $\omega_{\text{scl}} = 0.02$, yields final refinements.

**Coarse Correspondences.** Fig. 5.9 shows that the learned patch distribution is consistent across objects, establishing coarse correspondences between objects.

**Interpolation.** Due to the implicitly learned coarse correspondences, test objects can be encoded into object latent codes and then be linearly interpolated. Fig. 5.10 shows that interpolation of the latent codes leads to a smooth morph between the decoded shapes in 3D space.

**Generative Model.** The learned object latent space allows to turn ObjectNet into a generative model. Since auto-decoding does not yield an encoder that inputs a known distribution, the unknown input distribution needs to be estimated. Therefore, a multivariate Gaussian is fit to the object latent codes obtained at training time. New object latent codes

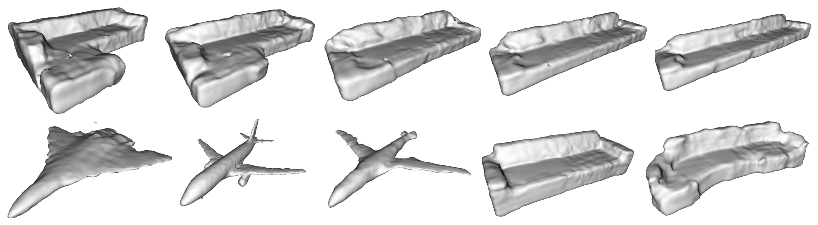


Figure 5.10: Interpolation (top). The left and right end points are encoded test objects. Generative Models (bottom). Object latents are sampled from ObjectNet's fitted prior.

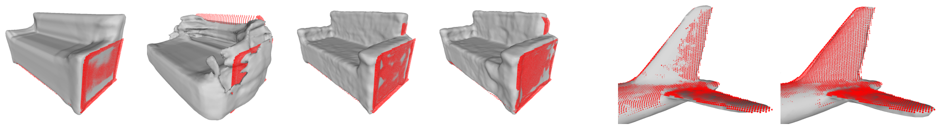


Figure 5.11: Shape Completion. (Sofa) from left to right: Baseline, DeepSDF, proposed unrefined, proposed refined. (Airplane) from left to right: Proposed unrefined, proposed refined.

can the be sampled from the fitted Gaussian and used to generate new objects, see Fig. 5.10.

**Partial Point Cloud Completion.** Given a partial point cloud, ObjectNet allows to optimize for the object latent code which best explains the visible region. ObjectNet acts as a prior which completes the missing parts of the shape. For the proposed method, PatchNet is pretrained on a different object category and then kept fixed, and then ObjectNet is trained on the target category, which makes this task more challenging for the proposed method. In contrast, for the baselines, the eight final layers are pretrained on all categories and finetuned on the target shape category. Evaluations of several other settings revealed this one to be the most competitive. See Sec. B.2 for more on surface reconstruction with object-level priors.

*Optimization*: The object latent code is initialized to the average of the object latent codes obtained at training time. Then, optimization runs for 600 iterations, starting with a learning rate of 0.01 and halving every 200 iterations. Since ObjectNet regresses the patch latent codes and extrinsics as an intermediate step, the result can be further refined by treating this intermediate patch-level representation as free variables. Specifically, the patch latent code is refined for the last 100 iterations with a learning rate of 0.001, while keeping the extrinsics fixed. This allows to integrate details not captured by the object-level prior. Fig. 5.11 demonstrates this effect. Optimization uses the reconstruction loss, the $L2$ regularizer, and the coverage loss. The other extrinsics losses have a detrimental effect on

| | sofas fixed | | sofas random | | airplanes fixed | | airplanes random | |
|---|---|---|---|---|---|---|---|---|
| | acc. | F-score | acc. | F-score | acc. | F-score | acc. | F-score |
| baseline | 0.094 | 43.0 | 0.092 | 42.7 | 0.069 | 58.1 | 0.066 | 58.7 |
| DeepSDF-based baseline | 0.106 | 33.6 | 0.101 | 39.5 | 0.066 | 56.9 | 0.065 | 55.5 |
| proposed | 0.091 | 48.1 | 0.077 | 49.2 | 0.058 | 60.5 | 0.056 | 59.4 |
| proposed+refined | **0.052** | **53.6** | **0.053** | **52.4** | **0.041** | **67.7** | **0.043** | **65.8** |

Table 5.6: Partial Point Cloud Completion from Depth Maps. Depth maps are completed from a fixed camera viewpoint and from per-scene random viewpoints.

patches that are outside the partial point cloud. Each iteration uses 8$k$ samples.

The partial point clouds are obtained from depth maps, similar to Park et al. (2019). The proposed method also employs their free-space loss, which encourages the network to regress positive values for samples between the surface and the camera, using 30% free-space samples. The depth maps can be from a fixed or from a per-scene random viewpoint. For shape completion, the F-score between the full ground-truth mesh and the reconstructed mesh is reported. Similar to Park et al. (2019), the mesh accuracy measures shape completion. It is the 90th percentile of shortest distances from the surface samples of the reconstructed shape to surface samples of the full ground truth. Tab. 5.6 shows how, due to local refinement on the patch level, ObjectNet outperforms the baselines everywhere.

### 5.3.4 *Articulated Deformation*

The proposed patch-level representation can model some articulated deformations by *only* modifying the patch extrinsics, without needing to adapt the patch latent codes. Given a template surface and patch extrinsics for this template, it first needs to be encoded into patch latent codes. After manipulating the patch extrinsics, an articulated surface can be obtained with the smooth blending from Eq. 5.11, as Fig. 5.12 demonstrates.

## 5.4 LIMITATIONS

The SDF is sampled using DeepSDF's sampling strategy, which might limit the level of detail. Generalizability at test time requires optimizing patch latent codes and extrinsics, a problem shared with other auto-

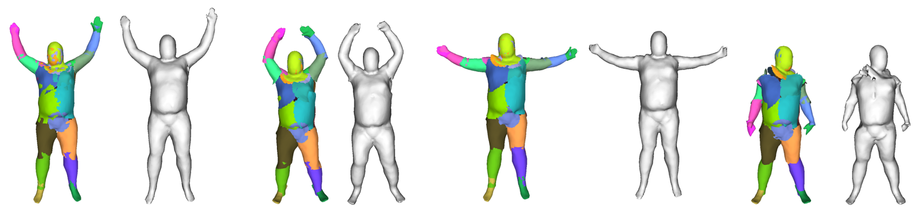


Figure 5.12: Articulated Motion. A template shape is encoded into patch latent codes (first pair). Then, the patch extrinsics are modified, while keeping the patch latent codes fixed, leading to non-rigid deformations (middle two pairs). The last pair shows a failure case due to large non-rigid deformations away from the template. Note that the colored patches move rigidly across poses while the mixture deforms non-rigidly.

decoders. Auto-encoding is slow at test time: Fitting the reduced test set takes 71 min due to batching, one object takes 10 min.

## 5.5 CONCLUSION

This chapter presents a mid-level geometry representation based on patches. This representation leverages the similarities of objects at patch level, leading to a highly generalizable neural shape representation. For example, the proposed representation, trained on one object category, can also represent other categories. PatchNets enable a large variety of downstream applications that go far beyond shape interpolation and point cloud completion.

This concludes the first half of the thesis, which is about representations. Two methods were introduced: DEMEA models the deformations and PatchNets models the geometry of general non-rigid objects with neural techniques. Next, this thesis turns from representing general non-rigid objects to the other part of 4D digitization: reconstruction.

Part II

# Reconstructing General Non-Rigid Objects

# 6 NON-RIGID NEURAL RADIANCE FIELDS: RECONSTRUCTION AND NOVEL VIEW SYNTHESIS OF A DYNAMIC SCENE FROM MONOCULAR VIDEO

The second half of this thesis is concerned with reconstruction from image data rather than representing general non-rigid objects. As before, neural techniques, in particular neural radiance fields, are used to this end.

This chapter presents Non-Rigid Neural Radiance Fields (NR-NeRF), a reconstruction and novel view synthesis approach for general non-rigid dynamic scenes (also published as Tretschk et al. (2021)). The proposed approach takes RGB images of a dynamic scene as input (*e.g.*, from a monocular video recording), and creates a high-quality space-time geometry and appearance representation. Experiments show that a single handheld consumer-grade camera is sufficient to synthesize sophisticated renderings of a dynamic scene from novel virtual camera views, *e.g.* a 'bullet-time' video effect. Further experiments examine the quality limit of the method via a straightforward (optional) extension to multi-view input. NR-NeRF disentangles the dynamic scene into a canonical volume and its deformation. Scene deformation is implemented as ray bending, where straight rays are deformed non-rigidly. This chapter also proposes a novel rigidity network to better constrain rigid regions of the scene, leading to more stable results. The ray bending and rigidity network are trained without explicit supervision. The proposed formulation enables dense correspondence estimation across views and time, and compelling video editing applications such as motion exaggeration. Overall, the approach proposed in this chapter enables free-viewpoint rendering of general deformable scenes with multiple objects and complex deformations with high visual fidelity, and yet does not rely on templates, 2D correspondences, and multi-view setups. Fig. 6.1 provides an overview of the results.

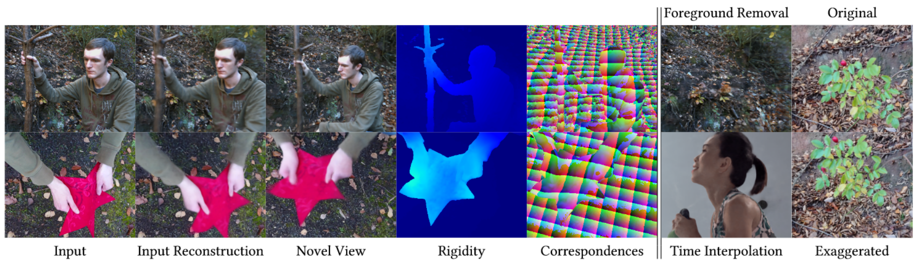


Figure 6.1: Given a monocular image sequence, NR-NeRF reconstructs a single canonical neural radiance field to represent geometry and appearance, and a per-time-step deformation field. The scene can be rendered into a novel spatio-temporal camera trajectory that significantly differs from the input trajectory. NR-NeRF also learns rigidity scores and correspondences without direct supervision on either. The rigidity scores can be used to remove the foreground, to supersample along the time dimension, and to exaggerate or dampen motion.

## 6.1 INTRODUCTION

Free viewpoint rendering is a well-studied problem due to its wide range of applications in movies and virtual/augmented reality (Collet et al., 2015; Miller et al., 2005; Smolic et al., 2006). This chapter focuses on dynamic scenes, which change over time, from novel user-controlled viewpoints. Traditionally, multi-view recordings are required for free viewpoint rendering of dynamic scenes (Oswald et al., 2014; Tung et al., 2009; Zhang et al., 2003). However, such multi-view captures are expensive and cumbersome. The goal of this chapter is to enable the setting in which a casual user records a dynamic scene with a single, moving consumer-grade camera. Access to only a monocular video of the deforming scene leads to a severely under-constrained problem. Most existing approaches thus limit themselves to a single object category, such as the human body (Habermann et al., 2019; Kocabas et al., 2020; Xiang et al., 2019) or face (Egger et al., 2020a). Some approaches allow for the reconstruction of general non-rigid objects (Garg et al., 2013; Kumar et al., 2018; Sidhu et al., 2020; Zollhöfer et al., 2018), but most methods only reconstruct the geometry without the appearance of the objects in the scene. In contrast, this chapter aims to reconstruct a general dynamic scene, including its appearance, such that it can be rendered from novel spatio-temporal viewpoints. Recent neural rendering approaches have shown impressive novel-view synthesis of general static scenes from multi-view input (Tewari et al., 2020). These approaches represent scenes using trained neural networks and rely on fewer constraints about the type of scene, compared to traditional approaches. The closest prior work to the proposed method is NeRF (Mildenhall et al., 2020), which learns a continuous volume of the scene encoded in a neural network using multiple camera views. However, NeRF assumes the scene to be static. Neural Volumes (Lombardi et al., 2019) is another closely related approach that uses multiple views of a deforming scene to enable free-viewpoint rendering. However, it uses a fixed-size voxel grid to represent the reconstruction of the scene, restricting the resolution. In addition, it requires multi-view input for training, which limits the applicability to in-the-wild outdoor settings or existing monocular footage. The proposed

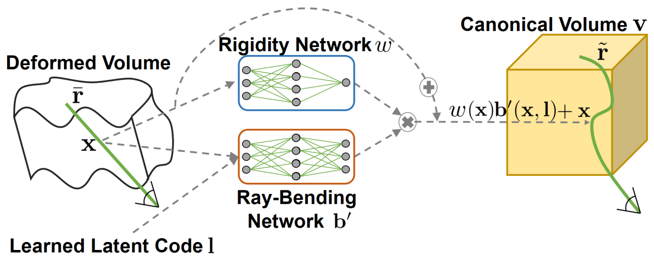

Canonical Volume $\mathbf{v}$
Rigidity Network $w$
Deformed Volume
$\bar{\mathbf{r}}$
$\mathbf{x}$
$w(\mathbf{x})\mathbf{b}'(\mathbf{x},\mathbf{l})+\mathbf{x}$
$\tilde{\mathbf{r}}$
Ray-Bending Network $\mathbf{b}'$
Learned Latent Code $\mathbf{l}$

Figure 6.2: NR-NeRF bends straight rays $\bar{\mathbf{r}}$ from the deformed volume using a deformation-dependent ray-bending network $\mathbf{b}'$ and a deformation-independent rigidity network $w$ into a single static canonical neural radiance field volume $\mathbf{v}$.

neural rendering approach instead targets the more challenging setting of using just a monocular video of a general dynamic scene. Due to the non-rigidity, each image of the video records a different, deformed state of the scene, violating the constraints of standard neural rendering approaches. The approach disentangles the observations in any image into a canonical scene and its deformations, without direct supervision on either. Several innovations are necessary to tackle this problem. The non-rigid scene is represented by two components, see Fig. 6.2: (1) a canonical neural radiance field for capturing geometry and appearance and (2) the scene deformation field. The canonical volume is a static representation of the scene encoded as a Multi-Layered Perceptron (MLP), which is not directly supervised. This volume is deformed into each individual image using the estimated scene deformation. Specifically, the scene deformation is implemented as ray bending, where straight camera rays can deform non-rigidly. The ray bending is modeled using an MLP that takes point samples on the ray as well as a latent code for each image as input. Both the ray bending and the canonical scene MLPs are jointly trained using monocular observations. Since the ray bending MLP deforms the entire space independent of camera parameters, the deforming volume can be rendered from static or time-varying novel viewpoints after training.

The ray bending MLP disentangles the geometry of the scene from the scene deformations. The disentanglement is an underconstrained problem, which is tackled with further innovations. The proposed method assigns a rigidity score to every point in the canonical volume, which allows for the deformations to not affect the static regions in the scene. This rigidity component is jointly learned without any direct supervision. Multiple regularizers act as additional soft-constraints: A regularizer on the deformation magnitude of the *visible* deformations encourages only sparse deformations of the volume, and thus helps to constrain the canonical volume. An additional divergence regularizer preserves the local shape, thereby constraining the representation of *hidden* (partially occluded) regions that are not visible throughout the full video.

Results in Sec. 6.3 show high-fidelity reconstruction and novel view synthesis for a wide range of non-rigid scenes. These experiments also compare NR-NeRF to several methods for neural novel view rendering. Sec. 6.4 discusses the limitations.

## 6.2 METHOD

This chapter proposes Non-Rigid Neural Radiance Field (NR-NeRF). It takes as input a set of $N$ RGB images $\{\hat{\mathbf{c}}_i\}_{i=0}^{N-1}$ of a non-rigid scene and their extrinsics $\{\mathbf{R}_i, \mathbf{t}_i\}_{i=0}^{N-1}$ and intrinsics $\{\mathbf{K}_i\}_{i=0}^{N-1}$. NF-NeRF then finds a single canonical neural radiance volume that can be deformed via ray bending to correctly render each $\hat{\mathbf{c}}_i$. Specifically, appearance and geometry information is collected in the static canonical volume $\mathbf{v}$ parametrized by weights $\theta$. Deformations are modeled by bending the straight rays sent out by a camera to obtain a deformed rendering of $\mathbf{v}$. This ray bending is implemented as a ray bending MLP $\mathbf{b}$ with weights $\psi$. It maps, conditioned on the current deformation, 3D points (*e.g.,* sampled from the straight rays) to 3D positions in $\mathbf{v}$. The deformation conditioning takes the form of auto-decoded latent codes $\{\mathbf{l}_i\}_{i=0}^{N-1}$ for each image $i$.

### 6.2.1 *Adaptations of NeRF for NR-NeRF*

This chapter assumes Lambertian materials and thus removes the view-dependent layers of rigid NeRF, *i.e.,* $\mathbf{c} = \mathbf{c}(\mathbf{x})$. Since each image corresponds to a different deformation of the volume in the non-rigid setting, a latent code is learned for each time step, which is then used as input for the ray bending network which parameterizes scene deformations. The weights of this network and the latent codes are shared between the coarse NeRF model $\mathbf{v}_c$ and the fine NeRF model $\mathbf{v}_f$ (see Sec. 3.4).

### 6.2.2 *Deformation Model*

The original NeRF method (Mildenhall et al., 2020) assumes rigidity and cannot handle non-rigid scenes. A naïve approach to modeling deformations in the NeRF framework would be to condition the volume on the deformation (*e.g.,* by conditioning it on time or a deformation latent code). Sec. 6.3.4 explores the latter option experimentally. As is shown there, apart from not providing hard correspondences, this naïve approach only leads to satisfying results when reconstructing the input camera path, but gives implausible results for novel view synthesis. Instead, NR-NeRF explicitly models the consistency of geometry and appearance across time by disentangling them from the deformation.

The proposed method accumulates geometry and appearance from all frames into a single, non-deforming canonical volume. General space warping (or ray bending) on top of the static canonical volume allows to model non-rigid deformations.

For an input image $\hat{\mathbf{c}}_i$ at training time, the goal is to render the canonical volume such that the image is reproduced. Thus, the deformation of the specific time step $i$ needs to be un-done by mapping the camera rays to the deformation-independent canonical volume. To this end, straight rays are sent from the input camera. To account for the deformation, they are then bent such that sampling and subsequently rendering the canonical volume along the bent rays yields $\hat{\mathbf{c}}_i$. NR-NeRF uses a very unrestricted parametrization of the ray bending, namely an MLP.

Specifically, ray bending is implemented as a ray bending network $\mathbf{b}(\mathbf{x}, \mathbf{l}_i) \in \mathbb{R}^3$. For a point $\mathbf{x}$, for example lying on a straight ray, the network regresses an offset under a deformation represented by $\mathbf{l}_i$. The offset is then added to $\mathbf{x}$, thereby bending the ray. Finally, the new, bent ray point is passed to the canonical volume, that is: $(\mathbf{c}, o) = \mathbf{v}(\mathbf{x} + \mathbf{b}(\mathbf{x}, \mathbf{l}_i))$. Note that $\mathbf{v}$ is not conditioned on $\mathbf{l}_i$, which leads to the disentanglement of deformation ($\mathbf{b}$ and $\mathbf{l}_i$) from geometry and appearance ($\mathbf{v}$). The bent version of the straight ray $\bar{\mathbf{r}}$ is denoted as $\tilde{\mathbf{r}}_{\mathbf{l}_i}(j) = \bar{\mathbf{r}}(j) + \mathbf{b}(\bar{\mathbf{r}}(j), \mathbf{l}_i)$.

**Rigidity Network.** However, rigid parts of the scene are insufficiently constrained by this formulation. NR-NeRF thus reformulates $\mathbf{b}(\mathbf{x}, \mathbf{l}_i) \in \mathbb{R}^3$ as the product of a raw offset $\mathbf{b}'(\mathbf{x}, \mathbf{l}_i)$ and a rigidity mask $w(\mathbf{x}) \in [0, 1]$, *i.e.*, $\mathbf{b}(\mathbf{x}, \mathbf{l}_i) = w(\mathbf{x})\mathbf{b}'(\mathbf{x}, \mathbf{l}_i)$. For rigid objects, deformations need to be prevented and hence $w(\mathbf{x}) = 0$ is desired, while for non-rigid objects, $w(\mathbf{x}) > 0$. This makes it easier for $\mathbf{b}'$ to focus on the non-rigid parts of the scene, which change over time, since rigid parts can get masked out by the rigidity network $w$, which is jointly trained. Because the rigidity network is not conditioned on the latent code $\mathbf{l}_i$, it is forced to share knowledge about the rigidity of regions in the scene across time steps, which also ensures that parts of the rigid background that can be unregularized at certain time steps are nonetheless reconstructed at all time steps without any deformation.

### 6.2.3 *Losses*

With the architecture specified, all parameters ($\theta$, $\psi$, $\{\mathbf{l}_i\}_i$) are next optimized jointly. The network weights are treated as usual but the latent codes $\mathbf{l}_i$ are auto-decoded (Park et al., 2019; Tan and Mayrovouniotis, 1995).

**Notation.** For ease of presentation, this section considers a single time step $i$ and a single straight ray $\bar{\mathbf{r}}$ with coarse ray points $\bar{C} = \{\bar{\mathbf{r}}(j)\}_{j \in C}$ for a set $C$ of uniformly sampled $j \in [j_n, j_f]$ and fine ray points $\bar{F} = \{\bar{\mathbf{r}}(j)\}_{j \in F}$ for a set $F$ of importance-sampled $j$. For a latent code $\mathbf{l}$, the bent ray $\tilde{\mathbf{r}}_\mathbf{l}$ gives $\tilde{C} = \{\tilde{\mathbf{r}}_\mathbf{l}(j)\}_{j \in C}$ and $\tilde{F} = \{\tilde{\mathbf{r}}_\mathbf{l}(j)\}_{j \in F}$. The actual training uses a batch of randomly chosen rays from the training images.

**Reconstruction Loss.** NR-NeRF adapts the data term from NeRF to the non-rigid setting as follows:

$$L_{data} = \|\mathbf{c}_c(\tilde{C}) - \hat{\mathbf{c}}(\mathbf{r})\|_2^2 + \|\mathbf{c}_f(\tilde{C} \,\dot{\cup}\, \tilde{F}) - \hat{\mathbf{c}}(\mathbf{r})\|_2^2 \ , \tag{6.1}$$

where $\hat{\mathbf{c}}(\mathbf{r})$ is the ground-truth color of the pixel and $\mathbf{c}(S)$ is the estimated ray color on the set $S$ of discrete ray points.

While this reconstruction loss yields satisfactory results along the space-time camera trajectory of the input recording, Sec. 6.3.3 will later show that it leads to undesirable renderings for novel views. It is thus necessary to regularize the bending of rays with further priors.

**Offsets Loss.** The offsets are regularized with a loss on their magnitude. Since the goal is for visually unoccupied space (*i.e.*, air) to be compressible and not hinder the optimization, the loss at each point is weighted by its opacity. However, this would still apply a high weight to completely occluded points along the ray, which leads to artifacts when rendering novel views. Thus, the loss is additionally weighted by transmittance:

$$L_{naïve\ offsets} = \frac{1}{|C|} \sum_{j \in C} \alpha_j \cdot \|\mathbf{b}(\bar{\mathbf{r}}(j), \mathbf{l})\|_2^{2 - w(\bar{\mathbf{r}}(j))}, \tag{6.2}$$

where each point is weighted by transmittance and occupancy $\alpha_j = V(j) \cdot o(\tilde{\mathbf{r}}(j))$. Gradients are not back-propagated into $\alpha_j$ to avoid undesirable local minima.

However, as Sec. 6.3 shows, applying the offsets loss to the masked offsets leads to an unstable background in novel views. Presumably, this is due to the multiplicative ambiguity between unmasked offsets and rigidity mask. Applying the loss to the regressed rigidity mask and raw offsets separately works better:

$$L_{offsets} = \frac{1}{|C|} \sum_{j \in C} \alpha_j \cdot \left(\|\mathbf{b}'(\bar{\mathbf{r}}(j), \mathbf{l})\|_2^{2 - w(\bar{\mathbf{r}}(j))} + \omega_{\text{rigidity}} w(\bar{\mathbf{r}}(j))\right), \tag{6.3}$$

where $\mathbf{b}'$ is penalized instead of $\mathbf{b}$. The exponent of the first term is a tweak to get two desirable properties that neither an $\ell_1$ nor an $\ell_2$ loss fulfills: For non-rigid objects ($w$ closer to 1), it becomes an $\ell_1$ loss, which has two advantages: (1) the gradient is independent of the magnitude of the offset, so unlike with an $\ell_2$ loss, small and large offsets/motions are treated equally, and (2) relative to an $\ell_2$ loss, it encourages sparsity in the offsets field, which fits the scenes considered in the experiments. For rigid objects ($w$ closer to 0), it becomes an $\ell_2$ loss, which tampers off in its gradient magnitude as the offset magnitude approaches 0, preventing noisy gradients that an $\ell_1$ loss has for the tiny offsets of rigid objects.

**Divergence Loss.** Since the offsets loss only constrains visible areas, additional regularization of hidden areas is required. Inspired by local, isometric shape preservation from computer graphics, like as-rigid-as-possible regularization for surfaces (Igarashi et al., 2005; Sorkine and Alexa, 2007) or volume preservation for volumes (Slavcheva et al., 2017), the regularization seeks to preserve the local shape after deformation. To this end, this chapter proposes to regularize the absolute value of the divergence of the offsets field, which Fig. 6.3 visualizes. The Helmholtz decomposition (Bhatia et al., 2012) allows to split any twice-differentiable 3D vector field on a bounded domain into a sum of a rotation-free and a divergence-free vector field. Thus, by penalizing the divergence, the vector field is encouraged to be composed primarily of translations and rotations, effectively preserving volume. The divergence loss is:

$$L_{divergence} = \frac{1}{|C|} \sum_{j \in C} w'_j \cdot |\mathrm{div}(\mathbf{b}(\bar{\mathbf{r}}(j), \mathbf{l}))|^2 \ , \qquad (6.4)$$

where no gradients are back-propagated into $w'_j = o(\tilde{\mathbf{r}}(j))$, and the divergence div of $\mathbf{b}$ is taken w.r.t. the position $\bar{\mathbf{r}}(j)$.

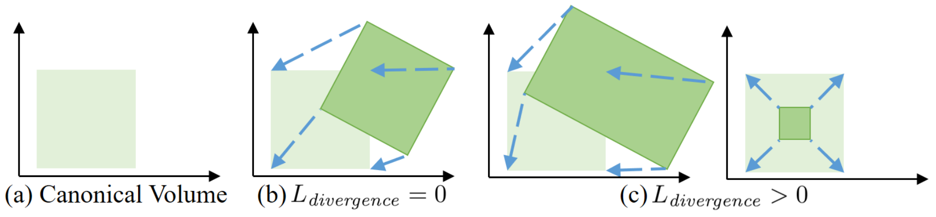


Figure 6.3: $L_{divergence}$ encourages the offsets field to (b) preserve local volume rather than (c) losing it while deforming.

NR-NeRF employs FFJORD's (Grathwohl et al., 2018) fast, unbiased divergence estimation, which is three times less computationally expensive than an exact computation. The divergence is defined as:

$$\mathrm{div}(\mathbf{b}(\mathbf{x})) = \mathrm{Tr}\left(\frac{\mathrm{d}\mathbf{b}(\mathbf{x})}{\mathrm{d}\mathbf{x}}\right) = \frac{\partial \mathbf{b}(\mathbf{x})_x}{\partial x} + \frac{\partial \mathbf{b}(\mathbf{x})_y}{\partial y} + \frac{\partial \mathbf{b}(\mathbf{x})_z}{\partial z}, \qquad (6.5)$$

where $\mathbf{b}(\mathbf{x})_k \in \mathbb{R}$ is the $k$-th component of $\mathbf{b}(\mathbf{x})$, $\mathrm{Tr}(\cdot)$ is the trace operator, and $\frac{\mathrm{d}\mathbf{b}(\mathbf{x})}{\mathrm{d}\mathbf{x}}$ is the $3 \times 3$ Jacobian matrix. Naïvely computing the divergence with PyTorch's automatic differentiation requires three backward passes, one for each term of the sum. Instead, the authors of FFJORD (Grathwohl et al., 2018) use Hutchinson's trace estimator (Hutchinson, 1989):

$$\mathrm{Tr}(\mathbf{A}) = \mathbb{E}_{\mathbf{e}}[\mathbf{e}^T \mathbf{A} \mathbf{e}]. \tag{6.6}$$

Here, **e** is Gaussian-distributed. The single-sample Monte-Carlo estimator implied by this expectation can be computed with a single backward pass.
**Full Loss.** All losses are combined to obtain the full loss:

$$L = L_{\text{data}} + \omega_{\text{offsets}} L_{\text{offsets}} + \omega_{\text{divergence}} L_{\text{divergence}}, \tag{6.7}$$

where the weights $\omega_{\text{rigidity}}$, $\omega_{\text{offsets}}$, and $\omega_{\text{offsets}}$ are scene-specific since NR-NeRF is applied to a variety of non-rigid scene types. The implementation uses the structure-from-motion method COLMAP (Schönberger and Frahm, 2016) to estimate the camera parameters. Sec. C.1 and Sec. C.2 contain further training and implementation details.

## 6.3 RESULTS

This section experimentally evaluates NR-NeRF. First, Sec. 6.3.1 provides information on data capture. Sec. 6.3.2 then presents qualitative results of the proposed method, including rigidity scores and correspondences, by rendering into input and novel spatio-temporal views. Turning to the inner workings, Sec. 6.3.3 investigates the crucial design choices made to improve novel view quality. Sec. 6.3.4 concludes the evaluation of NR-NeRF by comparing to prior work and a baseline approach. Finally, Sec. 6.3.5 shows simple scene-editing results. The appendix shows extensions to multi-view data and view-dependent effects (Sec. C.4).

### 6.3.1 *Data*

NR-NeRF is evaluated on a variety of scenes recorded with three different cameras: the Kinect Azure, a Blackmagic, and a phone camera. Since the RGB camera of the Kinect Azure exhibits strong radial distortions along the image border, the recorded RGB images are undistorted beforehand using the manufacturer-provided intrinsics and distortion parameters. Frames are extracted at 5 fps from the recordings, such that scenes usually consist of 80 to 300 images, at resolutions of $480 \times 270$ (Blackmagic, and Sony XZ2) or $512 \times 384$ (Kinect Azure).

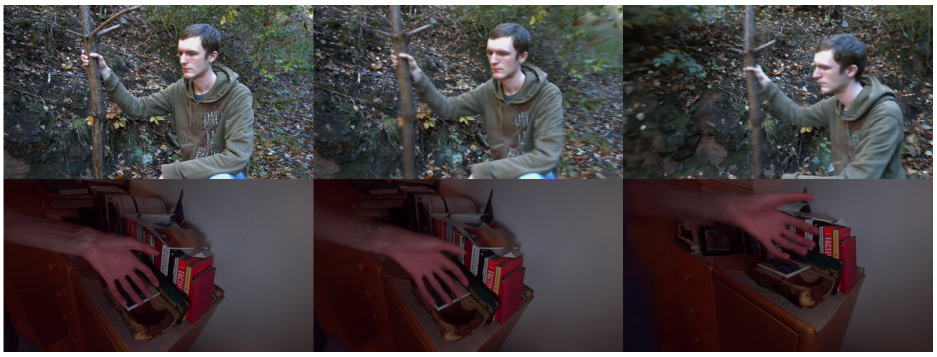


Figure 6.4: The input (left) is reconstructed by NR-NeRF (middle) and rendered into a novel view (right).

### 6.3.2 *Qualitative Results*

This section presents qualitative results of NR-NeRF by rendering the scene from input and novel spatio-temporal views. The additional outputs of the proposed approach are also visualized.

**Input Reconstruction and Novel View Synthesis.** Fig. 6.4 shows examples of input reconstruction and novel view synthesis with NR-NeRF. As the center column shows, the input is reconstructed faithfully. This enables high-quality novel view synthesis, example results can be found in the third column. The camera can be moved around freely in areas around the original camera paths, at any specific time step.

**Rigidity.** NR-NeRF estimates rigidity scores without supervision to improve background stability in novel-view renderings. In order to visualize the estimated rigidity, the rigidity of the ray associated with a pixel needs to be determined. Here, the rigidity of such a ray is defined as the rigidity of the point $j$ closest to an accumulated weight $\sum_{k=0}^{j-1} \alpha_k$ of 0.5, *i.e.*, closest to the median. In practice, this usually gives the rigidity at the first visible surface along the ray. Fig. 6.5 shows examples. The background is consistently scored as highly rigid, while the foreground is correctly estimated to be rather non-rigid.

**Correspondences.** Another side effect of the proposed approach is the ability to estimate consistent dense 3D correspondences into the canonical model across different camera views and time steps. To visualize correspondences, the canonical volume is treated as an RGB cube, *i.e.*, the xyz coordinate in canonical space is interpreted as an RGB color. Since this would result in very smooth colors, the canonical volume is split into a voxel grid of $100^3$ RGB cubes beforehand. The ray point that determines the pixel color is picked similar to the rigidity visualization. Fig. 6.5 shows examples.

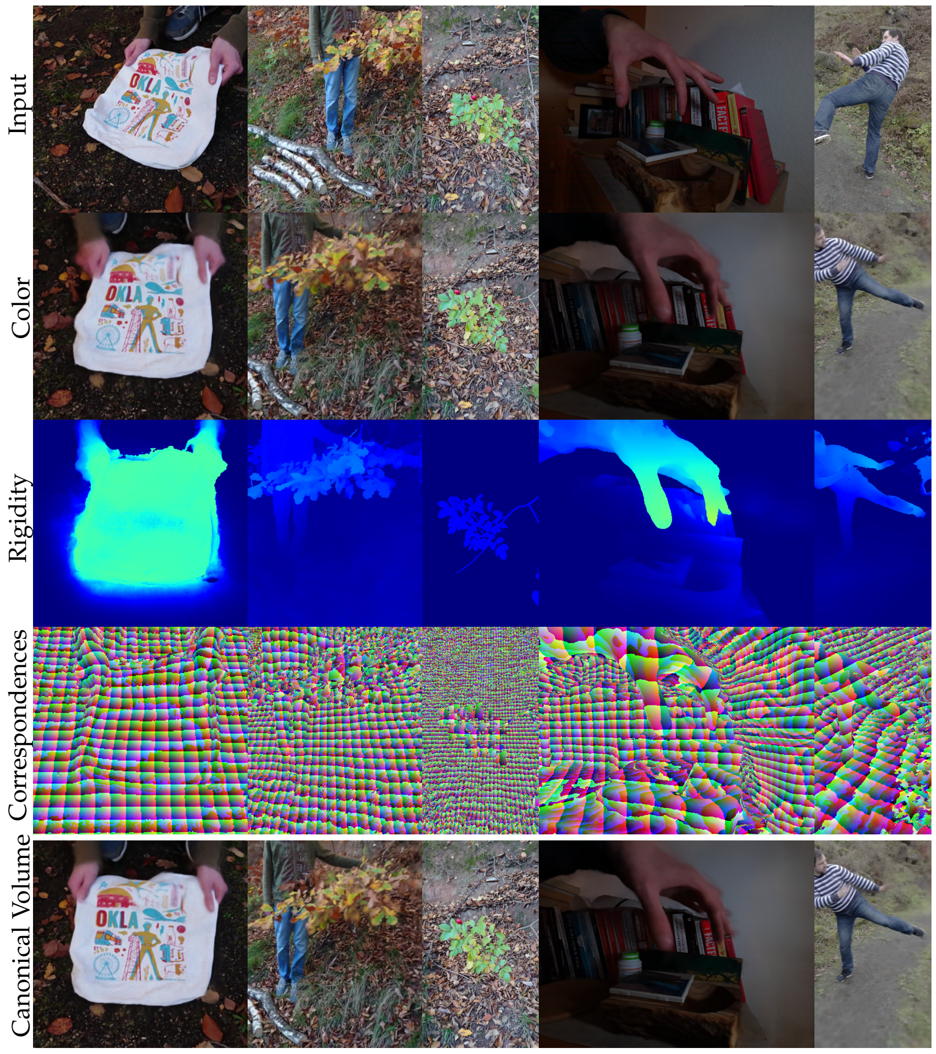


Figure 6.5: NR-NeRF can render a deformed state captured at a certain time step into a novel view. This is visualized here as novel-view rendering and additional output modalities as seen from the novel view, namely rigidity scores, correspondences, and the canonical volume. The canonical volume is a plausible state of the scene and does not show baked-in deformations.

**Canonical Volume.** Since the canonical volume is not supervised directly, it is conceivable that it could have baked-in deformations. The last row of Fig. 6.5 contains renderings of the canonical volume without any ray bending applied. The canonical volume is a plausible state of the scene and does not show baked-in deformations. It is thus sufficient to bias the optimization towards a desirable canonical volume by initializing the ray bending network to an identity map and by the regularization losses.

### 6.3.3 *Ablation Study*

After this look at the outputs of NR-NeRF, this section analyzes its internal workings further. Specifically, since the goal is convincing novel-view renderings, the impact of some of the design choices on the foreground and background stability of the novel-view results is investigated.
**Setup.** Removing each regularization loss individually and all of them at once assesses their necessity. Next, removing the rigidity network reveals its impact on background stability. Finally, the difference between applying the offsets loss separately on both the regressed rigidity and the unmasked offsets, *i.e.*, $L_{offsets}$ in NR-NeRF, or directly on the masked offsets, $L_{naïve\ offsets}$, is investigated.
**Results.** Fig. 6.6 shows that $L_{divergence}$ is crucial for stable deformations of the non-rigid objects in the foreground. On the other side, the interplay of *all* of the remaining design choices is necessary to stabilize the rigid background, as Fig. 6.7 shows.

### 6.3.4 *Comparisons*

Having only considered NR-NeRF in isolation so far, it is next compared to prior work and a baseline. In this section, the images are split into training and test sets by partitioning the temporally-ordered images into consecutive blocks of length 16 each, with the first twelve for the training set and the remaining four for the test set.
**Prior Work and Baseline.** The first baseline is the trivial baseline of rigid NeRF (Mildenhall et al., 2020), which cannot handle dynamic scenes. The experiments consider two variants: view-dependent rigid NeRF, as in the original method (Mildenhall et al., 2020), and view-independent rigid NeRF, where the view-direction conditioning is removed. Next, *naïve NR-NeRF* adds naïve support for dynamic scenes to rigid NeRF: It conditions the neural radiance fields volume on the latent code $\mathbf{l}_i$, *i.e.*, $(\mathbf{c}, o) = \mathbf{v}(\mathbf{x}, \mathbf{l}_i)$. For test images $i$, gradients are back-propagated into the corresponding latent code $\mathbf{l}_i$. NR-NeRF does the same in order to optimize for the test latent codes. Note that test images *solely* influence

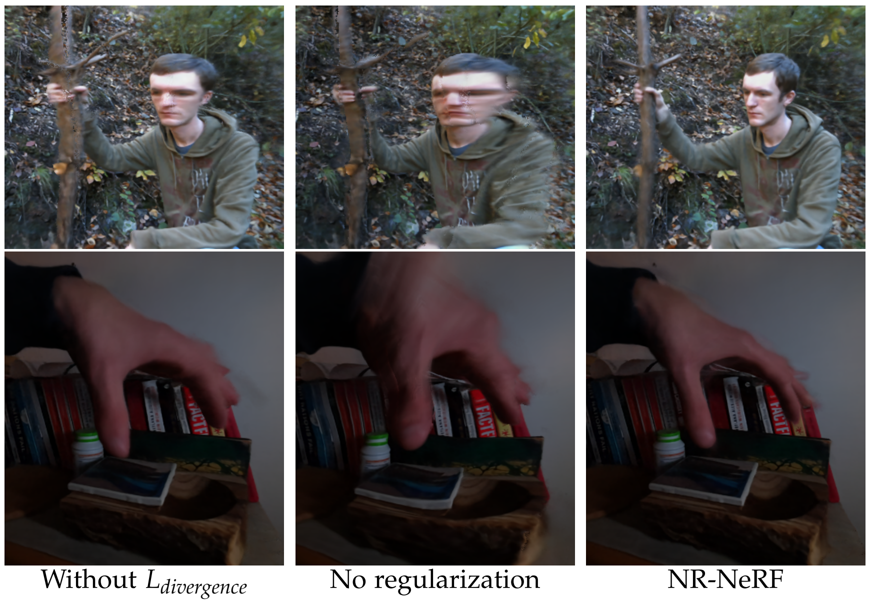


Figure 6.6: Ablation Study. The scene is rendered into novel views to determine the stability of the non-rigid part after removing the divergence loss, all regularization losses, and none of the losses.

test latent codes, as is typical for auto-decoding (Park et al., 2019). The final baseline is Neural Volumes (Lombardi et al., 2019), using the official code release. Two variants are considered: (1) as in Lombardi et al. (2019), the geometry and appearance template is conditioned on the latent code (*NV*), and (2) the geometry and appearance template are independent of the latent code (*modified NV*).

**Input Reconstruction.** Input reconstruction quality on the training set verifies the plausibility of the learned representations. See Fig. 6.8. Naïve NR-NeRF and both variants of Neural Volumes perform very well on this task, similar to the proposed method. However, rigid NeRF's not accounting for deformations leads to blur.

**Novel View Synthesis.** Next, the novel-view performance is evaluated qualitatively and quantitatively on the test sets. Fig. 6.8 contains novel-view results of all methods. Both versions of Neural Volumes give implausible results that are in some cases only barely recognizable. The two rigid NeRF variants show blurry, static results similar to the training reconstruction results earlier. While the still images in Fig. 6.8 show some undesirable artifacts like blurrier or less stable results compared to NR-NeRF, naïve NeRFs also exhibit temporal inconsistencies, especially on spatio-temporal trajectories different from the input.

After the qualitative overview, this section next quantitatively evaluates the novel-view results of the methods considered. The same three metrics

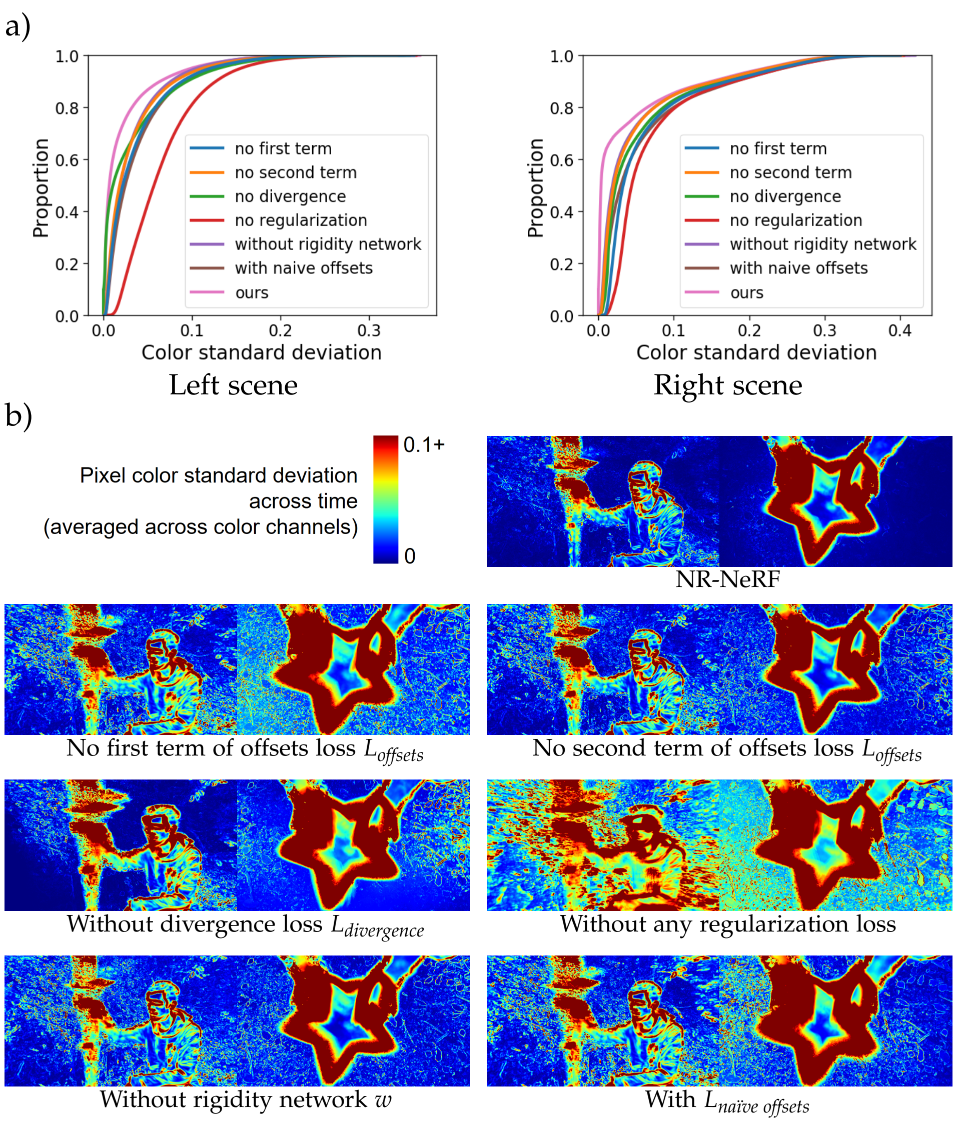


Figure 6.7: Ablation Study. The impact of the main design choices on background stability is quantified. To that end, the entire input sequence is rendered into a fixed novel view *for all time steps*. Then, the standard deviation of each pixel's color across time is computed to measure color changes and hence background stability. **a)** Cumulative plots across all pixels, where NR-NeRF (left-most curve) has the most stable background. **b)** The distribution of those instabilities in the scene. The results of NR-NeRF show the least instability in the background.

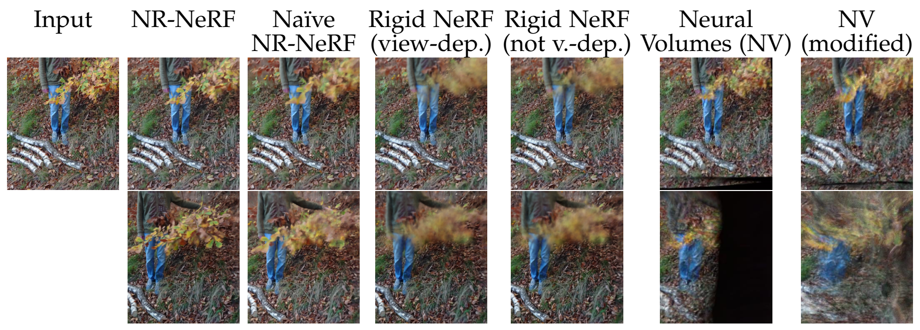


Figure 6.8: Comparison of input reconstruction quality (first row) and novel view synthesis quality (second row). Only NR-NeRF synthesizes sharp novel views.

| | NR-NeRF | Naïve | Rigid (cond.) | Rigid (no cond.) | NV | NV (mod.) |
|---|---|---|---|---|---|---|
| PSNR ↑ | *24.70* | **25.83** | 22.24 | 21.88 | 14.13 | 14.10 |
| SSIM ↑ | **0.758** | *0.738* | 0.662 | 0.659 | 0.259 | 0.263 |
| LPIPS ↓ | **0.197** | *0.226* | 0.309 | 0.313 | 0.580 | 0.583 |

Table 6.1: Quantitative Results Averaged Across Scenes. NR-NeRF, naïve NR-NeRF, rigid NeRF (Mildenhall et al., 2020) (1) with view conditioning and (2) without view conditioning, and Neural Volumes (Lombardi et al., 2019) (1) without and (2) with modifications are compared. For PSNR and SSIM (Wang et al., 2004), higher is better. For LPIPS (Zhang et al., 2018), lower is better.

as in NeRF (Mildenhall et al., 2020) are used: PSNR and SSIM (Wang et al., 2004) as conventional metrics for image similarity, where higher is better, and a learned perceptual metric, LPIPS (Zhang et al., 2018), where lower is better. Tab. 6.1 contains the quantitative results. NR-NeRF obtains the best SSIM and LPIPS scores, and the second-best PSNR after naïve NR-NeRF. As in the input reconstruction results, naïve NR-NeRF is competitive for settings that are close to the input spatio-temporal trajectory, as is the case for the test sets. Therefore, more challenging novel view scenarios are evaluated with a spatio-temporal trajectory significantly different from the input. Since there is no access to ground-truth novel view data, the evaluation focuses on background stability for a spatially fixed camera.

**Background Stability.** While a moving camera during rendering can obfuscate background instability, stabilizing the background for fixed novel-view renderings empirically matters for perceptual fidelity but is difficult to achieve. This section thus quantitatively evaluates this challenging task. Fig. 6.9 compares the background stability of NR-NeRF, naïve NR-NeRF, and the Neural Volumes variants. Rigid NeRF is excluded since it is

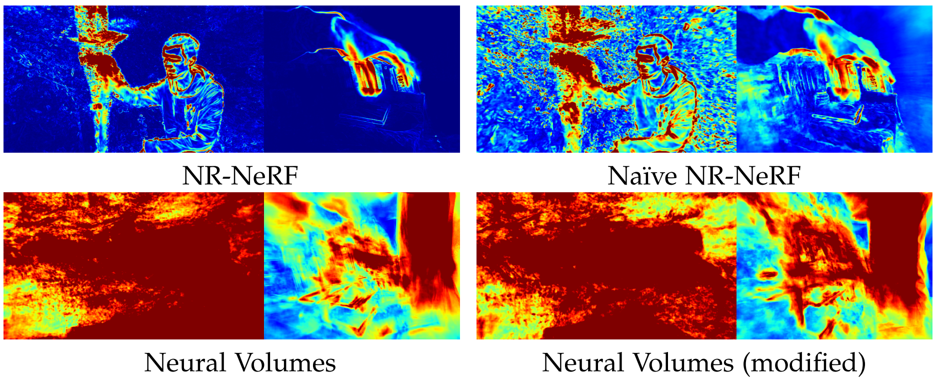


Figure 6.9: Background Stability. See Fig. 6.7 for an explanation. All test time steps are used here. The results of NR-NeRF show the least instability.

static by design. The proposed method leads to significantly more stable background synthesis than the other methods, and Sec. C.3 contains further results.

### 6.3.5 *Simple Scene Editing*

The learned model can be manipulated in several simple ways: foreground removal, temporal super-sampling, deformation exaggeration and dampening, and forced background stabilization.

**Foreground Removal.** The learned representation allows to remove a potentially occluding non-rigid object from the foreground, leaving only the unoccluded background. Assuming the rigidity network assigns higher scores to non-rigid objects than to rigid (background) objects, the scores can be thresholded at test time to segment the canonical volume into rigid and non-rigid parts. The non-rigid part can then be made transparent, see Fig. 6.11.

**Time Interpolation.** NR-NeRF can linearly interpolate between consecutive time steps to enable temporal super-resolution since it optimizes a latent code $\mathbf{l}_i$ for every time step $i$.

**Deformation Exaggeration/Dampening.** The deformation can be manipulated even further. Specifically, it is possible to exaggerate or dampen deformations relative to the canonical model by scaling all offsets with a constant $m \in \mathbb{R}$: $(\mathbf{c}, o) = \mathbf{v}(\mathbf{x} + m\mathbf{b}(\mathbf{x}, \mathbf{l}_i))$. Fig. 6.10 contains examples.

**Forced Background Stabilization.** Since NR-NeRF does not require any pre-computed foreground-background segmentation, it has to assign rigidity scores without supervision. Occasionally, this insufficiently constrains the background and leads to small motion. This can be fixed in some cases by enforcing a stable background at test time: the regressed

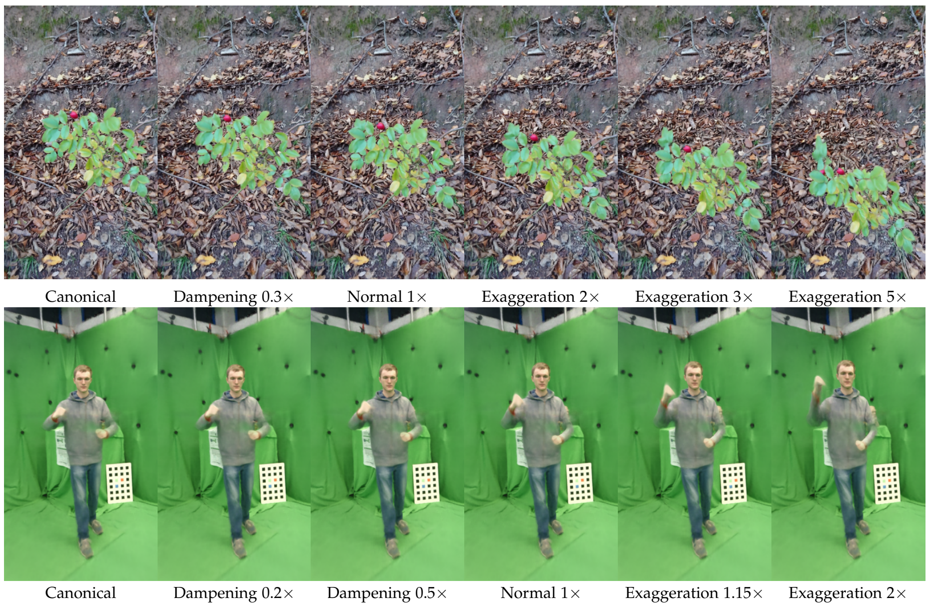


Figure 6.10: NR-NeRF allows to exaggerate or dampen the motion relative to the canonical model, and to render the result into a novel view.

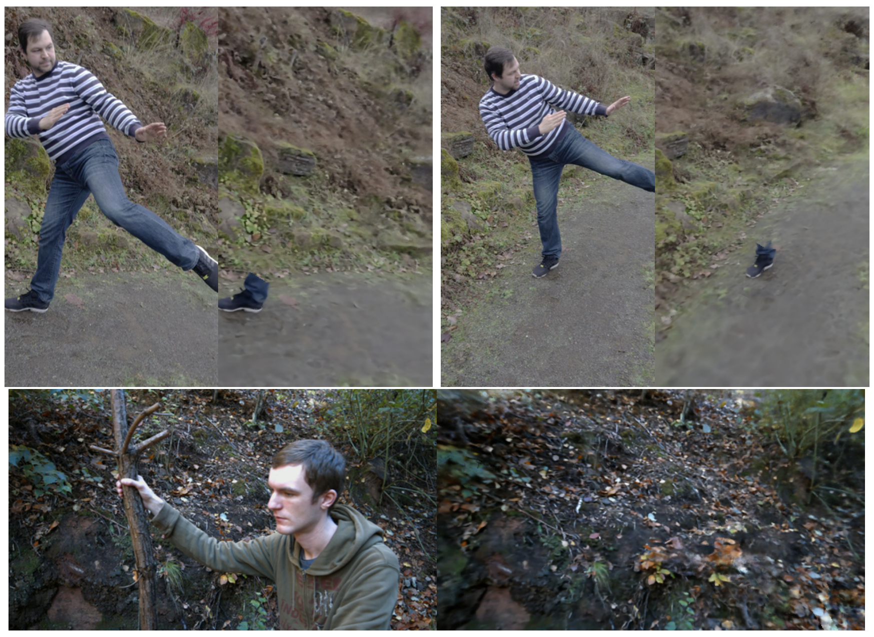


Figure 6.11: (Left) the ground-truth input image and (right) a rendering without non-rigid foreground.

score can be set to 0 if it is below some threshold $r_{min}$. If the rigid background has sufficiently small scores assigned to it relative to the non-rigid part of the scene, this forces the background to remain static for all time steps and views.

## 6.4 LIMITATIONS

For simplicity, the discrete integration along the bent ray uses the interval lengths given by the straight ray. As NR-NeRF builds on NeRF, it is similarly slow. All else being equal, ray bending increases runtime by about 20%. However, due to fewer rays and points sampled, training takes 6 hours. It is thus feasible to train multiple NR-NeRFs (to find appropriate loss weights) in a time similar to other NeRF-based methods (Mildenhall et al., 2020). Neural Sparse Voxel Fields (Liu et al., 2020) are a promising direction to speed up NeRF-like methods. The background needs to be fairly close to the foreground, an issue NR-NeRF "inherits" from NeRF and which could be addressed similarly to NeRF++ (Zhang et al., 2020). Since the proposed method uses a deformation model that does not go from the canonical space to the deformed space, it is not possible to obtain exact correspondences between images captured at different time steps, but instead a nearest neighbor approximation is necessary. NR-NeRF does not account for appearance changes that are due to deformation or lighting changes. For example, temporally changing shadowing in the input images is an issue, as Fig. 6.12 demonstrates. Foreground removal can fail if a part of the foreground is entirely static (*e.g.,* the foot in Fig. 6.11). Rendering parts of the scene barely or not at all observed in the training data would not lead to realistic results. Motion blur in input images is not modeled and would lead to artifacts. The background needs to be static and dominant enough for structure from motion (Schönberger and Frahm, 2016) to estimate correct extrinsics. Since the problem setting is severely under-constrained, NR-NeRF employs strong regularization, which leads to a trade-off between sharpness and stability on some scenes.

## 6.5 CONCLUSION

This chapter presents a method for free viewpoint rendering of a dynamic scene using just a monocular video as input. Several high-quality reconstruction and novel view synthesis results of general dynamic scenes, as well as unsupervised, yet plausible rigidity scores and dense 3D correspondences demonstrate the capabilities of the proposed method. The results suggest that space warping in the form of ray bending is a

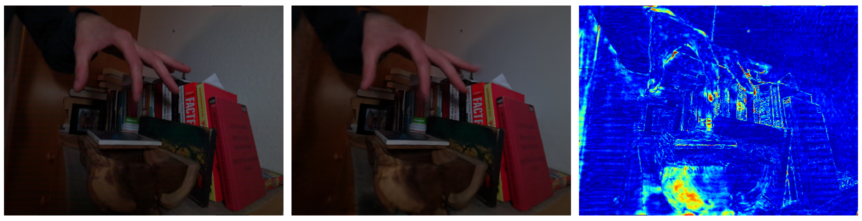


Figure 6.12: The input (left) is reconstructed by NR-NeRF (middle). The bottom of the image exhibits local shadowing absent at other time steps, which leads to a high reconstruction error (right).

promising deformation model for volumetric representations like NeRF. Furthermore, experiments demonstrate that background instability, a problem also noted by concurrent work (Park et al., 2021a), can be mitigated in an unsupervised fashion by learning a rigidity mask. The extensions to multi-view data and view dependence invite future work on more constrained settings for higher quality. Although rather rudimentary, this chapter also shows that NR-NeRF enables several scene-editing tasks, and may enable further work in the direction of editable neural representations.

The experimental results show that NR-NeRF is a promising step for reconstructing and encoding dynamic scenes in a neural manner. Still, several limitations remain, as discussed earlier. The next chapter addresses one particular shortcoming of NR-NeRF: It cannot reconstruct large motion.

# 7 SCENERFLOW: TIME-CONSISTENT RECONSTRUCTION OF GENERAL DYNAMIC SCENES

The fourth and last method proposed in this thesis builds on the previous chapter and extends it to also handle large motion. Existing methods for the 4D reconstruction of general, non-rigidly deforming objects focus on novel-view synthesis and neglect correspondences. However, time consistency enables advanced downstream tasks like 3D editing, motion analysis, or virtual-asset creation. This chapter proposes *SceNeRFlow* to reconstruct a general, non-rigid scene in a time-consistent manner (also published as Tretschk et al. (2024)). This dynamic-NeRF method takes multi-view RGB videos and background images from static cameras with known camera parameters as input. It then reconstructs the deformations of an estimated canonical model of the geometry and appearance in an online fashion. Since this canonical model is time-invariant, correspondences are preserved even for long-term, long-range motions. SceNeRFlow employs neural scene representations for its components. Like prior dynamic-NeRF methods, including the previous chapter, it uses a backwards deformation model. However, non-trivial adaptations of this model are necessary to handle larger motions: The deformations are decomposed into a strongly regularized coarse component and a weakly regularized fine component, where the coarse component also extends the deformation field into the space surrounding the object, which enables tracking over time. Experiments show that, unlike prior work that only handles small motion (at the order of at most dozens of centimeters), the proposed method enables the reconstruction of studio-scale motions (at the order of multiple meters).

## 7.1 INTRODUCTION

One criterion for the proper reconstruction of a deforming scene is time consistency, *i.e.* correspondences across time. Reconstructing general dynamic objects from RGB input in a time-consistent manner is a scarcely explored (Cagniart et al., 2010; Mustafa et al., 2016), but challenging and highly relevant research direction. Establishing correspondences is equivalent to factorizing the reconstruction into time-varying deformations and a time-invariant geometry model. High-level tasks that go beyond novel-view synthesis benefit immensely from such a deeper analysis of

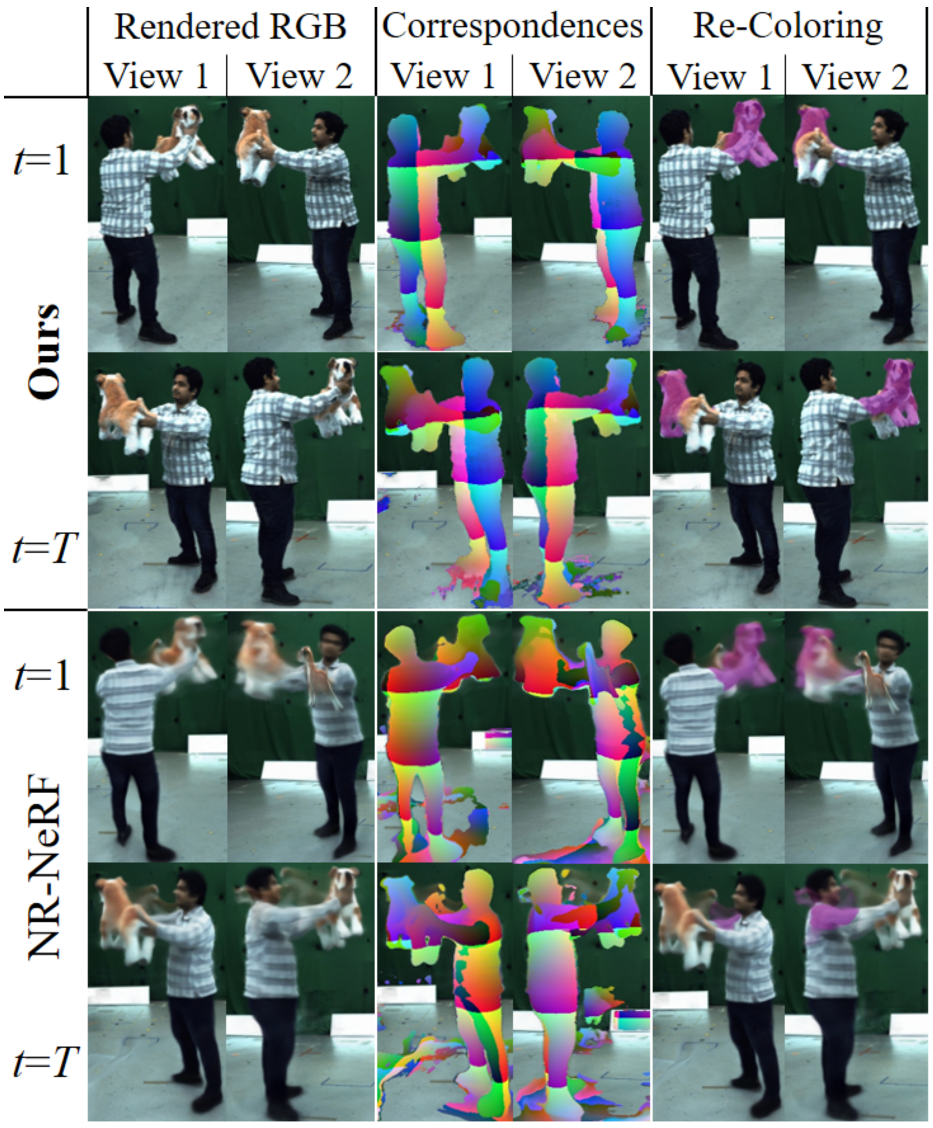


Figure 7.1: **SceNeRFlow.** The proposed NeRF-based method reconstructs a general non-rigid scene from multi-view videos *with time consistency*. Here, a person with a plush dog rotates 180° from time $t$=1 until $t$=$T$. Novel view 1 at $t$=1 is consistent with novel view 2 at $t$=$T$ (placed opposite of novel view 1) for SceNeRFlow, but not for NR-NeRF. This enables, *e.g.*, time- and view-consistent re-coloring. The correspondences are visualized by coloring them according to 3D positions in static canonical space, which differs between both methods.

the scene, namely a reconstruction with long-range, long-term dense 3D correspondences. For example, having access to such a virtual dynamic 3D object is a crucial step towards sophisticated 3D editing, estimating a motion model for virtual-asset creation (Chen et al., 2022), or 4D scene understanding such as motion analysis. Almost all existing work on reconstructing general dynamic objects either only obtains time consistency for small motion or relaxes it to very small time windows. This chapter explores the setting of time consistency for large motions.

Prior work (Du et al., 2021; Gao et al., 2021; Li et al., 2021b; Park et al., 2021a,b; Pumarola et al., 2021; Tretschk et al., 2021; Xian et al., 2021) for non-rigid 3D reconstruction based on NeRF (Mildenhall et al., 2020) focuses on novel-view synthesis and does not aim for long-range correspondences. Almost all dynamic-NeRF papers (except for Liu et al. (2022), Pumarola et al. (2021), and Tretschk et al. (2021), which only handle small motions) weaken the long-term consistency of the reconstruction by design by having a time-varying geometry and/or appearance model. Thus, only rather simple tasks like replaying the input scene under novel views are straightforward. Furthermore, as experiments will show, this time dependency improves the novel-view rendering quality but loosens long-term correspondences. In contrast, this chapter explores the other extreme, where *only* deformations are time-variant.

Classical, non-NeRF-based prior work on 4D reconstruction either only handles small motion (Bartoli et al., 2015; Kairanda et al., 2022; Kumar et al., 2018; Russell et al., 2012; Salzmann et al., 2007) or is not time-consistent (Bansal et al., 2020; Dou et al., 2016; Newcombe et al., 2015; Yoon et al., 2020). Similarly, scene-flow methods (Zhai et al., 2021) focus on the deformations and not a full reconstruction, and exhibit drift in their correspondences in the long run (Hung et al., 2013), similar to optical flow.

This chapter proposes *SceNeRFlow* (Scene Flow + NeRF) to tackle time-consistent reconstruction of a general, non-rigidly deforming scene; see Fig. 7.1. It is NeRF-based, trained per scene, and by design estimates each timestamp's deformation of a time-invariant canonical model, thereby counteracting correspondence drift. As is common for dynamic NeRFs (Park et al., 2021a; Pumarola et al., 2021; Tretschk et al., 2021), it employs a backward deformation model. Standard reconstruction techniques like online optimization, coarse-and-fine deformation decomposition, and rigidity regularization turn out to be insufficient for large motion. While category-specific priors like an articulated human skeleton would help by normalizing out large motion, this thesis tackles the general setting, where such prior knowledge is not available. Instead, carefully "extend-

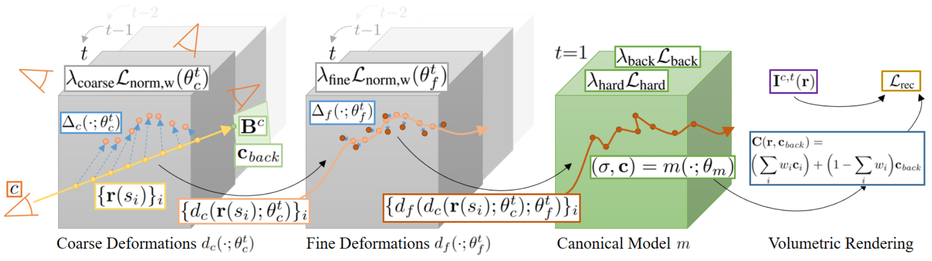

$t-2$
$t-1$
$t$
$\lambda_{\text{coarse}}\mathcal{L}_{\text{norm,w}}(\theta_c^t)$
$\Delta_c(\cdot;\theta_c^t)$
$\mathbf{B}^c$
$\mathbf{c}_{back}$
$c$
$\{\mathbf{r}(s_i)\}_i$
$\{d_c(\mathbf{r}(s_i);\theta_c^t)\}_i$
$\lambda_{\text{fine}}\mathcal{L}_{\text{norm,w}}(\theta_f^t)$
$\Delta_f(\cdot;\theta_f^t)$
$\{d_f(d_c(\mathbf{r}(s_i);\theta_c^t);\theta_f^t)\}_i$
$t{=}1$
$\lambda_{\text{back}}\mathcal{L}_{\text{back}}$
$\lambda_{\text{hard}}\mathcal{L}_{\text{hard}}$
$(\sigma,\mathbf{c}) = m(\cdot;\theta_m)$
$\mathbf{I}^{c,t}(\mathbf{r})$
$\mathcal{L}_{\text{rec}}$
$\mathbf{C}(\mathbf{r},\mathbf{c}_{back}) = \left(\sum_i w_i\mathbf{c}_i\right) + \left(1-\sum_i w_i\right)\mathbf{c}_{back}$
Coarse Deformations $d_c(\cdot;\theta_c^t)$
Fine Deformations $d_f(\cdot;\theta_f^t)$
Canonical Model $m$
Volumetric Rendering

Figure 7.2: **SceNeRFlow Overview.** *Input:* SceNeRFlow takes as input a multi-view RGB video $\{\mathbf{I}^{c,t}\}_{c,t}$ of a general dynamic scene from $C$ cameras with background images $\{\mathbf{B}^c\}_c$ and known camera parameters. *Output:* From this, it builds a NeRF-based (Mildenhall et al., 2020), time-consistent 4D reconstruction. *Model:* To render a ray $\mathbf{r}$ from camera $c$, discrete points $\{\mathbf{r}(s_i)\}_i$ along the straight ray (with background color $\mathbf{c}_{\text{back}}$ at the end) are first sampled. Then, the ray is coarsely bent with the coarse deformations $d_c$, and then finely bent with the fine deformations $d_f$. The *canonical model* $m$ is then queried at the resulting positions. $m$ represents geometry (opacity $\sigma$) and appearance (color $\mathbf{c}$) volumetrically for any 3D point, in a time-invariant manner. Finally, NeRF-style volumetric rendering yields the ray's color $\mathbf{C}$. All three components are parametrized with HashMLPs (Müller et al., 2022). *Training:* $m$ is constructed from time $t{=}1$ using a reconstruction loss ($\mathcal{L}_{\text{rec}}$ w.r.t. the ground-truth color $\mathbf{I}^{c,t}(\mathbf{r})$) and geometric regularizers ($\mathcal{L}_{\text{back}}$ and $\mathcal{L}_{\text{hard}}$). Then, $m$'s parameters $\theta_m$ are kept fixed for future timestamps, which leads to time consistency. For $t{>}1$, the deformations are split into a strongly regularized coarse component $\Delta_c$ and a weakly regularized fine component $\Delta_f$. An online optimization performs frame-wise tracking to handle large motion and the deformation parameters $\theta_c^t$ and $\theta_f^t$ at time $t$ are thus initialized with $\theta_c^{t-1}$ and $\theta_f^{t-1}$. To handle a peculiarity of backward deformation models in this tracking setting, $\Delta_c$ is regularized to "extend the deformation field" ($\mathcal{L}_{\text{norm,w}}$).

ing the deformation field" is the minimally necessary change to make backwards deformation modeling work for large non-rigid motion.

Since this is the first NeRF-based work with time consistency for large motion, it focuses on the core problem of long-term correspondences. It therefore uses multi-view input to avoid the need for correctly modeling the deformations of occluded geometry that arises in the monocular setting. Importantly, the recorded scenes exhibit significantly larger motion than any prior time-consistent general reconstruction method could handle. Experiments show how SceNeRFlow enables the time-consistent reconstruction even of studio-scale motion without any category-specific priorsj, *e.g.* without a human skeleton.

## 7.2 METHOD

SceNeRFlow takes as input multi-view RGB images of size $h \times w$ over $T$ consecutive timestamps from $C$ static cameras with known extrinsics and intrinsics, *i.e.* $\{\mathbf{I}^{c,t} \in [0,1]^{h\times w\times 3}\}_{c=1,t=1}^{C,T}$, and associated background images $\{\mathbf{B}^c\}_c$. In a time-consistent manner, it then reconstructs the general dynamic scene as a time-invariant, NeRF-style canonical model of the geometry and appearance, with time-dependent deformations. The optimization proceeds in an online manner: After building a canonical model from the first timestamp (Sec. 7.2.1), it tracks it frame-by-frame through the temporal input sequence (Sec. 7.2.2). Fig. 7.2 provides an overview of the pipeline.

### 7.2.1 *Constructing the Canonical Model*

**Canonical Model.** The canonical model $m$ encodes the geometry and appearance in a time-independent manner. Specifically, a *HashMLP* (Müller et al., 2022) represents opacity $\sigma$ and RGB color $\mathbf{c} \in [0,1]^3$ for any 3D point $\mathbf{x}$: $(\sigma, \mathbf{c}) = m(\mathbf{x})$. A HashMLP combines a *hash grid* $v$ with a subsequent shallow MLP $M$: $m(\mathbf{x}) = M(v(\mathbf{x}))$, which is significantly faster than NeRF's (Mildenhall et al., 2020) pure MLP. A hash grid consists of about a dozen voxel grids of increasing resolution, each containing learnable features. To evaluate, each grid is queried via trilinear interpolation and the resulting features from all grids are concatenated. Crucially, each voxel grid is implemented via hashed indexing into an array of feature vectors rather than as a dense voxel grid.

**Rendering.** Since SceNeRFlow takes 2D images as input, it is necessary to render $m$ into 2D. To this end, SceNeRFlow follows the quadrature discretization of volumetric rendering from NeRF (Mildenhall et al., 2020) for a ray $\mathbf{r}(s) = \mathbf{o} + s\mathbf{d}$ with origin $\mathbf{o} \in \mathbb{R}^3$ and direction $\mathbf{d} \in \mathbb{R}^3$:

$$\mathbf{C}(\mathbf{r}) = \sum_{i=1}^{S} w_i \mathbf{c}_i, \text{ with } w_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)(1 - \exp(-\sigma_i \delta_i)), \tag{7.1}$$

where $i$ indexes $S$ discrete samples $\{\mathbf{r}(s_i)\}_i$ along the ray; $(\sigma_i, \mathbf{c}_i) = m(\mathbf{r}(s_i))$ are the opacity and color of the $i$-th sample, respectively; and $\delta_i = s_{i+1} - s_i$. Like NeRF (Mildenhall et al., 2020), SceNeRFlow uses stratified sampling of $S$ evenly sized intervals between the near plane and far plane, yielding $\{s_i\}_i$.

When a background color $\mathbf{c}_{back}$ is provided (*e.g.* from $\mathbf{B}^c$), it can be composited at the end of the ray:

$$\mathbf{C}(\mathbf{r}, \mathbf{c}_{back}) = \mathbf{C}(\mathbf{r}) + \left(1 - \sum_i w_i\right) \mathbf{c}_{back}. \tag{7.2}$$

**Losses.** To obtain the canonical model, SceNeRFlow iteratively optimizes it on images $\{\mathbf{I}^{c,1}\}_c$ from $t$=1 for 20$k$ iterations. Each iteration uses a batch of $R$ rays with an $\ell_1$ reconstruction loss w.r.t. the ground-truth color $\mathbf{I}^{c,1}(\mathbf{r}_r)$:

$$\mathcal{L}_{\text{rec}} = \frac{1}{R}\sum_{r=1}^{R} \|\mathbf{C}(\mathbf{r}_r, \mathbf{c}_{back,r}) - \mathbf{I}^{c,1}(\mathbf{r}_r)\|_1. \tag{7.3}$$

However, using only $\mathcal{L}_{\text{rec}}$ leads to a canonical model with floating geometry artifacts. They are removed by steering the model to commit to foreground *or* background by encouraging $\sum_i w_i$ to be 1 or 0 via the beta distribution (Lombardi et al., 2019):

$$\mathcal{L}_{\text{back}} = \frac{1}{R}\sum_r \log\Big(\sum_i w_{r,i}\Big) + \log\Big(1 - \sum_i w_{r,i}\Big), \tag{7.4}$$

where $w_{r,i}$ is $w_i$ of ray $r$. Furthermore, transparent geometry is discouraged by encouraging each $w_i$ to be 0 or 1 via a mixture of two Laplacian distributions (Rebain et al., 2022):

$$\mathcal{L}_{\text{hard}} = -\frac{1}{RS}\sum_r\sum_i \log\Big(e^{-w_{r,i}} + e^{-(1-w_{r,i})}\Big). \tag{7.5}$$

Note that these losses do not use foreground masks; the canonical model needs to learn on its own whether it can be transparent and use the background image $\mathbf{B}$ or not.

The total loss for the canonical model is thus:

$$\mathcal{L}_{\text{canon}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{back}}\mathcal{L}_{\text{back}} + \lambda_{\text{hard}}\mathcal{L}_{\text{hard}}, \tag{7.6}$$

where $\lambda_{\text{back}}, \lambda_{\text{hard}} \in \mathbb{R}$ are loss weights. Since the canonical model is meant to be time-consistent, its parameters are kept fixed after constructing it from the first timestamp.

### 7.2.2 *Optimizing per Timestamp*

Given the canonical model from $t$=1, SceNeRFlow next reconstructs the deformations for the remaining timestamps $t$>1.

**Space Warping.** SceNeRFlow models the deformations at time $t$ like prior dynamic-NeRF work: It uses backwards space warping $d_c(\mathbf{x}; \theta_c^t) = \mathbf{x} + \Delta_c(\mathbf{x}; \theta_c^t) = \mathbf{x}'$, where $\mathbf{x}$ is a 3D point in deformed world space, $\Delta_c$ outputs a coarse offset (fine offsets $\Delta_f$ will be introduced later), $\mathbf{x}'$ is in undeformed canonical space, and $\theta_c^t$ are the time-dependent parameters of $d_c/\Delta_c$. $\Delta_c$ is parametrized by a HashMLP, *i.e.* there is one HashMLP per timestamp for coarse deformations. To query the canonical

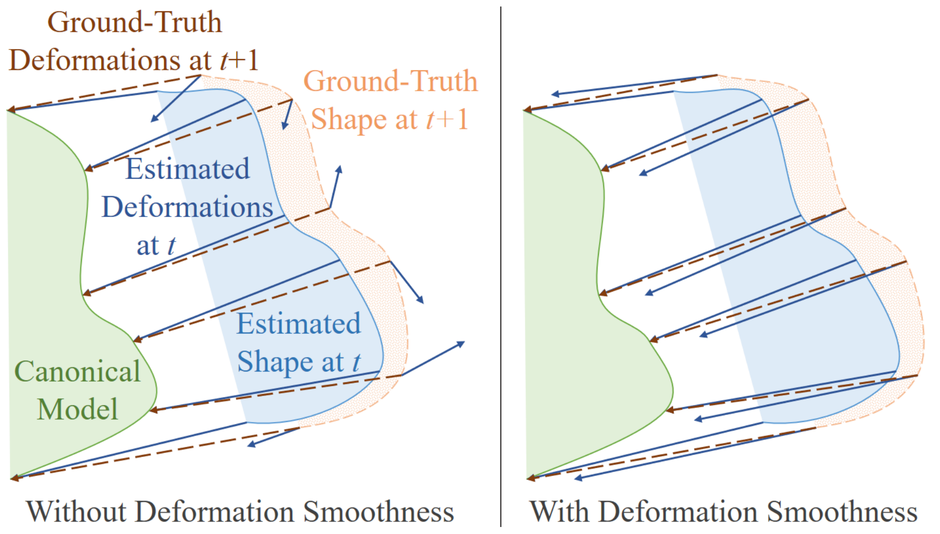

Ground-Truth Deformations at $t$+1
Ground-Truth Shape at $t$+1
Estimated Deformations at $t$
Estimated Shape at $t$
Canonical Model
Without Deformation Smoothness
With Deformation Smoothness

Figure 7.3: **Extending the Deformation Field for Tracking.** Without smoothness, the estimated backwards deformations at time $t$ give a bad initialization for $t$+1. Smoothness initializes the deformations closer to the ground truth.

model from world space at $t>1$, SceNeRFlow first undoes the deformation: $(\sigma(\mathbf{x}, t), \mathbf{c}(\mathbf{x}, t)) = m(d_c(\mathbf{x}; \theta_c^t))$. When rendering, this leads to view-consistent ray bending: $\{\mathbf{r}(s_i)\}_i$ in world space becomes $\{d_c(\mathbf{r}(s_i); \theta_c^t)\}_i$ in canonical space. Then, Eq. 7.1 or Eq. 7.2 can be applied to the bent ray to render, which in turn allows to use $\mathcal{L}_{\text{rec}}$ to optimize for the deformation parameters at time $t$ using $\{\mathbf{I}^{c,t}\}_c$.

**Frame-Wise Tracking.** However, naïvely reconstructing the entire scene by optimizing all timestamps at once does not converge to a recognizable canonical model when the scene contains large motion. Furthermore, keeping all $\{\mathbf{I}^{c,t}\}_{c,t}$ in memory at once is expensive in practice. This chapter thus proposes online, timestamp-by-timestamp tracking. Since $t$=1 has, by construction, zero offsets, the last layer of $\Delta_c(\cdot; \theta_c^1)$ is set to zeros (Tretschk et al., 2021) and not optimized at $t$=1. After reconstructing time $t$, $\theta_c^t$ is fixed and SceNeRFlow proceeds with $t$+1, where $\theta_c^{t+1}$ is initialized with $\theta_c^t$.

**Extending the Deformation Field.** Unfortunately, naïvely employing a HashMLP for $d_c$ fails to reconstruct the scene for any deformed state that significantly differs from the canonical model. This is because, at time $t$+1, the dynamic object resides in a slightly different place in world space, part of which was empty space at time $t$ and is hence not well initialized by $d_c(\cdot; \theta_c^t)$. This becomes especially prevalent when the object has undergone large motion. This chapter proposes to mitigate this failure of the backward model by extending the deformations into the area surrounding the dynamic object at time $t$, as Fig. 7.3 illustrates.

To this end, two means help: (1) Reducing the resolution of the coarsest grid of the deformation hash grid to $32^3$ stabilizes tracking. Presumably, when the object enters a previously empty voxel in a fine grid, the uninitialized latent codes of this voxel are too quickly too influential in the trilinear interpolation of the latent codes. These uninitialized latent codes thus do not obtain the right values before being relevant but rather negatively impact the deformations, leading to artifacts and a lack of large-scale smoothness. (2) However, this structural change gives an unpredictable, badly-controlled smoothness, similar to an MLP. SceNeRFlow encourages well-behavedness via a smoothness loss, which uses the inherent smoothness of the coarse grid to propagate its influence. **Smoothness Loss.** To stabilize the tracking, this chapter proposes to impose a smoothness loss on the deformations. Because the deformation model is continuous and fully differentiable, it is not necessary to discretize the loss. Instead, the local behavior can be directly influenced via the (spatial) Jacobian $\mathbf{J} \in \mathbb{R}^{3\times3}$ of the deformations: $\mathbf{J} = \mathbf{J}_\mathbf{x} = \frac{\partial d_c(\mathbf{x};\theta_t)}{\partial \mathbf{x}}$. As is common for general reconstruction methods (Tretschk et al., 2023), SceNeRFlow assumes the object to deform locally in an as-rigid-as-possible manner (Sorkine and Alexa, 2007). Specifically, SceNeRFlow takes inspiration from Nerfies's elastic loss on $\mathbf{J}$ (Park et al., 2021a). However, their loss is computationally expensive because it needs to compute all rows of $\mathbf{J}_{\mathbf{r}_r(s_i)}$ for each sample $\mathbf{r}_r(s_i)$ (which takes three backward passes during the forward pass of the loss computation) and then performs an SVD of $\mathbf{J}_{\mathbf{r}_r(s_i)}$. Denoting the identity matrix by $\mathbf{I}$, the SVD can be avoided (allowing for trivial computation of gradients) by relaxing the constraint from the rotation group $SO(3) = \{\mathbf{A} \in \mathbb{R}^{3\times3} | \mathbf{A}^\top \mathbf{A}{=}\mathbf{I}, \det \mathbf{A}{=}1\}$ to the orthogonal group $O(3) = \{\mathbf{A} \in \mathbb{R}^{3\times3} | \mathbf{A}^\top \mathbf{A}{=}\mathbf{I}\}$, which additionally allows reflections since $\det \mathbf{A}{=}\pm 1$ for $\mathbf{A} \in O(3)$. It is then possible to encourage rigidity via:

$$\mathcal{L}_\text{rigid} = \frac{1}{9RS} \sum_r \sum_i \sum_{j,k} \left| (\mathbf{J}^\top_{\mathbf{r}_r(s_i)} \mathbf{J}_{\mathbf{r}_r(s_i)} - \mathbf{I})_{j,k} \right|, \qquad (7.7)$$

where $\mathbf{A}_{j,k} \in \mathbb{R}$ is the entry of matrix $\mathbf{A}$ at index $(j,k)$.

This prior can be computed even faster by reducing the need for three backward passes to just one, lowering overall training time by a factor of $2{-}3$. To this end, first note that norm preservation is an equivalent definition of $O(3)$:

$$\mathbf{A} \in O(3) \iff \forall \mathbf{e} \in \mathbb{R}^3 : \|\mathbf{A}\mathbf{e}\|_2 = \|\mathbf{e}\|_2. \qquad (7.8)$$

Next, SceNeRFlow exploits the fact that automatic differentiation (Griewank and Walther, 2008; Paszke et al., 2017; Speelpenning, 1980) computes Jacobian-vector products $\mathbf{J}^\top \mathbf{e}$ in a single backward pass, where $\mathbf{e}$ is an arbitrary vector. Since $\mathbf{J}^\top = \mathbf{J}$ for $\mathbf{J} \in O(3)$, the norm of $\mathbf{Je}$ can be computed as $\|\mathbf{Je}\|_2 = \|\mathbf{J}^\top \mathbf{e}\|_2$. SceNeRFlow thus replaces $\mathcal{L}_{\text{rigid}}$ with a norm-preserving loss that only requires one backward pass:

$$\mathcal{L}_{\text{norm}} = \frac{1}{RS} \sum_r \sum_i \mathbb{E}_{\mathbf{e}} \left[ \left| \|\mathbf{J}^\top_{\mathbf{r}_r(s_i)} \mathbf{e}\|_2 - 1 \right| \right], \tag{7.9}$$

where $\mathbf{e}$ is distributed uniformly on the unit sphere and hence $\|\mathbf{e}\|_2 = 1$. Note that it is sufficient to only consider vectors on the unit sphere due to linearity. In practice, this expectation is approximated with one random sample.

**Weighting the Smoothness Loss.** To extend the deformation field as discussed earlier, one could naïvely encourage smoothness uniformly everywhere. However, this restricts empty space too much, leading it to push back against object deformations. SceNeRFlow thus focuses on the object by weighting $\mathcal{L}_{\text{norm}}$ by $\hat{\sigma}_{r,i} = \exp(-\sigma_{r,i}\delta)$, where $\sigma_{r,i}$ is $\sigma_i$ of the $r$-th ray and gradients are not backpropagated into the opacity (Tretschk et al., 2021).

The space *around* the object needs to be regularized. However, this space is hard to locate efficiently and SceNeRFlow approximates it by max-pooling over $\{\hat{\sigma}_{r,i}\}_i$ along ray $r$, with a window size of 1% of the ray length.

This still makes the space around the object too stiff. SceNeRFlow thus weakens the regularization by dividing the resulting weight by $u$ if $\hat{\sigma}_{r,i}$ and the weight after max-pooling differ by at least a factor of $u$ (empirically set to $u$=10).

Finally, regularizing very small offsets, which are widespread during early timestamps, tends to collapse the network. Hence, only deformations that are not very small are regularized. This weighted loss is denoted as $\mathcal{L}_{\text{norm,w}}(\theta^t_c)$. Sec. D.4 contains a full mathematical description.

**Fine Deformations.** In addition to enabling tracking by extending the deformation field, SceNeRFlow also uses the smoothness loss to stabilize the surface by strongly regularizing its deformations $\Delta_c$. However, this leads to a loss of detail because $\Delta_c$ can now only represent *coarse* deformations. To counteract this, the proposed method adds *fine* deformations $\Delta_f(\cdot;\theta^t_f)$ on top, after normalizing out (*i.e.*, applying) the coarse deformations: $\mathbf{x}'' = d_f(\mathbf{x}';\theta^t_f) = \mathbf{x}' + \Delta_f(\mathbf{x}';\theta^t_f)$, where $\mathbf{x}' = d_c(\mathbf{x};\theta^t_c)$ for any 3D point $\mathbf{x}$ in world space and we query $m$ at $\mathbf{x}''$. Crucially, details can be modeled by using a much weaker weight $\lambda_{\text{fine}}$ for $\mathcal{L}_{\text{norm,w}}(\theta^t_f)$, where the Jacobian is w.r.t. $\mathbf{x}'$. $\Delta_f$ is parametrized with a HashMLP.

**Frame-Wise Tracking Revisited.** SceNeRFlow applies tracking to the coarse and fine deformations as follows: At $t$=1, the last layers of $\Delta_c$ and $\Delta_f$ are initialized to zeros and not optimized. At any $t$>1, $\theta^t_c$ is initialized with the final $\theta_c^{t-1}$, and SceNeRFlow optimizes for $\Delta_c(\cdot;\theta_c^t)$ for $5k$ iterations while setting $\Delta_f = \mathbf{0}$. Then, $\theta_c^t$ is fixed, $\theta_f^t$ is initialized with the final $\theta_f^{t-1}$, and $\Delta_f(\cdot;\theta_f^t)$ are optimized for $5k$ iterations.

With $\lambda_{\text{coarse}}, \lambda_{\text{fine}} \in \mathbb{R}$, the total loss for any $t>1$ is:

$$\mathcal{L}_{\text{time}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{coarse}}\mathcal{L}_{\text{norm,w}}(\theta_c^t) + \lambda_{\text{fine}}\mathcal{L}_{\text{norm,w}}(\theta_f^t). \tag{7.10}$$

### 7.2.3 *Implementation Details*

This section describes how to speed up the proposed method and how to correct vignetting effects, and provides optimization details. Fig. 7.4 visualizes some of these methods.

**Foreground-Focused Batches.** To speed up training, the batches focus on the foreground. Background subtraction w.r.t. $\mathbf{B}^c$ yields a rough foreground mask $\mathbf{F}^{c,t}$ for each $\mathbf{I}^{c,t}$. Then, 80% of the rays in the batch are picked from the foreground and the rest from the background.

**Pruning.** To speed up rendering, any sample $\mathbf{r}(s_i)$ in world space that does not contain any foreground is pruned. To determine this for time $t$, a binary voxel grid $g_t$ overlaid over the scene is queried. $g_t$ is 1 for voxels that are potentially in the foreground and 0 otherwise. $g_t$ is filled via space carving (Kutulakos and Seitz, 2000) with $\{\mathbf{F}^{c,t}\}_c$. To be conservative and to let the deformation field extend into the surrounding area, both the foreground masks and the foreground in the resulting voxel grid are dilated. To clear out geometry artifacts in empty space, pruning is not used when constructing the canonical model, only when optimizing for $t>1$. Pruning removes $\sim$80% of samples, making SceNeRFlow four times faster.

**Vignetting Correction.** Cameras collect less light around the image border, which leads to darkening in the input images. The proposed method models this vignetting with a radial model (Bal and Palus, 2023; Stumpfel et al., 2006):

$$\mathbf{C}_{vig}(\mathbf{r}) = \mathbf{C}(\mathbf{r})(1 + k_1 p(\mathbf{r}) + k_2 p(\mathbf{r})^2 + k_3 p(\mathbf{r})^3), \tag{7.11}$$

where $p(\mathbf{r}) \in \mathbb{R}$ is the squared distance to the camera center in pixel space (the distance is divided by $w$ for resolution invariance), and $k_1, k_2, k_3 \in \mathbb{R}$ are correction parameters. These parameters are initialized to zeros. They are optimized for when constructing the canonical model, and kept fixed for $t>1$. The same set of parameters is used across all cameras and timestamps. $\mathbf{C}_{vig}$ is used in place of $\mathbf{C}$ in Eq. (7.2). The quality improves since the canonical model no longer needs to use artifacts to account for vignetting effects.

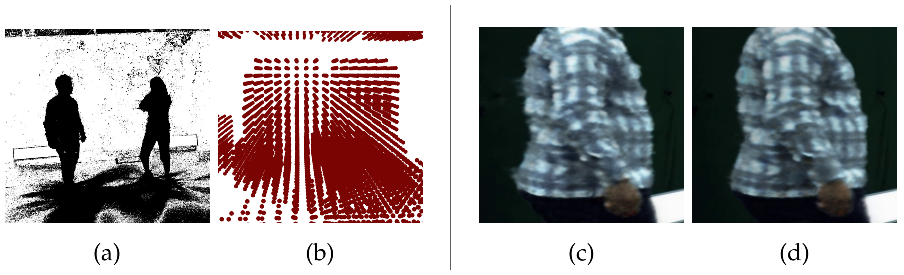


Figure 7.4: **Implementation.** (a) Foreground $\mathbf{F}^{c,t}$. (b) Pruning grid for (a). (c) Without and (d) with vignetting correction.

**Optimizer.** The optimization uses AdamW (Loshchilov and Hutter, 2019) with weight decay of 0.01 to stabilize the training. Auto-decoded parameters like hash grids get sparsely non-zero gradients in any given training iteration, which degrades the momentum accumulation. SceNeRFlow thus uses a modified version of AdamW: Rather than treating all parameters $\{\theta_i \in \mathbb{R}\}_i$ in a tensor the same, it separately keeps track of AdamW's parameters $\theta_i^{opt}$ (*e.g.* the number of iterations) for each individual parameter $\theta_i$. In any given iteration, only $\theta_i$ whose derivative is non-zero have their AdamW parameters $\theta_i^{opt}$ updated and AdamW applied. Sec. D.4 describes the learning rates in detail.

**Hyperparameters.** All scenes use the same settings: A batch size of $R$=1024, $S$=3072 samples per ray, and loss weights $\lambda_{\text{back}}$=0.001, $\lambda_{\text{hard}}$=1, $\lambda_{\text{coarse}}$=1000, and $\lambda_{\text{fine}}$=30. The architecture uses LeakyReLUs (Maas et al., 2013) for the deformations and ReLUs for the canonical model.

**Code.** SceNeRFlow is implemented in PyTorch (Paszke et al., 2019) using tiny-cuda-nn (Müller, 2021) via its Python wrappers for its fast implementation of hash grids.

## 7.3 RESULTS

This section evaluates the reconstruction quality and time consistency of the proposed method, performs ablations, shows simple scene editing, and discusses limitations and future work.

**Prior Work.** The experiments compare SceNeRFlow mainly to its closest work, NR-NeRF, which is the main prior time-consistent dynamic-NeRF method. Comparisons against PREF (Song et al., 2022), a NeRF-based method to estimate correspondences, allow to evaluate the time consistency. Further experiments compare with the time-consistent D-NeRF Pumarola et al., 2021 and DeVRF Liu et al., 2022.

**Variants.** The reconstruction quality is evaluated by using novel-view synthesis *as a proxy*. Unlike most works, this chapter targets *time-consistent*

| Name | Description | Frames |
|---|---|---|
| Seq. 1 | two people playing with a plush dog | 125 |
| Seq. 2 | two people holdings hands | 125 |
| Seq. 3 | one person rotating, one person walking | 300 |
| Seq. 4 | a person dancing while wearing a dark dress | 100 |
| Seq. 5 | a person walking like a zombie while wearing a light dress | 125 |
| Seq. 6 | a person playing with a plush dog | 125 |
| Seq. 7 | a person standing with a brown dress | 125 |
| Seq. 8 | a person doing squads (from NR-NeRF) | 5 |

Table 7.1: **Dataset Description.**

reconstruction and not novel-view synthesis, which leads to a trade-off with novel-view quality.

Most dynamic-NeRF papers condition the canonical model on time. Therefore, two variants of SceNeRFlow are additionally evaluated that do the same and, hence, synthesize better novel views at the cost of correspondences. The first variant, *SNF-A*, only makes the appearance time-varying, while the second variant, *SNF-AG*, makes the appearance and canonical geometry time-varying. Sec. D.4 has further details.

This chapter does not aim for these variants to be competitive with state-of-the-art 4D reconstruction methods that neglect correspondences and focus solely on novel-view synthesis. Their design is not tailored to this setting.

**Data.** For ease of recording, SceNeRFlow is evaluated on real-world scenes of one or two people. Crucially, the recordings also contain a plush dog and loose clothing. Furthermore, the proposed method reconstructs scenes with multiple people *without any knowledge that there are multiple entities in the scene nor that they are human*. These aspects demonstrate its generality. The scenes use a total of $C$=117 static cameras in a studio, for a total of six scenes of 4−5sec and one of 12sec at 25fps. For the test sets, the same two cameras are used for all scenes. Training is done at an input resolution of $h$=1504 and $w$=2056. In addition, the short multi-view scene from NR-NeRF, which contains 16 camera pairs, is used. Tab. 7.1 contains a description and the length in frames of each scene.

**Runtime.** On an Nvidia A100 GPU, the canonical model trains at 8iter/sec, the coarse deformations at 16iter/sec, and the fine deformations at 14iter/sec. For efficiency, all images are rendered at half the input resolution, at 10−15sec/image.

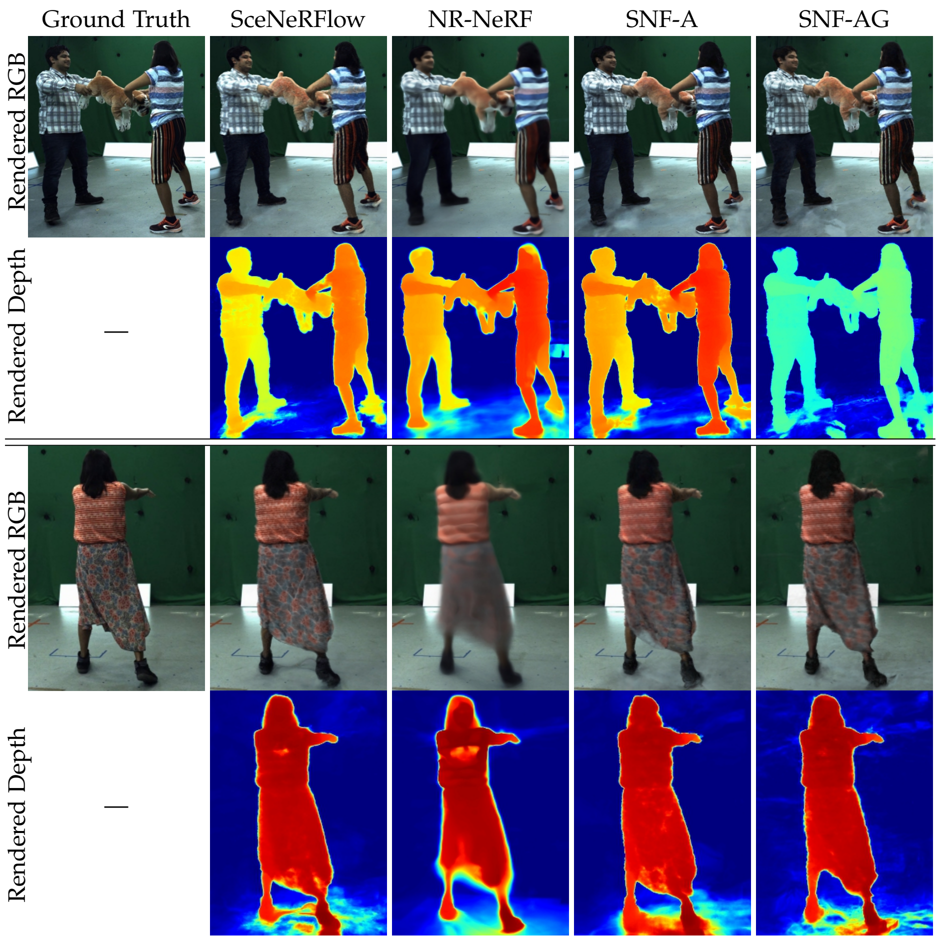


Figure 7.5: **Novel-View Synthesis.** (Top) Seq. 1 at $t=T$. (Bottom) Seq. 5 at $t=38$. The depth color differences between methods are due to normalization.

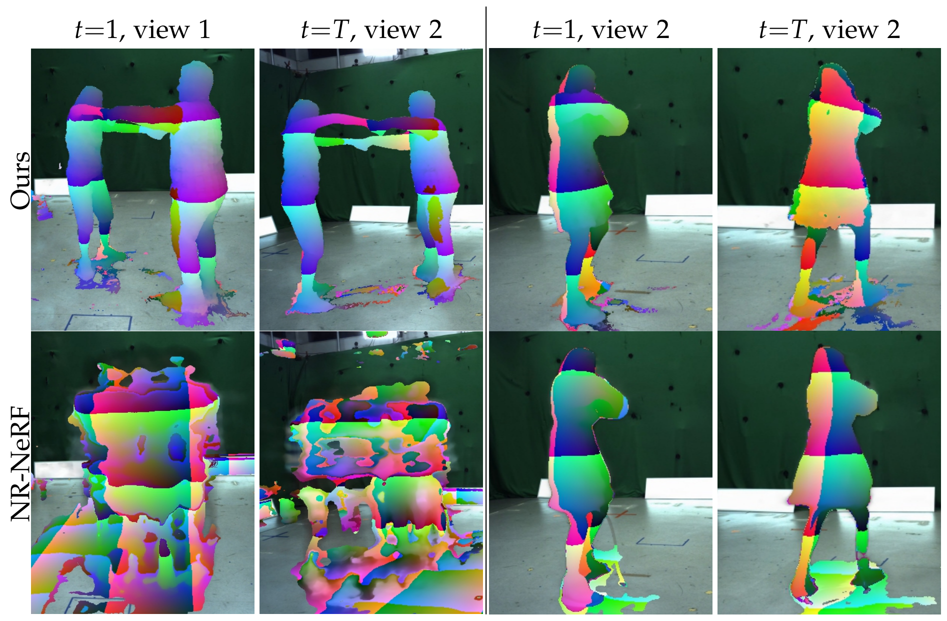


Figure 7.6: **Correspondences.** (Left) Seq. 2, opposite views. NR-NeRF fails. (Right) Seq. 4, same view. Only SceNeRFlow's correspondences follow the 90° left turn; see orange at $t$=1 & $T$.

### 7.3.1 *Qualitative Results*

First the reconstructions are evaluated qualitatively at any one point in time, and then their consistency over time is examined.

#### 7.3.1.1 *Volumetric Scene Reconstruction*

Fig. 7.5 shows novel views of the reconstruction and its geometry via depth maps. While NR-NeRF gives blurry results, the proposed method yields high-quality reconstructions. Although blurrier, SNF-A matches the pattern on the dress better than SceNeRFlow. SNF-AG reconstructs the non-rigid geometry more accurately as it loosens the correspondences.

#### 7.3.1.2 *Time Consistency*

**Correspondence Visualization.** Time consistency enables 3D correspondences over time: Fig. 7.6 shows that the estimated correspondences are temporally stable for SceNeRFlow. However, on scenes with large motion, NR-NeRF fails to converge to a recognizable model since this requires simultaneous correspondence estimation in this highly challenging setting.

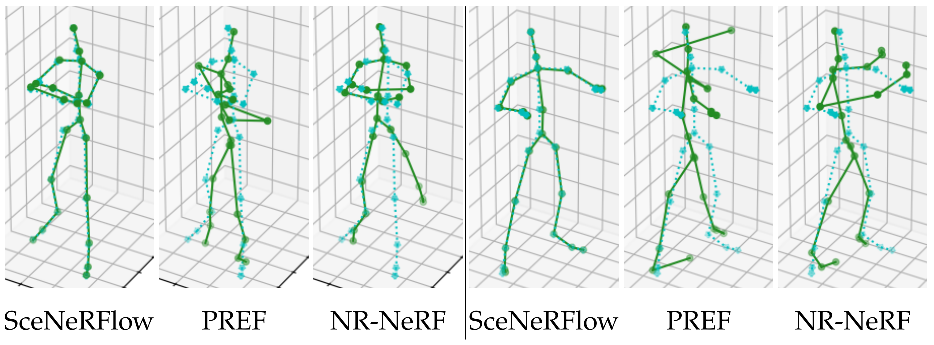


Figure 7.7: **Time Consistency.** (Left) Seq. 4. (Right) Seq. 6. The solid skeleton is the tracking estimate at $t{=}T$. The dotted skeleton is the pseudo-ground truth at $t{=}T$.

**Comparisons with Prior Work.** To compare with prior work, further evaluation follows PREF (Song et al., 2022), which estimates world-space 3D human joint positions over time: $\{\tilde{\mathbf{p}}_j^t \in \mathbb{R}^3\}_{t,j}$, where $j$ indexes the $J{=}23$ joints. Off-the-shelf commercial systems (Captury, 2023) exploit strong human-specific priors to reliably estimate 3D joints, which makes for excellent pseudo-ground-truth long-term 3D correspondences $\{\hat{\mathbf{p}}_j^t\}_{t,j}$. (The joints are used only for evaluation; SceNeRFlow does not use human priors.)

To track the joints with SceNeRFlow, the estimated positions $\{\tilde{\mathbf{p}}_j^1\}_j$ are first initialized with the ground-truth joints $\{\hat{\mathbf{p}}_j^1\}_j$ at $t{=}1$, which are the positions in the canonical model. Then, the backwards deformation field needs to be inverted for $t{>}1$. To this end, for each joint $j$ at time $t$, $d_f(d_c(\cdot;\theta_c^t);\theta_f^t)$ is evaluated on a voxel grid (with a resolution of $128^3$ and a side length of 40cm) centered on the estimated position at $t{-}1$, $\tilde{\mathbf{p}}_j^{t-1}$. Then, the voxel center that lands closest to the ground-truth joint $\hat{\mathbf{p}}_j^1$ in the canonical model is used as the estimated world-space joint position $\tilde{\mathbf{p}}_j^t$. Sec. D.5 contains tracking details for NR-NeRF and PREF.

Fig. 7.7 shows results at $t{=}T$. The proposed method yields stable correspondences, while NR-NeRF does not. PREF's correspondences drift strongly over time due to error accumulation from chaining frame-to-frame correspondences.

**Variants.** Next, the time consistency of the proposed method in contrast with its variants is analyzed. To this end, Fig. 7.8 visualizes the canonical model at two different timestamps. Unlike SceNeRFlow, the canonical model of the variants changes over time, *i.e.*, they lack inherent correspondences. However, these additional degrees of freedom improve the variants' novel-view synthesis, showing a trade-off between time consistency and novel-view synthesis.

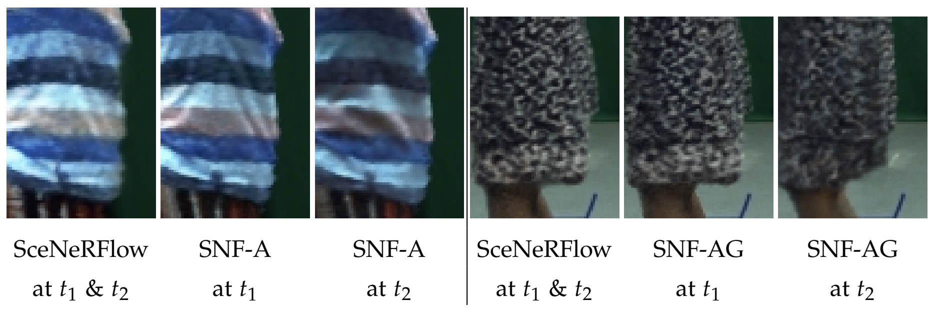


Figure 7.8: **Canonical Model.** SceNeRFlow uses a static canonical model, while those of SNF-A(G) vary over time.

| Sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Mean |
|---|---|---|---|---|---|---|---|---|
| SceNeRFlow | **0.9** | **0.9** | **1.8** | **1.8** | **2.8** | **0.9** | 1.4 | **1.5** |
| PREF | 11.0 | 13.9 | 47.8 | 8.9 | 13.4 | 12.1 | 3.9 | 15.9 |
| NR-NeRF | 2.2 | 46.4 | 110.9 | 6.1 | 8.5 | 14.8 | **1.2** | 27.2 |
| D-NeRF | 40.7 | 46.2 | 110.1 | 12.2 | 81.9 | 35.3 | 70.0 | 56.6 |

Table 7.2: **Time Consistency.** The table shows per-scene and mean MPJPE in cm. Lower is better.

### 7.3.2 *Quantitative Results*

**Time Consistency.** Like PREF (Song et al., 2022), this section measures time consistency via 3D joint tracking. The mean per-joint position error (MPJPE) (Sigal et al., 2010) over all timestamps $t>1$ and the $J$ joints is reported: MPJPE $= \frac{1}{(T-1)J}\sum_{t=2}^{T}\sum_{j=1}^{J}\|\hat{\mathbf{p}}_j^t - \tilde{\mathbf{p}}_j^t\|_2$. Lower is better. For scenes with two people, the average across both is reported. All scenes are evaluated, except for the 5-frame Seq. 8 used in NR-NeRF. Tab. 7.2 contains the results. SceNeRFlow has the lowest error with only marginal drift. PREF's frame-wise approach leads to large drift. D-NeRF cannot handle large motion.

**Reconstruction.** The reconstruction quality is quantified by using novel-view quality as a proxy. Like NeRF (Mildenhall et al., 2020), the latter is measured with PSNR and SSIM (Wang et al., 2004) (for both, higher is better), and the neural perceptual metric LPIPS (Johnson et al., 2016) (lower is better). To focus on the dynamic scene part, (1) these metrics are also provided w.r.t. the static background image, and (2) *masked scores* are reported where the renderings are masked out with foreground masks estimated from the ground truth. The results are in Tab. 7.3. SceNeRFlow outperforms NR-NeRF. The variants have artifacts in empty space and only outperform the proposed method on the masked scores. Sec. D.2

| | | | SceNeRFlow | NR-NeRF | SNF-A | SNF-AG | Background |
|---|---|---|---|---|---|---|---|
| Unmasked | PSNR | ↑ | 30.32 | 28.77 | **30.88** | 30.34 | 21.69 |
| | SSIM | ↑ | 0.939 | 0.922 | **0.940** | 0.929 | 0.920 |
| | LPIPS | ↓ | **0.054** | 0.099 | 0.057 | 0.089 | 0.101 |
| Masked | PSNR | ↑ | 32.38 | 30.80 | **33.31** | 33.22 | — |
| | SSIM | ↑ | 0.976 | 0.965 | **0.978** | 0.977 | — |
| | LPIPS | ↓ | 0.018 | 0.041 | **0.016** | 0.018 | — |

Table 7.3: **Novel-View Synthesis.** Mean PSNR, SSIM, and LPIPS across scenes. SNF-A(G) are variants of SceNeRFlow.

| | | | SceNeRFlow | D-NeRF | DeVRF |
|---|---|---|---|---|---|
| Unmasked | PSNR | ↑ | **29.98** | 24.89 | 25.24 |
| | SSIM | ↑ | **0.939** | 0.913 | 0.905 |
| | LPIPS | ↓ | **0.054** | 0.107 | 0.125 |
| Masked | PSNR | ↑ | **32.03** | 26.13 | 27.66 |
| | SSIM | ↑ | **0.975** | 0.951 | 0.957 |
| | LPIPS | ↓ | **0.017** | 0.060 | 0.041 |

Table 7.4: **Novel-View Synthesis.** Mean PSNR, SSIM, and LPIPS across scenes, except for Seq. 8 (due to memory limitations from the high resolution).

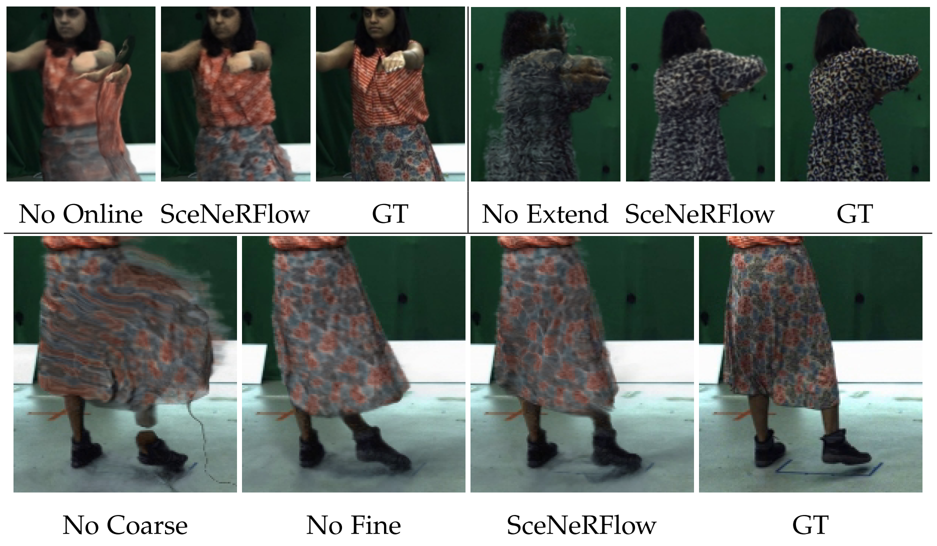


Figure 7.9: **Ablations.** (Top left) Online optimization, (top right) extending the deformations, (bottom) coarse and fine deformations.

has per-scene results. Tab. 7.4 contains further comparisons, without Seq. 8 due to memory constraints. SceNeRFlow outperforms D-NeRF and DeVRF, which both fail on large motion.

## 7.3.3 *Ablations*

This section analyzes the core components of the proposed method by ablating them. Specifically, the relevance of (1) optimizing in an online manner (by optimizing all timestamps at once), (2) "extending the deformation field" (by using a coarse resolution of $512^3$ and $\mathcal{L}_{\text{norm,w}}$ without max-pooling), (3) the coarse deformation model (by setting $\Delta_c=\mathbf{0}$ at training time), and (4) the fine deformation model (by setting $\Delta_f=\mathbf{0}$ at test time) is investigated. Fig. 7.9 contains qualitative results and Tab. 7.5 confirms these quantitatively: (1) Optimizing online simplifies implicit correspondence estimation, avoiding ghosting artifacts; (2) extending the deformation field helps the online optimization by initializing the deformations for the next timestamp well; (3) the coarse deformations stabilize the reconstruction; and (4) the fine deformations add details.

## 7.3.4 *Simple Editing*

The time consistency enables simple editing of the geometry and appearance. As a proof of concept, Fig. 7.10 shows that SceNeRFlow allows to re-color scene parts (*e.g.*, of Seq. 3) or make them transparent in a

| | | | SceNeRFlow | No Online | No Extend | No Coarse | No Fine |
|---|---|---|---|---|---|---|---|
| Unmasked | PSNR | ↑ | 30.32 | 30.36 | 28.53 | **30.40** | 28.95 |
| | SSIM | ↑ | **0.939** | 0.929 | 0.927 | 0.936 | 0.933 |
| | LPIPS | ↓ | **0.054** | 0.074 | 0.067 | 0.056 | 0.056 |
| Masked | PSNR | ↑ | 32.38 | **32.61** | 30.20 | 32.45 | 30.45 |
| | SSIM | ↑ | **0.976** | 0.974 | 0.966 | 0.974 | 0.970 |
| | LPIPS | ↓ | **0.018** | 0.020 | 0.027 | 0.020 | 0.021 |
| | MPJPE | ↓ | **1.5** | 33.4 | 15.6 | 10.4 | **1.5** |

Table 7.5: **Ablations.** Mean PSNR, SSIM, LPIPS, and MPJPE across scenes.

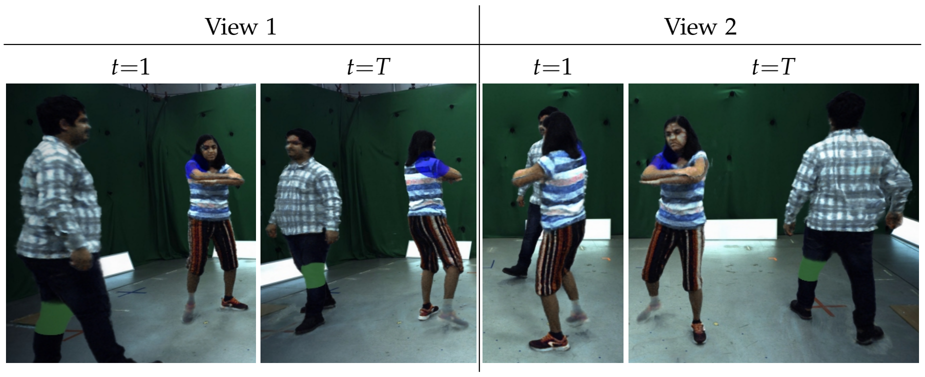


Figure 7.10: **Scene Editing.** (1) Coloring the left knee of person 1 green, (2) blending the right shoulder of person 2 with blue, and (3) making the right foot of person 2 transparent.

straightforward manner—by modifying the static canonical model. The deformations then consistently propagate these changes to all timestamps.

## 7.4 LIMITATIONS

While the proposed method design makes a few assumptions (*e.g.* multi-view input or having access to background images), they are exploited as little as possible, *e.g.* no accurate segmentation masks, depth estimates, or 3D estimates are used. This makes it easier to extend SceNeRFlow to other input settings.

Adjacent work offers promising remedies for the assumptions. Most dynamic-NeRF methods are monocular (but not time-consistent for large motion) and some static-NeRF works (Tewari et al., 2022; Yu et al., 2021) use only few cameras. Adapting these ideas might reduce the number of cameras required by the proposed method. Since the region of interest is the dynamic foreground, the background is excluded in a soft manner. The background could be included via a static NeRF as some existing dynamic-NeRF approaches do. Naïvely modeling time-varying appearance loosens correspondences. Sophisticated regularization from Shape-from-Shading (Barron and Malik, 2014; Zhang and Tsai, 1999; Zhang et al., 2021) might help. To preserve time consistency, SceNeRFlow does not update the canonical model with newly visible geometry from $t{>}1$. However, multi-view input mitigates the effect of occlusions at $t{=}1$.

## 7.5 CONCLUSION

SceNeRFlow demonstrates the great potential of modern neural scene representations for time-consistent reconstruction of general dynamic objects. In particular, this chapter shows how backwards deformation models can be adapted to tracking large motion. As a proof of concept, experiments demonstrate how SceNeRFlow enables simple edits that are consistent over time. The results also show how conditioning the canonical model on time trades off novel-view and correspondence quality. As discussed in Sec. 7.4, this chapter makes some simplifying assumptions but exploits them as little as possible, paving the way for future extensions to weaker inputs.

This concludes the second half of the thesis, which focused on reconstruction. First, the previous chapter extended the seminal NeRF method to dynamic scenes and then this chapter modified it to enable large motions.

# 8 CONCLUSION

This thesis is about neural digitization of general non-rigid objects and scenes. To this end, it introduced several new methods for representing and reconstructing general non-rigid objects, each of which benefits immensely from neural techniques: Chapter 4 presents a low-dimensional model for deformations using graph convolutions, Chapter 5 presents a generalizable model for geometry using coordinate-based neural networks, Chapter 6 presents a reconstruction method for general non-rigid objects using neural radiance fields, and Chapter 7 presents a reconstruction method that extends the method from the previous chapter to also handle large motion. Each chapter already contained some specific conclusions, but there are also some overarching insights and future steps that can be extracted from their sum total, as this concluding chapter of the thesis discusses.

## 8.1 INSIGHTS

The main insight that this thesis hints at is the use of coordinate-based networks as replacements for traditional parametrizations in classical non-learning-based optimization. Several facets of this point are discussed in the following.

### 8.1.1 *Domain Knowledge in the Small-Data Regime*

Recent foundation models strongly suggest that enough data and compute is sufficient to model language, image, audio, and, soon perhaps, video tasks, without the need for expert knowledge. However, researchers working in the 3D domain do not have access to such an internet-scale abundance of data. In this small-data regime, integrating domain knowledge from graphics into machine-learning pipelines remains beneficial, as Chapter 4 most explicitly shows. Chapter 5 similarly exploits the basic idea of local over global representations for generalization from little data. Furthermore, many tasks in computer graphics like material editing or deformation control become much more artist-friendly when factoring out deformations from the geometry and appearance as much as possible. This basic idea of factoring out the deformations allows the method in Chapter 6 to move beyond 'volumetric-video NeRFs' and to instead propose a truly 'deformable NeRF', thereby enabling much more challenging novel-view synthesis than concurrent work.

### 8.1.2 *Classical Optimization with Neural Networks*

Closely related to the previous point is the usage of deep learning in this thesis. Only Chapter 4 applies deep learning in its usual sense. In contrast, Chapter 5, Chapter 6, and Chapter 7 can be thought of as employing standard optimization for function fitting that uses MLPs to parametrize certain parts of the problem. Chapter 5 is somewhat in between, as PatchNets learn a parametrizable function that can represent different local geometries via a latent code (which acts as parameters). However, its method design does not lead its MLP to learn any truly high-level knowledge about geometry, rather its generalizability can be explained by the local nature of PatchNets. In a more clear-cut manner, Chapter 6 and Chapter 7 by design use their MLPs for scene-specific function fitting. Why then do these methods perform so well? Because the neural parametrization has an inductive smoothness bias and great flexibility, and enables arbitrary resolution, which are the crucial advantages over classical representations during optimization. Therefore, neural networks should be considered as strong alternatives to classical parametrizations (like polynomial ones) in non-deep-learning optimization problems.

### 8.1.3 *The Unreasonable Effectiveness of Neural Radiance Fields*

Starting with their surprising performance on static scenes, neural radiance fields remain an unreasonably powerful technique. However, extending them to general non-rigid objects should violate some of their core assumptions. In particular, the volumetric rendering acts as a 'depth voting' scheme, where each sample point's gradient upvotes the ground-truth pixel color at its position in space. When multiple rays intersect and agree at a position in space, that color is upvoted strongly, thereby slowly establishing correspondences. It is natural to integrate dynamics into this framework by factoring out the deformations and thereby allowing for accumulation of geometry and appearance information over the entire sequence. However, this deformation network adds a lot of freedom and could thereby conceivably hurt the optimization. Yet, it empirically can handle deformations of several centimeters, which is far larger than what would be expected from the purely local (pixel-wise) gradients of the reconstruction loss. This is not only due to the regularization loss terms but also because the underlying MLP representation lends itself to easier optimization, *e.g.*, due to higher flexibility, inherent smoothness, and its infinite resolution. Related to the previous point, it is noteworthy that these advantages are not related to deep learning (at least in any straightforward sense). Overall, since they can handle more challenging scenes than non-neural shape-from-template and non-rigid strucutre-from-motion methods, dynamic neural radiance fields are currently the most promising path towards general non-rigid reconstruction from RGB input.

## 8.2 OUTLOOK

Beyond the insights and contributions of this thesis, a lot of work remains. As this thesis deals with general non-rigid objects, the main difficulty lies with representing their deformations. Several future steps for this aspect are presented first. Then, the synergies with and integration of 2D methods for the reconstruction problem are discussed.

### 8.2.1 *Backwards Deformation Modeling*

Backwards deformation models are the natural choice for explicitly modeling deformations with volumetric representations like the density field of neural radiance fields. However, they suffer not only from violating standard intuition from forward models but also from being impractical for downstream tasks like editing. Searching for alternative deformation models for NeRF-style representations thus seems like a worthwhile endeavor. Ideally, future work would find a way to apply forward models, which is highly non-trivial at the time of this writing. Perhaps differentiable iso-surface extraction methods can be used to combine a NeRF-style canonical model with neural forward deformations applied to an extracted mesh, preserving the best of geometry and deformation modeling from both worlds.

### 8.2.2 *More Sophisticated Deformation Models*

Orthogonal to the previous point, current deformation models used for general non-rigid objects can be extended in the future to include even more desirable properties. For example, Chapter 4 learns a low-dimensional latent space but this space lacks semantics. Intuitive control thus remains difficult for general objects, in stark contrast to category-specific models for faces or bodies. On a different note, Chapter 7 shows that hierarchical deformation models have a place in the general dynamic NeRF field. Integrating semantic knowledge from 2D vision (without over-committing) into more hierarchical, perhaps skeleton-driven, models might allow for more standard editing and motion analysis. Ultimately, physics-based models would be desirable for certain tasks relying on simulation (like high-end video games, visual effects, or visualization in engineering), but they still remain too restricted and computationally expensive for easy and useful adaptation in NeRF-style methods.

### 8.2.3 *Synergies with Novel-View Synthesis*

Novel-view synthesis has many useful applications, especially in the areas of virtual and augmented reality as well as visual effects. However, many methods (*e.g.*, image-based ones or those using features on a 3D scaffold) focus solely on the novel-view synthesis task and forego the reconstruction aspect, *i.e.* 3D coherency is no longer available. While this makes for an easier problem to tackle, with many inspiring results, it unfortunately neglects more sophisticated downstream tasks like virtual-asset creation or robotics, which require 3D coherency. Thus, leveraging the countless insights from these kinds of novel-view synthesis methods for 3D reconstruction is a promising avenue for future work. For example, perhaps novel-view synthesis with 2D neural rendering on 3D *feature* volumes can be annealed back into a 3D-coherent neural scene representation of the standard graphics components of geometry and appearance. This might allow to tackle even harder reconstruction tasks by starting with an optimization problem that is even more relaxed than NeRF (which, for example, is so robust because it uses density even for surfaces) and then slowly tightening the relaxation to arrive at a 3D-coherent representation.

### 8.2.4 *Integrating Vision Models*

The methods in Chapter 6 and Chapter 7 are intentionally designed to not rely on additional, automatically extracted inputs annotations in order to avoid issues with general objects outside the training distribution of automatic annotation methods. However, the recent progress in depth estimation, optical-flow estimation, semantic segmentation, and other vision tasks promises that these tasks can now be relied upon for general dynamic objects without too many outliers and mistakes. The near future might see even more robustness in this regard, enabling reconstruction methods to exploit these rich 2D annotations without much effort. Similarly, self-supervised features learned by vision foundation models could perhaps soon allow for powerful annotations that effectively act like object coordinates. This would make it easier for reconstruction methods to find long-range correspondences for general non-rigid objects, which has not been feasible so far. Finally, diffusion models could be used for pseudo-multi-view supervision thanks to their novel-view synthesis capabilities, thereby stabilizing the optimization.

Since the field of representing and reconstructing general non-rigid objects is far from solved, many other avenues for future work exist. For example, time-varying appearance has barely been touched upon. Hopefully, researchers in the field will tackle this and the many other challenges successfully in the future.

# A APPENDIX FOR CHAPTER 4

This appendix expands on several points from Chapter 4. Sec. A.1 shows more examples of the artifacts seen in purely convolutional architectures. Sec. A.2 describes the normalization of depth maps and meshes (for reconstruction from real depth data). An expanded version of Tab. 4.4 is in Sec. A.3. Sec. A.4 contains more results of FCA and CA. Sec. A.5 shows artifacts due to coarse embedded graphs.

## A.1 ARTIFACTS

Tab. A.1 and A.2 show additional examples of artifacts that occur when not integrating the embedded graph into the network. Even the most competitive network (*i.e.* the ablation) suffers from visually unpleasant artifacts due to large non-rigid deformations, which most visibly occur on the hands and feet of DFaust. However, due to the localized nature of the artifacts, they do not have a large impact on the quantitative errors.

## A.2 NORMALIZATION

DEPTH All depth-to-mesh networks rescale the depth values of the input depth map from between $0.3m$ and $7m$ to $[-1, 1]$.

BODIES: DEPTH For the depth-to-mesh network on bodies, a number of additional normalization steps are helpful to focus on non-rigid reconstruction. First, a segmentation mask is used to filter out the background. The depth value of background pixels is set to 2. The foreground is cropped tightly and bilinearly sampled, to isotropically rescale the crop to $256 \times 256$. Given such a depth crop, the average (foreground) depth value is subtracted from the input. Such normalization necessitates normalizing the network output, which is described next.

BODIES: MESHES Subtracting from each mesh vertex the average vertex position normalizes out the global translation from the meshes. Since scale information is also lost, the scale of the meshes is fixed by normalizing their approximate spine length. To that end, the approximate spine length of the template mesh and of each mesh in the dataset is first computed. Then all the meshes are isotropically rescaled to the same spine

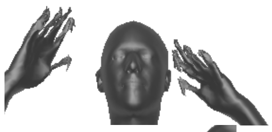
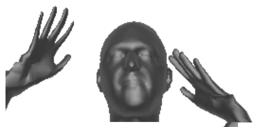
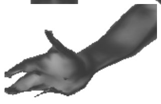
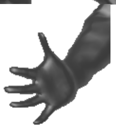
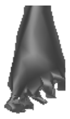
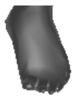
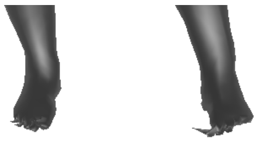
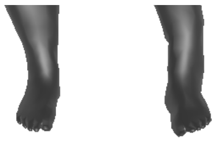
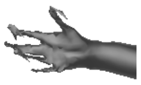
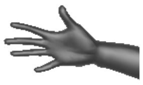
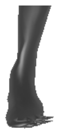
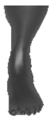
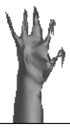
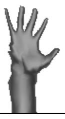
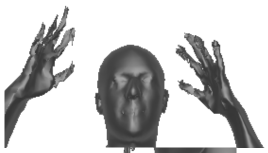
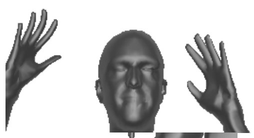
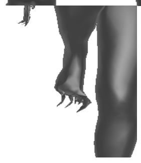
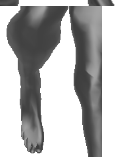
| Convolutional Ablation | DEMEA |
|---|---|
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table A.1: Artifacts. The top rows use 32 latent dimensions, the last two rows are for 8.

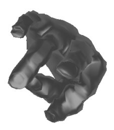
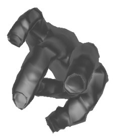
| Convolutional Ablation 8 | DEMEA 8 | Convolutional Ablation 32 | DEMEA 32 |
|---|---|---|---|
|  | |  | |
| | | | |

Table A.2: Artifacts on SynHand5M. In contrast to CA, DEMEA yields a smooth index finger in the examples shown above.

| | DFaust | | SynHand5M | | Cloth | | CoMA | |
|---|---|---|---|---|---|---|---|---|
| | 8 | 32 | 8 | 32 | 8 | 32 | 8 | 32 |
| FCA | 6.51 ± 2.45 | 2.17 ± 0.82 | 15.10 ± 4.06 | 2.95 ± 0.69 | 15.63 ± 7.18 | 5.99 ± 1.86 | 1.77 ± 0.57 | **0.67 ± 0.22** |
| FCED | 6.26 ± 2.35 | 2.14 ± 0.86 | 14.61 ± 3.95 | 2.75 ± 0.63 | 15.87 ± 7.73 | **5.94 ± 1.81** | 1.81 ± 0.71 | 0.73 ± 0.20 |
| CA | 6.35 ± 2.40 | **2.07 ± 0.73** | 8.12 ± 1.77 | 2.60 ± 0.60 | **11.21 ± 4.58** | 6.50 ± 1.85 | **1.17 ± 0.47** | 0.72 ± 0.22 |
| MCA | **6.21 ± 2.48** | 2.13 ± 0.79 | **8.11 ± 1.77** | 2.67 ± 0.60 | 11.64 ± 4.58 | 6.59 ± 1.96 | 1.20 ± 0.46 | 0.71 ± 0.21 |
| DEMEA | 6.69 ± 2.76 | 2.23 ± 0.99 | 8.12 ± 1.73 | **2.51 ± 0.59** | 11.28 ± 4.65 | 6.40 ± 1.96 | 1.23 ± 0.41 | 0.81 ± 0.22 |

Table A.3: Average per-vertex errors on the test sets of DFaust (in *cm*), SynHand5M (in *mm*), textureless cloth (in *mm*) and CoMA (in *mm*) for 8 and 32 latent dimensions, including standard deviations across the test sets.

length as the template mesh. The depth-to-mesh body reconstruction errors in Chapter 4 are reported for these normalized meshes.

## A.3 STANDARD DEVIATIONS IN TAB. 4.4

Tab. A.3 contains an expanded version of Tab. 4.4 that also has standard deviations.

## A.4 FCA AND CA RESULTS

Tab. A.4 contains qualitative results for FCA and Tab. A.5 shows artifacts when using FCA. Tab. A.6 contains qualitative results for CA. Tab. A.7 shows depth-to-mesh results.

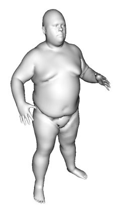
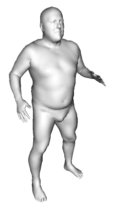
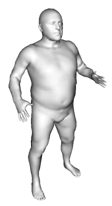
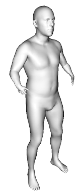
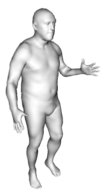
| Ground Truth | FCA 32 | DEMEA 32 | FCA 8 | DEMEA 8 |
|---|---|---|---|---|
|  |  |  |  |  |

Table A.4: Qualitative results on FCA.

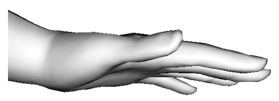
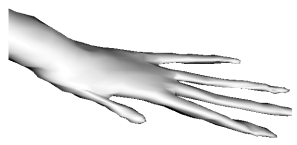
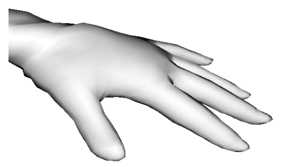
| Ground Truth | FCA 32 | DEMEA 32 |
|---|---|---|
|  |  |  |

Table A.5: Artifacts on FCA. While the reconstruction by DEMEA only matches the ground truth as well as FCA, it is a significantly more plausible shape.

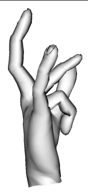
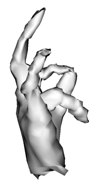
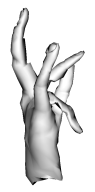
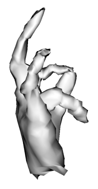
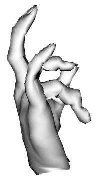
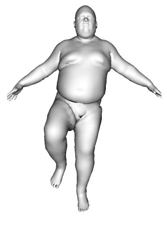
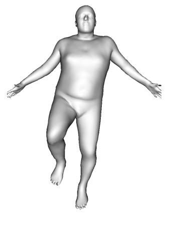
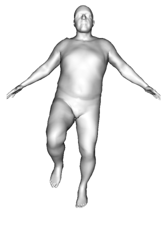
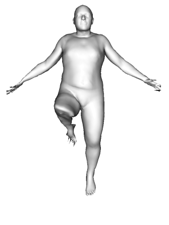
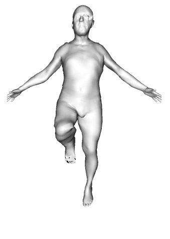
| Ground Truth | CA 32 | DEMEA 32 | CA 8 | DEMEA 8 |
|---|---|---|---|---|
|  |  |  |  |  |
|  |  |  |  |  |

Table A.6: Qualitative results on CA.

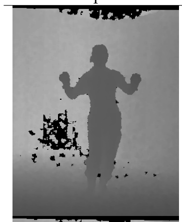
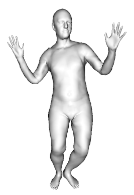
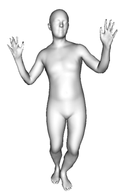
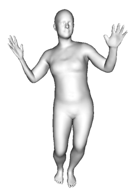
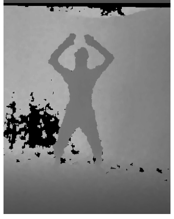
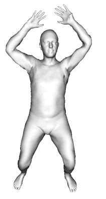
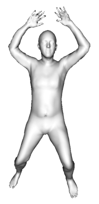
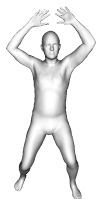
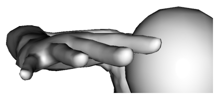
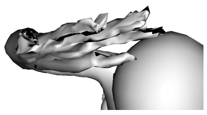
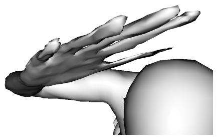
| Depth | DEMEA | CA | FCA |
|---|---|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table A.7: Depth-to-mesh results for CA and FCA. The bottom row shows artifacts that DEMEA avoids.

## A.5 COARSE EMBEDDED GRAPHS

Tab. A.8 shows how an embedded graph can lead to over-smoothing and a loss of detail.

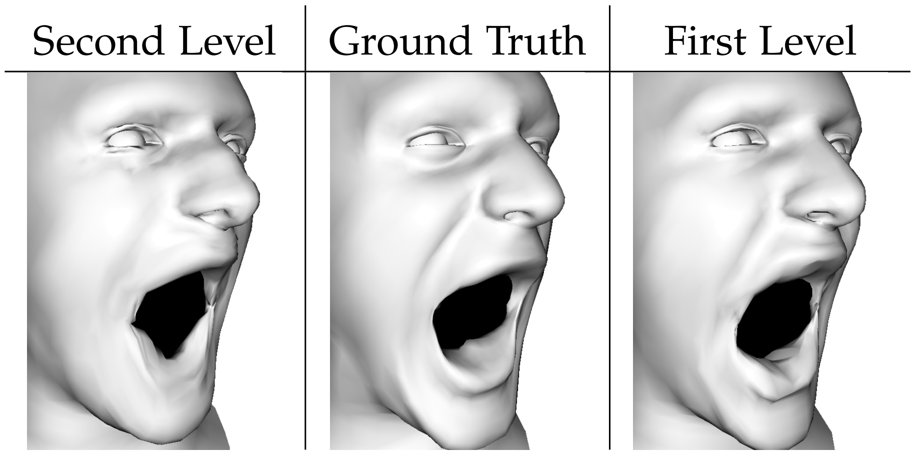


Table A.8: Coarse embedded graphs. Note the lips. An embedded graph on the second level of the mesh hierarchy instead of the first level can lead to over-smoothing.

# B APPENDIX FOR CHAPTER 5

This appendix expands on several points from Chapter 5. Sec. B.1 compares error measures on the reduced and full test sets. Sec. B.2 contains more experiments using object-level priors. Next, Sec. B.3 measures the performance under synthetic noise. Sec. B.4 shows preliminary results on a large scene.

## B.1 REDUCED TEST SET

The reduced test set on ShapeNet consists of 50 randomly chosen test shapes per category. Tab. B.1 shows how well the error measures on this reduced test set approximate the error measures on the full test set.

## B.2 OBJECT-LEVEL PRIORS

### B.2.1 *Surface Reconstruction*

This section reports surface reconstruction errors using object-level priors (see Sec. 5.3.3). Note that the experiments in Sec. 5.3.3 use the *most* competitive setting of the global-patch baseline (*i.e.*, pretrained on all categories and then refined) and the *least* competitve setting of PatchNet (*i.e.*, pretrained on one category and not refined). This demonstrates how well the proposed PatchNet generalizes. For consistency, for the DeepSDF-based baseline, the same setting as for the global-patch baseline are used. Note that that setting is virtually on par with the most competitive DeepSDF setting (*i.e.*, pretrained on one category and then refined).

#### B.2.1.1 *Settings*

Both PatchNets and the baselines consist of a four-layer ObjectNet and the standard final eight FC layers. The final eight FC layers are pretrained either on the reduced training set of all categories or on all shapes from the *Cabinets* category training set. Then, those pretrained weights are either kept fixed fixed while training ObjectNet or they are allowed to change for further refinement. While at training time, each phase lasts 1000 epochs, this is reduced to 800 epochs at test time.

| | IoU | | | | | | Chamfer | | | | | | F-score | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | DeepSDF | | Baseline | | PatchNets | | DeepSDF | | Baseline | | PatchNets | | DeepSDF | | Baseline | | PatchNets | |
| | full | red. | full | red. | full | red. | full | red. | full | red. | full | red. | full | red. | full | red. | full | red. |
| airplane | 84.9 | 84.0 | 65.3 | 64.2 | 91.1 | 90.7 | 0.012 | 0.023 | 0.077 | 0.084 | 0.004 | 0.006 | 93.0 | 92.3 | 72.9 | 71.6 | 97.8 | 97.5 |
| bench | 78.3 | 77.1 | 68.0 | 65.7 | 85.4 | 83.7 | 0.021 | 0.015 | 0.065 | 0.043 | 0.006 | 0.006 | 91.2 | 90.4 | 80.6 | 80.1 | 95.7 | 94.9 |
| cabinet | 92.2 | 89.1 | 88.8 | 84.8 | 92.9 | 91.6 | 0.033 | 0.027 | 0.055 | 0.047 | 0.110 | 0.119 | 91.6 | 90.3 | 86.4 | 84.3 | 91.2 | 91.8 |
| car | 87.9 | 88.4 | 83.6 | 84.3 | 91.7 | 92.6 | 0.049 | 0.057 | 0.070 | 0.074 | 0.049 | 0.050 | 82.2 | 82.1 | 74.5 | 74.4 | 87.7 | 87.8 |
| chair | 81.8 | 80.1 | 72.9 | 70.3 | 90.0 | 88.6 | 0.042 | 0.041 | 0.110 | 0.118 | 0.018 | 0.013 | 86.6 | 86.0 | 75.5 | 74.8 | 94.3 | 93.5 |
| display | 91.6 | 92.9 | 86.5 | 89.1 | 95.2 | 95.5 | 0.030 | 0.010 | 0.061 | 0.034 | 0.039 | 0.049 | 93.7 | 95.1 | 87.0 | 89.8 | 97.0 | 97.3 |
| lamp | 74.9 | 72.3 | 63.0 | 63.4 | 89.6 | 88.0 | 0.566 | 2.121 | 0.438 | 0.257 | 0.055 | 0.063 | 82.5 | 79.9 | 69.4 | 70.1 | 94.9 | 94.0 |
| rifle | 79.0 | 78.0 | 68.5 | 66.0 | 93.3 | 93.1 | 0.013 | 0.012 | 0.039 | 0.046 | 0.002 | 0.001 | 90.9 | 90.7 | 82.3 | 80.4 | 99.3 | 99.3 |
| sofa | 92.5 | 92.2 | 85.4 | 84.5 | 95.0 | 95.1 | 0.054 | 0.075 | 0.226 | 0.236 | 0.014 | 0.012 | 92.1 | 91.3 | 84.2 | 83.0 | 95.3 | 95.3 |
| speaker | 91.9 | 90.5 | 86.7 | 84.9 | 92.7 | 90.8 | 0.050 | 0.060 | 0.094 | 0.121 | 0.243 | 0.242 | 87.6 | 84.7 | 79.4 | 75.7 | 88.5 | 85.1 |
| table | 84.2 | 83.4 | 71.9 | 69.5 | 89.4 | 90.3 | 0.074 | 0.043 | 0.156 | 0.169 | 0.018 | 0.017 | 91.1 | 91.5 | 79.2 | 79.1 | 95.0 | 96.1 |
| telephone | 96.2 | 96.0 | 95.0 | 94.1 | 98.1 | 98.0 | 0.008 | 0.010 | 0.016 | 0.016 | 0.003 | 0.004 | 97.7 | 97.3 | 96.2 | 94.7 | 99.4 | 99.3 |
| watercraft | 85.2 | 84.9 | 79.1 | 78.5 | 93.2 | 93.1 | 0.026 | 0.019 | 0.041 | 0.031 | 0.009 | 0.006 | 87.8 | 88.2 | 90.2 | 80.6 | 96.4 | 96.6 |
| mean | 86.2 | 85.3 | 78.1 | 76.9 | 92.1 | 91.6 | 0.075 | 0.193 | 0.111 | 0.098 | 0.044 | 0.045 | 89.9 | 89.2 | 80.6 | 79.9 | 94.8 | 94.5 |

Table B.1: Reduced Test Set vs. Full Test Set. The computed metrics on the reduced test set of ShapeNet are a good approximation of the computed metrics on the full test set. This is an extended version of Tab. 5.1.

| | | baseline | | | | DeepSDF-based | | | | PatchNets | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | one | | all | | one | | all | | one | | all | |
| | | fix. | ref. | fix. | ref. | fix. | ref. | fix. | ref. | fix. | ref. | fix. | ref. |
| airplanes | IoU | 35.9 | 70.9 | 60.2 | 73.3 | 47.0 | 75.6 | 69.9 | 74.1 | 67.5 | 68.5 | 71.9 | 74.2 |
| | Chamfer | 0.710 | 0.146 | 0.218 | 0.147 | 0.546 | 0.049 | 0.127 | 0.050 | 0.203 | 0.182 | 0.179 | 0.170 |
| | F-score | 37.5 | 76.0 | 63.6 | 78.3 | 49.1 | 82.5 | 76.4 | 81.6 | 71.7 | 74.1 | 77.9 | 79.7 |
| sofas | IoU | 76.1 | 81.8 | 76.3 | 84.3 | 76.4 | 79.7 | 82.4 | 76.6 | 85.3 | 86.2 | 84.9 | 86.0 |
| | Chamfer | 0.416 | 0.159 | 0.398 | 0.171 | 0.467 | 0.178 | 0.282 | 0.406 | 0.118 | 0.139 | 0.236 | 0.082 |
| | F-score | 69.0 | 75.2 | 71.8 | 77.9 | 70.1 | 72.3 | 77.5 | 71.8 | 79.0 | 80.7 | 79.5 | 79.9 |

Table B.2: Surface Reconstruction with ObjectNet. The final eight layers are pretrained either on one category (*one*) or on all categories (*all*). Then, they are either kept fixed (*fix.*) or are refined (*ref.*).

#### B.2.1.2 *Results*

Tab. B.2 contains the quantitative results. The baselines do not generalize well if they are kept fixed. Refinement improves error measures.

### B.2.2 *Ablation Study*

Next, the extrinsics losses are ablated in the context of surface reconstruction with object-level priors. PatchNets is pretrained on the *Cabinets* category and was not refined. The ablation study is performed on the *Sofas* category.

| | IoU | Chamfer | F-score |
|---|---|---|---|
| no $\mathcal{L}_{\text{sur}}$ | 87.6 | 0.076 | 82.6 |
| no $\mathcal{L}_{\text{scl}}$ | 75.5 | 0.154 | 54.2 |
| no $\mathcal{L}_{\text{var}}$ | 71.8 | 0.269 | 47.3 |
| PatchNets | 85.3 | 0.118 | 79.0 |
| PatchNets with $\mathcal{L}_{\text{recon}}$ on mixture | 84.9 | 0.116 | 78.1 |

Table B.3: Ablation Study with Object-level Priors. Each of the extrinsics losses is removed.

| | sofas fixed | | sofas random | | airplanes fixed | | airplanes random | |
|---|---|---|---|---|---|---|---|---|
| | acc. | F-score | acc. | F-score | acc. | F-score | acc. | F-score |
| baseline | 0.094 | 43.0 | 0.092 | 42.7 | 0.069 | 58.1 | 0.066 | 58.7 |
| DeepSDF-based baseline | 0.106 | 33.6 | 0.101 | 39.5 | 0.066 | 56.9 | 0.065 | 55.5 |
| proposed (Chapter 5) | 0.091 | 48.1 | 0.077 | 49.2 | 0.058 | 60.5 | 0.056 | 59.4 |
| proposed+refined (Chapter 5) | **0.052** | 53.6 | **0.053** | 52.4 | **0.041** | 67.7 | **0.043** | 65.8 |
| proposed (baseline-matched) | 0.088 | 47.5 | 0.074 | 50.0 | 0.052 | 64.8 | 0.050 | 64.3 |
| proposed+refined (baseline-matched) | 0.061 | **54.7** | 0.056 | **53.5** | 0.045 | **70.3** | 0.044 | **69.9** |

Table B.4: Partial Point Cloud Completion from Depth Maps. Depth maps are completed from a fixed camera viewpoint and from per-scene random viewpoints.

The quantitative results are in Tab. B.3. The network failed to reconstruct without $\mathcal{L}_{\text{cov}}$.

### B.2.3 *Partial Point Cloud Completion*

This section reports additional depth-map completion results using the same settings for PatchNets that were used for the baselines in Chapter 5 (pretrained on all categories and refined). Note that Chapter 5 reports the shape-completion results of the most disadvantageous version of the proposed method (according to Tab. B.2). Tab. B.4 contains the quantitative results. In all cases, PatchNets after local refinement yields the best results.

## B.3 SYNTHETIC NOISE

Finally, the robustness of PatchNet to noise is investigated by adding Gaussian noise to the ground-truth SDF values of the reduced test set. PatchNet is trained with default settings, which also means that it has only seen unperturbed SDF data during training. The Gaussian noise has zero mean and different standard deviations $\sigma$. For reference, the mesh

| | IoU | Chamfer | F-score |
|---|---|---|---|
| $\sigma = 0.1$ | 81.2 | 0.037 | 85.3 |
| $\sigma = 0.01$ | 90.3 | 0.045 | 94.3 |
| $\sigma = 0.001$ | 91.5 | 0.047 | 94.4 |
| $\sigma = 0$ (proposed) | 91.6 | 0.045 | 94.5 |

Table B.5: Synthetic Noise at Test Time.

fits tightly into the unit sphere, as mentioned in Sec. 5.2.2. The results are in Tab. B.5.

## B.4 PRELIMINARY RESULTS ON ICL-NUIM

Once trained, a PatchNet can be used for any number of patches at test time. Here, some preliminary results on the large living room from ICL-NUIM (Handa et al., 2014) are presented.

Since the scene is already watertight, the depth fusion step of the preprocessing method can be skipped. The standard deviation used to generate SDF samples is reduced by a factor of 100 to account for scaling differences. Overall, 50 million SDF samples are sampled.

PatchNet uses 800 patches. The extrinsics are kept fixed at their initial values to improve the reconstruction. The optimization lasts for $10k$ iterations, halving the learning rate every $2k$ iterations. During optimization, $25k$ SDF samples are used per iteration. The baselines are trained with the same modified settings.

The results are in Fig. B.1. Note that due to our extrinsics initialization (Sec. 5.2.2) and $\mathcal{L}_{\text{var}}$, all patches have similar sizes, which leads to a wasteful distribution.

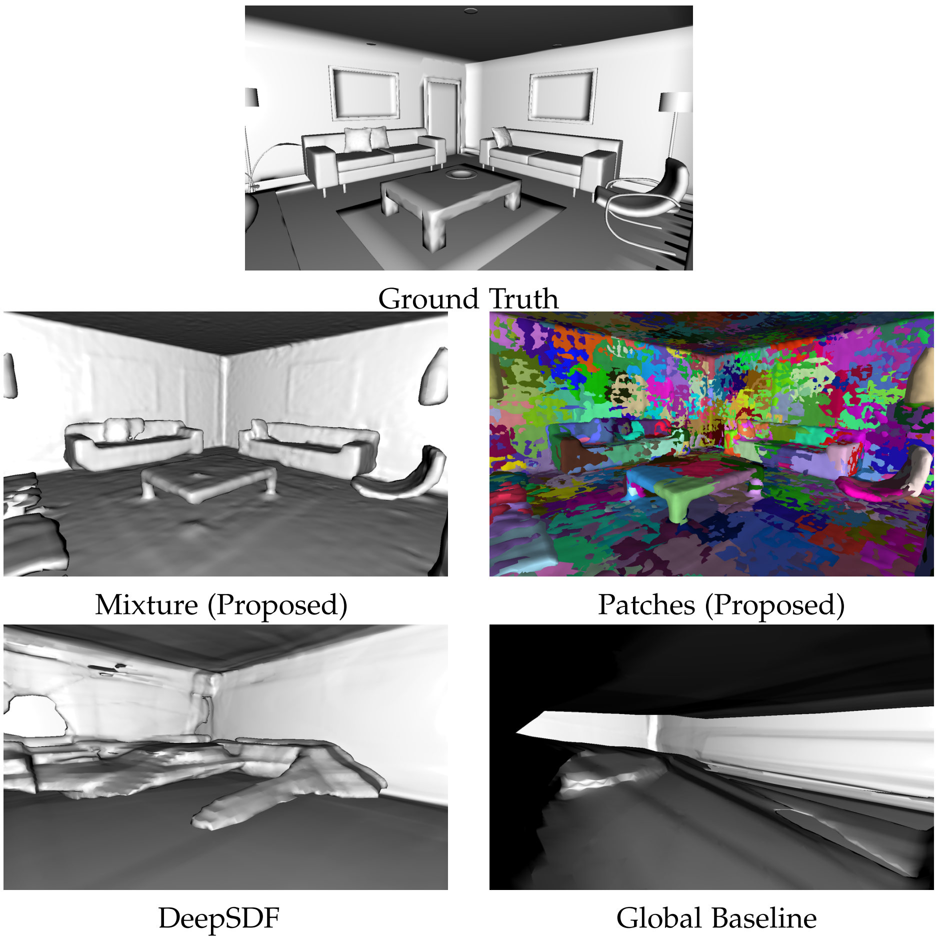


Figure B.1: Preliminary Results on ICL-NUIM.

# C APPENDIX FOR CHAPTER 6

This appendix expands on several points from Chapter 6. Sec. C.1 discusses a number of training details. Next, Sec. C.2 provides some implementation details. Sec. C.3 provides more details on the experimental settings of the comparisons and some additional results. Sec. C.4 presents the extensions to multi-view data and view-dependent effects. Finally, Sec. C.5 contains some preliminary qualitative comparisons to a concurrent, non-peer-reviewed work.

## C.1 TRAINING DETAILS

While the network weights are optimized as usual, the latent codes $\mathbf{l}_i$ are auto-decoded, *i.e.*, they are treated as free variables that are directly optimized for, similar to network weights, instead of being regressed. This is based on the auto-decoding framework used in DeepSDF and earlier works (Park et al., 2019; Tan and Mayrovouniotis, 1995).

The latent codes $\{\mathbf{l}_i\}_i$ are initialized to zero vectors. The radiance field is implemented with the same architecture as in NeRF (Mildenhall et al., 2020). The ray bending network is a 5-layer MLP with 64 hidden dimensions and ReLU activations, the last layer of which is initialized with all weights set to zero. The rigidity network is a 3-layer MLP with 32 hidden dimensions and ReLU activations, with the last layer initialized to zeros. The output of the last layer of the rigidity network is passed though a tanh activation function and then shifted and rescaled to lie in $[0, 1]$. Training usually takes $200k$ iterations with a batch of $1k$ randomly sampled rays. At training and at test time, rendering uses 64 coarse and 64 fine samples per ray in most cases. The optimization uses Adam (Kingma and Ba, 2015) and exponentially decays the learning rate to 10% from the initial $5 \cdot 10^{-4}$ over $250k$ iterations. For dark scenes, it is empirically necessary to introduce a warm-up phase that linearly increases the learning rate starting from $\frac{1}{20}$th of its original value over 1000 iterations. The latent codes are of the dimension 32. Training lasts between six and seven hours on a single Quadro RTX 8000 GPU.

NR-NeRF needs scene-specific weights for each loss term due to the variety of types of non-rigid objects/scenes and deformations. The following ranges are sufficient for a wide range of scenarios: $\omega_{\text{rigidity}}$ lies in $[0.01, 0.001]$ and typically is 0.003, $\omega_{\text{offsets}}$ lies in $[60, 600]$ and typically is 600, and $\omega_{\text{divergence}}$ lies in $[1, 30]$ and typically is 3 or 10. NR-NeRF is empirically fairly insensitive to $\omega_{\text{offsets}}$. Rather rigid objects benefit from higher $\omega_{\text{divergence}}$, while fairly non-rigid objects need lower $\omega_{\text{divergence}}$. Finally, $\omega_{\text{rigidity}}$ should be increased whenever the background is unstable. The training starts with each weight set to $\frac{1}{100}$th of its value, and then exponentially increases it until it reaches its full value at the end of training.

## C.2 IMPLEMENTATION DETAILS

The code is based on a faithful PyTorch (Paszke et al., 2019) port (Yen-Chen, 2020) of the official Tensorflow NeRF code (Mildenhall et al., 2020). Eq. 6.6 is estimated using the official FFJORD implementation (Grathwohl et al., 2018). If the camera extrinsics and intrinsics are not given, they are estimated using the Structure-from-Motion (SfM) implementation of COLMAP (Schönberger and Frahm, 2016; Schönberger et al., 2016). COLMAP is quite robust to non-rigid 'outliers'. Since NR-NeRF aims for smooth deformations, positional encoding is only applied to the input of the canonical NeRF volume, not to the input of the ray bending network.

## C.3 COMPARISONS

This section provides more details on the experimental settings of the comparisons.

### C.3.1 *Prior Work and Baseline*

The starting point is the trivial baseline of rigid NeRF (Mildenhall et al., 2020), which cannot handle dynamic scenes. Two variants are considered: view-dependent rigid NeRF, as in the original method (Mildenhall et al., 2020), and view-independent rigid NeRF, where the view-direction conditioning is removed.

Next, *naïve NR-NeRF* adds naïve support for dynamic scenes to rigid NeRF: It conditions the neural radiance fields volume on the latent code. Thus, for latent code $\mathbf{l}_i$, it computes $(\mathbf{c}, o) = \mathbf{v}(\mathbf{x}, \mathbf{l}_i)$. This allows the neural radiance fields volume to output time-varying color and occupancy. Unlike NR-NeRF's ray bending, naïve NR-NeRF does not have an explicit, separate deformation model. Instead, the volume needs to account for appearance, geometry, and deformation at once. Note that for test images $i$, gradients are backpropagated into the corresponding latent code $\mathbf{l}_i$.

Neural Volumes (Lombardi et al., 2019) is the final baseline, which uses the official code release. The standard settings are used as a starting point. However, the number of training iterations is set to $100k$, which leads to a training time of about two days on an RTX 8000 GPU. Neural Volumes uses an image encoder to regress a latent code that conditions the geometry, appearance, and deformation regression on the current time step. Since this design assumes a multi-view setup, it needs to be adapted to the monocular setting. Instead of picking three fixed camera views that are always input into the encoder, the single image of the current time step is used as input. In particular, at test time, the test image is used as input. Since there is no access to a background image, the estimated background image is set to an all-black image. Two variations are considered: (1) following the original Neural Volumes method, the geometry and appearance template is conditioned on the latent code (*NV*), and (2) the geometry and appearance template is independent of the latent code (*modified NV*). In the latter case, the latent code only conditions the warp field, which is similar to the proposed method.

### C.3.2 *Training/Test Split*

Quantitative evaluation requires a test set. In the comparison section, the images are split into training and test images by partitioning the temporally-ordered images into consecutive blocks, each of length 16. The first twelve images of each block are used as training data, while the remaining four are used for testing. In the monocular setting, test images still require corresponding latent codes to represent the deformations. Therefore, test images are treated like training images except that gradients are not backpropagated into the canonical volume or the ray bending network. However, the gradients from test images do optimize the corresponding latent codes. (Note that test images *solely* influence test latent codes, as is typical for auto-decoding (Park et al., 2019). [1]) All other results, outside of the comparisons, treat all images as training images since the reconstruction are achieved by training scene-specific networks.

### C.3.3 *Additional Results*

See Fig. C.1 for more qualitative results and Fig. C.4 for quantitative results on background stability under novel views.

1 The only tweak NR-NeRF adds is to align the optimization landscapes of the training and test latent codes by optimizing the test latent codes jointly with the training latent codes during training. This does not lead to any information leakage from the test images to any component except for the test latent codes.

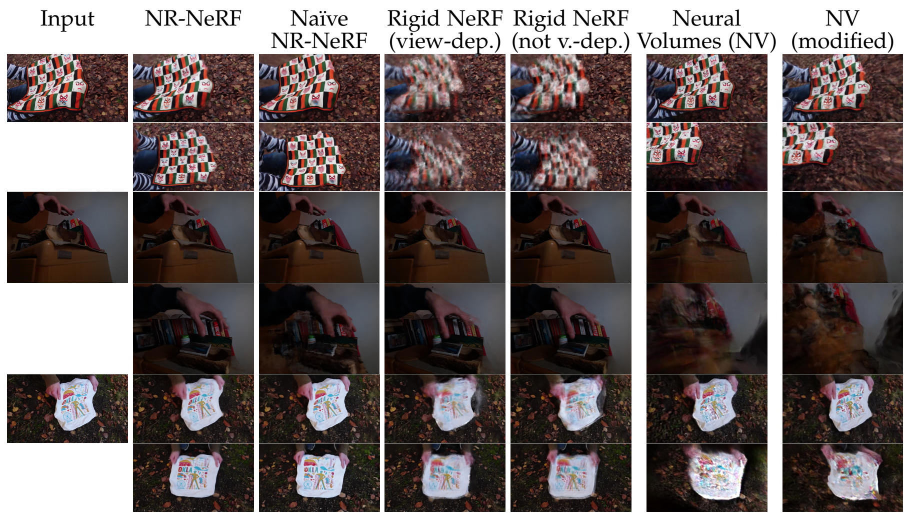


Figure C.1: Visualization of one time step each from each sequence, for input reconstruction quality (first row) and novel view synthesis quality (second row).

## C.4 EXTENSIONS

As mentioned in Sec. 6.3, NR-NeRF can be extended easily to work with multi-view data and view-dependent effects.

### C.4.1 *Multi-View Data*

The proposed approach naturally handles multi-view data. Although the experiments mainly consider monocular data, multi-view data is useful to investigate the upper quality bound of NR-NeRF under ideal real-world conditions.

METHOD Instead of each image having its own time step and hence latent code, images taken at the same time step share the same latent code. This ensures that the canonical volume deforms consistently within each time step.

DATA AND SETTINGS The multi-view dataset has 16 camera pairs evenly distributed around the scene, which sufficiently constrains the optimization such that the training does not need any regularization losses. Training happens at the original resolution of $5120 \times 3840$ for 2 million training iterations with 4096 rays per batch and 256 coarse and 128 fine samples. These highest-quality settings lead to a training time of 11 days on 4 RTX 8000 GPUs, and a rendering time of about 10 minutes per frame on the same hardware.

RESULTS Fig. C.2 shows results on five consecutive time steps.

### C.4.2 *View Dependence*

NR-NeRF can optionally handle view-dependent effects, like specularities.

METHOD Determining the view direction or ray direction is not as trivial as for the straight rays. Instead, the direction in which the bent ray passes through a point in the canonical volume is required. NR-NeRF considers two options of doing so: exact and slower, or approximate and faster.

*Exact:* The direction of the bent ray $\tilde{\mathbf{r}}$ at a point $\tilde{\mathbf{r}}(j)$ is obtained via the chain rule as $\nabla_j \tilde{\mathbf{r}}(j) = \frac{\partial \tilde{\mathbf{r}}(j)}{\partial \bar{\mathbf{r}}(j)} \cdot \frac{\partial \bar{\mathbf{r}}(j)}{\partial j} = J \cdot \mathbf{d}$, where $J$ is the $3 \times 3$ Jacobian and $\mathbf{d}$ is the direction of the straight ray. $J$ takes three backward passes to compute (one for each output dimension), which is computationally expensive.

*Approximate:* To reduce computation, the direction at the ray sample can be approximated via finite differences as the normalized difference vector between the current point $\tilde{\mathbf{r}}(j)$ and the previous point $\tilde{\mathbf{r}}(j-1)$ along the bent ray (which is closer to the camera).

RESULTS On multi-view data, conditioning on the viewing direction reduces the presence of subtle, smoke-like artifacts, which the canonical volume typically employs to model view-dependent effects without view conditioning. This is especially visible for the specularities on the face and the handle of the kettlebell. Without view-dependent effects, the reconstructed face still appears to exhibit specularities, but these are *incorrectly* modeled via smoke-like artifacts in the surrounding air. See Fig. C.2 for results.

For quantitative results on monocular sequences, see Tab. C.4.2. However, as Fig. C.3 shows, the proposed formulation leads to artifacts in some cases. Presumably, the combination of both significant motion and novel views significantly different from input views is too underconstrained for view-dependent effects. For example, non-rigid NeRF might incorrectly overfit to subtle correlations between deformation and camera position at training time. However, better formulations and regularization in future work may make view-dependent effects work in these challenging scenarios.

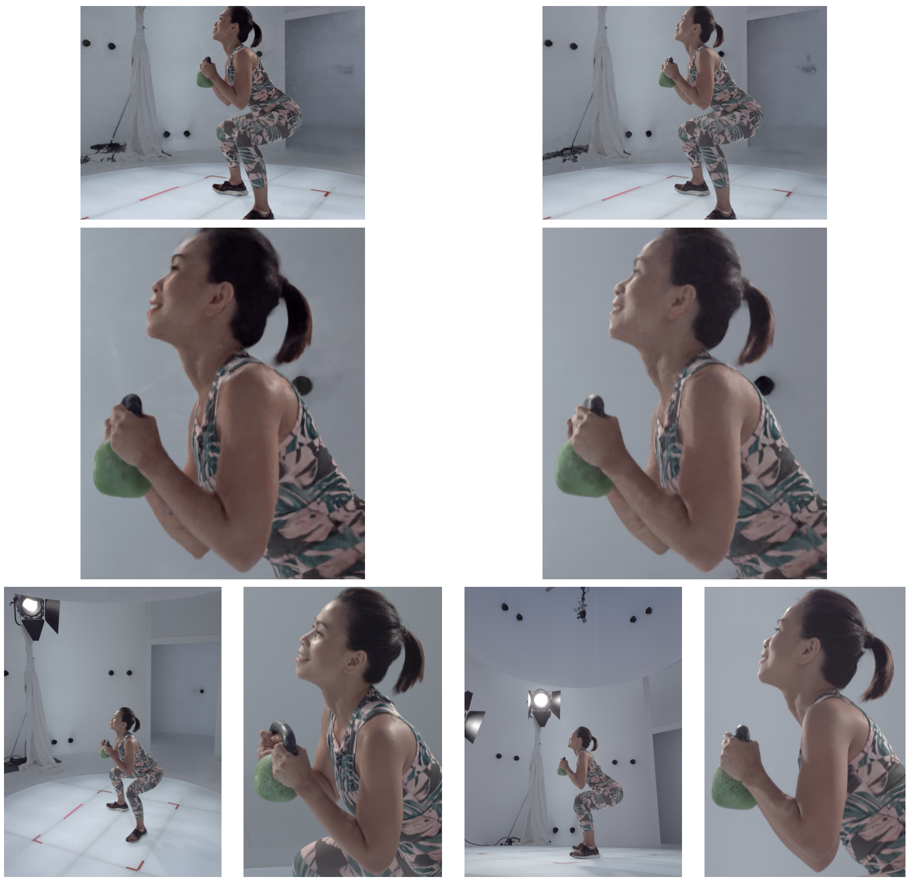


Figure C.2: A highly controlled multi-view setting allows to explore the upper quality bound of the proposed method. NR-NeRF can be extended to handle view-dependent effects. Results on the left are without view dependence, while those on the right are with view dependence. A full rendering by NR-NeRF (first row), zoom-ins thereof (second row), and input images from the two closest input cameras.

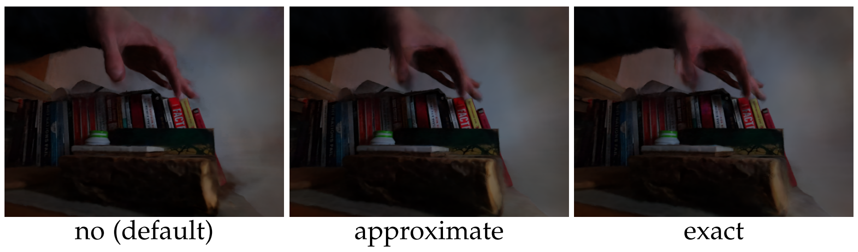


Figure C.3: While NR-NeRF extended with view-dependent effects (approximate or exact) gives similar results to the default NR-NeRF for many monocular scenes, it sometimes leads to artifacts for difficult novel views.

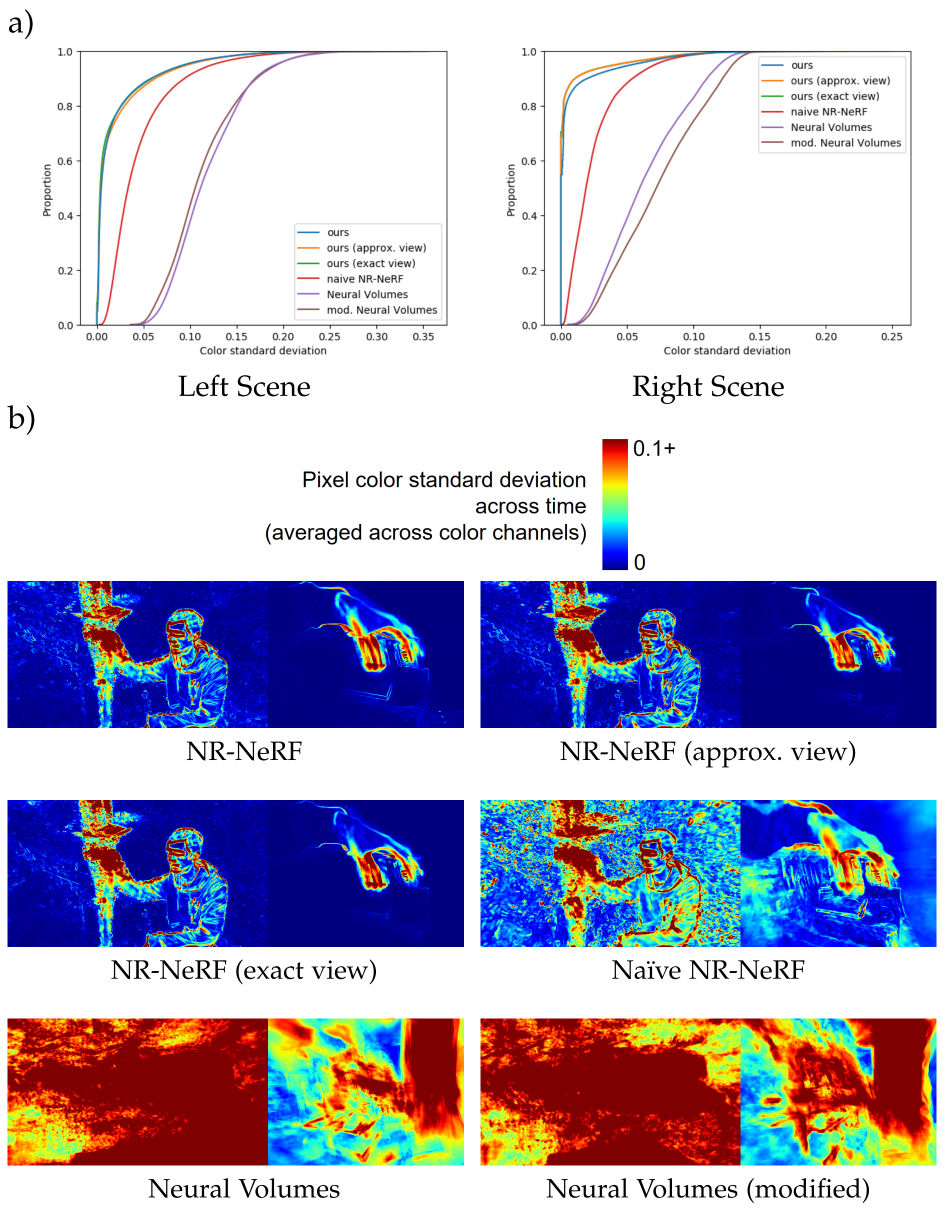


Figure C.4: Background Stability. This figure quantifies the difference in background stability between the proposed method, its variants with view dependence, naïve NR-NeRF, and Neural Volumes. To that end, all test time steps of the input sequence are rendered into a fixed novel view. Then, the standard deviation of each pixel's color across time is computed to measure color changes and hence background stability. **a)** Cumulative plots across all pixels, where NR-NeRF and its variants (left-most curves) have the most stable background. **b)** Distribution of those instabilities in the scene. The results of NR-NeRF and its variants show the least instability in the background.

| | NR-NeRF | NR-NeRF (appx.) | NR-NeRF (exact) | Naïve | Rigid (cond.) | Rigid (no cond.) | NV | NV (mod.) |
|---|---|---|---|---|---|---|---|---|
| PSNR | 24.70 | *25.15* | 25.07 | **25.83** | 22.24 | 21.88 | 14.13 | 14.10 |
| SSIM | 0.758 | **0.766** | *0.765* | 0.738 | 0.662 | 0.659 | 0.259 | 0.263 |
| LPIPS | 0.197 | *0.191* | **0.190** | 0.226 | 0.309 | 0.313 | 0.580 | 0.583 |

Table C.1: Quantitative Results Averaged Across Scenes. The following methods are evaluated: NR-NeRF (1) without view conditioning, (2) with approximate view conditioning, and (3) with exact view conditioning, naïve NR-NeRF, rigid NeRF (Mildenhall et al., 2020) (1) with view conditioning and (2) without view conditioning, and Neural Volumes (Lombardi et al., 2019) (1) without and (2) with modifications. For PSNR and SSIM (Wang et al., 2004), higher is better. For LPIPS (Zhang et al., 2018), lower is better. As in Tab. 6.1, 18 scenes are used here, with an average length of 146 frames and a minimum of 41 and a maximum of 453 frames.

## C.5 ADDITIONAL COMPARISONS

This section shows preliminary qualitative comparisons to the concurrent, non-peer-reviewed work Neural Scene Flow Fields (Li et al., 2021b) in Fig. C.5.

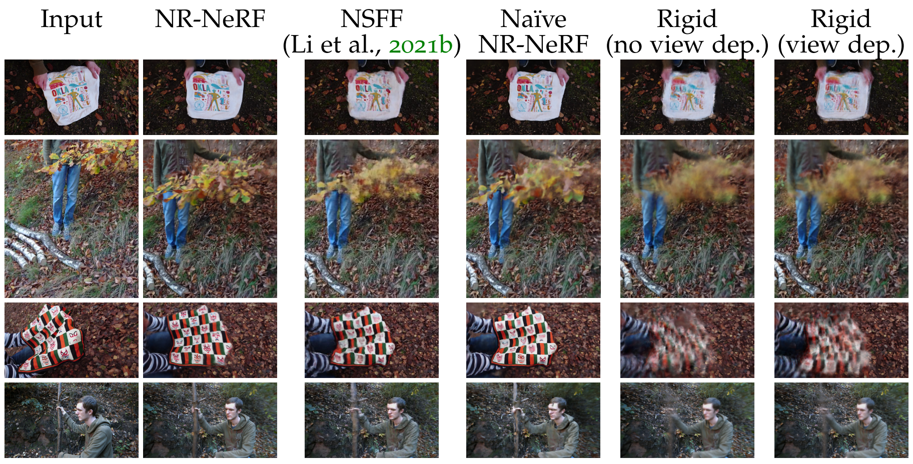


Figure C.5: Under challenging novel view scenarios, NR-NeRF benefits from the geometry and appearance information that the canonical volume has accumulated from all time steps, which allows NR-NeRF to output sharp results. Both the concurrent, non-peer-reviewed Neural Scene Flow Fields (Li et al., 2021b) and naïve NR-NeRF however entangle deformation with geometry and appearance by conditioning the 'canonical' volume on a time-dependent deformation latent code. This makes sharing information across time more difficult, leading to blurrier results in challenging novel view scenarios compared to NR-NeRF's results. Finally, rigid NeRF shows a blurry mix of the deformations observed over the entire input sequence, which highlights the need to account for deformations in the scene.

# D APPENDIX FOR CHAPTER 7

This appendix expands on several points from Chapter 7. Sec. D.1 provides details on how correspondences are visualized. Sec. D.2 shows per-scene quantitative novel-view-synthesis results. Sec. D.3 presents more qualitative joint-tracking and novel-view results. Sec. D.4 contains further architecture and training details. Sec. D.5 provides details on how the baselines are adapted to joint tracking. Sec. D.6 gives more details on the foreground masks used for evaluation.

## D.1 CORRESPONDENCE VISUALIZATION DETAILS

The experiments follow NR-NeRF's visualization and replace the appearance in the canonical model with a voxel grid of RGB cubes. Like prior work (Park et al., 2021a; Tretschk et al., 2021), that sample $i'$ of the ray is picked as the surface point $\mathbf{r}(s_{i'})$ that is closest to an accumulated transmittance $\sum_{i=1}^{i'} w_i$ of 0.5. For better visualization, rays with a total accumulated transmittance below 0.4 (*i.e.* those hitting the background) are filtered out and visualized as transparent. Fig. D.1 shows an example.

## D.2 PER-SCENE QUANTITATIVE RESULTS

Tab. D.1 collects the quantitative results per scene for novel-view synthesis.

## D.3 MORE QUALITATIVE JOINT-TRACKING AND NOVEL-VIEW RESULTS

Fig. D.2 contains the qualitative joints estimated at $t{=}T$ for all sequences not shown in Chapter 7. Fig. D.3 and Fig. D.4 contain more novel-view results of all methods for four scenes at $t{=}\frac{T}{2}$.

## D.4 FURTHER ARCHITECTURE AND TRAINING DETAILS

For all methods, the scene is normalized into the unit cube by tightly fitting an axis-aligned bounding box to all near and far plane samples of all images.

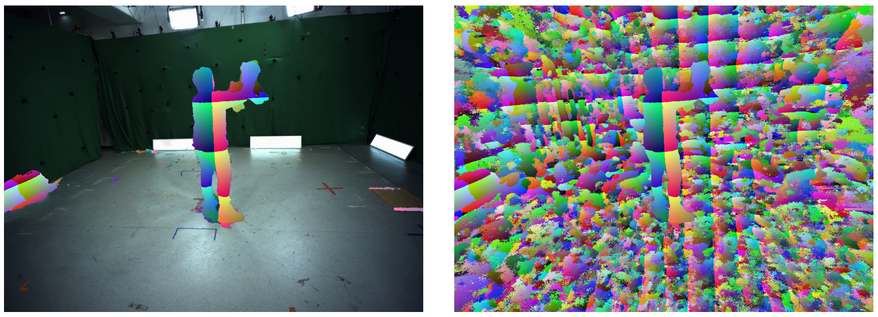


Figure D.1: **Filtering for Correspondence Visualization.** For better visualization, rays with an accumulated transmittance below 0.4 are filtered out.

### D.4.1 *SceNeRFlow*

**Architecture.** Both the coarse and fine deformation fields use the same architecture. The hash grid consists of 16 levels, with two feature dimensions per level. The coarsest level has a resolution of $32^3$, and each subsequent level has a 1.3819 times higher resolution. The hashmap has a size of $2^{20}$. The shallow MLP has one hidden layer with 64 hidden dimensions.

The canonical model uses a hash grid with 13 levels, two feature dimensions per level, a coarsest resolution of $128^3$, and a scaling factor of 1.3819. The shallow MLP that outputs the opacity has one hidden layer with 64 hidden dimensions. A second MLP outputs the appearance and takes as input a 15-dimensional vector additionally regressed by the first shallow MLP. The second MLP has two hidden layers with 64 hidden dimensions.

**Weighting Scheme for Smoothness Loss.** Each sample $i$ on ray $r$ is weighted depending on its closeness to the object. Mathematically, the initial weight is $\hat{\sigma}_{r,i} = \exp(-\sigma_{r,i}\delta)$, where $\sigma_{r,i}$ is the opacity of the $i$-th sample on ray $r$. Next, max-pooling with window size $k = \lfloor f \cdot S \rfloor$ is applied, where $f$=0.005 is empirically set:

$$\hat{\sigma}'_{r,i} = \max_{i' \in [i-k,\dots,i+k]} \hat{\sigma}_{r,i'}. \tag{D.1}$$

Then, the regularization is weakened on empty space by $u$=10:

$$\hat{\sigma}''_{r,i} = \begin{cases} \frac{1}{u}\hat{\sigma}'_{r,i} & \text{if } \hat{\sigma}'_{r,i} > u \cdot \hat{\sigma}_{r,i} \\ \hat{\sigma}'_{r,i} & \text{else.} \end{cases} \tag{D.2}$$

| | | | SceNeRFlow | NR-NeRF | SNF-A | SNF-AG | Background |
|---|---|---|---|---|---|---|---|
| Seq. 1 | Unmasked | PSNR ↑ | 27.49 | 26.84 | **27.84** | 26.73 | 18.64 |
| | | SSIM ↑ | **0.928** | 0.915 | **0.928** | 0.906 | 0.894 |
| | | LPIPS ↓ | **0.074** | 0.117 | 0.076 | 0.138 | 0.139 |
| | Masked | PSNR ↑ | 29.73 | 29.38 | **30.02** | 29.89 | — |
| | | SSIM ↑ | 0.970 | 0.966 | **0.971** | 0.970 | — |
| | | LPIPS ↓ | 0.021 | 0.036 | **0.017** | 0.018 | — |
| Seq. 2 | Unmasked | PSNR ↑ | 27.93 | 21.64 | **28.24** | 26.99 | 18.57 |
| | | SSIM ↑ | **0.929** | 0.875 | **0.929** | 0.907 | 0.898 |
| | | LPIPS ↓ | **0.069** | 0.183 | 0.074 | 0.136 | 0.130 |
| | Masked | PSNR ↑ | 29.65 | 22.75 | **30.23** | 29.96 | — |
| | | SSIM ↑ | 0.970 | 0.931 | **0.972** | 0.970 | — |
| | | LPIPS ↓ | 0.019 | 0.081 | **0.017** | 0.019 | — |
| Seq. 3 | Unmasked | PSNR ↑ | **27.72** | 21.48 | 27.50 | 26.61 | 18.94 |
| | | SSIM ↑ | **0.923** | 0.874 | 0.920 | 0.904 | 0.895 |
| | | LPIPS ↓ | **0.075** | 0.181 | 0.089 | 0.136 | 0.140 |
| | Masked | PSNR ↑ | **28.95** | 22.25 | 28.90 | 28.47 | — |
| | | SSIM ↑ | **0.961** | 0.925 | **0.961** | 0.958 | — |
| | | LPIPS ↓ | **0.029** | 0.089 | **0.029** | 0.033 | — |
| Seq. 4 | Unmasked | PSNR ↑ | 31.59 | 31.79 | 31.92 | **32.08** | 24.68 |
| | | SSIM ↑ | 0.950 | 0.948 | **0.951** | 0.949 | 0.943 |
| | | LPIPS ↓ | **0.034** | 0.055 | 0.036 | 0.044 | 0.070 |
| | Masked | PSNR ↑ | 33.68 | 33.67 | 34.15 | **34.41** | — |
| | | SSIM ↑ | 0.980 | 0.978 | **0.981** | **0.981** | — |
| | | LPIPS ↓ | **0.011** | 0.031 | 0.012 | 0.014 | — |
| Seq. 5 | Unmasked | PSNR ↑ | 32.25 | 31.73 | 32.58 | **32.77** | 23.33 |
| | | SSIM ↑ | 0.946 | 0.944 | **0.947** | 0.945 | 0.938 |
| | | LPIPS ↓ | **0.042** | 0.057 | 0.044 | 0.049 | 0.076 |
| | Masked | PSNR ↑ | 34.08 | 33.29 | 34.49 | **34.95** | — |
| | | SSIM ↑ | 0.976 | 0.973 | **0.978** | **0.978** | — |
| | | LPIPS ↓ | **0.017** | 0.033 | **0.017** | **0.017** | — |
| Seq. 6 | Unmasked | PSNR ↑ | 28.52 | 26.69 | **29.05** | 27.70 | 19.76 |
| | | SSIM ↑ | **0.938** | 0.917 | 0.937 | 0.918 | 0.916 |
| | | LPIPS ↓ | **0.060** | 0.123 | 0.069 | 0.120 | 0.105 |
| | Masked | PSNR ↑ | 30.24 | 28.85 | **31.19** | 30.99 | — |
| | | SSIM ↑ | 0.978 | 0.969 | **0.980** | 0.979 | — |
| | | LPIPS ↓ | 0.015 | 0.035 | **0.013** | **0.013** | — |
| Seq. 7 | Unmasked | PSNR ↑ | 34.38 | 34.79 | 35.32 | **35.43** | 26.08 |
| | | SSIM ↑ | 0.959 | 0.958 | **0.960** | 0.956 | 0.950 |
| | | LPIPS ↓ | **0.026** | 0.032 | **0.026** | 0.036 | 0.057 |
| | Masked | PSNR ↑ | 37.88 | 38.01 | 39.33 | **39.83** | — |
| | | SSIM ↑ | 0.991 | 0.990 | **0.992** | **0.992** | — |
| | | LPIPS ↓ | 0.005 | 0.007 | **0.003** | 0.004 | — |
| Seq. 8 | Unmasked | PSNR ↑ | 32.67 | **35.18** | 34.58 | 34.38 | 23.53 |
| | | SSIM ↑ | 0.937 | 0.942 | **0.944** | 0.941 | 0.926 |
| | | LPIPS ↓ | 0.046 | **0.041** | 0.042 | 0.047 | 0.088 |
| | Masked | PSNR ↑ | 34.82 | **38.17** | 38.13 | 37.26 | — |
| | | SSIM ↑ | 0.980 | 0.984 | **0.986** | 0.984 | — |
| | | LPIPS ↓ | 0.022 | 0.017 | **0.013** | 0.018 | — |

Table D.1: **Novel-View Synthesis.** The table reports per-scene PSNR, SSIM, and LPIPS.

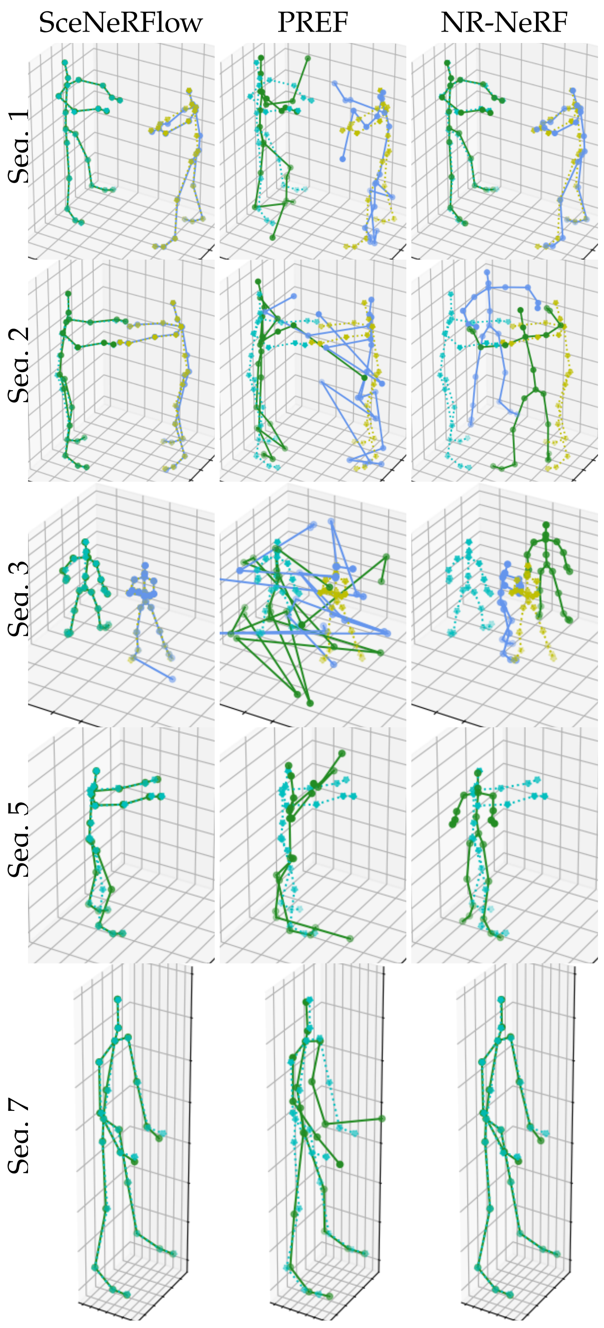


Figure D.2: **Time Consistency.** The solid skeleton is the tracking estimate at $t=T$. The dotted skeleton is the pseudo-ground truth at $t=T$.

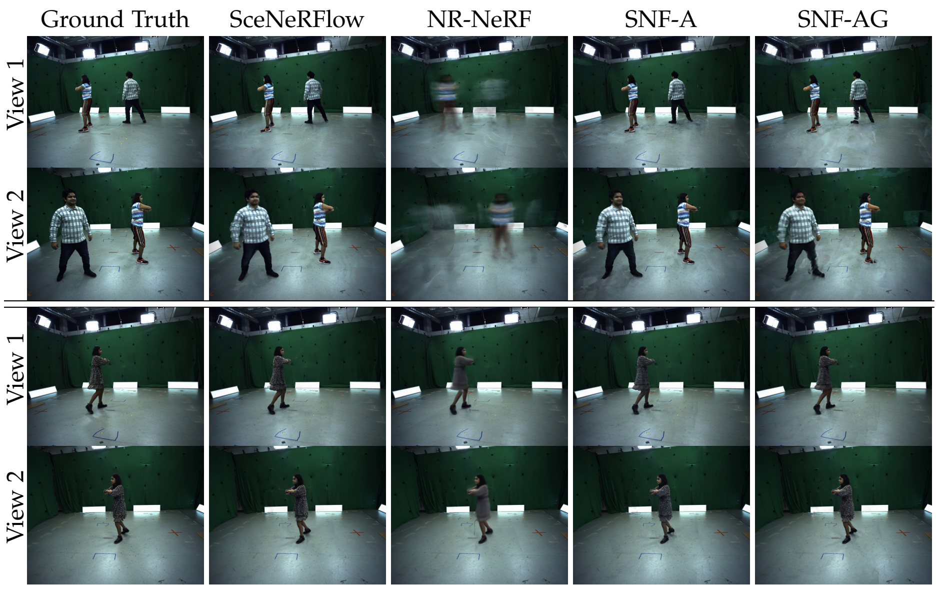


Figure D.3: **Novel-View Synthesis.** (First row) Seq. 3 at $t{=}\frac{T}{2}$. (Second row) Seq. 4 at $t{=}\frac{T}{2}$.

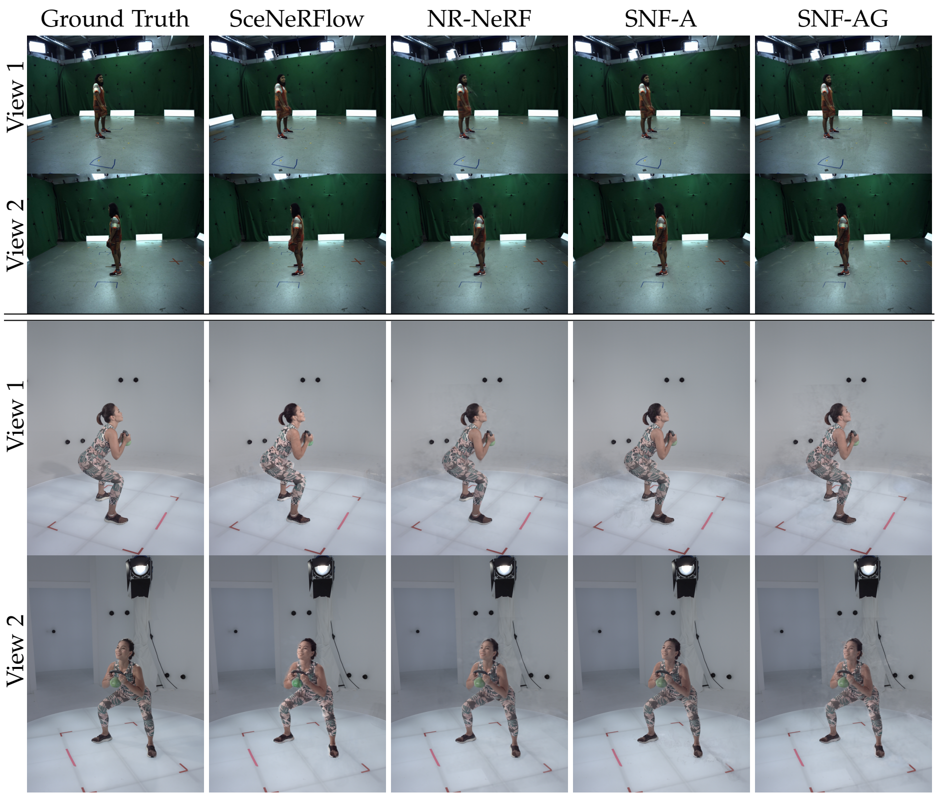


Figure D.4: **Novel-View Synthesis.** (First row) Seq. 7 at $t{=}\frac{T}{2}$. (Second row) Seq. 8 at $t{=}T$.

Finally, the regularization is weakened on very small offsets $\Delta \in \mathbb{R}^3$ with a soft threshold of $s_t$=0.001:

$$\hat{\sigma}'''_{r,i}(\Delta) = \text{sig}\left(\frac{4\|\Delta\|_2}{s_t} - 2\right)\hat{\sigma}''_{r,i}, \tag{D.3}$$

where $\text{sig}(x)=\frac{1}{1+\exp(-x)}$ is the sigmoid function. The weighting thus is:

$$\mathcal{L}_{\text{norm,w}} = \frac{1}{RS}\sum_r\sum_i \hat{\sigma}'''_{r,i}(\Delta)\mathbb{E}_{\mathbf{e}}\left[\left|\|\mathbf{J}^\top_{\mathbf{r}_r(s_i)}\mathbf{e}\|_2 - 1\right|\right], \tag{D.4}$$

where $\Delta=\Delta_c(\mathbf{r}_r(s_i))$ when regularizing the coarse deformations, and $\Delta=\Delta_f(d_c(\mathbf{r}_r(s_i)))$ when regularizing the fine deformations.

**Learning Rates.** The optimization uses exponential decay for the learning rates when constructing the canonical model and for each timestamp $t>1$. For $t=1$, they are decayed by a factor of 0.01 for all parameters. For $t>1$, the coarse deformations decay by 0.1 when they get optimized, and the fine deformations by 0.1 when they get optimized. All parameters of the canonical model have an initial learning rate of $10^{-2}$. All deformation parameters have an initial learning rate of $10^{-3}$. The vignetting parameters have an initial learning rate of $10^{-2}$ and are the only parameters with no weight decay.

During the first 1000 iterations when building the canonical model, the learning rates are warmed up with an additional factor that exponentially increases from 0.01 to 1.

### D.4.2 *Variants*

The experiments compare against two variants of the proposed method that use time-varying canonical models and thus neglect correspondences.

The first variant, SNF-A, allows the appearance but not the geometry of the canonical model to change for $t>1$. This is implemented via a separate appearance model that has the same HashMLP architecture as the standard canonical model. Then, the standard canonical model, which now only predicts the geometry, is kept fixed for $t>1$ while allowing the appearance to vary.

The second variant, SNF-AG, has time-varying appearance and geometry. Thus, the standard canonical model can be used. The geometry regularization losses $\mathcal{L}_{\text{back}}$ and $\mathcal{L}_{\text{hard}}$ are applied to the canonical model at all timestamps $t \geq 1$.

For both variants, as much of the reconstruction as possible should be explained via the deformations, such that the canonical model needs to change as little as possible. Thus, the $10k$ iterations per timestamp are split into three equally long phases: only coarse deformations (as before), then only fine deformations (as before), then only the canonical model. *I.e.*, the canonical model gets optimized only during the third phase, when the deformations are fixed.

Empirically, these variants are unstable to train, with frequent divergence, but the following remedies help: The canonical hash grid(s) use the settings from Instant NGP (Müller et al., 2022) (a coarsest resolution of 16, with the resolution of each finer level being 1.5 times finer than the previous level, with a total of 16 levels). During the third phase, Instant NGP's Huber loss is used with a threshold of 0.1 as reconstruction loss instead of SceNeRFlow's $\ell_1$ reconstruction loss. The learning rate exponentially decays for the third phase, going from $10^{-2}$ to $10^{-3}$ (except for the 300-frame Seq. 3, where a ten times lower learning rate is used for long-term training stability).

### D.4.3 *NR-NeRF*

NR-NeRF is adapted to the multi-view setting as follows: Since background images are provided to NR-NeRF, its rigidity network is not necessary and hence removed. Furthermore, due to the large-motion setting, its offsets loss that encourages the deformations to remain small is removed. To keep runtime reasonable, pruning is used. Doing so also allows to always sample densely and thus hierarchical sampling is not applied. Foreground-focused batches and vignetting correction are also used. Since the recommended (Tretschk et al., 2021) training time of $200k$ iterations is only shown for very short scenes, NR-NeRF is instead trained for longer on scenes in these experiments. Specifically, NR-NeRF's training time is extended to that of SceNeRFlow and thus NR-NeRF trains for one third of the number of iterations of SceNeRFlow. For Seq. 8, NR-NeRF is trained for $200k$ iterations, as that is longer.

### D.4.4 *PREF*

Like for NR-NeRF, background images are supplied, pruning is applied, foreground-focused batches are used, and the vignetting correction is learned. Since the evaluation follows the authors of PREF and splits the long scenes into chunks of 25 frames (see Sec. D.5), the training time is kept the same.

## D.5 JOINT EVALUATION DETAILS

This section provides details on how PREF and NR-NeRF are adapted to long-term 3D joint tracking.

**PREF.** To train on a longer sequence, PREF (Song et al., 2022) splits the sequence into chunks of 25 frames and trains on each chunk independently. The experiments thus do the same with the proposed scenes. To obtain long-term correspondences across chunks, the chunks are overlapped for three frames.

**NR-NeRF.** Unlike SceNeRFlow, NR-NeRF's canonical model does not coincide with the world space at $t$=1. It is therefore not possible to directly use $\{\hat{\mathbf{p}}_j^1\}_j$ as the target canonical positions. Instead, the backward deformation model is applied at $t$=1 to $\{\hat{\mathbf{p}}_j^1\}_j$ to obtain their positions in canonical space. These are then the joint positions in canonical space and the world-space positions at $t$=1, which allows to apply the same tracking procedure as for SceNeRFlow.

## D.6 FOREGROUND MASKS FOR EVALUATION

Since the variants are not tuned beyond making their training stable, they exhibit some significant undesired artifacts in empty space. In addition to scores on the full images, masked scores are therefore also reported that are focused on the actual dynamic object of interest. To this end, foreground masks are required.

Training uses very coarse and inaccurate foreground masks. However, more accurate foreground masks are used when computing masked scores during evaluation. To also consider reconstructions that are slightly off, these masks are dilated to include the areas surrounding the dynamic foreground in pixel space. Fig. D.5 shows example masks.

The following procedure yields these more accurate foreground masks. First, two foreground masks are computed separately: (1) $m_b$ using background subtraction with a threshold $\Delta_t$ followed by a morphological opening (*i.e.*, first erosion, then dilation) of the foreground, and (2) $m_\sigma$. $m_\sigma$ very roughly detects shadows by determining whether the pixel in $\mathbf{I}^{c,t}$ is a scaled version of the background pixel in $\mathbf{B}^c$ (*i.e.*, whether a naïve brightness change of the background has occurred). To this end, each of the three channels of the pixel in $\mathbf{I}^{c,t}$ is divided by the respective channel of the background pixel. Then, the standard deviation of the resulting three factors is thresholded at $\sigma_t$. Finally, an opening is applied to obtain the final $m_\sigma$. Since both masks use a very generous opening, they are combined into a single foreground mask by considering those pixels foreground that are foreground in both masks. $\Delta_t$, $\sigma_t$, and the opening sizes are manually tuned for each sequence.

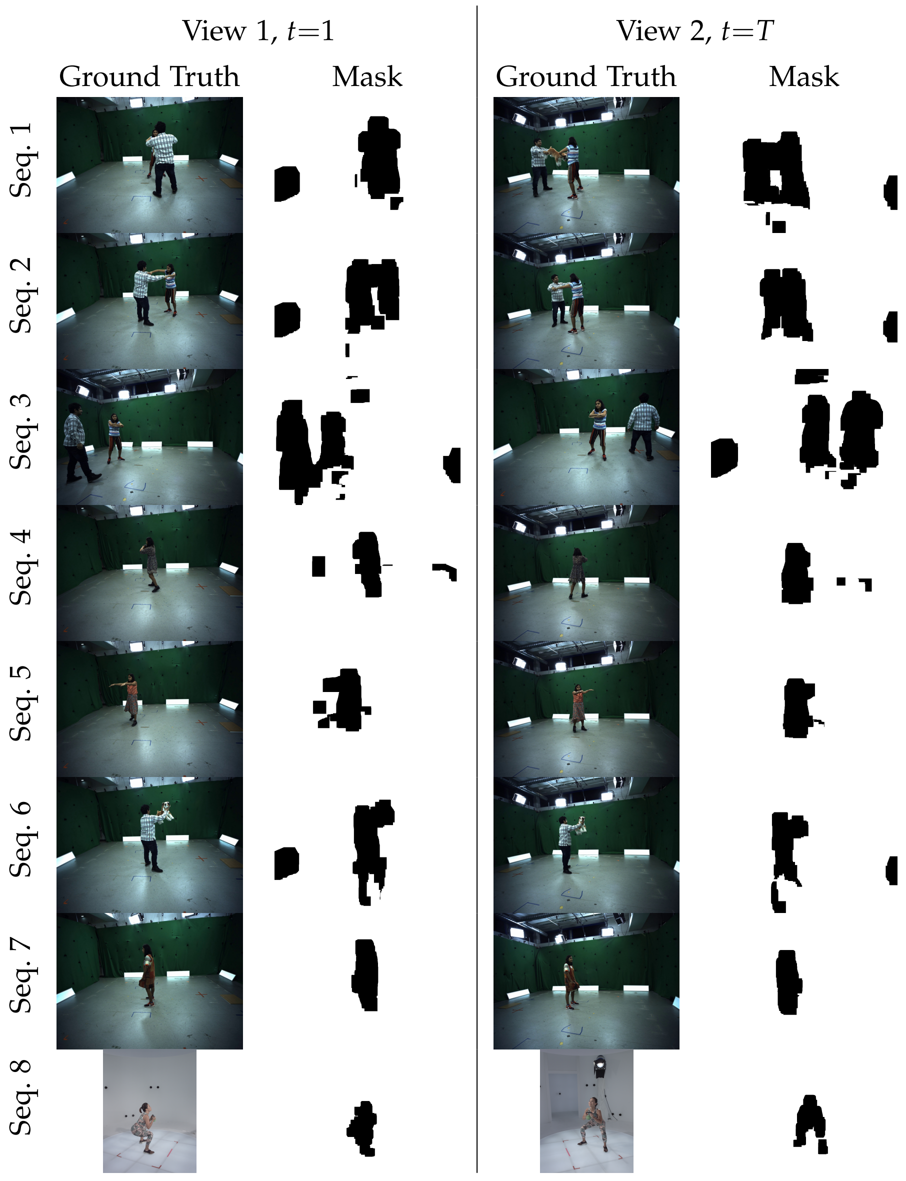


Figure D.5: **Foreground Masks for Evaluation.**

# BIBLIOGRAPHY

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. URL: https://www.tensorflow.org/.

Agarwal, Sameer, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski (2011). "Building Rome in a Day." In: *Commun. ACM* 54.10, 105–112.

Attal, Benjamin, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole (2021). "TöRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Atzmon, Matan and Yaron Lipman (2020). "SAL: Sign Agnostic Learning of Shapes From Raw Data." In: *Computer Vision and Pattern Recognition (CVPR)*.

Bagautdinov, Timur, Chenglei Wu, Jason Saragih, Yaser Sheikh, and Pascal Fua (2018). "Modeling Facial Geometry using Compositional VAEs." In.

Bal, Artur and Henryk Palus (2023). "Image Vignetting Correction Using a Deformable Radial Polynomial Model." In: *Sensors*.

Balakrishnan, Guha, Amy Zhao, Adrian V. Dalca, Frédo Durand, and John V. Guttag (2018). "Synthesizing Images of Humans in Unseen Poses." In: *Computer Vision and Pattern Recognition (CVPR)*.

Bansal, Aayush, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan (2020). "4d visualization of dynamic events from unconstrained multi-view videos." In: *Computer Vision and Pattern Recognition (CVPR)*.

Barron, Jonathan T and Jitendra Malik (2014). "Shape, illumination, and reflectance from shading." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Bartoli, Adrien, Yan Gérard, François Chadebecq, Toby Collins, and Daniel Pizarro (2015). "Shape-from-Template." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Basha, Tali, Yael Moses, and Nahum Kiryati (2013). "Multi-view scene flow estimation: A view centered variational approach." In: *International Journal of Computer Vision (IJCV)*.

Bednařík, J., P. Fua, and M. Salzmann (2018). "Learning to Reconstruct Texture-less Deformable Surfaces." In: *International Conference on 3D Vision (3DV)*.

Bhatia, Harsh, Gregory Norgard, Valerio Pascucci, and Peer-Timo Bremer (2012). "The Helmholtz-Hodge decomposition—a survey." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19.8, pp. 1386–1404.

Bhatia, Harshil, Edith Tretschk, Zorah Lähner, Marcel Benkner, Michael Möller, Christian Theobalt, and Vladislav Golyanik (2023). "CCuantuMM: Cycle-Consistent Quantum-Hybrid Matching of Multiple Shapes." In: *Computer Vision and Pattern Recognition (CVPR)*.

Blanz, Volker and Thomas Vetter (1999). "A morphable model for the synthesis of 3D faces." In: *Proc. Conference on Computer Graphics and Interactive Techniques*.

Bogo, Federica, Javier Romero, Gerard Pons-Moll, and Michael J. Black (2017). "Dynamic FAUST: Registering Human Bodies in Motion." In: *Computer Vision and Pattern Recognition (CVPR)*.

Boscaini, Davide, Jonathan Masci, Emanuele Rodoià, and Michael Bronstein (2016). "Learning Shape Correspondence with Anisotropic Convolutional Neural Networks." In: *International Conference on Neural Information Processing Systems (NIPS)*.

Bouritsas, Giorgos, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou (2019). "Neural 3D Morphable Models: Spiral Convolutional Networks for 3D Shape Representation Learning and Generation." In: *International Conference on Computer Vision (ICCV)*.

Bozic, Aljaz, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner (2020a). "Neural non-rigid tracking." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Bozic, Aljaz, Michael Zollhofer, Christian Theobalt, and Matthias Nießner (2020b). "Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data." In: *Computer Vision and Pattern Recognition (CVPR)*.

Bronstein, Michael M, Joan Bruna, Taco Cohen, and Petar Veličković (2021). "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges." In: *arXiv preprint*.

Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun (2013). "Spectral Networks and Locally Connected Networks on Graphs." In: *CoRR* abs/1312.6203.

Buehler, Chris, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen (2001). "Unstructured lumigraph rendering." In: *SIGGRAPH*.

Cagniart, Cedric, Edmond Boyer, and Slobodan Ilic (2010). "Probabilistic deformable surface tracking from multiple videos." In: *European Conference on Computer Vision (ECCV)*.

Cai, Hongrui, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang (2022). "Neural Surface Reconstruction of Dynamic Scenes with Monocular RGB-D Camera." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Captury, The (2023). *Captury Studio*. https://captury.com/.

Carceroni, Rodrigo L. and Kiriakos N. Kutulakos (2002). "Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape and Reflectance." In: *International Journal of Computer Vision* 49.2, pp. 175–214.

Casillas-Perez, David, Daniel Pizarro, David Fuentes-Jimenez, Manuel Mazo, and Adrien Bartoli (2021). "The Isowarp: the template-based visual geometry of isometric surfaces." In: *International Journal of Computer Vision (IJCV)*.

Chan, Caroline, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros (2019). "Everybody Dance Now." In: *International Conference on Computer Vision (ICCV)*.

Chang, Angel X, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. (2015). "ShapeNet: An information-rich 3D model repository." In: *arXiv preprint arXiv:1512.03012*.

Chen, Hsiao yu, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner (2022). "Virtual Elastic Objects." In: *Computer Vision and Pattern Recognition (CVPR)*.

Chen, Zhiqin and Hao Zhang (2019). "Learning implicit fields for generative shape modeling." In: *Computer Vision and Pattern Recognition (CVPR)*.

Choy, Christopher B, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese (2016). "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction." In: *European Conference on Computer Vision (ECCV)*.

Cignoni, Paolo, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia (2008). "MeshLab: an Open-Source Mesh Processing Tool." In: *Eurographics Italian Chapter Con-*

*ference*. Ed. by Vittorio Scarano, Rosario De Chiara, and Ugo Erra. The Eurographics Association. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

Collet, Alvaro, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan (2015). "High-quality streamable free-viewpoint video." In: *ACM Transactions on Graphics (ToG)* 34.4, p. 69.

Corona, Enric, Tomas Hodan, Minh Vo, Francesc Moreno-Noguer, Chris Sweeney, Richard Newcombe, and Lingni Ma (2022). "LISA: Learning implicit shape and appearance of hands." In: *Computer Vision and Pattern Recognition (CVPR)*.

Curless, Brian and Marc Levoy (1996). "A volumetric method for building complex models from range images." In: *Proc. Conference on Computer Graphics and Interactive Techniques*.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering." In: *International Conference on Neural Information Processing Systems (NIPS)*.

Deng, Bailin, Yuxin Yao, Roberto M Dyke, and Juyong Zhang (2022). "A Survey of Non-Rigid 3D Registration." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*.

Deng, Boyang, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi (2019). "Cvxnets: Learnable convex decomposition." In: *Advances in Neural Information Processing Systems Workshops*.

Deng, Boyang, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi (2020). "NASA: Neural Articulated Shape Approximation." In.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database." In: *Computer Vision and Pattern Recognition (CVPR)*.

Deprelle, Theo, Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry (2019). "Learning elementary structures for 3D shape generation and matching." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Dou, Mingsong, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. (2016). "Fusion4d: Real-time performance capture of challenging scenes." In: *ACM Transactions on Graphics*.

Du, Yilun, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu (2021). "Neural radiance flow for 4d view synthesis and video processing." In: *International Conference on Computer Vision (ICCV)*.

Egger, Bernhard, William A. P. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter (Aug. 2020a). "3D Morphable Face Models - Past, Present and Future." In: *ACM Transactions on Graphics* 39.5.

Egger, Bernhard, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. (2020b). "3d morphable face models—past, present, and future." In: *ACM Transactions on Graphics*.

Eslami, SM Ali, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. (2018). "Neural scene representation and rendering." In: *Science* 360.6394, pp. 1204–1210.

Fang, Jiemin, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian (2022). "Fast Dynamic Radiance Fields with Time-Aware Neural Voxels." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*.

Flynn, John, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker (2019). "Deep-View: View Synthesis with Learned Gradient Descent." In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fuentes-Jimenez, David, David Casillas-Perez, Daniel Pizarro, Toby Collins, and Adrien Bartoli (2018). "Deep Shape-from-Template: Wide-Baseline, Dense and Fast Registration and Deformable Reconstruction from a Single Image." In: *arXiv e-prints*.

Gao, Chen, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang (2021). "Dynamic view synthesis from dynamic monocular video." In: *International Conference on Computer Vision (ICCV)*.

Gao, Hang, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa (2022). "Monocular Dynamic View Synthesis: A Reality Check." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Gao, Lin, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L. Rosin, Weiwei Xu, and Shihong Xia (2018). "Automatic Unpaired Shape Deformation Transfer." In: *ACM Transactions on Graphics (TOG)*.

Garg, Ravi, Anastasios Roussos, and Lourdes Agapito (2013). "Dense Variational Reconstruction of Non-rigid Surfaces from Monocular Video." In: *Computer Vision and Pattern Recognition (CVPR)*.

Garland, Michael and Paul S. Heckbert (1997). "Surface Simplification Using Quadric Error Metrics." In: *ACM SIGGRAPH*.

Genova, Kyle, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser (2020). "Local Deep Implicit Functions for 3D Shape." In: *Computer Vision and Pattern Recognition (CVPR)*.

Genova, Kyle, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser (2019). "Learning shape templates with structured implicit functions." In: *International Conference on Computer Vision (ICCV)*.

Goldluecke, Bastian and Marcus Magnor (2004). "Space-time isosurface evolution for temporally coherent 3D reconstruction." In: *Computer Vision and Pattern Recognition (CVPR)*.

Golyanik, Vladislav, Soshi Shimada, Kiran Varanasi, and Didier Stricker (2018). "HDM-Net: Monocular Non-rigid 3D Reconstruction with Learned Deformation Model." In: *International Conference on Virtual Reality and Augmented Reality (EuroVR)*.

Gortler, Steven J., Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen (1996). "The Lumigraph." In: *SIGGRAPH*, 43–54.

Graßhof, Stella and Sami Sebastian Brandt (2022). "Tensor-Based Non-Rigid Structure from Motion." In: *Winter Conference on Applications of Computer Vision (WACV)*.

Grathwohl, Will, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud (2018). "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models." In: *International Conference on Learning Representations (ICLR)*.

Griewank, Andreas and Andrea Walther (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics (SIAM).

Groueix, Thibault, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry (2018). "AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation." In: *Computer Vision and Pattern Recognition (CVPR)*.

Gu, Xiuye, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang (2019). "Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds." In: *Computer Vision and Pattern Recognition (CVPR)*.

Guo, Kaiwen, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, and et al. (2019). "The Relightables: Volumetric Performance Capture of Humans with Realistic Relighting." In: *ACM Trans. Graph.* 38.6.

Guo, Kaiwen, Feng Xu, Yangang Wang, Yebin Liu, and Qionghai Dai (2015). "Robust non-rigid motion tracking and surface reconstruction using l0 regularization." In: *International Conference on Computer Vision (ICCV)*.

Guo, Kaiwen, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu (2017). "Real-Time Geometry, Albedo, and Motion Reconstruction Using a Single RGB-D Camera." In: *ACM Transactions on Graphics*.

Guo, Xiang, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding (2022). "Neural Deformable Voxel Grid for Fast Optimization of Dynamic View Synthesis." In: *Asian Conference on Computer Vision (ACCV)*.

Habermann, Marc, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt (2020). "DeepCap: Monocular Human Performance Capture Using Weak Supervision." In: *Computer Vision and Pattern Recognition (CVPR)*.

Habermann, Marc, Weipeng Xu, Michael Zollhöfer, Gerard Pons-Moll, and Christian Theobalt (Mar. 2019). "LiveCap: Real-Time Human Performance Capture From Monocular Video." In: *ACM Trans. Graph.* 38.2, 14:1–14:17. ISSN: 0730-0301.

Handa, A., T. Whelan, J.B. McDonald, and A.J. Davison (2014). "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM." In: *International Conference on Robotics and Automation (ICRA)*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition." In: *Computer Vision and Pattern Recognition (CVPR)*.

Hedman, Peter, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow (Dec. 2018). "Deep Blending for Free-viewpoint Image-based Rendering." In: *ACM Trans. Graph.* 37.6, 257:1–257:15. ISSN: 0730-0301.

Huguet, Frédéric and Frédéric Devernay (2007). "A variational method for scene flow estimation from stereo sequences." In: *International Conference on Computer Vision (ICCV)*.

Hung, Chun Ho, Li Xu, and Jiaya Jia (2013). "Consistent binocular depth and scene flow with chained temporal profiles." In: *International Journal of Computer Vision (IJCV)*.

Hutchinson, Michael F (1989). "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines." In: *Communications in Statistics-Simulation and Computation* 18.3, pp. 1059–1076.

Igarashi, Takeo, Tomer Moscovich, and John F Hughes (2005). "As-rigid-as-possible shape manipulation." In: *ACM transactions on Graphics (TOG)* 24.3, pp. 1134–1141.

Innmann, Matthias, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger (2016). "VolumeDeform: Real-time Volumetric Non-rigid Reconstruction." In.

Iwase, Shun, Shunsuke Saito, Tomas Simon, Stephen Lombardi, Timur Bagautdinov, Rohan Joshi, Fabian Prada, Takaaki Shiratori, Yaser Sheikh, and Jason Saragih (2023). "RelightableHands: Efficient Neural Relighting of Articulated Hand Models." In: *Computer Vision and Pattern Recognition (CVPR)*.

Jack, Dominic, Jhony K. Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frederic Maire, and Anders Eriksson (2018). "Learning Free-Form Deformations for 3D Object Reconstruction." In: *Asian Conference on Computer Vision (ACCV)*.

Johnson, Erik C.M., Marc Habermann, Soshi Shimada, Vladislav Golyanik, and Christian Theobalt (2022). "Unbiased 4D: Monocular 4D Reconstruction with a Neural Deformation Model." In: *arXiv:2206.08368*.

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual losses for real-time style transfer and super-resolution." In: *European Conference on Computer Vision (ECCV)*.

Kairanda, Navami, Edith Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik (2022). "$\phi$-SfT: Shape-from-Template with a Physics-Based Deformation Model." In: *Computer Vision and Pattern Recognition (CVPR)*.

Kanazawa, Angjoo, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik (2018). "Learning Category-Specific Mesh Reconstruction from Image Collections." In: *European Conference on Computer Vision (ECCV)*.

Kato, Hiroharu, Yoshitaka Ushiku, and Tatsuya Harada (2018). "Neural 3D Mesh Renderer." In: *Computer Vision and Pattern Recognition (CVPR)*.

Kim, Hyeongwoo, Mohamed Elgharib, Hans-Peter Zollöfer Michael Seidel, Thabo Beeler, Christian Richardt, and Christian Theobalt (2019). "Neural Style-Preserving Visual Dubbing." In: *ACM Transactions on Graphics (TOG)* 38.6, 178:1–13.

Kim, Hyeongwoo, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollöfer, and Christian Theobalt (2018). "Deep Video Portraits." In: *ACM Transactions on Graphics (TOG)* 37.

Kingma, Diederick P and Jimmy Ba (2015). "Adam: A method for stochastic optimization." In: *International Conference on Learning Representations (ICLR)*.

Kocabas, Muhammed, Nikos Athanasiou, and Michael J. Black (June 2020). "VIBE: Video Inference for Human Body Pose and Shape Estimation." In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 5252–5262.

Kokkinos, Filippos and Iasonas Kokkinos (2021). “Learning Monocular 3D Reconstruction of Articulated Categories From Motion.” In: *Computer Vision and Pattern Recognition (CVPR).*

Kumar, Suryansh, Anoop Cherian, Yuchao Dai, and Hongdong Li (2018). “Scalable Dense Non-rigid Structure-from-Motion: A Grassmannian Perspective.” In: *Computer Vision and Pattern Recognition (CVPR).*

Kurenkov, Andrey, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher Choy, and Silvio Savarese (2018). “DeformNet: Free-Form Deformation Network for 3D Shape Reconstruction From a Single Image.” In: *Winter Conference on Applications of Computer Vision (WACV).*

Kutulakos, Kiriakos N and Steven M Seitz (2000). “A theory of shape by space carving.” In: *International Journal of Computer Vision (IJCV).*

Larsen, E Scott, Philippos Mordohai, Marc Pollefeys, and Henry Fuchs (2007). “Temporally consistent reconstruction from multiple video streams using enhanced belief propagation.” In: *International Conference on Computer Vision (ICCV).*

Lawrence, Jason, Danb Goldman, Supreeth Achar, Gregory Major Blascovich, Joseph G Desloge, Tommy Fortes, Eric M Gomez, Sascha Häberling, Hugues Hoppe, Andy Huibers, et al. (2021). “Project starline: a high-fidelity telepresence system.” In: *ACM Transactions on Graphics.*

Levoy, Marc and Pat Hanrahan (1996). “Light Field Rendering.” In: *SIGGRAPH*, 31–42.

Li, Guannan, Chenglei Wu, Carsten Stoll, Yebin Liu, Kiran Varanasi, Qionghai Dai, and Christian Theobalt (2013). “Capturing relightable human performances under general uncontrolled illumination.” In: *Comput. Graph. Forum* 32.2, pp. 275–284.

Li, Tianye, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. (2022). “Neural 3D Video Synthesis From Multi-View Video.” In: *Computer Vision and Pattern Recognition (CVPR).*

Li, Xueqian, Jhony Kaesemodel Pontes, and Simon Lucey (2021a). “Neural scene flow prior.” In: *Advances in Neural Information Processing Systems (NeurIPS).*

Li, Xueting, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz (2020). “Online Adaptation for Consistent Mesh Reconstruction in the Wild.” In: *Advances in Neural Information Processing Systems (NeurIPS).*

Li, Zhengqi, Simon Niklaus, Noah Snavely, and Oliver Wang (2021b). “Neural scene flow fields for space-time view synthesis of dynamic scenes.” In: *Computer Vision and Pattern Recognition (CVPR).*

Lin, Wenbin, Chengwei Zheng, Jun-Hai Yong, and Feng Xu (2022). "OcclusionFusion: Occlusion-aware Motion Estimation for Real-time Dynamic 3D Reconstruction." In.

Liu, Jia-Wei, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou (2022). "DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, Lingjie, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt (2020). "Neural sparse voxel fields." In: *Advances in Neural Information Processing Systems* 33.

Liu, Lingjie, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt (2021). "Neural Actor: Neural Free-view Synthesis of Human Actors with Pose Control." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*.

Liu, Shichen, Shunsuke Saito, Weikai Chen, and Hao Li (2019a). "Learning to infer implicit surfaces without 3D supervision." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, Xingyu, Charles R Qi, and Leonidas J Guibas (2019b). "FlowNet3D: Learning Scene Flow in 3D Point Clouds." In: *Computer Vision and Pattern Recognition (CVPR)*.

Liu, Yebin, Qionghai Dai, and Wenli Xu (2010). "A Point-Cloud-Based Multiview Stereo Algorithm for Free-Viewpoint Video." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 16.3, pp. 407–418.

Lombardi, Stephen, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh (2019). "Neural volumes: Learning dynamic renderable volumes from images." In: *ACM Transactions on Graphics (TOG)*.

Lombardi, Stephen, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih (2021). "Mixture of volumetric primitives for efficient neural rendering." In: *ACM Transactions on Graphics*.

Loper, Matthew, Naureen Mahmood, and Michael J. Black (2014). "MoSh: Motion and Shape Capture from Sparse Markers." In: *ACM Transactions on Graphics (TOG)*.

Loper, Matthew, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black (2015). "SMPL: A Skinned Multi-person Linear Model." In: *ACM Transactions on Graphics (TOG)*.

Lorensen, William E and Harvey E Cline (1987). "Marching cubes: A high resolution 3D surface construction algorithm." In: *Conference on Computer Graphics and Interactive Techniques*.

Loshchilov, Ilya and Frank Hutter (2019). "Decoupled Weight Decay Regularization." In: *International Conference on Learning Representations (ICLR)*.

Ma, Liqian, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz (2018). "Disentangled Person Image Generation." In: *Computer Vision and Pattern Recognition (CVPR)*.

Maas, Andrew L, Awni Y Hannun, Andrew Y Ng, et al. (2013). "Rectifier nonlinearities improve neural network acoustic models." In: *International Conference on Machine Learning (ICML)*.

Malik, Jameel, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kiarash Tamaddon, Alexis Héloir, and Didier Stricker (2018). "DeepHPS: End-to-end Estimation of 3D Hand Pose and Shape by Learning from Synthetic Depth." In: *International Conference on 3D Vision (3DV)*.

Martin Brualla, Ricardo, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, Sean Fanello, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, and Anastasia Tkach (2018). "LookinGood: Enhancing performance capture with real-time neural re-rendering." In: vol. 37.

Masci, Jonathan, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst (2015). "Geodesic Convolutional Neural Networks on Riemannian Manifolds." In: *International Conference on Computer Vision Workshop (ICCVW)*.

Matsuyama, Takashi, Xiaojun Wu, Takeshi Takai, and Toshikazu Wada (2004). "Real-time dynamic 3-D object shape reconstruction and high-fidelity texture mapping for 3-D video." In: *IEEE Transactions on Circuits and Systems for Video Technology* 14.3, pp. 357–369.

Menapace, Willi, Stéphane Lathuilière, Aliaksandr Siarohin, Christian Theobalt, Sergey Tulyakov, Vladislav Golyanik, and Elisa Ricci (2022). "Playable Environments: Video Manipulation in Space and Time." In: *Computer Vision and Pattern Recognition (CVPR)*.

Mescheder, Lars, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger (2019). "Occupancy networks: Learning 3D reconstruction in function space." In: *Computer Vision and Pattern Recognition (CVPR)*.

Meshry, Moustafa, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla (2019). "Neural Rerendering in the Wild." In: *Computer Vision and Pattern Recognition (CVPR)*.

Michalkiewicz, Mateusz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson (2019). "Implicit surface representations as layers in neural networks." In: *International Conference on Computer Vision (ICCV)*.

Mildenhall, Ben, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng (2020). "Nerf: Representing scenes as neural radiance fields for view synthesis." In: *European Conference on Computer Vision (ECCV)*.

Miller, Graham, Adrian Hilton, and Jonathan Starck (2005). "Interactive free-viewpoint video." In: *IEEE European Conf. on Visual Media Production*, pp. 50–59.

Mittal, Himangi, Brian Okorn, and David Held (2020). "Just Go With the Flow: Self-Supervised Scene Flow Estimation." In: *Computer Vision and Pattern Recognition (CVPR)*.

Monti, Federico, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein (2017). "Geometric deep learning on graphs and manifolds using mixture model cnns." In: *Computer Vision and Pattern Recognition (CVPR)*.

Müller, Thomas (2021). *tiny-cuda-nn*. Version 1.7. URL: `https://github.com/NVlabs/tiny-cuda-nn`.

Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller (2022). "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding." In: *ACM Transactions on Graphics*.

Mustafa, Armin, Hansung Kim, Jean-Yves Guillemaut, and Adrian Hilton (2016). "Temporally coherent 4d reconstruction of complex dynamic scenes." In: *Computer Vision and Pattern Recognition (CVPR)*.

Nagano, Koki, Graham Fyffe, Oleg Alexander, Jernej Barbiç, Hao Li, Abhijeet Ghosh, and Paul Debevec (2015). "Skin Microstructure Deformation with Displacement Map Convolution." In: *ACM Trans. Graph.* 34.4.

Neverova, Natalia, Riza Alp Güler, and Iasonas Kokkinos (2018). "Dense Pose Transfer." In: *ECCV*.

Newcombe, Richard A, Dieter Fox, and Steven M Seitz (2015). "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time." In: *Computer Vision and Pattern Recognition (CVPR)*.

Ngo, Dat Tien, Sanghyuk Park, Anne Jorstad, Alberto Crivellaro, Chang D. Yoo, and Pascal Fua (2015). "Dense Image Registration and Deformable Surface Reconstruction in Presence of Occlusions and Minimal Texture." In: *International Conference on Computer Vision (ICCV)*.

Nguyen-Phuoc, Thu H, Chuan Li, Stephen Balaban, and Yongliang Yang (2018). "RenderNet: A deep convolutional network for differentiable rendering from 3D shapes." In: *Advances in Neural Information Processing Systems (NIPS)*.

Niemeyer, Michael, Lars Mescheder, Michael Oechsle, and Andreas Geiger (2019). "Occupancy flow: 4D reconstruction by learning particle dynamics." In: *International Conference on Computer Vision (CVPR)*.

– (2020). "Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision." In: *Computer Vision and Pattern Recognition (CVPR)*.

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov (2016). "Learning Convolutional Neural Networks for Graphs." In: *International Conference on Machine Learning (ICML)*.

Nieto, Jesús R and Antonio Susín (2012). "Cage based deformations: a survey." In: *Deformation Models: Tracking, Animation and Applications*. Springer.

Ohtake, Yutaka, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel (2003). "Multi-level partition of unity implicits." In: *ACM Transactions on Graphics (TOG)*.

Orts-Escolano, Sergio, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. (2016). "Holoportation: Virtual 3d teleportation in real-time." In: *Annual Symposium on User Interface Software and Technology*.

Oswald, Martin R., Jan Stühmer, and Daniel Cremers (2014). "Generalized Connectivity Constraints for Spatio-temporal 3D Reconstruction." In: *European Conference on Computer Vision (ECCV)*.

Parashar, Shaifali, Mathieu Salzmann, and Pascal Fua (2020). "Local Non-Rigid Structure-From-Motion From Diffeomorphic Mappings." In: *Computer Vision and Pattern Recognition (CVPR)*.

Park, Jeong Joon, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove (2019). "DeepSDF: Learning continuous signed distance functions for shape representation." In: *Computer Vision and Pattern Recognition (CVPR)*.

Park, Keunhong, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla (2021a). "Nerfies: Deformable Neural Radiance Fields." In: *International Conference on Computer Vision (ICCV)*.

Park, Keunhong, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz (2021b). "HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields." In: *ACM Transactions on Graphics*.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic differentiation in pytorch." In: *Advances in Neural Information Processing Systems (NeurIPS) Workshops*.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Perriollat, Mathieu, Richard Hartley, and Adrien Bartoli (2011). "Monocular Template-based Reconstruction of Inextensible Surfaces." In: *International Journal of Computer Vision (IJCV)*.

Pfister, Hanspeter, Matthias Zwicker, Jeroen van Baar, and Markus Gross (2000). "Surfels: Surface Elements as Rendering Primitives." In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 335–342. ISBN: 1581132085.

Pumarola, Albert, Antonio Agudo, Lorenzo Porzi, Alberto Sanfeliu, Vincent Lepetit, and Francesc Moreno-Noguer (2018). "Geometry-Aware Network for Non-Rigid Shape Prediction From a Single View." In: *Computer Vision and Pattern Recognition (CVPR)*.

Pumarola, Albert, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer (2021). "D-nerf: Neural radiance fields for dynamic scenes." In: *Computer Vision and Pattern Recognition (CVPR)*.

Qi, Charles Ruizhongtai, Li Yi, Hao Su, and Leonidas J Guibas (2017). "PointNet++: Deep hierarchical feature learning on point sets in a metric space." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Qian, Neng, Jiayi Wang, Franziska Mueller, Florian Bernard, Vladislav Golyanik, and Christian Theobalt (2020). "HTML: A Parametric Hand Texture Model for 3D Hand Reconstruction and Personalization." In: *European Conference on Computer Vision (ECCV)*.

Quiroga, Julian, Thomas Brox, Frédéric Devernay, and James Crowley (2014). "Dense semi-rigid scene flow estimation from rgbd images." In: *European Conference on Computer Vision (ECCV)*.

Ranjan, Anurag, Timo Bolkart, Soubhik Sanyal, and Michael J. Black (2018). "Generating 3D Faces using Convolutional Mesh Autoencoders." In: *European Conference on Computer Vision (ECCV)*.

Rebain, Daniel, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi (2022). "LOLNeRF: Learn from One Look." In.

Riegler, Gernot and Vladlen Koltun (2020). "Free View Synthesis." In: *European Conference on Computer Vision (ECCV)*.

Riegler, Gernot, Ali Osman Ulusoy, and Andreas Geiger (2017). "OctNet: Learning deep 3D representations at high resolutions." In: *Computer Vision and Pattern Recognition (CVPR)*.

Romero, Javier, Dimitrios Tzionas, and Michael J. Black (2017). "Embodied Hands: Modeling and Capturing Hands and Bodies Together." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*.

Russell, Chris, João Fayad, and Lourdes Agapito (2012). "Dense Non-rigid Structure from Motion." In: *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*.

Saito, Shunsuke, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li (2019). "PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization." In: *International Conference on Computer Vision (ICCV)*.

Salimans, Tim and Durk P Kingma (2016). "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Salzmann, Mathieu, Vincent Lepetit, and Pascal Fua (2007). "Deformable surface tracking ambiguities." In: *Computer Vision and Pattern Recognition (CVPR)*.

Sarkar, Kripasindhu, Dushyant Mehta, Weipeng Xu, Vladislav Golyanik, and Christian Theobalt (2020). "Neural Re-Rendering of Humans from a Single Image." In: *European Conference on Computer Vision (ECCV)*.

Schönberger, Johannes Lutz and Jan-Michael Frahm (2016). "Structure-from-Motion Revisited." In: *Computer Vision and Pattern Recognition (CVPR)*.

Schönberger, Johannes Lutz, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm (2016). "Pixelwise View Selection for Unstructured Multi-View Stereo." In: *European Conference on Computer Vision (ECCV)*.

Seelbach Benkner, Marcel, Maximilian Krahn, Edith Tretschk, Zorah Lähner, Michael Moeller, and Vladislav Golyanik (2023). "QuAnt: Quantum Annealing with Learnt Couplings." In: *International Conference on Learning Representations (ICLR)*.

Shade, Jonathan, Steven Gortler, Li-wei He, and Richard Szeliski (1998). "Layered Depth Images." In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 231–242. ISBN: 0897919998.

Shimada, Soshi, Vladislav Golyanik, Christian Theobalt, and Didier Stricker (2019). "IsMo-GAN: Adversarial Learning for Monocular Non-Rigid 3D Reconstruction." In: *Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Shimada, Soshi, Vladislav Golyanik, Edith Tretschk, Didier Stricker, and Christian Theobalt (2019). "DispVoxNets: Non-Rigid Point Set Alignment with Supervised Learning Proxies." In: *International Conference on 3D Vision (3DV)*.

Shysheya, Aliaksandra, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, Alexander Vakhitov, and Victor Lempitsky (2019). "Textured Neural Avatars." In: *Computer Vision and Pattern Recognition (CVPR)*.

Sidhu, Vikramjit, Edith Tretschk, Vladislav Golyanik, Antonio Agudo, and Christian Theobalt (2020). "Neural Dense Non-Rigid Structure from Motion with Latent Space Constraints." In: *European Conference on Computer Vision (ECCV)*.

Sigal, Leonid, Alexandru O Balan, and Michael J Black (2010). "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion." In: *International Journal of Computer Vision (IJCV)*.

Sinha, Ayan, Asim Unmesh, Qixing Huang, and Karthik Ramani (2017). "SurfNet: Generating 3D Shape Surfaces Using Deep Residual Networks." In: *Computer Vision and Pattern Recognition (CVPR)*.

Sitzmann, Vincent, Justus Thies, Felix Heide, Matthias Niessner, Gordon Wetzstein, and Michael Zollhofer (2019a). "DeepVoxels: Learning Persistent 3D Feature Embeddings." In: *Computer Vision and Pattern Recognition (CVPR)*.

Sitzmann, Vincent, Michael Zollhöfer, and Gordon Wetzstein (2019b). "Scene representation networks: Continuous 3D-structure-aware neural scene representations." In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Slavcheva, Miroslava, Maximilian Baust, Daniel Cremers, and Slobodan Ilic (2017). "Killingfusion: Non-rigid 3d reconstruction without correspondences." In: *Computer Vision and Pattern Recognition (CVPR)*.

Slavcheva, Miroslava, Maximilian Baust, and Slobodan Ilic (2018). "SobolevFusion: 3D Reconstruction of Scenes Undergoing Free Non-Rigid Motion." In: *Computer Vision and Pattern Recognition (CVPR)*.

Smolic, Aljoscha, Karsten Mueller, Philipp Merkle, Christoph Fehn, Peter Kauff, Peter Eisert, and Thomas Wiegand (2006). "3D video and free viewpoint video-technologies, applications and MPEG standards." In: *IEEE International Conference on Multimedia and Expo*. IEEE, pp. 2161–2164.

Song, Liangchen, Xuan Gong, Benjamin Planche, Meng Zheng, David Doermann, Junsong Yuan, Terrence Chen, and Ziyan Wu (2022). "PREF:

Predictability Regularized Neural Motion Fields." In: *European Conference on Computer Vision (ECCV)*.

Sorkine, Olga and Marc Alexa (2007). "As-rigid-as-possible surface modeling." In: *Symposium on Geometry Processing (SGP)*.

Speelpenning, Bert (1980). *Compiling fast partial derivatives of functions given by algorithms*. University of Illinois at Urbana-Champaign.

Starck, Jonathan, Gregor Miller, and Adrian Hilton (2006). "Volumetric Stereo with Silhouette and Feature Constraints." In: *British Machine Vision Conference (BMVC)*.

Stumpfel, Jessi, Andrew Jones, Andreas Wenger, Chris Tchou, Tim Hawkins, and Paul Debevec (2006). "Direct HDR capture of the sun and sky." In: *ACM SIGGRAPH Courses*.

Stutz, David and Andreas Geiger (2018). "Learning 3D shape completion under weak supervision." In: *International Journal of Computer Vision (IJCV)*.

Sumner, Robert W., Johannes Schmid, and Mark Pauly (2007). "Embedded Deformation for Shape Manipulation." In: *ACM SIGGRAPH*.

Tan, Qingyang, Lin Gao, Yu-Kun Lai, and Shihong Xia (2018a). "Variational Autoencoders for Deforming 3D Mesh Models." In: *Computer Vision and Pattern Recognition (CVPR)*.

Tan, Qingyang, Lin Gao, Yu-Kun Lai, Jie Yang, and Shihong Xia (2018b). "Mesh-Based Autoencoders for Localized Deformation Component Analysis." In: *AAAI Conference on Artificial Intelligence (AAAI)*.

Tan, Shufeng and Michael L. Mayrovouniotis (1995). "Reducing data dimensionality through optimizing neural network inputs." In: *AIChE Journal* 41.6, pp. 1471–1480.

Teed, Zachary and Jia Deng (2021). "RAFT-3D: Scene flow using rigid-motion embeddings." In: *Computer Vision and Pattern Recognition (CVPR)*.

Tewari, Ayush, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B Goldman, and Michael Zollhöfer (2020). "State of the Art on Neural Rendering." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*.

Tewari, Ayush, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Nießner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhöfer, and Vladislav Golyanik (2022). "Advances in Neural Rendering." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*.

Theobalt, Christian, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel (2007). "Seeing People in Different Light-Joint Shape, Motion, and Reflectance Capture." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 13.4, 663–674.

Thies, Justus, Michael Zollhöfer, and Matthias Nießner (2019). "Deferred neural rendering: image synthesis using neural textures." In: *ACM Transactions on Graphics* 38.

Tretschk, Edith, Vladislav Golyanik, Michael Zollhöfer, Aljaz Bozic, Christoph Lassner, and Christian Theobalt (2024). "SceNeRFlow: Time-Consistent Reconstruction of General Dynamic Scenes." In: *International Conference on 3D Vision (3DV)*.

Tretschk, Edith, Navami Kairanda, Mallikarjun B R, Rishabh Dabral, Adam Kortylewski, Bernhard Egger, Marc Habermann, Pascal Fua, Christian Theobalt, and Vladislav Golyanik (2023). "State of the Art in Dense Monocular Non-Rigid 3D Reconstruction." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*.

Tretschk, Edith, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt (2021). "Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video." In: *International Conference on Computer Vision (ICCV)*.

Tretschk, Edith, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt (2020a). "PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations." In: *European Conference on Computer Vision (ECCV)*.

Tretschk, Edith, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt (2020b). "DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects." In: *European Conference on Computer Vision (ECCV)*.

Tulsiani, Shubham, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik (2017). "Learning shape abstractions by assembling volumetric primitives." In: *Computer Vision and Pattern Recognition (CVPR)*.

Tung, Tony, Shohei Nobuhara, and Takashi Matsuyama (2009). "Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo." In: *International Conference on Computer Vision (ICCV)*.

Vedula, Sundar, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade (1999). "Three-dimensional scene flow." In: *International Conference on Computer Vision (ICCV)*.

Verma, Nitika, Edmond Boyer, and Jakob Verbeek (2018). "FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis." In: *Computer Vision and Pattern Recognition (CVPR)*.

Vogel, Christoph, Konrad Schindler, and Stefan Roth (2015). "3d scene flow estimation with a piecewise rigid scene model." In: *International Journal of Computer Vision (IJCV)*.

Wang, Liao, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu (2022). "Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-time." In: *Computer Vision and Pattern Recognition (CVPR)*.

Wang, Nanyang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang (2018). "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images." In: *European Conference on Computer Vision (ECCV)*.

Wang, Yufu, Nikos Kolotouros, Kostas Daniilidis, and Marc Badger (2021). "Birds of a feather: Capturing avian shape models from images." In: *Computer Vision and Pattern Recognition (CVPR)*.

Wang, Zhou, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli (2004). "Image quality assessment: from error visibility to structural similarity." In: *IEEE Transactions on Image Processing*.

Waschbüsch, Michael, Stephan Würmlin, Daniel Cotting, Filip Sadlo, and Markus Gross (2005). "Scalable 3D video of dynamic scenes." In: *The Visual Computer* 21.8, pp. 629–638.

Williams, Francis, Jerome Parent-Levesque, Derek Nowrouzezahrai, Daniele Panozzo, Kwang Moo Yi, and Andrea Tagliasacchi (2020). "Voronoinet: General functional approximators with local support." In: *Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Wu, Shangzhe, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi (2021). "DOVE: Learning Deformable 3D Objects by Watching Videos." In: *arXiv preprint arXiv:2107.10844*.

Xian, Wenqi, Jia-Bin Huang, Johannes Kopf, and Changil Kim (2021). "Space-time neural irradiance fields for free-viewpoint video." In: *Computer Vision and Pattern Recognition (CVPR)*.

Xiang, Donglai, Hanbyul Joo, and Yaser Sheikh (2019). "Monocular total capture: Posing face, body, and hands in the wild." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10965–10974.

Xie, Yiheng, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar (2022). "Neural Fields in Visual Computing and Beyond." In: *Computer Graphics Forum (Eurographics State of the Art Reports)*.

Xu, Weipeng, Avishek Chatterjee, Michael Zollhöfer, Helge Rhodin, Dushyant Mehta, Hans-Peter Seidel, and Christian Theobalt (2018). "MonoPerfCap: Human Performance Capture From Monocular Video." In: *ACM Trans. Graph.* 37.2, 27:1–27:15.

Yang, Gengshan, Minh Vo, Neverova Natalia, Deva Ramanan, Vedaldi Andrea, and Joo Hanbyul (2022). “BANMo: Building Animatable 3D Neural Models from Many Casual Videos.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Yen-Chen, Lin (2020). *NeRF-pytorch*. `https://github.com/yenchenlin/nerf-pytorch/`.

Yi, Li, Hao Su, Xingwen Guo, and Leonidas Guibas (2017). “SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Yildirim, Ilker, Max H Siegel, Amir A Soltani, Shraman Ray Chaudhari, and Joshua B Tenenbaum (2023). “3D Shape Perception Integrates Intuitive Physics and Analysis-by-Synthesis.” In: *arXiv preprint*.

Yoon, Jae Shin, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz (2020). “Novel View Synthesis of Dynamic Scenes With Globally Coherent Depths From a Monocular Camera.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Yoon, Jae Shin, Ziwei Li, and Hyun Soo Park (2018). “3d semantic trajectory reconstruction from 3d pixel continuum.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Yu, Alex, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa (2021). “pixelNeRF: Neural Radiance Fields from One or Few Images.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Yu, Rui, Chris Russell, Neill D. F. Campbell, and Lourdes Agapito (2015). “Direct, Dense, and Deformable: Template-Based Non-Rigid 3D Reconstruction from RGB Video.” In: *International Conference on Computer Vision (ICCV)*.

Zhai, Mingliang, Xuezhi Xiang, Ning Lv, and Xiangdong Kong (2021). “Optical flow and scene flow estimation: A survey.” In: *Pattern Recognition*.

Zhang, Kai, Gernot Riegler, Noah Snavely, and Vladlen Koltun (2020). “NeRF++: Analyzing and Improving Neural Radiance Fields.” In: *arXiv preprint arXiv:2010.07492*.

Zhang, Li, Brian Curless, and Steven M Seitz (2003). “Spacetime stereo: Shape recovery for dynamic scenes.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Richard, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang (2018). “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric.” In: *Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Ruo and Ping-Sing Tsai (1999). “Shape-from-shading: a survey.” In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Zhang, Xiuming, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron (2021). “Nerfactor: Neural factorization of shape and reflectance under an unknown illumination." In: *ACM Transactions on Graphics*.

Zhao, Fuqiang, Yuheng Jiang, Kaixin Yao, Jiakai Zhang, Liao Wang, Haizhao Dai, Yuhui Zhong, Yingliang Zhang, Minye Wu, Lan Xu, et al. (2022). "Human Performance Modeling and Rendering via Neural Animated Mesh." In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*.

Zhu, Hao, Hao Su, Peng Wang, Xun Cao, and Ruigang Yang (2018). "View Extrapolation of Human Body from a Single Image." In: *Computer Vision and Pattern Recognition (CVPR)*.

Zollhöfer, Michael, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger (2014). "Real-time Non-rigid Reconstruction using an RGB-D Camera." In: *ACM Transactions on Graphics*.

Zollhöfer, Michael, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb (2018). "State of the art on 3D reconstruction with RGB-D cameras." In.

Zuffi, Silvia, Angjoo Kanazawa, David Jacobs, and Michael J. Black (2017). "3D Menagerie: Modeling the 3D Shape and Pose of Animals." In: *Computer Vision and Pattern Recognition (CVPR)*.